IBM

IBM 시스템 - iSeries

데이터베이스
데이터베이스 프로그래밍

버전 5 릴리스 4

IBM

IBM 시스템 - iSeries

데이터베이스
데이터베이스 프로그래밍

버전 5 릴리스 4

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 347 페이지의 『주의사항』의 정보를 읽으십시오.

제 7 판(2006년 2월)

이 개정판은 새 개정판에서 별도로 명시하지 않는 한 IBM i5/OS 버전 5, 릴리스 4, 수정 0(제품 번호 5722-SS1) 및 모든 후속 릴리스와 수정사항에 적용됩니다. 이 버전은 모든 축약 명령어 세트 컴퓨터(RISC) 모델 및 CISC 모델에서도 실행되지 않습니다.

목차

데이터베이스 프로그래밍

이 데이터베이스 프로그래밍 주제에는 iSeries™용 DB2 Universal Database™(iSeries용 DB2® UDB) 데이터베이스 관리 시스템에 관한 정보가 수록되어 있으며 기존의 시스템 인터페이스를 사용하여 iSeries 서버에서 데이터베이스를 설정하고 사용하는 방법을 설명합니다.

주: 해당 코드 예제를 사용하여 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

V5R4의 새로운 사항

이 주제에서는 V5R4에 대해 이 주제 콜렉션에서 변경된 내용을 강조표시합니다.

| 다음 주제는 갱신됩니다.

| • 9 페이지의 『데이터베이스 파일 *크기*』

| • 13 페이지의 『예: 데이터베이스 파일 *크기*』

| • 304 페이지의 『참조 무결성 및 CL 명령』

| • 306 페이지의 『데이터베이스의 자동 이벤트 트리거』

| • 307 페이지의 『트리거 프로그램 작성』

| • 307 페이지의 『iSeries Navigator를 사용하여 트리거 추가』

| • 308 페이지의 『트리거 프로그램 작업 방식』

| • 329 페이지의 『트리거 버퍼 필드 설명』

| • 332 페이지의 『트리거 추가』

| • 333 페이지의 『트리거 제거』

| • 334 페이지의 『실제 파일 트리거를 작동 가능 또는 작동 불가능하게 설정』

| • 334 페이지의 『트리거 및 CL 명령과의 관계』

| 다음 주제는 139 페이지의 『OPNQRYF(조회 파일 열기) 명령 사용』 주제에 추가됩니다.

| • 195 페이지의 『열린 자료 경로 고려사항』

| • 195 페이지의 『필드명』

| • 196 페이지의 『표현식』

| • 200 페이지의 『내장 함수』

| • 215 페이지의 『제한된 내장 함수』

새로운 사항 또는 변경된 내용을 보는 방법

기술적으로 변경된 내용을 표시하기 위해 이 정보는 다음을 사용합니다.

- » 이미지는 새로운 또는 변경된 정보가 시작되는 위치를 표시합니다.
- « 이미지는 새로운 또는 변경된 정보가 끝나는 위치를 표시합니다.

이 릴리스에서 새로운 사항이나 변경된 내용에 대한 기타 정보를 찾으려면 사용자 메모를 참조하십시오.

인쇄 가능한 PDF

인쇄 가능한 PDF를 사용하여 이 정보의 PDF를 보거나 인쇄할 수 있습니다.

이 문서의 PDF 버전을 보거나 다운로드하려면 데이터베이스 프로그래밍(약 4305KB)을 선택하십시오.

PDF 파일 저장

워크스테이션에서 보거나 인쇄할 PDF를 저장하려면 다음을 수행하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 단추로 클릭하십시오(위의 링크를 마우스 오른쪽 단추로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.
3. PDF를 저장할 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Reader 다운로드

이 PDF를 보거나 인쇄하려면 Adobe Reader를 시스템에 설치해야 합니다. Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html) 에서 사본을 무료로 다운로드할 수 있습니다.

데이터베이스 파일 개념

이 주제에서는 IBM® i5/OS™ 데이터베이스 파일 설정 또는 작업에 대한 몇 가지 기본 데이터베이스 개념을 제공합니다.

iSeries용 DB2 Universal Database

iSeries용 DB2 Universal Database는 i5/OS 오퍼레이팅 시스템의 통합 관계형 데이터베이스 관리자입니다.

iSeries용 DB2 UDB는 기본 오퍼레이팅 시스템의 일부로서 자료에 대한 액세스 및 보호를 제공합니다. 또한 참조 무결성 및 병렬 데이터베이스 처리와 같은 확장 기능을 제공합니다.

iSeries용 DB2 UDB에서는 사용자가 독립 보조 기억장치 풀(ASP) 또는 독립 디스크 풀을 사용하면 각 ASP 그룹과 연관된 하나 이상의 데이터베이스를 별도로 가질 수 있습니다. 1차 독립 디스크 풀을 사용하여 데이터베이스를 설정할 수 있습니다.

관련 개념

독립 디스크 풀

iSeries용 DB2 Universal Database에 대한 인터페이스

iSeries용 DB2 Universal Database는 데이터베이스에 대한 몇 가지 독립적 인터페이스를 제공합니다.

기존 시스템 인터페이스

i5/OS 기존 시스템 인터페이스는 사용자가 iSeries용 DB2 Universal Database 자료에 액세스하고 수정할 수 있게 하는 시스템 명령 및 기타 비SQL 기능의 전체 세트입니다.

기존의 시스템 인터페이스는 데이터베이스 오브젝트를 작성하는 데 사용되는 제어 언어(CL)를 제공합니다. 또한 시스템 인터페이스에는 자료 서술 스펙(DDS)이라고 하는 자료를 서술하는 통합된 기능이 있습니다.

IBM 라이센스 프로그램인 iSeries용 WebSphere® Development Studio는 자료를 서술하고 처리하는 다양한 유틸리티를 제공합니다. 자료 파일 유틸리티(DFU)는 RPG/400®, DDS 및 대화식 자료 서술 유틸리티(IDDU)로 서술한 데이터베이스 파일에 자료를 추가, 변경 및 삭제할 수 있습니다. 소스 입력 유틸리티(SEU)는 파일에서 자료를 지정하고 변경하는 데 사용됩니다.

SQL

SQL(구조화 조회 언어)은 호스트 프로그래밍 언어에 사용하거나 데이터베이스에서 대화식으로 정보를 액세스할 때 사용할 수 있는 언어입니다.

SQL은 관계형 데이터베이스 제품을 액세스하고 수정하는 데 사용되는 산업 표준 데이터베이스 인터페이스입니다. SQL은 관계형 자료 모델을 사용하므로 표에 있는 기존의 모든 자료를 인식합니다. iSeries용 DB2 UDB 데이터베이스에는 시스템 안으로 통합시킨 SQL 처리 기능이 있습니다. SQL문이 들어 있는 컴파일된 프로그램을 처리합니다. SQL 어플리케이션을 개발하려면 어플리케이션을 개발하는 시스템에 DB2 UDB 조회 관리자 및 SQL 개발 킷, IBM 라이센스 프로그램이 필요합니다.

대화식 SQL은 일괄처리 모드 대신 SQL문을 동적으로 실행할 수 있게 해주는 DB2 UDB 조회 관리자 및 SQL 개발 킷 라이센스 프로그램의 기능입니다. 모든 대화식 SQL문은 작업 스테이션으로부터 읽어와서 준비한 후 동적으로 실행됩니다.

관련 개념

SQL 프로그래밍

iSeries SQL(구조화 조회 언어)용 DB2 UDB 소개

iSeries Navigator

iSeries Navigator는 i5/OS 오퍼레이팅 시스템에 번들로 제공되는 Windows®용 무상 기능입니다. iSeries Navigator는 데이터베이스를 포함하여 공통 i5/OS 관리 기능에 대한 그래픽 Microsoft® Windows 인터페이스를 제공합니다.

iSeries Navigator를 사용하여 액세스할 수 있는 대부분의 데이터베이스 조작은 SQL(구조화 조회 언어) 기능에 기초하고 있습니다. 그러나 일부 조작은 제어 언어(CL) 명령과 같은 기존 시스템 인터페이스에 기초하고 있습니다.

관련 개념

iSeries Navigator

iSeries용 IBM 조회

iSeries용 IBM 조회는 보고서 및 기타 파일을 생성하기 위해 데이터베이스 파일에서 정보를 선택, 형식화 및 분석하기 위해 사용되는 IBM 라이센스 프로그램입니다.

데이터베이스 파일

데이터베이스 파일은 시스템 오브젝트 유형 *FILE의 몇 가지 유형 중 하나입니다. 데이터베이스 파일에는 내부 기억장치에서 입력 자료가 프로그램에 부여되는 방식과 프로그램에서 출력 자료가 내부 기억장치로 부여되는 방식에 대한 설명이 들어 있습니다.

데이터베이스 파일에는 멤버 및 레코드가 포함됩니다.

소스 파일

소스 파일에는 특정 유형의 오브젝트를 작성하는 데 필요한 컴파일되지 않은 프로그래밍 코드 및 입력 자료가 들어 있습니다. 소스 파일에는 고급 언어 프로그램 및 자료 서술 스펙(DDS)과 같은 항목을 위한 소스문이 들어 있을 수 있습니다. 소스 파일은 소스 실제 파일, 디스켓 파일, 테이프 파일 또는 인라인 자료 파일 등이 될 수 있습니다.

실제 파일(PF)

실제 파일(PF)은 어플리케이션 자료를 저장하는 데이터베이스 파일입니다. 여기에는 자료가 프로그램에 부여되거나 수신되는 방식과 자료가 데이터베이스에 실제로 저장되는 방식에 대한 설명이 들어 있습니다.

실제 파일(PF)은 다양한 길이 필드를 가질 수 있는 고정 길이 레코드로 구성됩니다. 실제 파일(PF)에는 하나의 레코드 형식과 하나 이상의 멤버가 들어 있습니다. SQL 인터페이스의 관점에서 실제 파일(PF)은 표와 같습니다.

논리 파일

논리 파일은 논리적으로 하나 이상의 실제 파일(PF)을 나타내는 데이터베이스 파일입니다. 여기에는 자료가 프로그램에 부여되거나 수신되는 방식에 대한 설명이 들어 있습니다. 이러한 유형의 데이터베이스 파일에는 자료가 없으며 하나 이상의 실제 파일(PF)에 대한 레코드 형식을 정의합니다.

논리 파일은 표현하는 실제 파일(PF)과는 다른 순서와 형식으로 사용자가 자료를 액세스하게 합니다. SQL 인터페이스의 관점에서 논리 파일은 뷰 및 색인과 같습니다.

멤버

멤버는 여러 가지 자료 세트이며, 각각 하나의 데이터베이스 파일 내에서 동일한 형식을 갖습니다. 파일에 입력이나 출력 조작을 수행하기 전에 파일에 최소한 하나의 멤버가 있어야 합니다.

일반적으로 데이터베이스 파일에는 파일이 작성될 때 작성된 한 멤버만 포함됩니다. 한 파일에 두 개 이상의 멤버가 있으면 각 멤버가 파일에 있는 자료의 서브세트로서 서비스를 제공합니다.

레코드

레코드는 파일 내에서 관련된 자료의 그룹입니다. SQL 인터페이스의 관점에서 레코드는 행과 같습니다.

관련 개념

17 페이지의『소스 파일 사용 이유』

명령 자체만으로는 오브젝트 작성에 필요한 정보를 충분히 제공할 수 없을 경우 소스 파일이 사용됩니다.

데이터베이스 파일 서술 방법

이 주제에서는 데이터베이스 파일에 레코드를 서술하는 방법을 설명합니다.

데이터베이스 파일의 레코드는 다음의 두 가지 방법으로 서술할 수 있습니다.

- 필드 레벨 설명. 레코드 내의 필드를 시스템에 서술합니다. 각 필드에 대해 서술할 수 있는 것으로는 이름, 길이, 자료 유형, 유효성 검사 및 텍스트 설명 등을 들 수 있습니다. 필드 레벨 설명으로 작성되는 데이터베이스 파일을 외부 서술 파일이라고 합니다.
- 레코드 레벨 설명. 파일 내의 레코드 길이만 시스템에 서술합니다. 파일의 필드에 대해서는 시스템이 알지 *못합니다*. 이러한 데이터베이스 파일을 프로그램 서술 파일이라고 합니다.

파일이 필드 레벨 또는 레코드 레벨로 서술되는지에 관계없이 해당 파일을 사용하는 프로그램을 컴파일하기 전에 파일을 서술하고 작성해야 합니다. 즉 파일을 사용하기 전에 그 파일이 시스템에 존재해야 합니다.

외부 및 프로그램 서술 자료

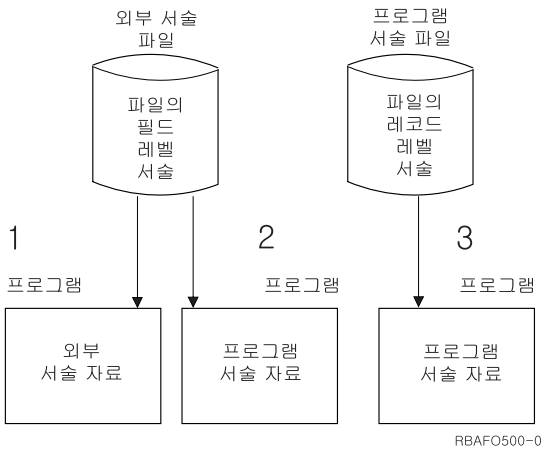
프로그램은 외부 서술 파일이나 프로그램 서술 파일 모두 사용할 수 있습니다.

프로그램은 다음의 두 가지 방법으로 파일 설명을 사용할 수 있습니다.

- 프로그램이 파일의 일부인 필드 레벨 설명을 사용합니다. 필드 설명이 프로그램 자체에 대해 외부적이기 때문에 자료를 외부 서술 자료라고 합니다.
- 프로그램이 프로그램 자체에 서술된 필드를 사용합니다. 따라서 프로그램 서술 자료라고 합니다. 레코드 레벨에만 서술된 파일 안의 필드는 파일을 사용하는 프로그램에 반드시 서술해야 합니다.

그러나 파일을 필드 레벨로 시작하면 시스템이 사용자를 위해 보다 많은 작업을 처리해줍니다. 예를 들어 프로그램을 컴파일할 때 시스템은 외부 서술 파일로부터 정보를 끌어내 프로그램에 필드의 정보를 자동적으로 포함시킵니다. 그러므로 파일을 사용하는 각 프로그램에 필드 정보를 코딩할 필요가 없습니다.

다음 도표는 iSeries 서버에서 파일과 프로그램간의 전형적인 관계를 보여주고 있습니다.



1 외부 서술 자료

프로그램이 시스템에 정의된 파일의 필드 레벨 설명을 사용합니다. 컴파일 시, 언어 컴파일러(language compiler)는 파일의 외부 설명을 프로그램 내에 복사합니다.

2 프로그램 서술 자료

프로그램이 필드 레벨로 시스템에 서술된 파일을 사용하나 실제 필드 설명을 사용하는 것은 아닙니다. 컴파일 시, 언어 컴파일러가 파일의 외부 설명을 프로그램 내에 복사합니다. 파일 내의 필드가 프로그램에 서술되어 있습니다. 이 경우 프로그램에서 사용되는 필드 속성(예: 필드 길이)은 외부 설명의 속성과 같아야 합니다.

3 프로그램 서술 자료

프로그램이 시스템의 레코드 레벨로만 서술된 파일을 사용합니다. 파일의 필드를 반드시 프로그램에 서술해야 합니다.

외부 서술 파일도 프로그램에 서술할 수 있습니다. 이전 시스템과의 호환성을 유지하기 위해 이와 같은 방법을 사용할 수 있습니다. 예를 들어, 종래의 파일 시스템으로부터 온 프로그램을 iSeries 서버에서 수행하려 할 경우이러한 프로그램은 프로그램 서술 자료를 사용하며 파일 자체는 레코드 레벨로만 서술됩니다. 나중에 시스템에서 제공되는 데이터베이스 기능을 좀더 사용하기 위해 필드 레벨로 파일을 서술합니다(외부 서술 파일). 프로그램 서술 자료가 들어 있는 기존 프로그램은 외부 서술 파일을 계속 사용할 수 있고, 새로운 프로그램은 파일의 일부인 필드 레벨 설명을 사용합니다. 여러 번에 걸쳐 필드 레벨 설명을 사용하기 위해서 기존의 여러 프로그램을 변경할 수 있습니다.

사전 서술 자료

프로그램 서술 파일이나 외부 서술 파일은 자료 사전에 저장된 레코드 형식 설명으로 정의할 수 있습니다.

레코드 형식 정보는 대화식 자료 정의 유틸리티(IDDU)를 사용하여 서술할 수 있습니다. 이 파일이 프로그램 서술 파일인 경우에도 iSeries용 IBM 조회, iSeries Access 및 DFU(Data File Utility)는 자료 사전에 저장되어 있는 레코드 형식 설명을 사용합니다.

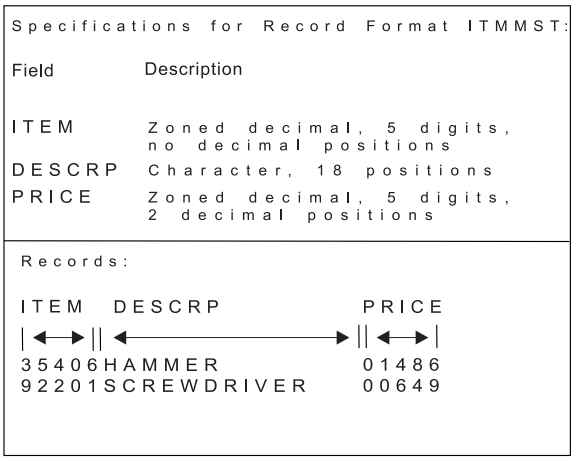
사용자가 IDDU를 사용하여 파일을 서술한 후 작성할 수 있습니다. 이렇게 작성된 파일이 외부 서술 파일입니다. 또한 외부 서술 파일에 저장된 파일 설명을 자료 사전으로 이동시킬 수 있습니다. 시스템은 항상 자료 사전 내의 정의와 외부 서술 파일에 저장된 설명을 동일한 것으로 만듭니다.

레코드 형식 설명

시스템에 데이터베이스 파일을 설명하는 경우 파일의 가장 중요한 두 부분, 즉 레코드 형식과 액세스 경로를 설명하게 됩니다. 레코드 형식은 각 레코드에 있는 필드 순서를 설명합니다.

또한 레코드 형식은 각 필드에 대해 길이, 자료 형태(예: 팩 십진 또는 문자), 유효성 검사, 텍스트 설명 및 기타 정보 등을 상세히 설명합니다.

다음 예는 실제 파일에서 레코드 형식과 레코드 사이의 관계를 보여주고 있습니다.



RBAFO501-0

레코드 형식 ITMMST를 위한 스펙의 예에는 세 개의 필드가 있습니다. 필드 *ITEM*은 존 십진, 소수 자리가 없는 다섯 자리 수입니다. 필드 *DESCRP*는 문자로서 18자리입니다. 필드 *PRICE*는 존 십진, 소수 두 자리의 다섯 자리 수입니다.

실제 파일은 1개의 레코드 형식만 가질 수 있습니다. 실제 파일의 레코드 형식은 자료가 실제로 저장되는 방법을 서술합니다.

논리 파일에는 자료가 들어 있지 않습니다. 논리 파일은 한 개 이상의 실제 파일에서 나온 자료를 각기 다른 형식과 순서로 배열하는 데 사용됩니다. 예를 들어, 논리 파일은 실제 파일의 필드 순서를 변경할 수 있으며, 실제 파일에 저장된 필드의 일부만 프로그램에 사용할 수 있습니다.

논리 파일 레코드 형식은 실제 파일에 저장된 필드의 길이 및 자료 형태를 변경할 수 있습니다. 실제 파일 필드 설명과 논리 파일 필드 설명 사이의 필요한 변환은 시스템이 수행합니다. 예를 들어, 실제 파일은 *FLDA* 필드를 5자리의 팩 십진 필드로 설명할 수 있고 *FLDA*를 사용하는 논리 파일은 이 필드를 7자리의 존 십진 필드로 다시 정의할 수 있습니다. 이 경우 프로그램이 논리 파일을 사용하여 레코드를 읽으면 시스템은 *FLDA*를 존 십진 형식으로 자동으로 변환(팩 해제)합니다.

액세스 경로 설명

시스템에 데이터베이스 파일을 서술할 때 그 파일의 중요한 두 부분 즉, 레코드 형식과 액세스 경로를 서술하게 됩니다. 액세스 경로는 레코드를 검색하는 순서를 서술합니다. 액세스 경로를 서술하는 경우 키순 액세스 경로인지 도달순 액세스 경로인지를 서술해야 합니다.

관련 개념

95 페이지의 『데이터베이스 파일에 대한 액세스 경로 설명』

이 주제에서는 데이터베이스 파일에 대한 액세스 경로를 설명하는 여러 가지 방법을 설명합니다.

데이터베이스 파일에 사용되는 명명 규칙

파일명, 레코드 형식명 및 필드명은 최대 10자까지 가능하며 시스템의 모든 명명 규칙을 따라야 합니다. 그러나 일부 고급 언어는 시스템에서보다 더 제한적인 명명 규칙을 사용합니다.

예를 들어, RPG/400 언어는 6자 이름만 허용하는 반면 시스템은 10자 이름을 허용합니다. 어떤 경우에는 시스템명을 고급 언어의 제한에 부합되는 이름으로 일시적으로 변경(재명명)할 수 있습니다. 프로그램에서 데이터베이스 필드 재명명에 대한 자세한 내용은 고급 언어 주제 콜렉션을 참조하십시오.

그 외에도 이름에는 다음과 같은 제한이 있습니다.

- 필드명은 레코드 형식 내에서 고유해야 합니다.
- 레코드 형식명과 멤버명은 파일 내에서 고유해야 합니다.
- 파일명은 라이브러리 내에서 고유해야 합니다.

데이터베이스 자료 보호 및 모니터링

시스템은 자료의 무결성 및 일관성을 향상시키는 피처를 제공합니다. 자료를 보호하는 두 가지 방법은 비즈니스 규칙과 자료 유형 규칙을 강제 적용하는 것입니다.

다음 메소드를 사용하여 비즈니스 규칙을 강제 적용할 수 있습니다.

- 참조 제한조건을 사용하여 상호 종속성을 갖도록 정의한 파일의 자료에 제어(제한조건)를 넣을 수 있습니다. 또한 참조 제한조건을 사용하여 파일에 변경을 수행할 때 따라야 할 규칙을 지정할 수 있습니다.
- 트리거(Trigger)는 파일이 변경될 때 조치를 취하고 변경사항을 평가하는 사용자 프로그램을 실행합니다. 사전 정의된 변경이 수행되거나 시도되면 트리거 프로그램이 실행됩니다.

한 예로 시스템은 특정 인스턴스를 체크 인하는 자료 유형을 수행함으로써 숫자 필드의 자료가 실제로 숫자인지 확인합니다.

또한 시스템은 다음 메소드를 사용하여 자료가 손상되는 것을 방지합니다.

- 저널링 및 확약 제어 기능
- 시스템 관리 액세스 경로 보호(SMAPP) 지원

관련 개념

291 페이지의 『참조 제한조건으로 자료 무결성 확인』
이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

306 페이지의 『데이터베이스의 자동 이벤트 트리거』
*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

257 페이지의 『데이터베이스 회복 및 복원』
이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

데이터베이스 파일 크기

iSeries 서버에서 파일을 설계할 경우 데이터베이스 파일 *크기*를 고려해야 합니다.

다음은 데이터베이스 파일의 최대 값을 나열하는 표입니다.

설명	최대값
한 레코드 내의 바이트 수	32 766바이트
한 레코드 형식 내의 필드 수	8 000필드
한 파일 내의 키 필드 수	120 필드
실제 및 논리 파일에 대한 키 *크기*	32 768문자[1]
ORDER BY(SQL) 및 KEYFLD(OPNQRYF)의 키 *크기*	10 000바이트
한 파일 멤버 내의 레코드 수	4 294 967 294레코드[2]
한 파일 멤버 내의 바이트 수	1 869 162 846 624바이트[3]
한 액세스 경로 내의 바이트 수	1 099 511 627 776바이트[3] [5]
한 실제 파일 멤버를 기초로 하여 작성되는 키순 논리 파일의 수	3686파일
한 논리 파일 멤버 내의 실제 파일 멤버의 수	32 멤버
결합될 수 있는 멤버의 수	256 멤버
문자나 DBCS 필드의 *크기*	32 766바이트[4]
존 십진 또는 팩 십진 필드의 *크기*	63 자릿수
한 번에 사용할 수 있는 고유한 데이터베이스 파일의 최대 수	~500 000
실제 또는 논리 파일의 최대 멤버 수	32 767
실제 파일 당 제한조건의 최대 수	300개의 제한조건
실제 파일 당 트리거의 최대 수	300 트리거
반복 삽입 및 갱신 트리거 호출의 최대 수	200

설명	최대값
[1] 파일에 대해 선변경 선출(FCFO) 액세스 경로가 지정된 경우 실제 및 논리 파일에 대한 키의 최대 *크기*는 ACCPTHSIZ(*MAX1TB)의 경우 32 763자, ACCPTHSIZ(*MAX4GB)의 경우 1995자입니다.	
[2] 키순 액세스 경로가 지정된 파일의 경우 한 멤버 내의 최대 레코드 수는 경우에 따라 달라지며 다음 공식으로 그 값을 추정할 수 있습니다. ACCPTHSIZ(*MAX4GB)가 지정되면 다음 공식을 사용하십시오. 2,867,200,000 10 + (.8 x 키 길이) ACCPTHSIZ(*MAX1TB)가 지정되면 다음 공식을 사용하십시오. 725,680,000,000 12 + (.8 x 키 길이) 이 값은 추정값입니다. 실제의 최대 레코드 수는 이 추정값과 *크게* 다를 수 있습니다.	
[3] 파일 멤버의 바이트 수와 액세스 경로의 바이트 수는 최대 시스템 오브젝트 *크기*에 도달했음을 나타내는 메세지 CPF5272가 전송될 때 볼 수 있습니다.	
[4] 가변 길이 문자나 DBCS 필드의 최대 *크기*는 32 740바이트입니다. DBCS 그래픽 필드 길이는 문자 수로 표현되므로 최대값은 16 383자(필드 길이) 및 16 370자(가변 길이)입니다.	
[5] 액세스 경로가 최대 *크기* 4기가바이트(GB), ACCPTHSIZ(*MAX4GB)로 작성되는 경우 최대값은 4 294 966 272바이트입니다.	

위의 값들은 최대값입니다. 실제 한계값은 위에 언급된 최대값보다 낮을 수도 있습니다. 예를 들어, 일부 고급 언어에서는 위에 기술된 값보다 더욱 한계값이 제한될 수 있습니다.

최대치에 가까운 값을 사용할수록 시스템 성능이 저하될 수 있음에 유의해야 합니다. 예를 들어, 하나의 실제 파일에 대해 많은 논리 파일을 작성할수록 시스템 성능이 저하될 가능성이 커집니다. (실제 파일의 자료를 자주 변경하는 경우 많은 논리 파일 액세스 경로를 변경시킬 수 있습니다.)

일반적으로 iSeries 데이터베이스 파일은 그 *크기*가 시스템에서 허용되는 최대치에 도달할 때까지 커질 수 있습니다. 시스템은 대개 모든 파일 공간을 동시에 할당하지 않고 파일이 커짐에 따라 추가 공간을 때때로 할당합니다. 이러한 기억장치 자동 할당 방식은 시스템 성능의 향상 및 효율적인 보조 기억장치 공간 관리 등을 제공합니다.

파일 *크기*, 기억장치 할당 및 파일이 보조 기억장치에 연결될 것인지를 제어하려면 CRTPF(실제 파일 작성) 및 CRTSRCPF(소스 실제 파일 작성) 명령에서 SIZE, ALLOCATE 및 CONTIG 매개변수를 사용하면 됩니다.

다음 공식을 사용하여 실제 파일과 논리 파일의 디스크 *크기*를 추정할 수 있습니다.

- 널(null) 허용 필드가 들어 있는 실제 파일(액세스 경로 제외)의 경우:

```
디스크 크기 = (유효한 및 삭제 레코드의 수 + 1) x
              (레코드 길이 + 1) + 20480 x (멤버 수) + 8192
```

실제 파일의 *크기*는 CRTPF 및 CRTSRCPF 명령의 SIZE와 ALLOCATE 매개변수에 따라 결정됩니다. ALLOCATE(*YES)를 지정하면 SIZE 키워드의 *초기 할당* 및 *증분 크기*를 레코드 수 대신 사용해야 합니다.

- 널(null) 허용 필드가 들어 있는 실제 파일(액세스 경로 제외)의 경우:

| 디스크 *크기* = (유효한 및 삭제 레코드의 수 + 1) x
| (레코드 길이 + 1) + 20480 x (멤버 수) +
| 8192 + ((형식에서의 필드 수 ÷ 8) 올림) x
| (유효한 삭제 레코드 수 + 1)

실제 파일의 *크기*는 CRTPF 및 CRTSRCPF 명령의 SIZE와 ALLOCATE 매개변수에 따라 결정됩니다. ALLOCATE(*YES)를 지정하면 SIZE 키워드의 *초기 할당* 및 *증분 크기*를 레코드 수 대신 사용해야 합니다.

- 논리 파일(액세스 경로 제외):

| 디스크 *크기* = (12288) x (멤버 수) + 8192

- 키순 액세스 경로의 경우 멤버 당 색인 *크기*의 일반화된 등식은 다음과 같습니다.

```
let a = (LimbPageUtilization - LogicalPageHeaderSize) *
            (LogicalPageHeaderSize - LeafPageUtilization - 2 * NodeSize)

let b = NumKeys * (TerminalTextPerKey + 2 * NodeSize) *
            (LimbPageUtilization - LogicalPageHeaderSize + 2 * NodeSize)
      + CommonTextPerKey * [ LimbPageUtilization + LeafPageUtilization
      - 2 * (LogicalPageHeaderSize - NodeSize) ]
          - 2 * NodeSize * (LeafPageUtilization - LogicalPageHeaderSize
          + 2 * NodeSize)

let c = CommonTextPerKey * [ 2 * NodeSize - CommonTextPerKey
      - NumKeys * (TerminalTextPerKey + 2 * NodeSize) ]


then NumberLogicalPages = ceil( [ -b - sqrt(b ** 2 - 4 * a * c) ]
                                                    / (2 * a))

and  TotalIndexSize = NumberLogicalPages * LogicalPageSize
```

이 등식은 다음과 같은 등식의 상수 세트를 변경하여 3바이트 및 4바이트 색인에 사용됩니다.

상수	3바이트 색인	4바이트 색인
NodeSize	**3**	**4**
LogicalPageHeaderSize	**16**	**64**
LimbPageUtilization	**.75 * LogicalPageSize**	**.75 * LogicalPageSize**
LeafPageUtilization	**.75 * LogicalPageSize**	**.80 * LogicalPageSize**

나머지 상수인 CommonTextPerKey 및 TerminalTextPerKey는 다음 공식을 사용하면 최적의 추정치를 구할 수 있습니다.

```
CommonTextPerKey = [ min(max(NumKeys - 256,0),256)
                                + min(max(NumKeys - 256 * 256,0),256 * 256)
                                + min(max(NumKeys - 256 * 256 * 256,0),
                                    256 * 256 * 256)
                                + min(max(NumKeys - 256 * 256 * 256 * 256,0),
                   256 * 256 * 256 * 256) ]
                                * (NodeSize + 1) / NumKeys

TerminalTextPerKey = KeySizeInBytes - CommonTextPerKey
```

따라서 색인의 유형(예: 3 또는 4바이트), 총 키 *크기* 및 키의 수에 대한 색인 *크기*를 계산하는 데 필요한 모든 것을 감소시켜야 합니다. 추정치는 공통 텍스트 추정치가 최소이므로 실제 색인 *크기*보다 커야 합니다.

색인 *크기*에 대해 이러한 일반 등식이 제공된 경우 LogicalPageSize는 다음과 같습니다.

표 1. LogicalPageSize 값

키 길이	*MAX4GB(3바이트) LogicalPageSize	*MAX1TB(4바이트) LogicalPageSize
1 - 500	**4096바이트**	**8192바이트**
501 - 1000	**8192바이트**	**16 384바이트**
1001 - 2000	**16 384바이트**	**32 768바이트**
2001 - 10 000	**N/A**	**65 536바이트**
10 001 - 18 000	**N/A**	**131 072바이트**
18 001 - 26 000	**N/A**	**262 144바이트**
26 001 - 32 768	**N/A**	**524 288바이트**

표 1의 LogicalPageSizes는 다음 LimbPageUtilizations를 생성합니다.

키 길이	*MAX4GB(3바이트) LimbPageUtilization	*MAX1TB(4바이트) LimbPageUtilization
1 - 500	**3072바이트**	**6144바이트**
501 - 1000	**6144바이트**	**12 288바이트**
1001 - 2000	**12 288바이트**	**24 576바이트**
2001 - 10 000	**N/A**	**49 152바이트**
10 001 - 18 000	**N/A**	**98 304바이트**
18 001 - 26 000	**N/A**	**196 608바이트**

키 길이	*MAX4GB(3바이트) LimbPageUtilization	*MAX1TB(4바이트) LimbPageUtilization
26 001 - 32 768	**N/A**	**393 216**바이트

12 페이지의 표 1의 LogicalPageSizes는 다음 LeafPageUtilizations를 생성합니다.

키 길이	*MAX4GB(3바이트) LeafPageUtilization	*MAX1TB(4바이트) LeafPageUtilization
1 - 500	**3072**바이트	**6554**바이트
501 - 1000	**6144**바이트	**13 107**바이트
1001 - 2000	**12 288**바이트	**26 214**바이트
2001 - 10 000	**N/A**	**52 428**바이트
10 001 - 18 000	**N/A**	**104 857**바이트
18 001 - 26 000	**N/A**	**209 715**바이트
26 001 - 32 768	**N/A**	**419 430**바이트

그런 다음 색인 크기에 대한 일반 등식을 간소화하려면 다음과 같습니다.

CommonTextPerKey = 0

위 등식은 다음과 같습니다.

TerminalTextPerKey = KeySizeInBytes

```
b = NumKeys * (KeySizeInBytes + 2 * NodeSize) *
        (LimbPageUtilization - LogicalPageHeaderSize + 2 * NodeSize)
        - 2 * NodeSize * (LeafPageUtilization - LogicalPageHeaderSize
        + 2 * NodeSize)

c = 0

NumberLogicalPages = ceil( [ -b - sqrt(b ** 2 ) ]
                                                    / (2 * a))
                   = ceil[ (-2 * b) / (2 * a) ]
                   = ceil[ -b/a ]
```

예: 데이터베이스 파일 크기

키 120바이트, 레코드가 500 000개인 *MAX1TB(4바이트) 액세스 경로의 경우 TotalIndexSize를 바이트로 계산하면 다음과 같습니다.

```
a = (LimbPageUtilization - LogicalPageHeaderSize) *
        (LogicalPageHeaderSize - LeafPageUtilization - 2 * NodeSize)
    = (6144 - 64) *
```

```
        (64 - 6554 - 2 * 4)
    = 6080 * -6498
    = -39,507,840

b = NumKeys * (KeySizeInBytes + 2 * NodeSize) *
        (LimbPageUtilization - LogicalPageHeaderSize + 2 * NodeSize)
        - 2 * NodeSize * (LeafPageUtilization - LogicalPageHeaderSize
        + 2 * NodeSize)
    = 500,000 * (120 + 2 * 4) *
        (6144 - 64 + 2 * 4)
        - 2 * 4 * (6554 - 64 + 2 * 4)
    = 500,000 * 128 *
        6088
        - 8 * 6498
    = 3.896319e+11

NumberLogicalPages = ceil[ -b/a ]
                   = ceil[ -3.896319e+11/-39507840 ]
                                    = 9863

TotalIndexSize = NumberLogicalPages * LogicalPageSize
                                = 9863 * 8192
                                = 80,797,696 bytes
```

이전 버전의 오퍼레이팅 시스템에서 색인 *크기*에 대한 등식은 다음 결과를 산출합니다.

```
TotalIndexSize = (키 수) * (키 길이 + 8) *
                                    (0.8) * (1.85) + 4096
                                = (NumKeys) * (KeySizeInBytes + 8) *
                                    (0.8) * (1.85) + 4096
                                = 500000 * 128 *
                                    .8 * 1.85 + 4096
                                = 94,724,096
```

이 추정치는 사용자의 파일과 *크게* 다를 수 있습니다. 키순 액세스 경로는 사용자 레코드의 자료에 따라 *크게* 달라집니다. 정확한 *크기*를 알 수 있는 유일한 방법은 자료를 로드하여 파일 설명을 표시하는 것입니다.

다음은 최소 파일 *크기*를 나열하는표입니다.

설명	최소 *크기*
멤버가 없는 실제 파일	8192바이트
멤버가 하나인 실제 파일	20 480바이트
키순 액세스 경로	12 288바이트

주: 도달순 액세스 경로에는 추가 공간이 필요없습니다.

시스템은 파일 *크기* 외에도 데이터베이스 파일에 대한 내부 형식 및 디렉토리를 유지보수합니다. (사용자 프로파일 QDBSHR이 이러한 내부 오브젝트를 소유합니다.) 다음은 이들 오브젝트의 *크기*를 추산하는 공식입니다.

- 다른 파일의 형식을 공유하지 않는 모든 파일:

  형식 *크기* = (144 x 필드 수) + 4096

- 다른 파일과 자체의 형식을 공유하는 파일:

| 형식 공유 디렉토리 *크기* = (16 x 형식을 공유하는 파일의 파일 수) + 4096

- 그 위에 작성된 논리 파일 또는 논리 파일 멤버가 있는 각 실제 파일 및 실제 파일 멤버:

| 자료 공유 디렉토리 *크기* = (16 x 자료를 공유하는 파일 또는
| 멤버 수) + 4096

- 논리 파일 멤버가 자체의 액세스 경로를 공유하는 각 파일 멤버:

| 액세스 경로 공유 디렉토리 *크기* = (16 x 액세스 경로를 공유하는
| 파일 또는 멤버 수) + 4096

데이터베이스 파일 설정

이 주제에서는 iSeries 데이터베이스 파일 설정 방법, 데이터베이스 파일의 액세스 경로 설명 방법 및 데이터베이스 보안 방법에 대해 설명합니다.

데이터베이스 파일 작성 및 설명

이 주제에서는 데이터베이스 파일, 라이브러리, 소스 파일 및 실제 파일을 작성하는 프로세스에 대한 개요를 제공합니다.

시스템은 데이터베이스 파일을 설명하고 작성하기 위한 다음과 같은 여러 가지 방법을 지원합니다.

- 대화식 자료 정의 유틸리티(IDDU)

  iSeries용 WebSphere Development Studio 라이센스 프로그램의 일부인 IDDU를 사용하여 데이터베이스 파일을 작성할 수 있습니다. IDDU를 사용하여 데이터베이스 파일을 서술하는 경우 IDDU를 사용하여 파일을 작성하는 것도 고려할 수 있습니다.

- 자료 서술 스펙(DDS)을 지정하기 위해 소스 입력 유틸리티(SEU) 또는 자료 파일 유틸리티(DFU)를 사용하는 제어 언어(CL).

  CL을 사용하여 데이터베이스 파일을 작성할 수 있습니다. 데이터베이스 파일 작성 CL 명령으로는 CRTPF(실제 파일 작성), CRTLF(논리 파일 작성) 및 CRTSRCPF(소스 실제 파일)이 있습니다. 데이터베이스 파일이 작성되면 다음에는 SEU 또는 DFU를 사용하여 파일에서 자료를 설명할 수 있습니다. SEU 및 DFU는 iSeries용 IBM WebSphere Development Studio 라이센스 프로그램의 일부입니다. 이러한 주제는 다음 메소드를 사용하는 파일 작성 방법을 집중적으로 설명합니다.

- 구조화 조회 언어(SQL)

  SQL문을 사용하여 데이터베이스 파일(표)을 작성하고 설명할 수 있습니다. SQL은 IBM 관계형 데이터베이스 언어로서, 데이터베이스 파일을 대화식으로 서술하고 작성하기 위해 iSeries에서 사용됩니다.

- iSeries Navigator

  iSeries Navigator를 사용하여 데이터베이스 파일(표)을 작성할 수도 있습니다.

관련 개념

SQL 프로그래밍

표 작성 및 사용

iSeries Navigator 시작

라이브러리 작성

라이브러리는 다른 오브젝트에 대한 디렉토리로서 사용되는 시스템 오브젝트입니다. 라이브러리는 관련 오브젝트를 그룹화하고 사용자가 이름별로 오브젝트를 찾는 것을 허용합니다.

오브젝트 유형의 시스템 인식 식별자는 *LIB입니다. 데이터베이스 파일을 작성하기 전에 저장할 라이브러리를 작성해야 합니다. 다음과 같은 방법으로 라이브러리를 작성할 수 있습니다.

- iSeries Navigator를 사용하여 라이브러리(SQL에서 스키마라고 함)를 작성할 수 있습니다.
- CRTLIB(라이브러리 작성) 명령을 사용하여 라이브러리를 작성할 수 있습니다.

라이브러리를 작성할 경우 라이브러리를 저장할 보조 기억장치 풀(ASP)을 지정할 수 있습니다. 이를 통해 별개의 여러 데이터베이스를 작성할 수 있습니다.

iSeries Navigator를 사용하여 라이브러리(스키마) 작성

iSeries Navigator를 사용하여 라이브러리(SQL에서는 스키마라고 함)를 작성하는 방법을 알아봅시다.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. 작업할 **데이터베이스**와 라이브러리를 확장하십시오.
3. **스키마**를 마우스 오른쪽으로 클릭하고 **새 스키마**를 클릭하십시오.
4. 새 스키마 창에서 스키마 이름을 지정하십시오.
5. 이 스키마를 표시되는 스키마 리스트에 추가하려면 **표시되는 스키마 리스트에 추가**를 선택하십시오.
6. 표준 스키마로 작성하려면 **표준 스키마로 작성**을 선택하십시오.
7. 자료 사전을 작성하려면 **자료 사전 작성**을 선택하십시오.
8. 스키마를 포함하기 위한 디스크 풀(pool)을 지정하십시오.
9. 설명을 지정하십시오.
10. **확인**을 클릭하십시오.

관련 참조

CRTLIB(라이브러리 작성) 명령

소스 파일 설정

이 주제에서는 소스 파일 설정 방법과 소스 파일을 사용하는 이유를 설명합니다.

관련 개념

276 페이지의 『소스 파일 사용』

이 주제에서는 소스 파일에 자료를 입력하고 유지보수하는 방법과 소스 파일을 사용하여 시스템에서 다른 오브젝트(예: 파일 또는 프로그램)를 작성하는 방법을 설명합니다.

소스 파일 사용 이유:

명령 자체만으로는 오브젝트 작성에 필요한 정보를 충분히 제공할 수 없을 경우 소스 파일이 사용됩니다.

소스 파일에는 특정 유형의 오브젝트를 작성하는 데 필요한 입력(소스) 자료가 들어 있습니다. 예를 들어, 제어 언어(CL) 프로그램을 작성하기 위해서는 명령 형태의 소스문이 들어 있는 소스 파일을 사용해야 하며, 논리 파일을 작성하려면 자료 서술 스펙(DDS)을 포함하는 소스 파일을 사용해야 합니다.

다음 오브젝트를 작성하는 경우 소스 파일이 필요합니다.

- 고급 언어 프로그램
- 제어 언어 프로그램
- 논리 파일
- 시스템간 통신 기능(ICF: intersystem communication function) 파일
- 명령

다음 오브젝트를 작성하는 경우 소스 파일이 사용될 수는 있으나 필수는 *아닙니다*.

- 실제 파일
- 표시장치 파일
- 프린터 파일
- 변환표

소스 파일은 데이터베이스 파일, 디스켓 파일, 테이프 파일 또는 인라인 자료 파일 등이 될 수 있습니다(인라인 자료 파일은 작업의 일부분으로 포함됨). 소스 데이터베이스 파일은 데이터베이스 파일의 한 유형입니다. 시스템에서 다른 데이터베이스 파일을 사용하는 것과 마찬가지로 소스 데이터베이스 파일을 사용할 수 있습니다.

관련 개념

4 페이지의『데이터베이스 파일』

데이터베이스 파일은 시스템 오브젝트 유형 *FILE의 몇 가지 유형 중 하나입니다. 데이터베이스 파일에는 내부 기억장치에서 입력 자료가 프로그램에 부여되는 방식과 프로그램에서 출력 자료가 내부 기억장치로 부여되는 방식에 대한 설명이 들어 있습니다.

소스 파일 작성:

이 주제에서는 소스 파일 작성 방법에 대해 설명합니다.

소스 파일을 작성하기 전에 먼저 라이브러리를 작성해야 합니다. 그런 다음 소스 파일을 작성하려면 다음 명령 중 하나를 사용하십시오.

- CRTSRCPF(소스 실제 파일 작성) 명령

  일반적으로 대부분의 매개변수가 소스 파일에 자주 사용되는 값으로 설정되므로 보통 CRTSRCPF 명령을 사용하여 소스 파일을 작성하게 됩니다.
- CRTPF(실제 파일 작성) 또는 CRTLF(논리 파일 작성) 명령

소스 파일을 작성하고 자료 서술 스펙(DDS)을 사용하여 레코드 형식 및 필드를 정의하려면 CRTPF(실제 파일 작성) 또는 CRTLF(논리 파일 작성) 명령을 사용하십시오.

소스 파일 작성에 대한 하나의 대안으로서 i5/OS 및 기타 라이센스 프로그램에서 제공하는 소스 파일을 사용할 수 있습니다.

관련 개념

16 페이지의 『라이브러리 작성』
라이브러리는 다른 오브젝트에 대한 디렉토리로서 사용되는 시스템 오브젝트입니다. 라이브러리는 관련 오브젝트를 그룹화하고 사용자가 이름별로 오브젝트를 찾는 것을 허용합니다.

관련 참조

실제 파일 작성(CRTPF) 명령

논리 파일 작성(CRTLF) 명령

CRTSRCPF(소스 실제 파일 작성) 명령을 사용하여 소스 파일 작성:

이 예는 CRTSRCPF(소스 실제 파일 작성) 명령과 명령의 디폴트를 사용하여 소스 파일을 작성하는 방법을 표시합니다.

```
CRTSRCPF FILE(QGPL/FRSOURCE) TEXT('Source file')
```

CRTSRCPF 명령은 실제 파일을 작성하지만 소스 실제 파일에 적합한 속성이 필요합니다. 예를 들어, 소스 파일의 디폴트 레코드 길이는 92(소스 자료 필드 길이는 80, 소스 순서 번호 필드 길이는 6, 소스 날짜 필드 길이는 6)입니다.

관련 참조

소스 실제 파일 작성(CRTSRCPF) 명령

DDS를 사용하여 소스 파일 작성:

자료 서술 스펙(DDS)으로 소스 파일을 작성하려면 CRTPF(실제 파일 작성) 또는 CRTLF(논리 파일 작성) 명령을 사용하십시오.

레코드 형식을 정의하는 데 필요한 소스 파일을 작성하려면 CRTPF 또는 CRTLF 명령을 사용하십시오. 소스 논리 파일을 작성하는 경우 중복 키를 막기 위해 논리 파일 멤버는 하나의 실제 파일 멤버만을 참조해야 합니다.

다음 예는 CRTPF 명령을 사용하여 소스 파일의 레코드 형식을 정의하는 데 필요한 DDS를 보여줍니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A*   R RECORD1
          A          F1             6S 2
          A          F2             6S
          A          F3            92A
```

관련 참조

실제 파일 작성(CRTPF) 명령

논리 파일 작성(CRTLF) 명령

DDS를 사용하지 않고 소스 파일 작성:

CRTSRCPF(소스 실제 파일 작성) 명령의 여러 매개변수가 소스 파일에 사용하려는 값을 디폴트로 사용할 경우 자료 서술 스펙(DDS)을 사용하지 않고 소스 파일을 작성할 수 있습니다.

DDS를 사용하지 않고 레코드 길이(RCDLEN 매개변수)를 지정하여 소스 실제 파일을 작성할 경우 작성된 소스에는 *SRCSEQ*, *SRCDAT* 및 *SRCDTA* 필드가 포함됩니다(레코드의 자료 부분의 길이가 레코드 길이에서 12를 뺀 값이 되도록 레코드 길이에는 순서 번호 및 최종 변경 날짜 필드에 12자가 포함되어야 함). 레코드의 자료 부분에는 두 개 이상의 필드가 포함되도록 정의될 수 있습니다(각 필드는 문자 또는 존 십진 필드여야 함). 레코드의 자료 부분에 두 개 이상의 필드가 포함되도록 정의하기 위해서는 DDS를 사용하여 필드를 정의해야 합니다.

CRTSRCPF 명령을 사용하여 작성된 소스 실제 파일에는 자동으로 다음 세 필드로 구성된 레코드 형식이 사용됩니다.

필드	이름	자료 유형 및 길이	설명
1	SRCSEQ	존 십진, 6자리, 소수 자릿수 2	레코드의 순서 번호
2	SRCDAT	존 십진, 6자리, 소수 자릿수 없음	레코드의 최종 갱신 날짜
3	SRCDTA	문자, 길이 제한 없음	레코드의 자료 부분(텍스트)

주: 모든 IBM 제공 데이터베이스 소스 파일에서 자료 부분의 길이는 80바이트입니다. IBM 제공 장치 소스 파일에서 자료 부분의 길이는 연관 장치의 최대 레코드 길이입니다.

IBM 제공 소스 파일:

사용자 편의를 위해 i5/OS 라이센스 프로그램 및 기타 라이센스 프로그램은 각 소스 유형에 데이터베이스 소스 파일을 제공합니다.

이 표에는 IBM이 제공하는 소스 파일이 표시됩니다.

파일명	라이브러리명	작성 시 사용
QCBLSRC	QGPL	System/38™ 호환 COBOL
QCSRC	QGPL	C 프로그램
QCLSRC	QGPL	CL 프로그램
QCMDSRC	QGPL	명령 정의문
QDDSSRC	QGPL	파일
QFMTSRC	QGPL	정렬 소스
QLBLSRC	QGPL	COBOL/400® 프로그램
QS36SRC	#LIBRARY	System/36™ 호환 COBOL 프로그램
QREXSRC	QGPL	프로시듀어 언어 400/REXX 프로그램
QRPGSRC	QRPG	RPG/400 프로그램
QAPLISRC	QPLI	PL/I 프로그램
QPLISRC	QGPL	PL/I 프로그램
QARPGSRC	QRPG38	System/38 환경 RPG
QRPG3SRC	QRPG38	System/38 환경 RPG
QRPG2SRC	#RPGLIB	System/36 호환 RPG II
QS36PRC	#RPGLIB	System/36 호환 RPG II
QS36SRC	#LIBRARY	System/36 호환 RPG II(설치 후)

파일명	라이브러리명	작성 시 사용
QPASSRC	QPAS	Pascal 프로그램
QTBLSRC	QGPL	변환표
QTXTSRC	QPDA	텍스트

위의 파일에 소스 멤버를 추가하거나 사용자가 소스 파일을 작성할 수 있습니다. 일반적으로, IBM 제공 파일과 같은 이름을 사용하여 다른 라이브러리에 사용자의 소스 파일을 작성합니다(시스템의 새로운 릴리스가 설치되면 IBM 제공 파일이 대체될 수도 있음). IBM 제공 소스 파일은 대응되는 작성 명령에 사용되는 파일명으로 작성됩니다(예를 들어, CRTCLPGM(CL 프로그램 작성) 명령은 QCLSRC 파일명을 디폴트로 사용함). 이 밖에 IBM 제공 프로그래머 메뉴도 동일한 디폴트 이름을 사용합니다. 사용자 고유의 소스 파일을 작성하는 경우 이를 IBM 제공 소스 파일이 들어 있는 라이브러리에 넣지 마십시오. (IBM 제공 파일명과 동일한 파일명을 사용하는 경우 라이브러리 리스트에서 사용자의 소스 파일이 들어 있는 라이브러리가 IBM 제공 소스 파일이 들어 있는 라이브러리 앞에 있도록 해야 합니다.)

소스 파일 속성:

이 주제에서는 대부분의 소스 파일에 대해 공통적인 속성과 이러한 속성 사용 시 제한사항에 대해 간단히 설명합니다.

소스 파일은 대개 다음과 같은 속성을 가집니다.

- 92자의 레코드 길이(6바이트의 순서 번호, 6바이트의 날짜 및 80바이트의 소스)
- 액세스 경로에서 고유 키를 지정하지 않는 경우라도 키(순서 번호)는 고유합니다. 사용자가 소스 파일에 대해 키를 지정할 필요가 없습니다. 디폴트 소스 파일은 키 없이 작성됩니다(도달순 액세스 경로). 도달순 액세스 경로로 작성된 소스 파일은 키순 액세스 경로가 지정된 소스 파일에 비해 기억장치가 적게 필요하며 저장/복원 시간도 짧게 걸립니다.
- 두 개 이상의 멤버가 있습니다.
- 이 멤버를 사용해 작성될 오브젝트명과 동일한 멤버명을 가집니다.
- 모든 레코드의 레코드 형식이 동일합니다.
- 대부분의 자료 파일에 비해 각 멤버 안의 레코드 수가 상대적으로 적습니다.

소스 파일에는 다음과 같은 몇 가지 제한사항이 있습니다.

- 키가 지정될 경우 소스 순서 번호가 키로 사용되어야 합니다.
- 키가 지정될 경우 반드시 오름차순으로 지정되어야 합니다.
- 액세스 경로는 고유 키를 지정할 수 없습니다.
- ALTSEQ 키워드는 소스 파일의 자료 서술 스펙(DDS)에 허용되지 않습니다.
- 첫 필드는 존 십진 자료 및 소수 자릿수가 2인 6자리의 순서 번호 필드여야 합니다.
- 두 번째 필드는 존 십진 자료 및 소수 자릿수가 없는 6자리의 날짜 필드여야 합니다.
- 두 번째 이후의 모든 필드는 존 십진이거나 문자 필드여야 합니다.

데이터베이스 파일 설명

이 주제는 iSeries 데이터베이스 파일 서술에 대한 몇 가지 방법을 소개하며 자료 서술 스펙(DDS)에 자료 정의를 위한 대부분의 옵션에 있으므로 DDS를 사용하여 데이터베이스 파일을 설명하는 방법을 중점적으로 살펴봅니다.

파일을 레코드 레벨로만 서술하려면 CRTPF(실제 파일 작성) 및 CRTSRCPF(소스 실제 파일 작성) 명령에 RCDLEN(레코드 길이) 매개 변수를 사용할 수 있습니다. 파일을 필드 레벨로 서술하려면 몇 가지 방법을 사용하여 자료를 대화식 자료 정의 유틸리티(IDDU), 구조화 조회 언어(SQL) 명령 또는 자료 서술 스펙(DDS) 등의 데이터베이스 시스템에 설명할 수 있습니다.

대화식 자료 정의 유틸리티(IDDU)

실제 파일은 IDDU를 사용하여 서술할 수 있습니다. IDDU는 메뉴 처리 방식의 대화식 자료 서술 방식이므로 IDDU를 사용할 수도 있습니다. System/36에서 IDDU를 사용하여 자료를 서술하는 것이 익숙할 수 있습니다. 또한 IDDU를 사용하면 조회, iSeries Access 및 자료 파일 유틸리티(DFU)와 함께 사용할 수 있도록 다중 형식 실제 파일을 설명할 수 있습니다.

IDDU를 사용하여 파일을 설명할 경우 파일 정의가 i5/OS 자료 사전의 일부가 됩니다.

iSeries용 DB2 Universal Database 구조화 조회 언어(SQL)

SQL은 iSeries 데이터베이스 파일을 서술하는 데 사용할 수 있습니다. SQL은 데이터베이스 파일의 필드를 서술하고 파일을 작성하는 명령문을 지원합니다.

SQL은 표준 및 일반 데이터베이스 언어의 필요를 충족시키기 위해 IBM에서 개발한 것입니다. 현재 이것은 모든 IBM DB2 플랫폼 및 타사의 다른 많은 데이터베이스에 사용되고 있습니다.

iSeries용 DB2 UDB SQL 언어를 사용하여 데이터베이스 파일이 작성될 때 SQL 콜렉션의 자료 사전에 자동으로 파일 설명이 추가됩니다. 자료 사전(또는 리스트)은 시스템에 의해 자동적으로 관리됩니다.

SQL은 다른 많은 플랫폼에 있는 데이터베이스를 액세스하기 위한 선택 언어입니다. 이는 분산 데이터베이스와 이기종(hetero genous) 시스템용으로는 유일한 언어입니다.

자료 서술 스펙(DDS)

외부 서술 자료 파일은 DDS를 사용하여 서술할 수 있습니다. DDS를 사용하여 필드, 레코드 및 파일 레벨의 정보를 설명합니다.

DDS는 프로그래머에게 데이터베이스 내의 자료를 설명하는 옵션을 가장 많이 제공하기 때문에 DDS를 사용할 수도 있습니다. 예를 들면 DDS만이 논리 파일 내의 키 필드를 서술할 수 있습니다.

DDS 양식은 자료를 외부적으로 서술하기 위한 일반 형식을 제공합니다. DDS 자료는 열(column)에 민감합니다. 이 안내서에 나오는 예들은 열 번호가 있으며 정확한 열 안에 자료를 표시하고 있습니다.

데이터베이스 파일을 설명한 다음에는 그 설명을 볼 수 있습니다.

관련 개념

IDDU 사용 PDF

SQL 프로그래밍

iSeries SQL용 DB2 UDB 참조서

245 페이지의『데이터베이스 파일 정보 표시』

iSeries Navigator에서 표 설명 표시 조작으로 데이터베이스 파일 및 장치 파일에 대한 파일 속성을 표시할 수 있습니다.

DDS를 사용하여 데이터베이스 파일 서술:

자료 서술 스펙(DDS)을 사용하여 데이터베이스 파일을 서술하는 경우 파일, 레코드 형식, 결합, 필드, 키 및 선택/생략 레벨에서 정보를 서술할 수 있습니다.

- 파일 레벨 DDS는 전체 파일에 대한 정보를 시스템에 제공합니다. 예를 들면 파일 내의 모든 키 필드 값이 고유해야 하는지를 지정할 수 있습니다.
- 레코드 형식 레벨 DDS는 파일 내의 특정 레코드 형식에 관한 정보를 시스템에 제공합니다. 예를 들면 파일 내의 모든 키 필드 값이 고유해야 하는지를 지정할 수 있습니다.
- 결합 레벨 DDS는 결합 논리 파일에서 사용되는 실제 파일에 대한 정보를 시스템에 제공합니다. 예를 들면 두 개의 실제 파일을 결합하는 방법을 지정할 수 있습니다.
- 필드 레벨 DDS는 레코드 형식 내의 각 필드에 관한 정보를 시스템에 제공합니다. 예를 들면 각 필드의 이름 및 속성을 지정할 수 있습니다.
- 키 필드 레벨 DDS는 파일의 키 필드에 관한 정보를 시스템에 제공합니다. 예를 들면 레코드 형식 내의 필드 중 키 필드로 사용될 필드를 지정할 수 있습니다.
- 선택/생략 키 레벨 DDS는 파일 처리시 프로그램으로 리턴되는 레코드에 관한 정보를 시스템에 제공합니다. 선택/생략 스펙은 논리 파일에만 적용됩니다.

관련 개념

실제 및 논리 파일용 DDS

예: DDS를 사용하여 실제 파일 서술:

이 예는 DDS를 사용하여 실제 파일을 서술하는 방법을 표시합니다.

다음 예에서처럼 실제 파일을 위한 DDS는 반드시 아래와 같은 순서로 작성되어야 합니다.

1 파일 레벨 항목(선택적). UNIQUE 키워드는 파일의 각 레코드 키 필드 값이 고유해야 함을 나타내기 위해 사용됩니다. 중복 키 값은 이 파일에서 허용되지 않습니다.

2 레코드 형식 레벨 항목. 선택적인 텍스트 설명과 더불어 레코드 형식명이 지정됩니다.

3 필드 레벨 항목. 각 필드에 대한 선택적인 텍스트 설명과 더불어 필드명 및 필드 길이가 지정됩니다.

4 키 필드 레벨 항목(선택적). 키 필드로 사용될 필드명이 지정됩니다.

5 주석(선택적)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A  5
```

```
A                                   1  UNIQUE
A      2 R ORDHDR                      TEXT('Order header record')
A        3 CUST           5   0       TEXT('Customer number')
A            ORDER          5   0       TEXT('Order number')
A            .
A            .
A            .
A          K CUST
A    4     K ORDER
```

다음 예에서는 키 필드가 지정되지 않은 도달순 액세스 경로의 실제 파일 ORDHDRP(주문 헤더 파일)와 그 파일을 서술하는 데 필요한 DDS를 보여줍니다.

실제 파일 ORDHDRP의 레코드 형식

Customer number(CUST) — 팩 십진 길이 5, 소수점 없음
Order number(ORDER) — 팩 십진 길이 5, 소수점 없음
Order date(ORDATE) — 팩 십진 길이 6, 소수점 없음
Purchase order number(CUSORD) — 팩 십진 길이 15, 소수점 없음
Shipping instructions(SHPVIA) — 문자 길이 15
Order status(ORDSTS) — 문자 길이 1
...
State(STATE) — 문자 길이 2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A          R ORDHDR                    TEXT('Order header record')
     A            CUST           5   0      TEXT('Customer Number')
     A            ORDER          5   0      TEXT('Order Number')
     A            ORDATE         6   0      TEXT('Order Date')
     A            CUSORD        15   0      TEXT('Customer Order No.')
     A            SHPVIA        15          TEXT('Shipping Instr')
     A            ORDSTS         1          TEXT('Order Status')
     A            OPRNME        10          TEXT('Operator Name')
     A            ORDAMT         9   2      TEXT('Order Amount')
     A            CUTYPE         1          TEXT('Customer Type')
     A            INVNBR         5   0      TEXT('Invoice Number')
     A            PRTDAT         6   0      TEXT('Printed Date')
     A            SEQNBR         5   0      TEXT('Sequence Number')
     A            OPNSTS         1          TEXT('Open Status')
     A            LINES          3   0      TEXT('Order Lines')
     A            ACTMTH         2   0      TEXT('Accounting Month')
     A            ACTYR          2   0      TEXT('Accounting Year')
     A            STATE          2          TEXT('State')
     A
```

17열의 R은 레코드 형식이 정의되고 있음을 나타냅니다. 레코드 형식명 ORDHDR은 19열에서 28열에 지정됩니다.

필드를 서술할 때 17열에는 아무 항목도 입력하지 않습니다. 19열에서 28열까지의 이름과 함께 17열의 공백은 필드명을 나타냅니다.

자료 형태는 35열에 지정됩니다. 유효한 자료 형태는 다음과 같습니다.

항목	의미
A	문자
P	팩 십진
S	존 십진
B	2진
F	부동 소수점
H	16진
L	날짜
T	시간
Z	시간소인

주:

1. 2바이트 문자 세트(DBCS) 자료 형태에 대해서는 337 페이지의 『2바이트 문자 세트 고려사항』을 참조하십시오.
2. iSeries 시스템은 존 십진보다 팩 십진 연산을 보다 효율적으로 수행합니다.
3. 부동 소수점 자료를 지원하지 않는 고급 언어도 있습니다.
4. 부동 소수점 필드 사용 시 적용되는 특별한 고려사항은 다음과 같습니다.
   - 부동 소수점 필드와 연관된 정밀도(단정밀도 또는 배정밀도)는 비트 수의 기능과 부동 소수점 값의 내부적 표시입니다. 이것은 유효부에서 지원되는 십진 숫자의 수와 부동 소수점 필드에서 나타낼 수 있는 최대값으로 변환됩니다.
   - 부동 소수점 필드가 지정된 필드에 의해 지원되는 자릿수보다 더 적은 자리로 정의된 경우 그 길이는 표시 길이일 뿐이며 내부 계산에 사용되는 정밀도에는 전혀 영향이 없습니다.
   - 부동 소수점 수의 정밀도가 7(단정밀도)이나 15(배정밀도)의 십진 자리까지 정확하다고 해도 사용자는 9 또는 17자리까지 지정할 수 있습니다. 여분의 자릿수는 내부 부동 소수점 형식에서 내부 비트 구성을 고유하게 설정하는 데 사용되어 내부 형식의 부동 소수점 수가 십진으로 변환되고 또 다시 내부 형식으로 변환되는 경우 동일한 결과를 얻을 수 있도록 합니다.

자료 형태(35열)가 지정되어 있지 않으면 소수 자릿수(숫자 필드용)가 자료 형태를 결정합니다. 소수 자릿수(36열에서 37열)가 공백일 경우 자료 유형은 문자(A)인 것으로 가정됩니다. 이 열에 0에서 31까지의 숫자가 있을 경우 자료 유형은 팩 십진수(P)로 가정됩니다.

필드 길이는 30열에서 34열 사이에 지정되며, 소수 자릿수(숫자 필드의 경우)는 36열과 37열에서 지정됩니다. 팩 십진 또는 존 십진 필드가 고급 언어 프로그램에서 사용되는 경우 필드 길이는 사용 중인 고급 언어에서 허용하는 길이로 제한되어야 합니다. 길이는 기억장치 내의 필드 길이가 아니라 기억장치로부터 외부적으로 지정된 숫자나 문자의 자릿수입니다. 예를 들어 5자리의 팩 십진 필드는 DDS에서 길이가 5로 지정되나, 기억장치에서는 3바이트만 차지합니다.

문자나 16진 자료는 VARLEN 필드 레벨 키워드를 지정하여 가변 길이로 정의할 수 있습니다. 예를 들면, 일반적으로 가변 길이 필드를 데이터베이스 내의 사원명으로 사용합니다. 보통 이름은 30바이트 필드에 저장할 수 있지만 긴 이름을 저장하려면 100바이트를 필요로 할 때도 있습니다. 사용자가 항상 100바이트로 긴 필드를 정의하면 기억장치가 낭비됩니다. 사용자가 항상 30바이트로 필드를 정의하면 어떤 이름은 잘립니다.

DDS VARLEN 키워드는 가변 길이로 문자 필드를 지정하는 경우에 사용할 수 있습니다. 사용자는 이 필드를 다음과 같이 정의할 수 있습니다.

- 할당된 길이가 없는 가변 길이. 이것은 필드 자료와 같은 바이트의 수(길이 값으로 필드당 2바이트와 레코드당 추가된 바이트)만을 사용하여 저장하도록 합니다. 그러나 모든 자료가 파일의 가변 부분에 저장되어, 검색을 위한 두 디스크 읽기 조작을 요구하기 때문에 성능에 영향을 미치게 됩니다.
- 자료의 *크기*와 가장 근접한 할당된 길이를 가진 가변 길이. 이것은 파일의 고정 부분에 대부분의 필드 자료를 저장하고, 고정 길이 필드 정의에 공통되는 사용되지 않은 기억장치 할당을 최소화합니다. 할당된 필드 길이보다 작은 길이의 필드 자료를 검색할 때에는 하나의 읽기 조작만 필요합니다. 할당 길이보다 긴 길이의 필드 자료는 파일의 가변 부분에 저장되며 자료 검색에 두 개의 읽기 조작을 필요로 합니다.

예: DDS를 사용하여 논리 파일 서술:

이 예는 DDS를 사용하여 논리 파일을 서술하는 방법을 표시합니다.

다음 예에서처럼 논리 파일을 위한 DDS는 반드시 아래와 같은 순서로 작성되어야 합니다.

1 파일 레벨 항목(선택적). 이 예에서 UNIQUE 키워드는 이 파일에서 각 레코드의 키 값이 고유해야 함을 나타내기 위해 사용됩니다. 중복 키 값은 허용되지 않습니다.

각 레코드 형식에서

2 레코드 형식 레벨 항목. 이 예에서는 레코드 형식명, 연관된 실제 파일명 및 선택적인 텍스트 설명 등이 지정됩니다.

3 필드 레벨 항목(선택적). 이 예에서는 레코드 형식 내에서 사용되는 각 필드명이 지정됩니다.

4 키 필드 레벨 항목(선택적). 이 예에서는 *Order* 필드가 키 필드로 사용됩니다.

5 선택/생략 필드 레벨 항목(선택적). 이 예에서 *Opnsts* 필드에 N 값이 들어 있는 모든 레코드는 파일의 액세스 경로에서 생략됩니다. 다시 말해 이 파일로부터 레코드를 읽는 프로그램은 *OPNSTS* 필드에 N 값이 들어 있는 레코드를 볼 수 없다는 것입니다.

6 주석

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A   6
     A                                  1  UNIQUE
     A     2  R ORDHDR                     PFILE(ORDHDRP)
     A       3  ORDER                      TEXT('Order number')
     A            CUST                      TEXT('Customer number')
     A            .
     A            .
```

```
A            .
A     4  K ORDER
A          O OPNSTS              5   CMP(EQ 'N')
A          S                          ALL
```

실제 파일을 기초로 하는 논리 파일은 모든 실제 파일이 작성된 후 작성되어야 합니다. 앞의 예에서 PFILE 키워드는 논리 파일의 기초가 되는 실제 파일을 지정하는 데 사용됩니다.

논리 파일 내의 레코드 형식은

- 실제 파일의 필드에 기초를 둔 새로운 레코드 형식이 될 수 있습니다.
- 이전의 실제 또는 논리 파일에서 서술된 것과 동일한 레코드 형식이 될 수 있습니다.

논리 파일의 레코드 형식 내의 필드는 하나 이상의 실제 파일의 레코드 형식에 있거나 논리 파일의 기초가 되는 실제 파일의 필드로부터 파생된 것이어야 합니다.

관련 개념

30 페이지의『데이터베이스 파일에서 기존 레코드 형식 설명 공유』
레코드 형식은 실제 파일이나 논리 파일(결합 논리 파일 제외)에서 한 번 서술하여 여러 파일에서 이를 사용할 수 있습니다. 새로운 파일을 서술할 때 기존 파일의 레코드 형식이 새로운 파일에 사용되도록 지정할 수 있습니다.

44 페이지의『논리 파일 설정』
이 주제에서는 논리 파일 작성에 대한 몇 가지 특수 고려사항에 대해 설명합니다.

DDS로 서술할 수 있는 추가 필드 정의 기능:

기능 키워드(DDS 양식의 45-80열)를 사용하여, 실제 및 논리 파일 레코드 형식의 필드에 대하여 추가의 정보를 서술할 수 있습니다.

사용자가 지정할 수 있는 것은 다음과 같습니다.

- 필드 자료가 표준에 부합되는지를 확인하기 위한 유효성 검사 키워드. 예를 들면, 필드가 500부터 900까지의 유효한 범위를 갖도록 서술할 수 있습니다. (이러한 검사는 자료가 키보드에서 표시장치로 입력될 때에만 수행됩니다.)
- 필드가 표시되거나 인쇄되는 방법을 제어하는 편집 키워드. 예를 들면, EDTCDE(Y) 키워드를 사용하여 날짜 필드가 MM/YY/DD로 표시되도록 지정할 수 있습니다. EDTCDE 및 EDTWRD 키워드는 편집을 제어하는 데 사용할 수 있습니다. (이 편집은 화면 파일이나 프린터 파일에서 사용될 때에만 수행됩니다.)
- 필드의 설명 및 이름을 제어하는 문서화, 표제 및 이름 제어 키워드. 예를 들어 각 필드의 설명을 문서화하기 위해 TEXT 키워드를 사용할 수 있습니다. 이러한 텍스트 설명은 프로그램에서 사용되는 파일을 보다 효율적으로 문서화하기 위해서 컴파일러 리스트에 포함됩니다. TEXT 및 COLHDG 키워드는 텍스트 및 열 제목 정의를 제어합니다. ALIAS 키워드는 필드에 보다 서술적인 이름을 제공하는 데 사용될 수 있습니다. 별명 또는 대체명은 프로그램에서 사용됩니다(고급 언어가 별명을 지원하는 경우).
- 필드에 대한 널값 내용과 디폴트 자료를 제어하기 위한 내용 및 디폴트 값 키워드. ALWNULL 키워드는 필드에 널값이 허용되는지의 여부를 지정합니다. ALWNULL이 사용되면 필드에 대한 디폴트는 널(null)입

니다. ALWNULL이 필드 레벨에서 표시되지 않으면 서로 다른 값을 지정하기 위해 DFT(디폴트 값) 키워드를 사용하는 경우를 제외하고는 널값이 허용되지 않으며 디폴트 문자와 16진 필드는 공백으로 되고 숫자 필드는 0으로 됩니다.

기존 필드 설명 및 필드 참조 파일을 사용하여 데이터베이스 파일 서술:

필드가 기존 파일에서 이미 서술되었고 그 필드 설명을 현재 설정중인 새로운 파일에서 사용하려는 경우 기존 설명을 새로운 파일 설명으로 복사해 주도록 시스템에 요청할 수 있습니다.

DDS 키워드 REF 및 REFFLD를 사용하면 기존 파일의 필드 설명을 참조할 수 있습니다. 이러한 기능은 DDS 문을 코딩하는 노력을 덜어줍니다. 또한 그 필드를 사용하는 모든 파일에서 필드 속성이 일관성있게 사용되도록 합니다.

그 외에도 실제 파일의 필드 설명만을 사용하기 위한 목적으로 실제 파일을 작성할 수 있습니다. 즉, 파일에는 자료가 들어있지 않으며 다른 파일의 필드 설명에 대한 참조로써만 사용됩니다. 이러한 유형의 파일을 필드 참조 파일이라고 합니다. *필드 참조 파일*은 자료는 포함하지 않고, 필드 서술만을 포함하고 있는 실제 파일(PF)입니다.

사용자는 필드 참조 파일을 사용하여 레코드 형식 설명을 단순하게 만들고 필드 설명이 일관성있게 사용되는지 확인합니다. 어플리케이션이나 모든 파일 그룹에 필요한 모든 필드를 참조 파일에 정의할 수 있습니다. 사용자는 DDS와 CRTPF(실제 파일 작성) 명령을 사용하여 필드 참조 파일을 작성할 수 있습니다.

필드 참조 파일이 작성되고 나면 각 파일의 각 필드 특성을 서술하지 않고도 이 파일로 실제 파일 레코드 형식을 빌드할 수 있습니다. 실제 파일 빌드 시 사용자는 필드 참조 파일을 참조하고(REF 및 REFFLD 키워드 사용) 변경사항을 지정하기만 하면 됩니다. 새로운 파일에 지정된 필드 설명 및 키워드에 대한 변경은 필드 참조 파일 내의 설명을 대체합니다.

다음 예에서는 필드 참조 파일 DSTREFP가 분배 어플리케이션을 위해 작성됩니다. 다음 예는 DSTREFP를 서술하기 위해 필요한 DDS를 보여줍니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* FIELD REFERENCE FILE (DSTREFP)
          A           R DSTREF                    TEXT('Field reference file')
     A
          A* FIELDS DEFINED BY CUSTOMER MASTER RECORD (CUSMST)
          A             CUST           5  0       TEXT('Customer numbers')
          A                                       COLHDG('CUSTOMER' 'NUMBER')
          A             NAME          20         TEXT('Customer name')
          A             ADDR          20         TEXT('Customer address')
     A
          A             CITY          20         TEXT('Customer city')
     A
          A             STATE          2         TEXT('State abbreviation')
          A                                       CHECK(MF)
          A             CRECHK         1         TEXT('Credit check')
          A                                       VALUES('Y' 'N')
          A             SEARCH         6  0       TEXT('Customer name search')
          A                                       COLHDG('SEARCH CODE')
          A             ZIP            5  0       TEXT('Zip code')
          A                                       CHECK(MF)
```

```
     A            CUTYPE         15         COLHDG('CUSTOMER' 'TYPE')
     A                                       RANGE(1 5)
A
     A* FIELDS DEFINED BY ITEM MASTER RECORD (ITMAST)
     A            ITEM           5          TEXT('Item number')
     A                                       COLHDG('ITEM' 'NUMBER')
     A                                       CHECK(M10)
     A            DESCRP        18          TEXT('Item description')
     A            PRICE          5  2       TEXT('Price per unit')
     A                                       EDTCDE(J)
     A                                       CMP(GT 0)
     A                                       COLHDG('PRICE')
     A            ONHAND         5  0       TEXT('On hand quantity')
     A                                       EDTCDE(Z)
     A                                       CMP(GE 0)
     A                                       COLHDG('ON HAND')
     A            WHSLOC         3          TEXT('Warehouse location')
     A                                       CHECK(MF)
     A                                       COLHDG('BIN NO')
     A            ALLOC     R                REFFLD(ONHAND *SRC)
     A                                       TEXT('Allocated quantity')
     A                                       CMP(GE 0)
     A                                       COLHDG('ALLOCATED')
A
     A* FIELDS DEFINED BY ORDER HEADER RECORD (ORDHDR)
     A            ORDER          5  0       TEXT('Order number')
     A                                       COLHDG('ORDER' 'NUMBER')
     A            ORDATE         6  0       TEXT('Order date')
     A                                       EDTCDE(Y)
     A                                       COLHDG('DATE' 'ORDERED')
     A            CUSORD        15          TEXT('Cust purchase ord no.')
     A                                       COLHDG('P.O.' 'NUMBER')
A            SHPVIA        15          TEXT('Shipping instructions')
     A            ORDSTS         1          TEXT('Order status code')
     A                                       COLHDG('ORDER' 'STATUS')
     A            OPRNME    R                REFFLD(NAME *SRC)
     A                                       TEXT('Operator name')
     A                                       COLHDG('OPERATOR NAME')
     A            ORDAMT         9  2       TEXT('Total order value')
     A                                       COLHDG('ORDER' 'AMOUNT')
A            INVNBR         5  0       TEXT('Invoice number')
     A                                       COLHDG('INVOICE' 'NUMBER')
     A            PRTDAT         6  0       EDTCDE(Y)
     A                                       COLHDG('PRINTED' 'DATE')
     A            SEQNBR         5  0       TEXT('Sequence number')
     A                                       COLHDG('SEQ' 'NUMBER')
     A            OPNSTS         1          TEXT('Open status')
     A                                       COLHDG('OPEN' 'STATUS')
     A            LINES          3  0       TEXT('Lines on invoice')
     A                                       COLHDG('TOTAL' 'LINES')
     A            ACTMTH         2  0       TEXT('Accounting month')
     A                                       COLHDG('ACCT' 'MONTH')
     A            ACTYR          2  0       TEXT('Accounting year')
     A                                       COLHDG('ACCT' 'YEAR')
A
     A* FIELDS DEFINED BY ORDER DETAIL/LINE ITEM RECORD (ORDDTL)
     A            LINE           3  0       TEXT('Line no. this item')
     A                                       COLHDG('LINE' 'NO')
```

```
     A            QTYORD         3  0       TEXT('Quantity ordered')
     A                                       COLHDG('QTY' 'ORDERED'
     A                                       CMP(GE 0)
     A            EXTENS         6  2       TEXT('Ext of QTYORD x PRICE')
     A                                       EDTCDE(J)
     A                                       COLHDG('EXTENSION')
A
     A* FIELDS DEFINED BY ACCOUNTS RECEIVABLE
     A            ARBAL          8  2       TEXT('A/R balance due')
     A                                       EDTCDE(J)
A
     A* WORK AREAS AND OTHER FIELDS THAT OCCUR IN MULTIPLE PROGRAMS
     A            STATUS        12          TEXT('status description')
     A            A
```

앞의 예에서 DDS를 소스 파일 FRSOURCE에 입력했다고 가정할 경우 멤버명은 DSTREFP입니다. 필드 참조 파일을 작성하려면 다음과 같이 CRTPF 명령을 사용하십시오.

```
CRTPF FILE(DSTPRODLB/DSTREFP)
           SRCFILE(QGPL/FRSOURCE) MBR(*NONE)
           TEXT('Distribution field reference file')
```

매개변수 MBR(*NONE)은 파일에 멤버를 추가하지 않도록 시스템에 지시합니다(필드 참조 파일에는 자료가 들어 있지 않으므로 멤버가 필요없기 때문).

DSTREFP를 참조하여 실제 파일 ORDHDRP를 서술하려면 다음과 같은 DDS를 사용하십시오.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP) - PHYSICAL FILE RECORD DEFINITION
     A                                       REF(DSTREFP)
     A          R ORDHDR                     TEXT('Order header record')
     A            CUST      R
     A            ORDER     R
     A            ORDATE    R
     A            CUSORD    R
     A            SHPVIA    R
     A            ORDSTS    R
     A            OPRNME    R
     A            ORDAMT    R
     A            CUTYPE    R
     A            INVNBR    R
     A            PRTDAT    R
     A            SEQNBR    R
     A            OPNSTS    R
     A            LINES     R
     A            ACTMTH    R
     A            ACTYR     R
     A            STATE     R
A
```

필드 참조 파일명(DSTREFP)이 지정된 REF키워드(45열에서 80열까지)는 사용될 필드 설명이 들어 있는 파일을 나타냅니다. 각 필드의 29열에 있는 R은 필드 설명이 참조 파일로부터 온다는 것을 나타냅니다.

ORDHDRP 파일을 작성할 때 시스템은 DSTREFP 파일을 사용하여 ORDHDR 레코드 형식 내에 포함된 필드의 속성을 결정합니다. ORDHDRP 파일을 작성하려면 CRTPF 명령을 사용하십시오. 앞의 예에서 DDS를 소스 파일 QDDSSRC에 입력했다고 가정할 경우 멤버명은 ORDHDRP입니다.

```
CRTPF FILE(DSTPRODLB/ORDHDRP)
              TEXT('Order Header physical file')
```

주: 이 주제 콜렉션의 일부 예에 사용된 파일은 이 필드 참조 파일을 참조합니다.

데이터베이스 파일의 필드 참조에 자료 사전 사용:

DDS 필드 참조 파일을 사용하는 대신 자료 사전과 대화식 자료 서술 유틸리티(IDDU)를 사용할 수 있습니다. IDDU를 사용하여 자료 사전 내에 필드를 정의할 수 있습니다.

관련 개념

IDDU 사용 PDF

데이터베이스 파일에서 기존 레코드 형식 설명 공유:

레코드 형식은 실제 파일이나 논리 파일(결합 논리 파일 제외)에서 한 번 서술하여 여러 파일에서 이를 사용할 수 있습니다. 새로운 파일을 서술할 때 기존 파일의 레코드 형식이 새로운 파일에 사용되도록 지정할 수 있습니다.

기존 레코드 형식 설명을 공유하면 새 파일의 레코드 형식을 설명하도록 코딩하는 DDS문의 수를 줄일 수 있으며 보조 기억장치 공간도 절약할 수 있습니다.

레코드 형식을 최초로 서술하는 파일은 레코드 형식을 공유하는 파일에 영향을 미치지 않고도 삭제할 수 있습니다. 레코드 형식을 사용하는 마지막 파일이 삭제된 후, 시스템은 자동적으로 그 레코드 형식을 삭제합니다.

다음 예는 두 개의 파일에 대한 DDS를 표시합니다. 첫 번째 파일은 레코드 형식을 설명하고, 두 번째 파일을 첫 번째 파일의 레코드 형식을 공유합니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RECORD1                   PFILE(CUSMSTP)
          A            CUST
          A            NAME
          A            ADDR
          A            SEARCH
          A          K CUST
    A
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RECORD1                   PFILE(CUSMSTP)
          A                                      FORMAT(CUSMSTL)
          A          K NAME
    A
```

첫 번째 예는 파일 CUSMSTL을 나타내며, 이 파일에서는 *Cust*, *Name*, *Addr* 및 *Search* 필드가 레코드 형식을 구성합니다. *Cust* 필드가 키 필드로 지정되었습니다.

두 번째 예의 DDS는 파일 CUSTMSTL1 나타내며, 여기서 FORMAT 키워드는 레코드 형식을 제공할 CUSMSTL을 명명합니다. 레코드 형식명은 반드시 RECORD1이어야 하며 첫 번째 예에 나오는 레코드 형식명과 같습니다. 그 이유는 두 파일 모두 레코드 형식 내에 *Cust*, *Name*, *Addr* 및 *Search* 필드를 가진 같은 형식을 공유하기 때문입니다그러나 파일 CUSMSTL1에는 다른 키 필드인 *Name*이 지정됩니다.

공유 레코드 형식에는 다음과 같은 제한사항이 적용됩니다.

- 실제 파일은 논리 파일의 형식을 공유할 수 없습니다.
- 결합 논리 파일은 다른 파일의 형식을 공유할 수 없으며, 다른 파일도 결합 논리 파일의 형식을 공유할 수 없습니다.
- 뷰는 다른 파일의 형식을 공유할 수 없으며, 다른 파일도 뷰의 형식을 공유할 수 없습니다. 구조화 조회 언어(SQL)에서 *뷰*는 하나 이상의 표에서 자료를 다른 방식으로 표시한 것입니다. 뷰에는 정의된 표에 들어 있는 열 전부 혹은 일부가 포함될 수 있습니다.

관련된 모든 파일을 삭제하고 원래의 파일과 관련된 모든 파일을 다시 작성하여 원래의 레코드 형식이 변경될 경우 이를 공유한 모든 파일에서도 변경됩니다. 원래 형식의 파일만 삭제되어 새로운 레코드 형식으로 다시 작성되었을 경우 파일 형식을 이전에 공유하고 있던 모든 파일은 계속해서 원래의 형식을 사용합니다.

논리 파일이 정의되었으나 필드 설명이 지정되지 않았고 키워드 FORMAT이 지정되지 않은 경우 첫 번째 실제 파일(논리 파일에 대해 PFILE 키워드에서 제일 먼저 지정된 파일)의 레코드 형식이 자동적으로 공유됩니다. 논리 파일에 지정된 레코드 형식명은 실제 파일에 지정된 레코드 형식명과 같아야 합니다.

파일이 다른 파일과 형식을 공유하는지 알아보려면 DSPDBR(데이터베이스 관계 표시) 명령에 RCDFMT 매개변수를 사용하십시오.

실제 및 논리 데이터베이스 파일 사이의 레코드 형식 관계:

CHGPF(실제 파일 변경) 명령으로 필드를 변경, 추가 및 삭제할 경우 동일한 레코드 형식을 공유하는 실제 및 논리 파일 사이에는 특별한 관계가 존재합니다.

- 실제 파일에서 필드의 길이를 변경하면 논리 파일의 필드 길이도 변경됩니다.
- 실제 파일에 필드를 추가하면 논리 파일에도 필드가 추가됩니다.
- 실제 파일에서 필드를 삭제하면 키가 지정된 필드나 선택/생략문과 같이 DDS에 다른 종속성이 없는 한 논리 파일에서도 그 필드가 삭제됩니다.

실제 및 논리 데이터베이스 파일에 대한 레코드 형식 공유 제한:

동일한 데이터베이스 오브젝트를 여러 번 복제할 경우 이 레코드 형식 공유 제한이 발생할 수 있습니다.

레코드 형식은 32K 오브젝만이 공유할 수 있습니다. 이 제한에 도달하면 오류 메세지가 발행됩니다.

주: 형식 공유는 복제된 파일에 대해 수행됩니다. 이 형식은 32 767번까지 공유할 수 있습니다. 이 형식을 공유하는 파일이 32 767번 이상으로 복제되면 복제된 파일에 새로운 형식이 작성됩니다.

데이터베이스 파일 및 멤버 속성 지정:

데이터베이스 파일 작성시, 데이터베이스의 속성은 파일 및 멤버와 함께 저장됩니다. 속성들은 데이터베이스 명령 매개변수로 지정합니다.

관련 개념

제어 언어(CL)

관련 참조

실제 파일 작성(CRTPF) 명령

논리 파일 작성(CRTLF) 명령

소스 실제 파일 작성(CRTSRCPF) 명령

실제 파일 멤버 추가(ADDPFM) 명령

논리 파일 멤버 추가(ADDLFM) 명령

CHGPF(실제 파일 변경) 명령

실제 파일 멤버 변경(CHGPFM) 명령

CHGLF(논리 파일 변경) 명령

논리 파일 멤버 변경(CHGLFM) 명령

CHGSRCPF(소스 실제 파일 변경) 명령

FILE 및 MBR(파일명 및 맴버명) 매개변수 지정:

데이터베이스 파일 작성 시 FILE 및 MBR 매개변수로 파일 및 멤버에 이름을 지정할 수 있습니다.

파일명은 작성 명령어의 FILE 매개변수에 지정합니다. 파일이 저장된 라이브러리에 이름을 지정할 수도 있습니다. 실제 파일이나 논리 파일 작성시, 시스템은 보통 그 파일과 같은 이름의 멤버를 작성합니다. 그러나 작성 명령어의 MBR 매개변수에 멤버명을 지정할 수 있습니다. 또한 작성 명령어에 MBR(*NONE)을 지정하여 어떤 멤버도 만들지 않을 수 있습니다.

주: 시스템은 소스 실제 파일에 대해 멤버를 자동으로 작성하지 *않습니다.*

DTAMBRS(실제 파일 맴버 제어) 매개변수 지정:

CRTLF(논리 파일 작성) 명령의 DTAMBRS 매개변수로 실제 파일 멤버 읽기를 제어할 수 있습니다.

다음과 같이 지정할 수 있습니다.

- 읽힐 실제 파일 멤버의 순서
- 사용될 실제 파일 멤버의 수

관련 개념

50 페이지의 『논리 파일 멤버 정의』

자료를 논리 그룹별로 분리하기 위해서 논리 파일에 멤버를 정의할 수 있습니다. 논리 파일 멤버는 한 개 또는 여러 개의 실제 파일 멤버와 연관될 수 있습니다.

SRCFILE 및 SRCMBR(소스 파일 및 소스 맴버) 매개변수 지정:

SRCFILE 및 SRCMBR 매개변수를 사용하여 DDS문을 포함하는 소스 파일명과 멤버명을 지정할 수 있습니다.

이름을 지정하지 않을 경우:

- 디폴트 소스 파일명은 QDDSSRC입니다.
- 디폴트 멤버명은 FILE 매개변수에 지정된 이름이 됩니다.

FILETYPE(데이터베이스 파일 유형) 매개변수 지정:

FILETYPE 매개변수로 데이터베이스 파일 유형을 지정할 수 있습니다.

데이터베이스 파일 유형은 자료(*DATA)나 소스(*SRC)가 됩니다. CRTPF(실제 파일 작성) 명령과 CRTLF(논리 파일 작성) 명령은 디폴트 자료 파일 유형(*DATA)을 사용합니다.

MAXMBRS(허용되는 최대 맴버 수) 매개변수 지정:

MAXMBRS 매개변수로 파일이 보유할 수 있는 최대 멤버 수를 지정할 수 있습니다.

실제 파일 및 논리 파일의 디폴트 최대 멤버 수는 한 개이며, 소스 실제 파일에 대한 디폴트는 *NOMAX입니다.

UNIT(자료 저장 위치) 매개변수 지정:

시스템이 보조 기억장치에서 그 파일이 위치할 장소를 찾습니다. UNIT 매개변수로 파일을 저장할 위치를 지정할 수 있습니다.

주: 버전 3 릴리스 6에 유효한 UNIT 매개변수는 다음 명령들에 대해 작동하지 않는(NOP) 함수입니다.

- 실제 파일 작성(CRTPF)
- 논리 파일 작성(CRTLF)
- CRTSRCPF(소스 실제 파일 작성)
- CHGPF(실제 파일 변경)
- CHGLF(논리 파일 변경)
- CHGSRCPF(소스 실제 파일 변경)

이 매개변수는 코드화될 수 있으며, 이 매개변수로 인해 오류가 발생하지 않습니다. 이 매개변수는 무시됩니다.

UNIT 매개변수는 다음과 같은 것을 지정합니다.

- 실제 파일(PF)에서 자료 레코드의 위치
- 실제 파일과 논리 파일에 대한 액세스 경로

다음과 같은 경우에 자료는 다른 장치에 위치합니다.

- 그 장치에 공백이 충분하지 않을 경우.

- 사용자의 시스템에서 그 장치가 사용 불가능할 경우

파일 멤버가 추가될 때 그 파일이 요청한 장치에 위치할 수 없음을 나타내는 정보용 메세지가 나옵니다. (파일 멤버가 확장될 때에는 메세지가 나오지 *않습니다*.)

UNIT 매개변수 추가 정보

일반적으로 UNIT 매개변수는 지정하면 안됩니다. 시스템이 선택한 디스크 장치에 파일을 저장하도록 하십시오. 일반적으로 이 방법이 성능이 더 좋고, 사용자가 보조 기억장치를 관리해야 하는 수고를 덜 수 있습니다. 장치 번호와 보조 기억장치 풀(ASP)을 지정할 경우 장치 번호는 무시됩니다.

관련 개념

독립 디스크 풀

FRCRATIO(보조 기억장치에 자료 기록 횟수) 매개변수 지정:

데이터베이스 파일 작성, 변경 또는 대체 명령에 강제 기록 수(FRCRATIO) 매개변수를 사용하여 보조 기억장치에 데이터베이스 변경 내용이 기록되는 시기를 제어할 수 있습니다.

보통, 변경된 자료를 언제 주 기억장치에서 보조 기억장치로 옮겨 적을 것인가는 시스템이 결정합니다. 파일 닫기(공유된 닫기는 제외)와 자료의 끝 강제(force-end-of-data) 조작은 보조 기억장치에 대해 나머지 갱신, 삭제 및 추가 조작을 강행합니다. 파일이 저널중인 경우 FRCRATIO 매개변수는 보통 *NONE이어야 합니다.

FRCRATIO 매개변수 추가 정보

FRCRATIO 매개변수를 사용할 때에는 시스템의 성능 및 회복상에서 고려할 사항이 있습니다.

관련 개념

257 페이지의『데이터베이스 회복 및 복원』

이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

FRCACCPTH(액세스 경로 작성 횟수) 매개변수 지정:

강제 액세스 경로(FRCACCPTH) 매개변수로 액세스 경로 작성 횟수를 지정할 수 있습니다. FRCACCPTH 매개변수는 액세스 경로가 보조 기억장치에 기록되는 시점을 제어합니다.

FRCACCPTH(*YES)를 지정하면 액세스 경로가 변경될 때마다 보조 기억장치에 액세스 경로가 강제로 기록됩니다. 이렇게 하면 시스템 실패 시 액세스 경로를 다시 작성해야 할 가능성이 줄어듭니다.

FRCACCPTH 매개변수 추가 정보

FRCACCPTH(*YES)를 지정하면 액세스 경로 변경시 성능이 감소될 수 있습니다. 강제 액세스 경로의 또 다른 방법은 액세스 경로를 저널링하는 것입니다.

관련 개념

257 페이지의 『데이터베이스 회복 및 복원』
이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

LVLCHK(레코드 형식 설명 변경 검사) 매개변수 지정:

파일이 열리면, 시스템은 데이터베이스 파일 정의의 변경사항에 대해 점검합니다. LVLCHK 매개변수로 레코드 형식 설명 변경 검사를 지정할 수 있습니다.

프로그램이 파일을 처리할 수 없는 정도로 파일이 변경된 경우 시스템은 프로그램에 알립니다. 디폴트 값은 레벨 검사를 수행합니다. 다음과 같은 경우 원한다면 레벨 검사를 지정할 수 있습니다.

- 파일 작성 시
- 데이터베이스 파일 변경 명령 사용 시

OVRDBF(데이터베이스 파일 대체) 명령으로 레벨 검사를 무시하고 시스템을 대체할 수 있습니다.

예: 레벨 검사

예를 들어, 두 달 전에 프로그램을 컴파일했는데 그 때 그 프로그램은 각 레코드에 필드가 세 개 있는 것으로 정의했다고 가정합시다. 그리고 지난 주 다른 프로그래머가 그 레코드 형식에 새로운 필드 하나를 추가하여 이제 각 레코드에는 4개의 필드가 있는 것으로 가정합시다. 프로그램이 파일을 열려고 할 때 지난번 프로그램이 컴파일된 이후로 파일 정의에 중요한 변경이 발생하였음을 시스템이 프로그래머에게 통보합니다. 이러한 통보를 레코드 형식 레벨 검사라고 합니다.

MAINT(현재 액세스 경로 유지보수) 매개변수 지정:

MAINT 매개변수로 닫힌 파일에 대한 액세스 경로의 유지보수 방법을 지정할 수 있습니다.

파일이 열려 있는 동안 파일의 자료가 변경되면 시스템은 이 파일에 대한 액세스 경로를 유지보수합니다. 그러나 동일한 자료에 대해 여러 액세스 경로가 있을 수 있으므로 한 파일의 자료를 변경하면 현재 열려 있지 않은(사용 중이 아닌) 다른 파일의 액세스 경로가 변경될 수 있습니다.

닫힌 파일에 대한 액세스 경로를 유지보수하는 세가지 방법은 다음과 같습니다.

- 액세스 경로의 **즉시** 유지보수는 파일이 열리고 닫힘에 관계없이 연관된 자료가 변경될 때 액세스 경로가 즉시 유지보수됨을 의미합니다. 참조 제한조건이 사용하는 액세스 경로는 항상 즉시 유지보수 상태로 있습니다.
- 액세스 경로의 **재구성 유지보수**는 액세스 경로가 파일이 닫혀 있을 때는 유지보수되지 않고 파일이 열려 있는 동안에만 유지보수되며, 다음 번에 파일이 열릴 때 액세스 경로가 재구성됨을 의미합니다. 리빌드 유지보수된 파일이 닫히면, 시스템은 액세스 경로의 유지보수를 중단합니다. 그 파일이 다시 열릴 때 액세스 경로는 완전히 재구성됩니다. 하나 이상의 프로그램이 리빌드 유지보수가 지정된 특정 파일 멤버를 열었을 경우 시스템은 마지막 사용자가 파일 멤버를 닫을 때까지 그 멤버에 대한 액세스 경로를 유지보수합니다.

- 액세스 경로의 **지연** 유지보수는 파일 멤버가 그 다음번에 열린 후, 열려 있는 동안에만 액세스 경로가 유지보수됨을 의미합니다. 그러나 액세스 경로는 리빌드 유지보수처럼 재구성되지는 않습니다. 액세스 경로에 대한 변경 내용은 멤버가 닫힌 시간부터 다시 열릴 때까지 수집됩니다. 멤버가 열릴 때 수집된 변경사항만이 액세스 경로에 병합됩니다.

파일에 대해 유지보수 유형을 지정하지 않으면 즉시 유지보수가 디폴트 값이 됩니다.

MAINT 매개변수 비교:

이 주제에서는 파일 열기 및 처리 시 즉시, 리빌드 및 지연 유지보수의 영향을 비교합니다.

표 2. MAINT 값

기능	즉시 유지보수	리빌드 유지보수	지연 유지보수
열기	액세스 경로가 최신 상태이므로 빨리 열림.	액세스 경로가 재구성되어야 하므로 늦게 열림.	액세스 경로가 변경되어야 하나 재구성될 필요는 없으므로 비교적 빨리 열림. 광범위한 변경이 필요한 경우에는 열리는 시간이 늦어짐.
처리	많은 즉시 유지보수 액세스 경로가 변경될 자료에 설정되어 있으면 갱신/출력 조작이 늦어짐(시스템이 액세스 경로를 유지보수해야 함).	많은 리빌드 유지보수 액세스 경로가 변경될 자료에 설정되어 있으며 열려 있지 않으면 갱신/출력 조작이 빨라짐(시스템이 액세스 경로를 유지보수할 필요가 없음).	많은 지연 유지보수 액세스 경로가 변경되는 자료에 설정되어 있고, 이들이 열려 있지 않으면 갱신/출력 조작이 조금 빨라짐(시스템이 변경사항을 기록하나, 액세스 경로 자체는 유지보수되지 않음).

주:

1. 지연 또는 리빌드 유지보수는 고유 키를 가진 파일에는 지정할 수 없습니다.
2. 리빌드 유지보수는 액세스 경로가 저널되고 있는 파일에는 지정할 수 없습니다.

MAINT 매개변수 추가 정보:

지정할 액세스 경로 유지보수 유형은 레코드 수와 파일이 닫혀 있는 동안 파일에 대한 추가, 삭제 또는 갱신 조작 발생 빈도에 따라 결정됩니다.

파일 멤버가 닫혀있는 동안 액세스 경로에 비교적 변경사항이 많지 않은 파일에 대해서는 지연 유지보수를 사용해야 합니다. 지연 유지보수는 즉시 유지보수되는 액세스 경로의 수를 줄임으로써 시스템의 과도한 업무를 줄여줍니다. 또한 액세스 경로를 재구성할 필요가 없으므로 열기 처리가 더 빨라질 수 있습니다.

즉시 유지보수는 자주 사용되는 액세스 경로에 대해 지정할 수 있으며, 파일이 열릴 때 액세스 경로의 재구성을 기다릴 수 없는 경우에도 이를 지정할 수 있습니다. 반면에 액세스 경로를 구성하는 레코드 키에 대해 변경이 자주 발생하지 않을 경우 자주 사용하지 않는 액세스 경로에 대해 지연 유지보수를 지정할 수 있습니다.

일반적으로 대화식으로 사용되는 파일의 경우 즉시 유지보수를 사용하면 응답 시간이 좋아집니다. 일괄처리 작업에 사용되는 파일의 경우 즉시, 지연 또는 리빌드 유지보수 모두 적당하나 멤버의 크기와 변경의 빈도에 따라 달라지게 됩니다.

RECOVER(회복) 매개변수 지정:

장애가 발생하면 보조 기억장치에 강제 기록되지 않았거나 RECOVER 매개변수로 저널되지 않은 변경된 액세스 경로는 다시 구축해야 합니다.

액세스 경로를 다시 구축하고 자료를 회복하려면 다음 명령에서 RECOVER 매개변수를 사용하면 됩니다. 이 명령들은 액세스 경로가 재구축되는 시기를 지정합니다.

- 실제 파일 작성(CRTPF)
- 논리 파일 작성(CRTLF)
- CRTSRCPF(소스 실제 파일 작성)

주: 액세스 경로는 초기 프로그램 로드(IPL) 시, IPL 이후 또는 파일이 열릴 때 재구성됩니다.

표 3은 중복 키 옵션과 유지보수 옵션의 가능한 조합을 나타냅니다.

표 3. 회복 옵션

이 중복 키 옵션 사용	이 유지보수 옵션	회복 옵션
고유함	즉시	IPL 중에 재구성(*IPL) IPL 이후 재구성(*AFTIPL, 디폴트) IPL 시 재구성하지 않고 첫 번째 열릴 때까지 대기(*NO)
고유하지 않음	즉시 또는 지연	IPL 중에 재구성(*IPL) IPL 이후 재구성(*AFTIPL) IPL시 재구성하지 않고 첫 번째 열릴 때까지 대기(*NO, 디폴트)
고유하지 않음	재구성	IPL시 재구성되지 않고 첫 번째 열릴 때까지 대기(*NO, 디폴트 값)

RECOVER 매개변수 추가 정보

IPL이 수동 모드인 경우 회복해야 할 액세스 경로가 있는 파일 리스트가 다음 IPL 시 '액세스 경로 재구성 편집' 화면에 표시됩니다. 화면에서 적절한 옵션을 선택하여 파일에 원래 지정된 회복 옵션을 편집할 수 있습니다. IPL이 완료된 후 EDTRBDAP(액세스 경로 재구성 편집) 명령을 사용하여 액세스 경로의 재구성 순서를 정할 수 있습니다. IPL이 무인(unattended)인 경우 '액세스 경로 재구성 편집' 화면이 표시되지 않고 액세스 경로는 RECOVER 매개변수에 의해 정해진 순서로 재구성됩니다. *AFTIPL과 *NO(열림) 액세스 경로만 표시됩니다.

관련 개념

257 페이지의『데이터베이스 회복 및 복원』

이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

백업 및 회복 PDF

SHARE(파일 공유) 매개변수 지정:

데이터베이스 시스템을 사용하면 여러 사용자가 동시에 파일을 액세스하고 변경할 수 있습니다. SHARE 매개변수로 파일 공유를 지정할 수 있습니다.

SHARE 매개변수는 동일한 작업 내에서 열린 파일을 공유하도록 합니다. 예를 들어, 작업 시 파일을 공유하면 작업 프로그램은 파일의 상태 레코드 위치 및 버퍼를 공유할 수 있습니다. 파일을 공유하면 다음과 같은 것을 줄여 성능이 향상됩니다.

- 작업이 필요로 하는 기억장치
- 파일을 열고 닫는 데 소요되는 시간

관련 개념

124 페이지의 『동일한 작업 또는 활성 그룹에서 데이터베이스 파일 공유』
기본적으로 데이터베이스 관리 시스템에서는 하나의 파일이 동시에 많은 사용자에 의해 읽히고 변경되도록 되어 있습니다. 그러나 SHARE 매개변수를 통해 동일한 작업이나 활성 그룹의 데이터베이스 파일을 공유할 수 있습니다.

WAITFILE 및 WAITRCD(잠긴 파일 및 레코드 대기 시간) 매개변수 지정:

파일 작성 시 다른 작업이 WAITFILE 및 WAITRCD 매개변수로 파일이나 레코드를 잠근 경우 프로그램이 그 파일이나 그 파일의 레코드를 대기해야 하는 시간을 지정할 수 있습니다.

파일 또는 레코드가 해제되기 전에 대기 시간이 종료될 경우 작업이 해당 파일을 사용할 수 없거나 레코드를 읽을 수 없었음을 알리는 메세지가 프로그램에 전달됩니다.

관련 개념

120 페이지의 『레코드 잠금』
iSeries용 DB2 Universal Database에는 레코드에 대한 내장 무결성이 있습니다. 예를 들어, PGMA가 갱신 조작을 위해 레코드를 읽으면 그 레코드가 잠깁니다. PGMA가 그 레코드를 해제할 때까지는 다른 프로그램이 갱신 조작을 위해 동일한 레코드를 읽을 수 없지만 조회용으로는 읽을 수 있습니다.

121 페이지의 『파일 잠그기』
일부 파일 조작은 처리 시간 동안 파일을 독점적으로 할당합니다. 파일이 배타적으로 할당된 경우 파일을 열려는 프로그램은 파일이 릴리스될 때까지 기다려야 합니다. 그러나 WAITFILE 매개변수를 사용하여 파일을 사용할 수 있도록 대기 시간을 설정할 수 있습니다.

AUT(공용 권한) 매개변수 지정:

공용 권한은 한 사용자가 파일(또는 시스템상의 다른 오브젝트)에 대해 특정 권한을 가지고 있지 않거나 그 파일에 대해 특정 권한을 가진 그룹에 속하지 않은 경우 파일에 대해 갖는 권한입니다. 파일 작성 시 AUT 매개변수로 공용 권한을 지정할 수 있습니다.

관련 개념

109 페이지의 『공용 권한 지정』
파일 작성 시 공용 권한을 지정하고 부여할 수 있습니다. 공용 권한에 대해 지정할 수 있는 값과 이를 부여하는 방법에 대해 읽어 보십시오.

SYSTEM(파일이 작성되는 시스템) 매개변수:

SYSTEM(파일이 작성되는 시스템) 매개변수를 지정할 수 있습니다. 파일이 분산 자료 관리(DDM)를 지원하는 로컬 시스템에서 작성되는지 아니면 리모트 시스템에서 파일이 작성되는지를 지정할 수 있습니다.

관련 개념

분산 자료 관리

TEXT(파일 및 맴버 텍스트) 매개변수 지정:

TEXT 매개변수로 작성하는 각 파일 및 멤버에 텍스트 설명을 지정할 수 있습니다. 텍스트 자료는 파일 및 멤버에 관한 정보를 서술하는 데 유용합니다.

CCSID(코드화 문자 세트 ID) 매개변수 지정:

CCSID 매개변수로 실제 파일에 코드화 문자 세트 ID(CCSID)를 지정할 수 있습니다. CCSID는 파일에 포함된 문자 유형 필드에 대한 문자 세트 및 코드화 체계를 설명합니다.

관련 개념

i5/OS 국제화

SRTSEQ(정렬 순서) 매개변수 지정:

SRTSEQ 매개변수로 실제 파일이나 논리 파일에 정렬 순서를 지정할 수 있습니다.

SRTSEQ, CCSID, LANGID 매개변수 값은 파일이 사용하는 정렬 순서 표를 결정합니다.

다음과 같이 지정할 수 있습니다.

- 시스템이 지원하는 유일 또는 공유 배열 가중치를 가진 정렬 순서표. 각 지원 언어에는 해당 정렬 순서표가 있음.
- 사용자가 작성한 모든 정렬 순서표
- 문자 세트에 있는 문자의 16진 값
- 현재 작업 정렬 순서나 ALTSEQ 매개변수에 지정된 작업의 정렬 순서. 정렬 순서표는 정렬 순서가 *HEX일 경우를 제외하고는 파일에 저장됩니다.

LANGID(언어 ID) 매개변수 지정:

SRTSEQ 매개변수 값이 *LANGIDSHR 또는 *LANGIDUNQ일 경우 시스템이 사용하는 언어 ID를 지정할 수 있습니다.

LANGID, CCSID 및 SRTSEQ 매개변수의 값은 파일이 사용하는 정렬 순서표를 결정합니다. 실제 파일과 논리 파일에 대해 LANGID 매개변수를 설정할 수 있습니다.

시스템에서 제공하는 언어 ID나 현재 작업에서 사용되는 언어 ID를 지정할 수 있습니다.

실제 파일 설정

이 주제에서는 실제 파일 작성 시 특수 고려사항에 대해 설명합니다.

관련 개념

95 페이지의『데이터베이스 파일에 대한 액세스 경로 설명』
이 주제에서는 데이터베이스 파일에 대한 액세스 경로를 설명하는 여러 가지 방법을 설명합니다.

관련 참조

22 페이지의『예: DDS를 사용하여 실제 파일 서술』
이 예는 DDS를 사용하여 실제 파일을 서술하는 방법을 표시합니다.

실제 파일 작성:

이 주제는 자료 서술 스펙(DDS)을 사용하여 실제 파일을 작성하는 방법을 표시합니다.

실제 파일을 작성하기 전에 라이브러리 및 소스 파일을 작성해야 합니다.

실제 파일을 작성하려면 다음 단계를 수행하십시오.

1. DDS를 사용하는 경우 실제 파일에 대한 DDS를 소스 파일에 입력합니다. 이것은 소스 입력 유틸리티(SEU)를 사용하여 수행될 수 있습니다. SEU는 iSeries용 IBM WebSphere Development Studio의 일부입니다.
2. 실제 파일을 작성합니다. CRTPF(실제 파일 작성) 명령 또는 CRTSRCPF(소스 실제 파일 작성) 명령을 사용할 수 있습니다.

다음 명령은 DDS를 사용하여 한 개의 멤버 파일을 작성하고, 그 파일을 DSTPRODLB라고 하는 라이브러리에 넣습니다.

```
CRTPF FILE(DSTPRODLB/ORDHDRP)
            TEXT('Order header physical file')
```

위에서 보는 바와 같이 이 명령은 디폴트 값을 사용합니다. SRCFILE 및 SRCMBR 매개변수에 대해 시스템은 QDDSSRC라고 하는 소스 파일과 ORDHDRP라고 하는 멤버(파일명과 동일함)에 있는 DDS를 사용합니다. 동일한 이름의 한 개 멤버를 갖는 파일 ORDHDRP가 라이브러리 DSTPRODLB에 위치하게 됩니다.

실제 파일과 유사한 것이 표입니다. 표는 iSeries Navigator나 CREATE TABLE SQL문으로 생성할 수 있습니다.

관련 개념

16 페이지의『라이브러리 작성』
라이브러리는 다른 오브젝트에 대한 디렉토리로서 사용되는 시스템 오브젝트입니다. 라이브러리는 관련 오브젝트를 그룹화하고 사용자가 이름별로 오브젝트를 찾는 것을 허용합니다.

17 페이지의『소스 파일 작성』
이 주제에서는 소스 파일 작성 방법에 대해 설명합니다.

276 페이지의『소스 파일에 대한 작업』
이 주제에서는 다양한 방법을 사용하여 자료를 입력하고 유지보수하는 방법을 설명합니다.

21 페이지의 『데이터베이스 파일 설명』
이 주제는 iSeries 데이터베이스 파일 서술에 대한 몇 가지 방법을 소개하며 자료 서술 스펙(DDS)에 자료 정의를 위한 대부분의 옵션에 있으므로 DDS를 사용하여 데이터베이스 파일을 설명하는 방법을 중점적으로 살펴봅니다.

iSeries Navigator 시작

관련 참조

실제 파일 작성(CRTPF) 명령

소스 실제 파일 작성(CRTSRCPF) 명령

CREATE TABLE

실제 파일 작성 시 실제 파일 및 멤버 속성 지정:

이 주제에서는 CRTPF(실제 파일 작성), CRTSRCPF(소스 실제 파일 작성), CHGPF(실제 파일 변경), CHGSRCPF(소스 실제 파일 변경), ADDPFM(실제 파일 멤버 추가) 및 CHGPFM(실제 파일 멤버 변경) 명령에서 실제 파일과 멤버에 지정할 수 있는 일부 속성에 대해 설명합니다.

만기일:

EXPDATE 매개변수는 파일의 각 멤버에 대한 만기일을 지정합니다(ADDPFM, CHGPFM, CRTPF, CHGPF, CRTSRCPF 및 CHGSRCPF 명령).

만기일이 지나면 시스템 오퍼레이터는 파일이 열릴 때 통보를 받게 됩니다. 그러면 시스템 오퍼레이터는 만기일을 대체하여 작업을 계속하거나 작업을 중단할 수 있습니다. 각 멤버는 서로 다른 만기일을 가질 수 있으며, 그 만기일은 멤버가 파일에 추가될 때 지정됩니다.

관련 개념

119 페이지의 『파일 만기일 검사』
시스템은 지정한 파일이 현재 유효한지를 확인할 수 있습니다. 그러나 EXPDATE 및 EXPCHK 매개변수를 사용하여 파일에 만료일을 지정할 수 있습니다.

실제 파일 맴버 크기:

SIZE 매개변수는 각 멤버 내에 들어갈 수 있는 최대 레코드 수를 지정합니다(CRTPF, CHGPF, CRTSRCPF 및 CHGSRCPF 명령).

최대 수를 결정하기 위해서는 다음과 같은 공식을 사용할 수 있습니다.

R + (I * N)

위에서

R 초기 레코드 수

I 매번 추가되는 레코드 수(증가분)

N 증가분의 추가 횟수

SIZE 매개변수의 디폴트 값은 다음과 같습니다.

R 10 000

I 1000

N 3(CRTPF 명령)

499(CRTSRCPF 명령)

예를 들어 RDL 5000 레코드에 각각 1000개의 레코드를 3번 증가할 수 있도록 작성된 파일이 있는 것으로 가정하십시오. 시스템은 초기 레코드 수인 5000개에 3번에 걸쳐 각기 1000개씩을 추가할 수 있으므로 총 최대 수는 8000개가 됩니다. 최대 수에 도달하면 시스템 오퍼레이터는 작업을 중단하거나 시스템으로 하여금 레코드의 다른 증가분을 추가하도록 하여 작업을 계속할 수 있습니다. 레코드의 증가분이 추가되면 시스템 이력 기록부에 메세지가 전달됩니다. 파일이 최대 *크기* 이상으로 확장될 경우 최소 확장 *크기*는 비록 이것이 지정된 증가분보다 *크*더라도 현재 파일 *크기*의 10%입니다. 디폴트 *크기*를 선택하거나 *크기*를 지정하는 대신 *NOMAX를 지정할 수도 있습니다.

관련 참조

9 페이지의『데이터베이스 파일 *크기*』

iSeries 서버에서 파일을 설계할 경우 데이터베이스 파일 *크기*를 고려해야 합니다.

기억장치 할당:

ALLOCATE 매개변수는 파일에 멤버가 추가될 때 멤버에 할당되는 기억장치를 제어합니다(CRTPF, CHGPF, CRTSRCPF 및 CHGSRCPF 명령).

할당되는 기억장치는 멤버에 대한 초기 레코드 수를 저장하기에 충분한 *크기*입니다. 멤버 추가 시 기억장치를 할당하지 않을 경우 시스템은 자동적으로 필요한 만큼 기억장치 할당을 늘립니다. SIZE 매개변수에 최대 *크기*를 지정한 경우에만 ALLOCATE 매개변수를 사용할 수 있습니다. SIZE(*NOMAX)가 지정되면 ALLOCATE(*YES)를 지정할 수 없습니다.

기억장치 할당 방법:

CONTIG 매개변수는 멤버에 대해 실제 기억장치를 할당하는 방법을 제어합니다(CRTPF 및 CRTSRCPF 명령).

기억장치를 할당할 경우 한 멤버의 초기 레코드 수에 대한 기억장치가 인접하도록 요구할 수 있습니다. 다시 말해 한 멤버 내의 모든 레코드가 실제로 함께 상주합니다. 인접 기억장치가 충분하지 않을 경우 기억장치가 인접해서 할당되지 않으며 이를 알리는 정보 메세지가 멤버 추가 시 할당을 요청한 작업에 전달됩니다.

주: 실제 파일이 처음 작성될 때 시스템은 항상 해당 초기 기억장치에 인접해서 할당되도록 노력합니다. CONTIG(*NO)와 CONTIG(*YES) 사이의 유일한 차이점은 CONTIG(*YES)를 사용하여 파일을 작성할 때 인접 기억장치가 할당될 수 없으면 시스템이 작업 기록부에 메세지를 전달한다는 점입니다. 작성 후에 파일이 확장될 때에는 CONTIG 매개변수에 지정된 값과는 관계없이 아무런 메세지도 전달되지 않습니다.

레코드 길이:

RCDLEN 매개변수는 파일의 레코드 길이를 지정합니다(CRTPF 및 CRTSRCPF 명령).

파일이 레코드 레벨로만 서술되면 사용자는 파일 작성 시 RCDLEN 매개변수를 지정합니다. 파일이 DDS, IDDU 또는 SQL을 사용하여 서술될 경우 이 매개변수를 지정할 수 없습니다. (시스템이 자동으로 필드 레벨 설명에서 파일의 레코드 길이를 결정하기 때문입니다.)

삭제 레코드:

DLTPCT 매개변수는 시스템이 시스템 이력 기록부에 메세지를 보내기 전에 파일에 포함될 수 있는 삭제 레코드의 비율을 지정합니다(CRTPF, CHGPF, CRTSRCPF 및 CHGSRCPF 명령).

파일이 닫히면 시스템은 삭제 레코드의 비율을 판별하기 위해 멤버를 검사합니다. 비율이 DLTPCT 매개변수에 지정된 값을 초과하면 이력 기록부로 메세지가 전달됩니다. 이력 기록부 처리에 대한 정보는 제어 언어(CL) 주제를 참조하십시오. 파일의 삭제 레코드가 지정된 비율에 도달하는 때를 알려는 이유 중의 하나는 삭제 레코드가 사용한 공간을 재생하려는 데 있습니다. 삭제 레코드에 관한 메세지를 수신한 후, RGZPFM(실제 파일 멤버 재구성) 명령을 수행하여 그 공간을 재생할 수 있습니다. 또한 DLTPCT 매개변수에 *NONE 값을 사용하여 삭제 레코드 검사를 생략하도록 바이패스할 수 있습니다. DLTPCT 매개변수의 디폴트는 *NONE입니다.

REUSEDLT 매개변수는 삭제 레코드 공간이 후속 기록 조작에 재사용되는지의 여부를 지정합니다(CRTPF 및 CHGPF 명령). REUSEDLT 매개변수에 *YES를 지정하면 그 파일에 대한 모든 삽입 요청이 삭제 레코드 공간의 재사용을 시도합니다. 삭제 레코드 공간의 재사용은 RGZPFM 명령을 수행하지 않고도 삭제 레코드가 사용한 공간을 재생할 수 있습니다. 삭제 레코드를 재사용하도록 CHGPF 명령이 파일 변경에 사용될 경우 명령은 특히 파일이 크고, 삭제 레코드가 많을 때 장시간 수행됩니다. 주의할 점은 다음과 같습니다.

- *입력순*이란 용어는 삭제 레코드 공간을 재사용하는 파일에 대해서는 그 의미를 상실합니다. 레코드는 삭제 레코드 공간이 재사용될 때 파일의 끝에 더 이상 삽입되지 않습니다.
- 새로운 실제 파일이 삭제 레코드 공간 재사용 속성으로 작성되고 파일이 키순이면, FIFO나 LIFO 액세스 경로 속성은 실제 파일에 지정될 수 없으며 FIFO나 LIFO 액세스 경로 속성을 가진 어떠한 키순 논리 파일도 실제 파일 위에 설정될 수 없습니다.
- 중복 키에 대해 FIFO나 LIFO 순서를 지정하는 실제 파일 위에 논리 파일이 있거나 실제 파일이 FIFO 또는 LIFO 중복 키 순서를 가지면, 삭제 레코드 공간을 재사용하기 위해 기존 실제 파일을 변경할 수 없습니다.
- 삭제 레코드 공간 재사용은 직접 파일로 처리되거나 상대 레코드 번호를 사용하여 처리되는 파일에 대해서는 지정되지 않아야 합니다.

관련 개념

239 페이지의『실제 파일 재구성』

이 주제에서는 i5/OS 오퍼레이팅 시스템에서 실제 파일을 재구성하는 방법과 재구성 조작을 일시중단하거나 취소하는 방법 및 선택할 수 있는 재구성 조작 유형에 대해 설명합니다. 실제 파일 재구성에 대한 고려사항도 포함됩니다.

115 페이지의 『삭제 레코드 재사용』
데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

실제 파일 기능:

파일 기능은 데이터베이스 파일 권한과는 독립적으로 데이터베이스 파일에 대해 어떤 입출력 조작이 허용되는지를 제어하는 데 사용됩니다. ALWUPD 및 ALWDLT 매개변수는 파일의 갱신 가능 및 삭제 가능 여부를 지정합니다(CRTPF 및 CRTSRCPF 명령).

관련 개념

105 페이지의 『데이터베이스 보안』
이 주제에서는 데이터베이스 보안을 위해 취할 수 있는 조치에 대해 설명합니다.

소스 유형:

SRCTYPE 매개변수는 소스 파일 내의 멤버에 소스 유형을 지정합니다(ADDPFM 및 CHGPFM 명령).

소스 유형은 멤버에 사용되는 구문 검사 프로그램, 프롬트 작업 및 형식화를 결정합니다. 고유 소스 유형(COBOL 및 RPG와 같은 iSeries 지원 유형 이외의 것) 지정시 사용자는 고유한 유형을 처리하기 위해 프로그래밍을 제공해야 합니다.

소스 유형이 변경되면 변경사항은 멤버가 그 다음에 열릴 때부터 반영됩니다. 즉, 현재 열려 있는 멤버는 영향을 받지 않습니다.

파일 작성 시 내재적 저널링:

실제 파일 작성 시 저널링을 자동으로 시작할 수 있습니다.

QDFTJRN이라는 자료 영역이 실제 파일(PF)이 작성되는 동일한 라이브러리에 있고 사용자에게 자료 영역에 대한 권한이 있는 경우 저널링은 다음 제한사항이 모두 참이면 자료 영역에 명명된 저널에서 시작됩니다.

- 실제 파일에 대해 식별된 라이브러리가 QSYS, QSYS2, QRECOVERY, QSPL, QRCL, QRPLOBJ, QGPL 또는 QTEMP일 수 없습니다.
- 자료 영역에 지정된 저널이 있어야 하며 사용자아게 저널에 대한 저널링을 시작할 권한이 있어야 합니다.
- 자료 영역의 첫 번째 10바이트에는 저널을 찾을 라이브러리명이 있어야 합니다.
- 두 번째 10바이트에는 저널명이 있어야 합니다.
- 세 번째 *n*바이트에는 값 *FILE이 있어야 합니다.*NONE 값은 저널링의 시작을 중지하는 데 사용될 수 있습니다.

논리 파일 설정

이 주제에서는 논리 파일 작성에 대한 몇 가지 특수 고려사항에 대해 설명합니다.

논리 파일 설정을 위한 많은 규칙들은 논리 파일의 모든 범주에 적용됩니다. 이 주제에서는 논리 파일의 한 범주에만 적용되는 규칙이 참조할 범주를 식별합니다. 논리 파일의 모든 범주에 적용되는 규칙은 논리 파일의 특정한 범주를 식별하지 않습니다.

논리 파일 작성

이 주제는 자료 서술 스펙(DDS)을 사용하여 논리 파일을 작성하는 방법을 표시합니다.

논리 파일을 작성하기 전에 그 논리 파일의 기초가 되는 실제 파일이 존재해야 합니다.

논리 파일을 작성하려면 다음 단계를 수행하십시오.

1. 논리 파일에 대한 DDS를 소스 파일에 입력하십시오. 이 작업은 소스 입력 유틸리티(SEU) 또는 다른 방법을 사용하여 수행될 수 있습니다. 다음 예는 논리 파일 ORDHDRL(주문 헤더 파일)에 대한 DDS를 표시합니다.

   ```
   |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
             A* ORDER HEADER LOGICAL FILE (ORDHDRL)
             A          R ORDHDR                    PFILE(ORDHDRP)
             A          K ORDER
   ```

   이 파일은 액세스 경로로 키 필드 *Order*(주문 번호)를 사용합니다. 레코드 형식은 연관된 실제 파일 ORDHDRP와 동일합니다. 논리 파일의 레코드 형식명은 필드 설명이 정의되지 않았기 때문에 실제 파일의 레코드 형식명과 동일해야 합니다.

2. 논리 파일을 작성하십시오. CRTLF(논리 파일 작성) 명령을 사용할 수 있습니다. 다음 예는 CRTLF 명령 입력 방법을 표시합니다.

   ```
   CRTLF   FILE(DSTPRODLB/ORDHDRL)
                 TEXT('Order header logical file')
   ```

위에서 표시된 것처럼 이 명령은 몇 가지 디폴트 값을 사용합니다. 예를 들어, SRCFILE 매개변수와 SRCMBR 매개변수가 지정되지 않았으므로 시스템은 IBM 제공 소스 파일 QDDSSRC에 있는 DDS를 사용하고 소스 파일 멤버명은 ORDHDRL입니다(CRTLF 명령에서 지정된 파일명과 같음). 같은 이름을 가진 하나의 멤버로 구성된 파일 ORDHDRL이 라이브러리 DSTPRODLB에 위치합니다.

하나의 실제 파일에서 여러 개의 논리 파일을 작성할 수 있습니다. 하나의 실제 파일에서 작성될 수 있는 논리 파일의 최대 수는 32K입니다.

논리 파일과 유사한 것이 뷰입니다. 뷰는 iSeries Navigator 또는 CREATE VIEW SQL문으로 작성할 수 있습니다.

관련 개념

276 페이지의『소스 파일에 대한 작업』
이 주제에서는 다양한 방법을 사용하여 자료를 입력하고 유지보수하는 방법을 설명합니다.

227 페이지의『복수 형식 파일에 추가할 레코드 형식 식별』
어플리케이션이 데이터베이스에 추가될 레코드에 대해 레코드 형식명 대신 파일명을 사용하는 경우 또한 사용된 파일이 둘 이상의 레코드 형식을 가진 논리 파일인 경우 데이터베이스의 레코드 위치를 결정하기 위해 형식 선택 프로그램을 작성해야 합니다.

보기 작성 및 사용

관련 참조

논리 파일 작성(CRTLF) 명령

CREATE VIEW

둘 이상의 레코드 형식이 있는 논리 파일 작성:

복수 형식의 논리 파일은 한 개의 논리 파일로 두 개 이상의 실제 파일에서 관련된 레코드를 사용하도록 합니다. 각 레코드 형식은 항상 한 개 이상의 실제 파일과 연관되어 있습니다. 동일한 실제 파일이 두 개 이상의 레코드 형식으로 사용될 수 있습니다.

다음 예는 필드 참조 파일에서 작성된 실제 파일 ORDDTLP에 대한 자료 서술 스펙(DDS)을 표시합니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* ORDER DETAIL FILE (ORDDTLP) - PHYSICAL FILE RECORD DEFINITION
          A                                     REF(DSTREF)
          A           R ORDDTL                  TEXT('Order detail record')
          A             CUST      R
          A             ORDER     R
          A             LINE      R
          A             ITEM      R
          A             QTYORD    R
          A             DESCRP    R
          A             PRICE     R
          A             EXTENS    R
          A             WHSLOC    R
          A             ORDATE    R
          A             CUTYPE    R
          A             STATE     R
          A             ACTMTH    R
          A             ACTYR     R
      A
```

다음은 실제 파일 ORDHDRP에 대한 DDS입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* ORDER HEADER FILE (ORDHDRP) - PHYSICAL FILE RECORD DEFINITION
          A                                     REF(DSTREFP)
          A           R ORDHDR                  TEXT('Order header record')
          A             CUST      R
          A             ORDER     R
          A             ORDATE    R
          A             CUSORD    R
          A             SHPVIA    R
          A             ORDSTS    R
          A             OPRNME    R
          A             ORDMNT    R
          A             CUTYPE    R
          A             INVNBR    R
          A             PRTDAT    R
          A             SEQNBR    R
          A             OPNSTS    R
          A             LINES     R
```

```
          A            ACTMTH    R
          A            ACTYR     R
          A            STATE     R
     A
```

다음의 예는 두 개의 레코드 형식을 가진 논리 파일 ORDFILL의 작성 방법입니다. 한 레코드 형식은 주문 헤더 레코드를 위해 실제 파일 ORDHDRP를 정의하고, 다른 레코드 형식은 주문 세부사항 레코드로 실제 파일 ORDDTLP를 정의합니다.

논리 파일 레코드 형식 ORDHDR은 하나의 키 필드, *Order*를 사용하여 순번 처리합니다. 논리 파일 레코드 형식 ORDDTL은 두 개의 키 필드 *Order* 및 *Line*을 사용하여 순번을 처리합니다.

다음은 논리 파일 ORDFILL에 대한 DDS입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* ORDER TRANSACTION LOGICAL FILE (ORDFILL)
          A          R ORDHDR                    PFILE(ORDHDRP)
          A          K ORDER
     A
          A          R ORDDTL                    PFILE(ORDDTLP)
          A          K ORDER
          A          K LINE
     A
```

연관된 두 개의 실제 파일로 논리 파일 ORDFILL을 작성하려면 다음과 같이 CRTLF(논리 파일 작성) 명령을 사용하십시오.

```
CRTLF  FILE(DSTPRODLB/ORDFILL)
              TEXT('Order transaction logical file')
```

DDS 소스는 QDDSSRC 파일의 멤버 ORDFILL에 들어 있습니다. 같은 이름을 가진 하나의 멤버로 구성된 파일 ORDFILL은 DSTPRODLB 라이브러리에 위치합니다. 논리 파일 멤버 ORDFILL의 액세스 경로가 ORDHDRP 파일과 ORDDTLP 파일로부터 레코드를 배열합니다. 두 실제 파일의 레코드 형식에는 공통 필드인 *Order*가 키로 지정되어 있습니다. 논리 파일 설명에서 정의된 순서 때문에 첫 번째로 검색된 헤더 파일 ORDHDRP와 두 번째로 검색된 상세 파일 ORDDTLP 사이의 중복이 *Order* 순서로 병합됩니다FIFO, LIFO 또는 FCFO가 지정되지 않았으므로 한 파일 내의 중복 키 검색 순서는 보장되지 않습니다.

주: 어떤 환경에서는 복수 형식 논리 파일보다 복수 논리 파일을 사용하는 것이 더 좋습니다. 예를 들어 복수 형식 논리 파일과 함께 키순 액세스 경로를 사용하는 경우 파일 중 하나에 레코드가 거의 들어 있지 않으면 성능이 아주 저하될 수 있습니다. 복수 형식일지라도 논리 파일은 각각의 실제 파일에서 나온 하나의 색인만 가집니다. 어플리케이션 프로그램의 처리 종류(예를 들면, 작은 파일을 처리하기 위해 키로 RPG SETLL 및 READE 사용)에 따라 시스템은 작은 파일에서 항목을 찾기 위해 모든 색인 항목들을 탐색해야 하는 경우도 있습니다. 색인에 항목이 많으면 각 파일의 키 수와 색인 내의 키 순서에 따라 색인 탐색에 장시간이 소요될 수도 있습니다. (작은 파일에 레코드가 없으면 시스템이 빠른 경로를 택해 색인 탐색을 피할 수 있으므로 성능에는 영향을 미치지 않습니다.)

복수 형식 파일에서 레코드 검색 방법 제어:

두 개 이상의 레코드 형식을 가진 논리 파일에서는 키 필드 정의가 필요합니다. 각각의 레코드 형식은 고유 키를 가지고 있으며 서로 다른 형식의 레코드를 병합하기 위해 레코드 형식 키 필드를 정의할 수 있습니다. 각 레코드 형식에 키로 지정된 모든 키 필드가 있을 필요는 없습니다.

다음 레코드를 참조하십시오.

표 4. 헤더 레코드 형식

레코드	Order	Cust	Ordate
1	41882	41394	050688
2	32133	28674	060288

표 5. 상세 레코드 형식

레코드	Order	Line	Item	Qtyord	Extens
A	32133	01	46412	25	125000
B	32133	03	12481	4	001000
C	41882	02	46412	10	050000
D	32133	02	14201	110	454500
E	41882	01	08265	40	008000

자료 서술 스펙(DDS)에서 헤더 레코드 형식은 상세 레코드 형식보다 먼저 정의합니다. 액세스 경로로 *Order* 필드를 두 개의 레코드 형식에 첫 번째 키 필드로 사용하고, *Line* 필드를 두 번째 키 필드로 두 번째 레코드 형식에서만 사용하는 경우(모두 오름차순) 액세스 경로의 레코드 순서는 다음과 같습니다.

- 레코드 2
- 레코드 A
- 레코드 D
- 레코드 B
- 레코드 1
- 레코드 E
- 레코드 C

주: 중복 키 값을 가진 레코드는 지정된 실제 파일의 순서대로 배열됩니다. 그런 후, 레코드 형식 내에 여전히 중복이 존재할 경우 중복 레코드는 FIFO, LIFO 또는 FCFO 키워드에서 지정한 순서대로 배열됩니다. 예를 들어 논리 파일에 DDS 키워드 FIFO를 지정한 경우 형식 내의 중복 레코드는 FIFO(선입선출) 순서로 배열됩니다.

하나 이상의 레코드 형식을 가진 논리 파일의 경우 키 필드에 *NONE DDS 기능을 사용하여 한 레코드 형식을 동일한 액세스 경로상의 다른 레코드 형식과 분리할 수 있습니다. 일반적으로 모든 레코드 형식의 레코드는 키 값을 기준으로 병합됩니다. 그러나 DDS에서 키 필드에 *NONE이 지정되면 *NONE 이전의 모든 레코드 형식에서 키 필드를 가진 레코드들만 병합됩니다. 그러한 레코드를 하나 이상의 레코드 형식으로부터 나온 키로 검색할 때는 *NONE 앞의 모든 레코드 형식에 나오는 키 필드만 사용됩니다. 사용되는 키 필드의 수를 증가시키려면 고려해야 할 레코드 형식의 수를 제한하십시오.

다음 예는 논리 파일에 3개의 레코드 형식이 들어 있으며 각 레코드 형식은 서로 다른 실제 파일과 연관되어 있음을 나타냅니다.

레코드 형식	실제 파일(PF)	키 필드
EMPMSTR	EMPMSTR	Empnbr(사원 번호) **1**
EMPHIST	EMPHIST	Empnbr, Empdat(고용일) **2**
EMPEDUC	EMPEDUC	Empnbr, Clsnbr(등급 번호) **3**

주: 모든 레코드 형식은 공통되는 키 필드를 하나만 가집니다(*Empnbr*).

위의 예에 대한 DDS는 다음과 같습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A
            A          K EMPNBR  1
      A
            A          K EMPNBR  2
            A          K EMPDAT
      A
            A          K EMPNBR  3
            A          K *NONE
            A          K CLSNBR
      A
```

EMPMSTR의 두 번째 키와 세 번째 키 필드에 그리고 EMPHIST의 세 번째 키 필드에 *NONE이 가정됩니다. 그 이유는 이 키 필드 위치 다음에 키 필드가 없기 때문입니다.

다음은 레코드 배열을 나타내는 표입니다.

Empnbr	Empdat	Clsnbr	레코드 형식명
426			EMPMSTR
426	6/15/74		EMPHIST
426		412	EMPEDUC
426		520	EMPEDUC
427			EMPMSTR
427	9/30/75		EMPHIST
427		412	EMPEDUC

*NONE은 레코드 형식 EMPHIST와 EMPEDUC에 대해 분리자 역할을 합니다. EMPHIST에서 동일한 *Empnbr* 필드를 가진 모든 레코드는 함께 그룹화되며 *Empdat* 필드에 의해 분류됩니다. EMPEDUC에서 동일한 *Empnbr* 필드를 가진 모든 레코드는 함께 그룹화되며 *Clsnbr*필드에 의해 분류됩니다.

주: 위의 순서를 보장하기 위해 키순 액세스 경로에 키 필드 값이 추가로 들어갔기 때문에 중복 키 값은 예측할 수 없습니다.

관련 개념

DDS 개념

복수 형식 파일에 레코드 추가 방법 제어:

복수 형식 논리 파일에 레코드를 추가하려면 레코드를 기록할 기준 실제 파일의 멤버를 확인해야 합니다.

사용 중인 어플리케이션이 하나의 형식 내에 특정 멤버를 지정할 수 없지만 논리 파일 내의 각 형식은 하나의 실제 파일 멤버와 연관되어야 합니다. 하나 이상의 기준 실제 파일에 둘 이상의 멤버가 포함될 경우 『논리 파일 멤버 정의』에 서술된 DTAMBRS 매개변수를 사용하여 한 멤버를 각 형식과 연결해야 합니다. 마지막으로, 복수 형식 논리 파일에 있는 각 형식에 고유명을 지정하십시오. 이러한 방법으로 복수 형식 논리 파일이 정의되면 추가로 형식명을 지정할 때는 레코드가 추가될 특정 실제 파일 멤버를 지정하십시오.

복수 형식 논리 파일에 레코드를 추가하고 어플리케이션 프로그램이 레코드 형식명 대신 파일명을 사용하는 경우 사용자는 형식 선택 프로그램을 작성해야 합니다.

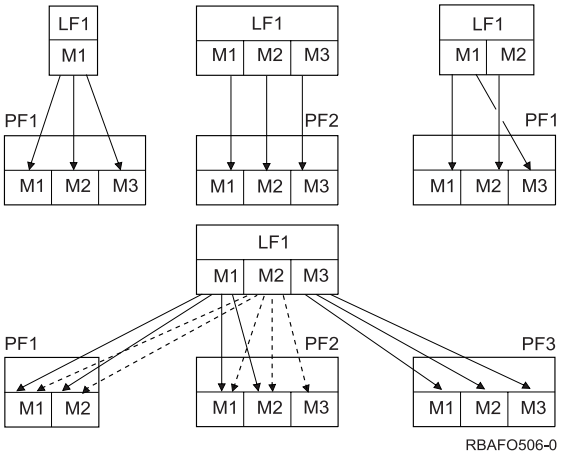
관련 개념

227 페이지의 『복수 형식 파일에 추가할 레코드 형식 식별』
어플리케이션이 데이터베이스에 추가될 레코드에 대해 레코드 형식명 대신 파일명을 사용하는 경우 또한 사용된 파일이 둘 이상의 레코드 형식을 가진 논리 파일인 경우 데이터베이스의 레코드 위치를 결정하기 위해 형식 선택 프로그램을 작성해야 합니다.

논리 파일 멤버 정의:

자료를 논리 그룹별로 분리하기 위해서 논리 파일에 멤버를 정의할 수 있습니다. 논리 파일 멤버는 한 개 또는 여러 개의 실제 파일 멤버와 연관될 수 있습니다.

다음 도표는 이러한 개념을 설명합니다.



논리 파일 내의 모든 논리 멤버에 사용되는 레코드 형식은 파일 작성 시 자료 서술 스펙(DDS)에 정의되어야 합니다. 새로운 레코드 형식이 필요하면 또 다른 논리 파일이나 레코드 형식이 작성되어야 합니다.

액세스 경로 속성은 논리 파일 작성 시 DDS 및 명령에 지정된 정보에 따라 결정됩니다. 자료 멤버의 선택은 CRTLF(논리 파일 작성) 및 ADDLFM(논리 파일 멤버 추가) 명령의 DTAMBRS 매개변수에 지정됩니다.

논리 파일이 정의될 때 그 논리 파일이 사용하는 실제 파일은 레코드 레벨 PFILE이나 JFILE 키워드로 DDS에 지정됩니다. 복수 레코드 형식이 DDS에 정의되는 경우 각 레코드 형식에 PFILE 키워드가 지정되어야 합니다. 각각의 PFILE 키워드에 한 개 이상의 실제 파일을 지정할 수 있습니다.

논리 파일이 작성되거나 멤버가 파일에 추가되면 CRTLF 또는 ADDLFM 명령의 DTAMBRS 매개변수를 사용하여 논리 파일에 사용되는 실제 파일의 어떤 멤버를 자료에 사용할 것인지 지정할 수 있습니다. 실제 파일의 어떤 멤버도 자료로 사용하지 않을 경우 실제 파일의 멤버명으로 *NONE을 지정하면 됩니다.

다음 예에서는 논리 파일이 정의된 두 개의 레코드 형식을 가지고 있습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
00010A          R LOGRCD2                   PFILE(PF1 PF2)
          A            .
          A            .
          A            .
00020A          R LOGRCD3                   PFILE(PF1 PF2 PF3)
          A            .
          A            .
          A            .
     A
```

다음과 같이 CRTLF 또는 ADDLFM 명령에 DTAMBRS 매개변수가 지정된 경우:

```
DTAMBRS((PF1 M1) (PF2 (M1 M2)) (PF1 M1) (PF2 (*NONE)) (PF3 M3))
```

레코드 형식 LOGRCD2는 PF1의 실제 파일 멤버 M1 및 파일 PF2의 M1 및 M2와 연관됩니다. PF2의 어떤 멤버도 LOGRCD3과는 연관되지 않습니다. 두 개 이상의 PFILE 키워드에 동일한 실제 파일명이 지정되면 실제 파일명이 발생될 때마다 서로 다른 실제 파일로서 처리됩니다. 두 개 이상의 PFILE 키워드에 동일한 실제 파일명이 지정되면 실제 파일명이 발생될 때마다 서로 다른 실제 파일로서 처리됩니다.

PFILE 키워드의 파일에 라이브러리명이 지정되지 않으면 논리 파일 작성 시 실제 파일을 찾는 데 라이브러리 리스트가 사용됩니다. 그리고 실제 파일명과 라이브러리명은 논리 파일 설명의 일부가 됩니다. DTAMBRS 매개변수에 지정된 실제 파일명과 라이브러리명은 논리 파일 설명에 저장된 이름과 동일해야 합니다.

DTAMBRS 매개변수에서 파일명에 라이브러리명이 규정화되지 않으면 라이브러리명은 디폴트 *CURRENT로 설정되고, 시스템은 각각의 실제 파일명으로 논리 파일 설명에 지정된 라이브러리명을 사용합니다. 이 라이브러리명은 PFILE DDS 키워드에서 파일에 대해 지정된 라이브러리명이거나 논리 파일 작성 시 사용한 라이브러리 리스트 내의 라이브러리명입니다.

논리 파일에 멤버를 추가할 때에는 다음과 같이 자료 멤버를 지정할 수 있습니다.

- 연관된 실제 파일 멤버를 지정하지 않습니다(디폴트 DTAMBRS *ALL). 논리 파일 멤버는 논리 파일 DDS에서 지정된 PFILE 키워드의 모든 실제 파일의 모든 실제 파일 멤버와 연관됩니다.
- 연관된 실제 파일 멤버를 지정합니다(DTAMBRS 매개변수). 라이브러리명을 지정하지 않으면 논리 파일이 라이브러리를 결정합니다. 실제 파일에 두 개 이상의 실제 파일 멤버가 지정되는 경우 이들 멤버에 중복 키 값이 발생할 때 레코드가 검색될 순서대로 멤버명을 지정해야 합니다. 특정한 실제 파일의 멤버를 포함하지 않을 경우 실제 파일명을 지정하지 않거나 실제 파일명을 지정하고 멤버명에 *NONE을 지정하십시오. 이 방법은 논리 파일에 정의된 레코드 형식의 일부분인 논리 파일 멤버를 정의하는 데 사용할 수 있습니다.

논리 파일 작성시 첫 번째 멤버를 작성하기 위해 CRTLF(논리 파일 작성) 명령을 사용할 수 있습니다. 후속 멤버들은 ADDLFM(논리 파일 멤버 추가) 명령을 사용하여 추가해야 합니다. 그러나 여러 멤버를 추가하려면 CRTLF 명령어의 MAXMBRS 매개변수에 1보다 큰 숫자를 지정해야 합니다. 논리 파일에 멤버를 추가하는 다음 예는 CRTLF 명령을 사용합니다.

```
CRTLF   FILE(DSTPRODLB/ORDHDRL)
               MBR(*FILE)  DTAMBRS(*ALL)
               TEXT('Order header logical file')
```

*FILE은 MBR 매개변수의 디폴트이며, 멤버명이 파일명과 동일하다는 의미입니다. 연관된 실제 파일(ORDHDRP)의 모든 멤버는 논리 파일(ORDHDRL) 멤버에서 사용됩니다. 텍스트 설명은 멤버의 텍스트 설명입니다.

논리 파일 레코드 형식 설명

자료 서술 스펙(DDS)으로 서술되는 각 논리 파일 레코드 형식의 경우 레코드 형식명과 PFILE 키워드(단순 및 복수 형식 논리 파일의 경우) 또는 JFILE 키워드(결합 논리 파일의 경우)를 지정해야 합니다.

PFILE이나 JFILE 키워드에서 지정된 파일명은 논리 파일의 기초가 되는 실제 파일입니다. 단순 또는 복수 형식의 논리 파일 레코드 형식은 다음 중의 한 가지 방법으로 DDS를 작성합니다.

- 단순 논리 파일 레코드 형식에서는 레코드 형식명과 PFILE 키워드만 지정하십시오. PFILE 키워드에서 지정된 유일한(또는 첫 번째) 실제 파일의 레코드 형식은 논리 파일의 레코드 형식입니다. 논리 파일에 지정된 레코드 형식명은 유일한(또는 첫 번째) 실제 파일에서의 레코드 형식과 동일해야 합니다. 아래와 같은 단순 논리 파일의 예를 고려하십시오.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
          A          R ORDDTL                    PFILE(ORDDTLP)
     A
```

- 포함시키려는 필드명을 열거하여 사용자의 레코드 형식을 서술합니다. 다른 순서로 필드명을 지정하고, RENAME 키워드를 사용하여 필드를 재명명하고, CONCAT 키워드를 사용하여 필드를 결합하고, SST 키워드를 사용하여 필드의 지정 위치를 사용할 수 있습니다. 또한 논리 파일에 다른 속성을 지정하여 필드의 속성을 대체할 수도 있습니다. 아래와 같은 단순 논리 파일의 예를 고려하십시오.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
          A          R ORDHDR                    PFILE(ORDHDRP)
          A            ORDER
          A            CUST
          A            SHPVIA
     A
```

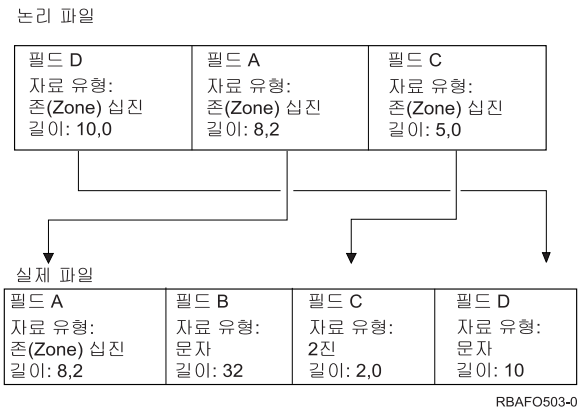
- FORMAT 키워드의 파일명에 대해 데이터베이스 파일명을 지정하십시오. 서술중인 논리 파일이 이 데이터베이스 파일의 레코드 형식을 공유됩니다. 파일명은 라이브러리명에 의해 규정화될 수 있습니다. 라이브러리명이 지정되지 않은 경우 실제 파일을 찾기 위해 라이브러리 리스트가 사용됩니다. 서술 중인 파일이 작성될 때 파일은 반드시 존재해야 합니다. 그 외에도, 논리 파일에 지정한 레코드 형식명은 FORMAT 키워드에 지정한 파일에 들어 있는 형식명 중의 하나와 동일해야 합니다. 다음 예를 고려하십시오.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
          A          R CUSRCD                    PFILE(CUSMSTP)
          A                                      FORMAT(CUSMSTL)
     A
```

아래 예에서 프로그램은 다음을 필요로 합니다.

- 다른 순서로 놓여진 필드
- 실제 파일에서 온 필드의 서브세트
- 일부 필드에 대해 변경된 자료 유형
- 일부 필드에 대해 변경된 필드 길이

이러한 변경을 위해 논리 파일을 사용할 수 있습니다.



논리 파일에 대한 DDS는 다음과 같습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
          A          R LOGREC                    PFILE(PF1)
          A            D             10S 0
          A            A
          A            C              5S 0
     A
```

실제 파일에 대한 DDS는 다음과 같습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
          A          R PHYREC
          A            A              8S 2
          A            B             32
          A            C              2B 0
          A            D             10
     A
```

논리 파일로부터 레코드가 읽히면, 실제 파일의 필드는 논리 파일 설명에 일치하도록 변경됩니다. 프로그램이 레코드를 갱신하거나 추가하는 경우 필드는 원래의 상태로 다시 변경됩니다. 논리 파일을 사용한 추가 및 갱신 조작의 경우 프로그램은 논리 파일에서 사용된 형식과 일치하는 자료를 제공해야 합니다.

다음 도표에서는 실제 파일과 논리 파일 사이에서 유효한 유형의 자료 맵핑을 보여주고 있습니다.

실제 파일 자료 유형	논리 파일 자료 유형							
	문자 또는 16진	**존**	**팩**	**2진**	**부동 소수점**	**날짜**	**시간**	**시간소인**
문자 또는 16진	유효함	주 1 참조	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
존	주 1 참조	유효함	유효함	주 2 참조	유효함	유효하지 않음	유효하지 않음	유효하지 않음
팩	유효하지 않음	유효함	유효함	주 2 참조	유효함	유효하지 않음	유효하지 않음	유효하지 않음
2진	유효하지 않음	주 2 참조	주 2 참조	주 3 참조	주 2 참조	유효하지 않음	유효하지 않음	유효하지 않음
부동 소수점	유효하지 않음	유효함	유효함	주 2 참조	유효함	유효하지 않음	유효하지 않음	유효하지 않음
날짜	유효하지 않음	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음	유효하지 않음
시간	유효하지 않음	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음
시간소인	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효함

주:

1. 문자 또는 바이트 수가 자릿수와 같은 경우에만 유효함.
2. 2진 필드의 소수 자릿수가 0인 경우에만 유효함.
3. 2진 필드 모두가 소수 자릿수를 가진 경우에만 유효함.

관련 개념

337 페이지의『2바이트 문자 세트 고려사항』

이 주제에서는 iSeries 시스템에 데이터베이스를 적용할 경우 2바이트 문자 세트(DBCS) 고려사항을 설명합니다.

논리 파일의 필드 사용 서술:

논리 파일의 필드를 입력 전용, 입/출력용 또는 비입/출력용으로 지정할 수 있습니다.

38열에 다음 항목 중 하나를 지정하면 됩니다.

항목 의미

공백 단순 또는 복수 형식 논리 파일의 경우 디폴트는 B(입/출력용)이고, 결합 논리 파일에 대해 디폴트는 I(입력 전용).

B 입/출력이 모두 허용되지만 결합 논리 파일에는 유효하지 않음.

I 입력 전용(읽기 전용).

N 입/출력 모두 허용되지 않지만 결합 논리 파일에만 유효함.

주: 사용 값(38열)은 참조 기능에서 사용되지 않습니다. 다른 파일이 논리 파일의 필드를 참조할 경우(REF 또는 REFFLD 키워드를 사용하여), 사용 값은 그 파일에 복사되지 않습니다.

논리 파일의 필드 사용 설명: 둘다:

입출력용 필드는 입력 및 출력 조작에 모두 사용할 수 있습니다. 프로그램은 필드로부터 자료를 읽고 자료를 그 필드에 기록할 수 있습니다. 결합 논리 파일은 읽기 전용 파일이므로 입/출력용 필드는 결합 논리 파일에는 유효하지 않습니다.

논리 파일의 필드 사용 설명: 입력 전용:

입력 전용 필드는 입력 조작에만 사용할 수 있습니다. 프로그램은 필드에서 자료를 읽을 수는 있으나 파일의 필드를 갱신할 수는 없습니다.

일반적인 입력 전용 필드로는 키 필드(키 필드 값의 변경을 막음으로써 액세스 경로의 유지보수를 줄임), 사용자가 볼 수는 있지만 갱신할 수 없는 중요한 필드(예를 들면, 급여), 변환표(TRNTBL) 키워드나 서브스트링(SST) 키워드가 지정된 필드가 있습니다.

입력 전용으로 지정된 필드가 있는 레코드를 프로그램이 갱신할 경우 입력 전용 필드는 파일에서 변경되지 않습니다. 입력 전용 필드를 가진 레코드를 프로그램이 추가할 경우 입력 전용 필드는 디폴트를 선택하게 됩니다(DFT 키워드).

논리 파일에 필드 사용 설명: 둘 다 아님:

비입출력용 필드는 입력이나 출력으로 사용되지 않습니다. 결합 논리 파일의 경우에만 두 필드 모두 유효하지 않습니다. 비입/출력용 필드는 결합 논리 파일에서 결합 필드로 사용될 수 있으나 프로그램은 비입/출력용 필드를 읽거나 갱신할 수 없습니다.

실제 파일 내의 결합 필드들의 속성이 일치하지 않을 때 비입출력용 필드를 사용하십시오. 이 경우 하나 또는 양쪽의 결합 필드가 다시 정의되어야 합니다. 그러나 이렇게 다시 정의된 필드는 레코드 형식에 포함될 수 없습니다. (어플리케이션 프로그램은 다시 정의된 필드를 사용하지 않습니다.) 따라서 다시 정의된 결합 필드는 N으로 코드되어 레코드 형식에 나오지 않도록 할 수 있습니다.

38열에 N이 지정된 필드는 사용자 프로그램에서 사용하는 버퍼에 나오지 않습니다. 그러나 DSPFFD(파일 필드 설명 표시) 명령을 사용하면 필드 설명이 표시됩니다.

비입출력용 필드는 선택/생략 필드나 키 필드로서 사용될 수 없습니다.

관련 참조

82 페이지의『레코드 형식에 나오지 않는 필드 서술(예 5)』
비입출력용으로 결합 논리 파일에 사용될 수 있는(38열에 N이 지정됨) 두 필드 모두 레코드 형식에 포함될 수 없습니다. 이 예는 레코드 형식에 나오지 않는 필드 서술 방법을 표시합니다.

기존 필드에서 새로운 필드 파생:

논리 파일의 필드는 그 논리 파일의 기초가 되는 실제 파일의 필드나 동일한 논리 파일의 필드로부터 파생될 수 있습니다.

예를 들어 CONCAT 키워드로 실제 파일에 있는 2개 이상의 필드를 연결하여 논리 파일에서는 한 개의 필드처럼 보이도록 할 수 있습니다. 마찬가지로 SST 키워드로 실제 파일의 한 필드를 여러 개로 나누어 논리 파일에서는 여러 개의 필드처럼 보이도록 할 수 있습니다.

연결 필드:

CONCAT 키워드를 사용하면 실제 파일 레코드 형식에 있는 두 개 이상의 필드를 결합하여 논리 파일 레코드 형식에 한 개의 필드를 만들 수 있습니다.

예를 들어 실제 파일 레코드 형식에 *Month*, *Day*, *Year*라는 필드들이 있습니다 이 경우 논리 파일에서 이 필드들을 *Date*라는 한 개의 필드로 연결할 수 있습니다.

연결된 결과 필드의 필드 길이는 포함된 필드의 길이를 합친 것입니다. (그러나 실제 파일의 필드가 2진수이거나 팩 십진수인 경우를 제외하고는 이 필드들은 존 십진수로 변경됩니다.) 결과 필드의 필드 길이는 시스템에 의해 자동적으로 계산됩니다. 연결 필드는 다음 정보를 가질 수 있습니다.

- 열 표제
- 유효성 검사
- 텍스트 설명
- 편집 코드 또는 편집 단어(숫자 연결 필드에만 해당)

주: 이러한 편집 및 유효성 검사 정보는 데이터베이스 관리 시스템에 의해 사용되지 않으나 데이터베이스 파일의 필드 설명이 화면 파일이나 프린터 파일에 참조될 때 이들 정보가 검색됩니다.

필드가 연결될 때 그 자료 형태가 변경될 수 있습니다. (결과 자료 형태는 시스템에 의해 자동적으로 결정됩니다.) 이 경우 아래와 같은 규칙과 제한이 적용됩니다.

- 오퍼레이팅 시스템은 연결 중인 필드의 자료 유형에 따라 자료 유형을 할당합니다.
- 연결 필드의 최대 길이는 연결 필드의 자료 형태와 연결 중인 필드의 길이에 따라 다릅니다. 연결 필드가 존 십진(S)이면 총 길이는 31바이트를 초과할 수 없고, 문자(A)이면 총 길이가 32766바이트를 초과할 수 없습니다.
- 결합 논리 파일의 경우 연결될 필드는 동일한 실제 파일에서 온 필드여야 합니다. 사용될 실제 파일은 CONCAT 키워드에 지정된 첫 번째 필드로 식별됩니다. 그러므로 첫 번째 필드는 논리 파일의 기초가 되는 실제 파일들 사이에서 고유해야 하거나 사용될 실제 파일을 지정하기 위해서는 JREF 키워드를 지정해야 합니다.
- 연결 필드가 가변 길이이면 연결 필드의 용도는 입력 전용(I)이어야 합니다. 그렇지 않으면 용도는 B(입출력용)여야 합니다.
- 자료 유형이 O 또는 J인 연결 필드에는 REFSHIFT가 할당될 수 없습니다.
- 임의의 필드에 널(null)이 들어 있으면 연결의 결과는 널입니다.

주: DBCS 필드의 연결에 대해 자세히 알려면 337 페이지의 『2바이트 문자 세트 고려사항』을 참조하십시오.

숫자 필드만 연결될 경우 그룹에서 마지막 필드의 부호가 연결 필드의 부호로 사용됩니다.

주:

1. 0이 아닌 십진 정밀도를 가진 숫자 필드는 연결 필드에 포함될 수 없습니다.
2. 날짜, 시간, 시간소인 및 부동 소수점 필드는 연결 필드에 포함될 수 없습니다.

다음 예는 연결용 자료 서술 스펙(DDS)의 필드 설명을 표시합니다. (CONCAT 키워드는 연결할 필드를 지정하는 데 사용됩니다.)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
00101A            MONTH
00102A            DAY
00103A            YEAR
00104A            DATE                      CONCAT(MONTH DAY YEAR)
     A
```

위의 예에서 논리 파일 레코드 형식은 연결된 *Date* 필드뿐만 아니라 *Month*, *Day* 및 *Year*와 같은 개별 필드도 포함합니다다음 형식을 사용할 수 있습니다.

- *Month*, *Day* 및 *Year*의 개별 필드를 가진 형식
- 연결된 *Date* 필드만 있는 형식
- *Month*, *Day* 및 *Year*의 개별 필드와 연결된 *Date* 필드가 있는 형식

개별 필드와 연결 필드 모두가 형식에 있을 경우 필드에 대한 갱신은 DDS에 지정된 순서에 따라 처리됩니다. 앞의 예에서 *Date* 필드에 103188이 들어 있고 *Month* 필드가 12로 변경될 경우 레코드가 갱신될 때 *Date* 필드의 월(month)을 사용합니다. 갱신된 레코드에는 103188이 들어 있게 됩니다. *Date* 필드가 먼저 지정된 경우라면 갱신된 레코드에는 123188이 들어 있게 됩니다.

연결 필드는 키 필드 및 선택/생략 필드로 사용될 수도 있습니다.

서브스트링 필드:

SST 키워드를 사용하여 서브스트링에 들어갈 필드(문자, 16진 또는 존 십진 필드)를 지정할 수 있습니다. 또한 논리 파일의 자료 유형으로 S(존 십진수)를 지정하여 실제 파일에서 팩 필드가 있는 서브스트링을 사용할 수 있습니다.

예를 들어, 실제 파일 *PF1*에서 *Date* 필드 길이를 6자로 정의했다고 가정하십시오. 그러면 각각 길이가 두 자인 세 개의 필드를 갖는 논리 파일을 서술할 수 있습니다. SST 키워드를 사용하여 *Date* 필드의 위치 1에서부터 시작하는 2문자를 MM으로 지정하고 *Date* 필드의 위치 3에서 시작되는 2문자를 DD로 정의하며, *Date* 필드의 위치 5에서 시작하는 2문자를 YY로 정의할 수 있습니다.

다음 예는 이러한 서브스트링 필드용 자료 서술 스펙(DDS)에 필드 설명을 표시합니다. SST 키워드는 서브스트링에 필드를 지정하기 위해 사용됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1                     PFILE(PF1)
     A
          A            MM                 I      SST(DATE 1 2)
          A            DD                 I      SST(DATE 3 2)
          A            YY                 I      SST(DATE 5 2)
     A
```

주: 서브스트링의 시작 위치는 파일 내의 위치가 아니라 수행 중인 필드 내의 위치에 따라(*Date*) 지정됨에 유의하십시오. 사용 열의 I는 입력 전용을 표시합니다.

서브스트링 필드는 키 필드 및 선택/생략 필드로 사용될 수도 있습니다.

이름 변경 필드:

RENAME 키워드를 사용하여 논리 파일의 필드명을 실제 파일의 필드명과 다르게 지정할 수 있습니다. 프로그램이 다른 필드명을 사용하여 작성되었거나 원래의 필드명이 사용 중인 고급 언어의 명명 규칙을 따르지 않기 때문에 논리 파일의 필드를 재명명해야 할 경우가 있습니다.

변환 필드:

TRNTBL 키워드를 사용하여 필드에 대한 변환표를 지정할 수 있습니다. 논리 파일 레코드를 읽고 변환표가 논리 파일의 한 개 이상의 필드에 대해 지정된 경우 시스템은 실제 파일의 필드 값을 변환표에 의해 결정된 값으로 자료를 변환합니다.

논리 파일에서 부동 소수점 필드 서술:

논리 필드에서 부동 소수점 필드를 맵핑된 필드로 사용할 수 있습니다. 단정밀도 또는 배정밀도의 부동 소수점 필드는 존, 팩, 0 정밀도 2진 또는 다른 부동 소수점 필드에 또는 필드로부터 맵핑될 수 있습니다. 부동 소수점 필드는 0이 아닌 정밀도 2진 필드, 문자 필드, 16진 필드 또는 2바이트 문자 세트(DBCS) 필드와는 맵핑될 수 없습니다.

단정밀도나 배정밀도와 같은 서로 다른 정밀도를 가진 부동 소수점 필드간의 맵핑이나 부동 소수점 필드와 다른 숫자 필드간의 맵핑은 정밀도의 손실 또는 반올림을 초래할 수 있습니다. 배정밀도 부동 소수점 수를 단정밀도 부동 소수점 수로 맵핑할 때에는 포함된 특정 수나 내부 표시에 따라 반올림을 초래할 수 있습니다. 반올림은 가장 가까운(짝수) 비트로 이루어 집니다. 그 결과는 항상 가능한 한 최대의 정밀도를 갖게 됩니다. 정밀도의 자릿수가 줄어들 경우 두 개의 십진수 사이에서도 정밀도가 유실될 수 있습니다.

프로그램에서 명시적으로 변경하지 않은 필드의 값이 우연하게 변경될 수 있습니다. 부동 소수점 필드의 경우 이러한 일은 실제 파일에 논리 파일의 단정밀도 필드에 맵핑되는 배정밀도 필드가 있고, 논리 파일을 통해 레코드를 갱신할 경우에 발생할 수 있습니다. 부동 소수점 수의 내부 표시로 인해 이 수가 논리 파일로 맵핑될 때 반올림이 발생하면 논리 레코드 갱신 시 실제 파일 내에서 정밀도가 영구적으로 손실됩니다. 반올림된 숫자가 실제 레코드의 키이면 실제 파일의 레코드 순서가 변경될 수도 있습니다.

정밀도가 논리 파일에서 감소되면 고정 소수점 숫자 필드도 우연히 갱신될 수 있습니다.

논리 파일에 대한 액세스 경로 서술

여러 가지 방법 중 하나를 사용하여 논리 파일에 대한 액세스 경로를 서술합니다.

논리 파일 레코드 형식에 대한 액세스 경로는 다음 중 한 가지 방법으로 지정할 수 있습니다.

- 키순 액세스 경로 스펙. 마지막 레코드 또는 필드 레벨 스펙 다음에 키 필드를 지정합니다. 키 필드명은 레코드 형식 내에 있어야 합니다. 결합 논리 파일의 경우 키 필드는 첫 번째 실제 파일 또는 1차 실제 파일에서 온 것이어야 합니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R CUSRCD                    PFILE(CUSMSTP)
          A          K ARBAL
          A          K CRDLMT
     A
```

- 인코드된 벡터 액세스 경로 스펙. SQL CREATE INDEX 명령문을 사용하여 인코드된 벡터 액세스 경로를 정의합니다.
- 도달순 액세스 경로 스펙. 키 필드를 지정하지 않습니다. PFILE 키워드에 하나의 실제 파일(논리 파일 멤버를 추가할 때는 단 하나의 실제 파일 멤버)만을 지정할 수 있습니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R CUSRCD                    PFILE(CUSMSTP)
```

- 이전에 정의된 키순 액세스 경로 스펙(단순 및 복수 형식 논리 파일의 경우에만 해당). 이 논리 파일에 해당 액세스 경로 및 선택/생략 스펙을 복사시킬 데이터베이스 파일을 지정하기 위해서 파일 레벨에 REFACCPTH 키워드를 사용하십시오. REFACCPTH 키워드와 함께 개별적인 키나 선택/생략 필드를 지정할 수는 없습니다.

  주: 지정된 파일의 액세스 경로 스펙이 사용되더라도, 시스템은 어떤 파일의 액세스 경로가 실제로 공유될 것인지 판정합니다. 시스템은 REFACCPTH 키워드의 사용에 관계없이 항상 액세스 경로를 공유하려 합니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R CUSRCD                    PFILE(CUSMSTP)
```

다른 파일의 액세스 경로에 대한 키 필드 스펙을 공유하는 논리 파일에 대해 레코드 형식을 정의할 경우(DDS 키워드 REFACCPTH를 사용하여), 연관된 실제 파일 레코드 형식으로부터 모든 필드를 사용할 수 있습니다. 이들 필드는 액세스 경로를 서술하는 파일에서 사용해서는 안됩니다. 그러나 액세스 경로를 서술하는 파일에서 사용되는 모든 키와 선택/생략 필드는 새로운 레코드 형식에서 사용되어야 합니다.

관련 참조

CREATE INDEX

논리 파일을 사용하여 레코드 선택 및 생략:

시스템은 논리 파일 사용 시 레코드를 선택하고 생략할 수 있습니다. 이로 인해 처리상의 편의나 보안을 위해 파일 내의 레코드를 제외시킬 수 있습니다.

레코드 선택 및 생략 과정은 논리 파일에 대한 DDS 양식의 17열에서 식별되는 비교에 의해 이루어지며, 고급 언어 프로그램에서 코딩된 일련의 비교와 유사합니다. 예를 들어 주문 상세 레코드가 있는 논리 파일에서 주문

량이 선적량보다 큰 레코드만을 사용하도록 지정할 수 있습니다. 그 외의 모든 레코드는 액세스 경로에서 생략됩니다. 생략된 레코드는 실제 파일에 그대로 있으나 논리 파일에서는 검색되지 않습니다. 레코드를 실제 파일에 추가하는 경우 모든 레코드가 추가됩니다. 그러나 선택/생략 액세스 경로를 통해서 선택/생략 기준에 맞는 레코드만이 선택됩니다.

DDS에서 선택이나 생략을 지정하려면 DDS 양식의 17열에 S(선택) 또는 O(생략)을 지정합니다. 그런 후 선택 또는 생략 과정에서 사용될 필드를 지정합니다(19열에서 28열 사이). 45열에서 80열 사이에는 비교를 지정합니다.

주: 선택/생략 스펙은 키 스펙 다음에 나옵니다(키가 지정되는 경우).

레코드는 다음과 같은 여러 가지 비교 유형에 의하여 선택 및 생략될 수 있습니다.

- VALUES. 필드의 내용에 100개까지 비교 값을 사용할 수 있습니다. 일치되는 값이 있을 경우 레코드가 선택되거나 생략됩니다. 다음 예에서는 VALUES 키워드에 지정된 값 중 하나가 *Itmnbr* 필드에 있을 경우 레코드가 선택됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          S ITMNBR                    VALUES(301542 306902 382101 422109 +
          A                                      431652 486592 502356 556608 590307)
     A
```

- RANGE. 필드의 내용이 최소값 및 최대값과 비교됩니다. 내용이 하한값 이상이거나 상한값 이하일 경우 레코드가 선택되거나 생략됩니다. 다음 예에서는 *Itmnbr* 필드에서 301000에서 599999 사이의 범위에 속하는 모든 레코드가 선택됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          S ITMNBR                    RANGE(301000 599999)
     A
```

- CMP. 필드의 내용이 다른 필드의 값이나 내용과 비교됩니다. 유효한 비교 코드는 EQ, NE, LT, NL, GT, NG, LE 및 GE입니다. 비교가 충족되면 레코드가 선택되거나 생략됩니다. 다음 예에서는 *Itmnbr* 필드가 599999 이하일 경우 레코드가 선택됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
         A          S ITMNBR                    CMP(LE 599999)
    A
```

CMP, VALUES 또는 RANGE 키워드가 지정된 숫자 필드에 대한 값은 필드에 지정된 소수 자릿수에 기준을 두고 배열되며 필요한 곳은 0으로 채워집니다. 필드에 대해 소수 자릿수가 지정되지 않은 경우 소수점은 값의 가장 오른쪽 자리의 우측에 위치하게 됩니다. 예를 들어 길이 5와 소수 자릿수 2를 가진 숫자 필드인 경우 값 1.2는 001.20으로 해석되고 값 100은 100.00으로 해석됩니다.

레코드 상태는 선택/생략문이 지정되어 있는 순서대로 그 명령문을 수행합니다. 선택이나 생략에 레코드가 해당될 경우 뒤에 나오는 명령문은 무시됩니다.

보통 선택 및 생략 비교는 독립적으로 처리되고 이 비교들은 OR로 연결됩니다. 즉 선택 또는 비교가 충족되면 해당 레코드가 선택되거나 생략됩니다. 조건이 충족되지 않을 경우 시스템은 그 다음 비교로 진행합니다. 비교

를 함께 연결하려면 DDS 양식의 17열을 공백으로 둡니다. 그러면 이러한 방법으로 연결한 모든 비교가 충족되어야 레코드가 선택, 생략됩니다. 즉 이 비교는 AND로 연결됩니다.

비교가 적을수록 작업은 효율적으로 이루어집니다. 따라서 여러 개의 선택/생략 비교가 있을 경우 가장 많은 레코드를 선택하거나 생략하는 비교를 먼저 지정하십시오.

다음 예에서는 선택/생략 기능을 코딩하는 방법을 보여줍니다. 다음 예에서 *Rep* 필드가 JSMITH인 레코드는 거의 없습니다. New York주에 있는 JSMITH라는 판매원에 대한 1988년 이전의 모든 레코드를 선택하기 위해 DDS를 사용하는 방법을 보여주고 있습니다. 효율성 면에서는 다르지만 모두 같은 결과를 나타냅니다. **3**이 가장 효율적입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
            A          S ST                       CMP(EQ 'NY')       1
            A            REP                      CMP(EQ 'JSMITH')
            A            YEAR                     CMP(LT 88)
      A
```

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
            A          O YEAR                     CMP(GE 88)         2
            A          S ST                       CMP(EQ 'NY')
            A            REP                      CMP(EQ 'JSMITH')
      A
```

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
            A          O REP                      CMP(NE 'JSMITH')   3
            A          O ST                       CMP(NE 'NY')
            A          S YEAR                     CMP(LT 88)
      A
```

1 모든 레코드는 생략되거나 선택되기 이전에 선택 필드 *St*, *Rep* 및 *Year*와 비교되어야 합니다.

2 모든 레코드가 *Year* 필드와 비교된 후, 1988 이전의 레코드를 *St* 및 *Rep* 필드와 비교해야 합니다.

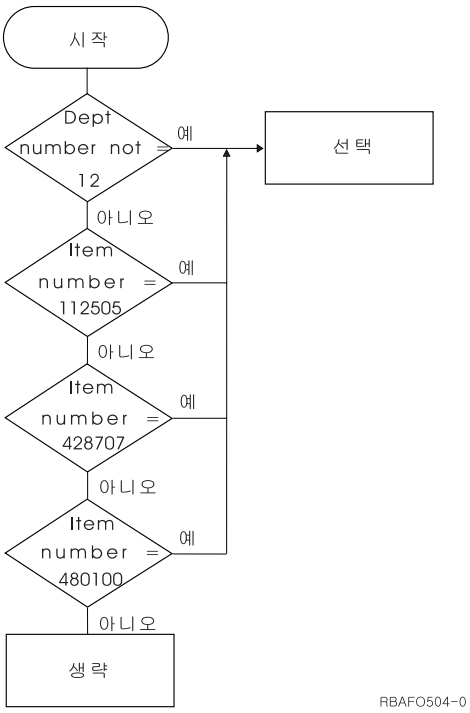
3 모든 레코드가 *Rep* 필드와 비교된 후, JSMITH에 대한 적은 수의 레코드만이 *St* 필드와 비교됩니다. 그 다음, 나머지 적은 수의 레코드가 *Year* 필드와 비교됩니다.

또 다른 예로서 다음 항목을 선택하는 것으로 가정하십시오.

- 부서 12 이외의 다른 부서에 대한 모든 레코드
- 부서 12에서 항목 번호가 112505, 428707 또는 480100인 모든 레코드. 부서 12에 대한 그 외의 레코드는 선택하지 않습니다.

정렬 순서표와 함께 앞의 예를 작성할 경우 선택/생략 필드는 비교 전에 정렬표에 따라 변환됩니다. 예를 들어, 대문자와 소문자에 공용의 가중치를 사용하는 정렬 순서표에서는 NY와 ny가 동일합니다.

다음 다이어그램은 이 예에 대한 논리를 나타냅니다.



이 예는 DDS 선택 및 생략 기능을 사용하여 다음과 같이 코딩됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          S DPTNBR                    CMP(NE 12)
          A          S ITMNBR                    VALUES(112505 428707 480100)
     A
```

선택/생략 값이 있는 액세스 경로를 가지고 파일을 입력순으로 처리할 수 있습니다. 예를 들어 고급 언어 프로그램에서는 키순 액세스 경로가 무시되도록 지정할 수 있습니다. 이 경우 모든 레코드는 파일에서 입력순으로 읽히나 파일에 지정된 선택/생략 값에 부합되는 레코드만 고급 언어 프로그램에 리턴됩니다.

키 필드와 선택/생략 값이 지정된 논리 파일은 입력순으로 처리될 수 있으며 또는 상대 레코드 번호를 임의로 사용하여 처리할 수도 있습니다. 선택/생략 값에 의해서 생략된 레코드는 처리되지 않습니다. 즉 생략된 레코드가 상대 레코드 번호에 의해 요구될 경우 그 레코드는 고급 언어 프로그램으로 리턴되지 않습니다.

논리 파일을 통해 추가나 변경이 이루어진 경우 그 레코드가 같은 논리 파일로 다시 액세스할 수 있도록 시스템이 보장하지는 않습니다. 예를 들어 논리 파일의 선택값에서 *Fld1*이 A인 레코드를 지정하고 프로그램에서는 *Fld1*이 B인 레코드를 갱신할 경우 프로그램이 이 논리 파일을 사용하여 해당 레코드를 다시 검색할 수 없습니다.

주: 부동 소수점 필드 값에 기준을 두고 선택하거나 생략할 수는 없습니다.

선택/생략 조작에는 액세스 경로 선택/생략 및 동적 선택/생략의 두 종류가 있습니다. 디폴트 값은 액세스 경로 선택/생략입니다. 선택/생략 스펙 자체는 양쪽 다 같으나 실제로 시스템은 각기 다른 시점에 레코드의 생략과 선택 작업을 수행합니다.

또한 레코드를 선택하거나 생략할 때 OPNQRF(조회 파일 열기) 명령을 사용할 수도 있습니다.

관련 개념

DDS 개념

액세스 경로 선택/생략:

액세스 경로 선택/생략의 경우 액세스 경로에는 논리 파일에서 지정된 선택/생략 값을 충족하는 키만 들어 있습니다.

파일에 대해 키 필드를 지정하는 경우 액세스 경로는 그 파일에 대해 유지되고 논리 파일이 사용하는 실제 파일에서 레코드를 추가 또는 갱신할 때 시스템에 의해 유지됩니다. 액세스 경로에서 색인 항목만이 선택/생략 값을 충족시키는 항목입니다.

동적 선택/생략:

동적 선택/생략 조작의 경우 프로그램이 파일에서 레코드를 읽으면 시스템은 선택/생략 값을 충족시키는 레코드만 리턴합니다. 즉 선택/생략의 실제적인 처리는 레코드가 추가되거나 변경될 때가 아닌, 레코드가 프로그램에 의해 읽힐 때 수행됩니다.

그러나 키순 액세스 경로는 선택된 레코드의 키뿐만 아니라 모든 키를 포함합니다. 동적 선택/생략을 사용하는 액세스 경로는 보다 많은 액세스 경로를 허용하며 이를 통해 성능을 향상시킬 수 있습니다.

동적 선택/생략을 지정하려면 동적 선택(DYNSLT) 키워드를 사용하십시오. 동적 선택/생략에서는 키 필드가 필요하지 않습니다.

자주 갱신되며 거의 읽히지 않는 파일의 경우 프로그램이 그 파일을 읽기 전까지는 선택/생략 목적으로서 액세스 경로를 갱신할 필요가 없습니다. 이런 경우에는 동적 선택/생략을 선택하는 것이 바람직합니다. 다음 예가 이 설명에 도움이 될 것입니다.

레코드를 선택 또는 생략하기 위해서는 자주 변경되지 않는 코드 필드(A=활동 중, I=비활동 중)를 사용합니다. 프로그램은 활동 레코드를 처리하며 레코드의 대부분(80% 이상)이 활동 중입니다. 이런 경우에는 코드 필드가 변경될 때 액세스 경로를 유지하기보다는 DYNSLT를 사용하여 처리 시점에 동적으로 레코드를 선택하는 것이 보다 효율적일 것입니다.

관련 개념

64 페이지의『기존 액세스 경로 사용』

두 개 이상의 파일이 동일한 실제 파일에 기초하고 동일한 키 필드가 동일한 순서를 가진 경우 이 파일들은 동일한 키순 액세스 경로를 자동적으로 공유하게 됩니다. 액세스 경로 공유 시, 액세스 경로 유지에 필요한 시스템 활동량과 파일에 의해 사용되는 보조 기억장치는 줄어듭니다.

OPNQRYF(조회 파일 열기) 명령으로 레코드 선택 및 생략:

또 다른 레코드 선택 방법으로는 OPNQRYF(조회 파일 열기) 명령에 QRYSLT 매개변수를 사용하는 것입니다. OPNQRYF 명령에 의해 작성된 열린 자료 경로는 임시 논리 파일과 유사합니다. 즉, 닫힐 때 자동으로 삭제됩니다. 반면에 논리 파일은 특별히 삭제하기 전까지는 계속 존재합니다.

관련 개념

139 페이지의 『OPNQRYF(조회 파일 열기) 명령 사용』
OPNQRYF(조회 파일 열기) 명령은 데이터베이스 파일에서 여러 자료 처리 기능을 수행할 수 있도록 하는 제어 언어(CL) 명령입니다. 이 주제에서는 OPNQRYF 명령을 사용한 조회 작성 방법, 주요 기능에 매개변수 지정 방법 및 고급 언어 프로그램에서 이를 사용하는 방법에 대해 설명합니다.

기존 액세스 경로 사용:

두 개 이상의 파일이 동일한 실제 파일에 기초하고 동일한 키 필드가 동일한 순서를 가진 경우 이 파일들은 동일한 키순 액세스 경로를 자동적으로 공유하게 됩니다. 액세스 경로 공유 시, 액세스 경로 유지에 필요한 시스템 활동량과 파일에 의해 사용되는 보조 기억장치는 줄어듭니다.

키순 액세스 경로를 가진 논리 파일이 작성될 때 시스템은 항상 기존 액세스 경로를 공유하려고 합니다. 액세스 경로 공유가 이루어지려면 시스템에 다음 조건을 충족시키는 액세스 경로가 존재해야 합니다.

- 추가될 논리 파일 멤버는 기존 액세스 경로가 있는 실제 파일 멤버에 기초를 두어야 합니다.
- 각 키 필드에 지정된 길이, 자료 유형 및 소수 자릿수는 새로운 파일과 기존 파일에서 같아야 합니다.
- FIFO, LIFO 또는 FCFO 키워드가 지정되지 않을 경우 새로운 파일은 기존 액세스 경로보다 키 필드를 적게 가질 수 있습니다. 즉 새로운 논리 파일은 키의 시작 부분이 같을 경우 기존 액세스 경로를 공유할 수 있습니다. 그러나 파일이 기존 액세스 경로에서 키 세트의 일부를 공유할 경우 공유 액세스 경로에 대한 키 세트의 일부인 필드에서 레코드를 갱신하면 해당 액세스 경로에서 레코드 위치가 변경될 수 있습니다.
- 액세스 경로의 속성(예: UNIQUE, LIFO, FIFO 또는 FCFO)과 키 필드의 속성(예:DESCEND, ABSVAL, UNSIGNED 및 SIGNED)은 같아야 합니다.

  예외:

  1. 액세스 경로에 대한 다른 모든 요구 조건이 충족될 경우 FIFO 액세스 경로는 UNIQUE 키워드가 지정된 액세스 경로를 공유할 수 있습니다.
  2. 액세스 경로 공유에 대한 다른 모든 요구 조건이 충족될 경우 UNIQUE 액세스 경로는 재구성(예를 들어, *REBLD 유지보수가 지정됨)되어야 할 FIFO 액세스 경로를 공유할 수 있습니다.
- 새로운 논리 파일에 선택/생략 스펙이 있는 경우 이는 기존 액세스 경로의 선택/생략 스펙과 동일해야 합니다. 그러나 새로운 논리 파일에 DYNSLT를 지정하면 기존 액세스 경로가 다음 중 하나일 경우 기존 액세스 경로를 공유할 수 있습니다.
  - 동적 선택(DYNSLT) 키워드가 지정됨
  - 선택/생략 키워드가 지정되지 않음
- 새로운 논리 파일 멤버의 대체 배열 순서(ALTSEQ 키워드)와 변환표(TRNTBL 키워드)는 기존 액세스 경로의 대체 배열 순서 및 변환표와 동일해야 합니다.

주: 연결 필드나 서브스트링 필드가 들어 있는 논리 파일은 실제 파일과 액세스 경로를 공유할 수 없습니다.

액세스 경로의 소유자는 그 액세스 경로를 원래 작성한 논리 파일 멤버입니다. 공유 액세스 경로의 경우 그 액세스 경로를 소유하는 논리 멤버가 삭제되면 해당 액세스 경로를 공유하는 첫 번째 멤버가 새로운 소유자가 됩니다. CRTLF(논리 파일 작성) 명령의 FRCACCPTH, MAINT 및 RECOVER 매개변수는 공유될 해당 액

세스 경로에 대한 기존 액세스 경로의 매개변수와 일치하지 않아도 됩니다. 논리 파일 멤버가 액세스 경로를 공유하고, FRCACCPTH, MAINT 및 RECOVER 매개변수가 일치하지 않을 경우 시스템은 공유 멤버에 지정된 각 매개변수의 가장 제한적인 값으로 액세스 경로를 관리합니다. 다음 리스트는 이러한 방법을 설명한 것입니다.

- MBRA는 다음을 지정합니다.

  ```
  FRCACCPTH (*NO)
  MAINT (*IMMED)
  RECOVER (*AFTIPL)
  ```

- MBRB는 다음을 지정합니다.
- ```
  FRCACCPTH (*YES)
  MAINT (*DLY)
  RECOVER (*NO)
  ```

- 시스템은 다음을 수행합니다.

  ```
  FRCACCPTH (*YES)
  MAINT (*IMMED)
  RECOVER (*AFTIPL)
  ```

액세스 경로의 공유는 멤버간의 공유에 제한을 받지 않습니다. 따라서 액세스 경로의 공유는 멤버가 삭제되는 순서를 제한하지 않습니다.

DSPFD(파일 설명 표시) 명령과 DSPDBR(데이터베이스 관계 표시) 명령은 액세스 경로 공유 관계를 보여줍니다.

관련 개념

102 페이지의『중복 키 배열』

자료 서술 스펙(DDS)에 UNIQUE 키워드를 지정하지 않을 경우 시스템에서 중복 키 값이 있는 레코드를 저장하는 방법을 지정할 수 있습니다(중복 키 값이 있을 경우).

예: 내재적으로 공유된 액세스 경로:

이 예는 내재적 공유 액세스 경로를 표시합니다.

두 개의 논리 파일, LFILE1과 LFILE2가 실제 파일 PFILE 위에 구축됩니다. 처음 작성된 LFILE1에는 두 개의 키 필드 KFD1과 KFD2가 있습니다. LFILE2에는 세 개의 키 필드 KFD1, KFD2 그리고 KFD3가 있습니다. 이와 같은 경우 두 논리 파일은 두 개의 동일한 키 필드를 사용하지만 세 개의 키 필드를 가진 논리 파일이 두 개의 키 필드를 가진 파일 다음에 작성되었기 때문에 액세스 경로가 공유되지 않습니다.

표 6. 저장 및 복원 전의 실제 및 논리 파일

	실제 파일(**PFILE**)	논리 파일 **1**(**LFILE1**)	논리 파일 **2**(**LFILE2**)
액세스 경로 필드	KFD1, KFD2, KFD3, A, B, C, D, E, F, G	KFD1, KFD2 KFD1, KFD2, KFD3, F, C, A	KFD1, KFD2, KFD3 KFD1, KFD2, KFD3, D, G, F, E
  ```

어플리케이션이 LFILE1을 사용하여 레코드를 액세스하고 KFD3 필드에 C가 있으면 공백으로 공백이면 C로 변경합니다. 이 어플리케이션은 액세스 경로를 공유하지 않기 때문에 사용자에게 예상치 못한 결과를 발생시키지는 않습니다. 그러나 실제 파일과 두 논리 파일을 저장 및 복원한 후, 프로그램은 아무것도 수행하지 않는 것처럼 보이며 처리하는 데에도 시간이 오래 걸립니다.

복원한 내용을 변경하기 위해 어떤 조치도 취하지 않을 경우 iSeries 시스템은 다음을 수행합니다.

- 우선 가장 많은 키를 가진 논리 파일을 복원합니다.
- 불필요한 액세스 경로를 빌드하지 않습니다.

세 개의 키 필드가 있기 때문에 LFILE2가 가장 먼저 복원됩니다. 복구 후 LFILE1은 LFILE2와 액세스 경로를 공유합니다. 내재적으로 액세스 경로가 공유된다는 것을 이해하지 못하는 사용자는 복구 후 LFILE1을 사용할 때 LFILE2의 키를 사용한다는 것을 알지 못합니다.

표 7. 저장 및 복원 후의 실제 및 논리 파일. 저장 및 복원 전과의 차이점은 동일한 액세스 경로를 공유한다는 것입니다.

	실제 파일(PFILE)	**논리 파일 1(LFILE1)**	**논리 파일 2(LFILE2)**
액세스 경로		KFD1, KFD2, KFD3	KFD1, KFD2, KFD3
필드	KFD1, KFD2, KFD3, A, B, C, D, E, F, G	KFD1, KFD2, KFD3, F, C, A	KFD1, KFD2, KFD3, D, G, F, E

테스트 및 변경된 레코드들의 내용은 다음과 같습니다.

상대 레코드	**KFD1**	**KFD2**	**KFD3**
001	01	01	<공백>
002	01	01	<공백>
003	01	01	<공백>
004	01	01	<공백>

첫 번째 레코드는 첫 번째 키 0101<공백>에 의해 읽히고 0101C로 변경됩니다. 이제 레코드는 다음과 같습니다.

상대 레코드	**KFD1**	**KFD2**	**KFD3**
001	01	01	C
002	01	01	<공백>
003	01	01	<공백>
004	01	01	<공백>

어플리케이션이 다음 키를 읽을 때 0101<공백> 바로 위의 키는 0101C입니다. 이것은 바로 전에 변경된 레코드입니다. 그러나 이때 어플리케이션은 KFD3 필드를 C에서 공백으로 바꿉니다.

사용자가 내재적 공유 액세스 경로를 이해하지 못하기 때문에 어플리케이션은 모든 레코드를 두 번씩 액세스하여 변경하게 됩니다. 결과적으로, 어플리케이션은 처리에 오랜 시간이 소요되며, 레코드들은 변경되지 않은 것처럼 보입니다.

결합 논리 파일 설정

이 주제에서는 결합 논리 파일 설정 방법을 보여 주는 예를 표시합니다. 또한 결합 논리 파일에 관한 성능, 무결성 및 규칙 요약이 포함됩니다.

일반적으로, 이 주제의 예에는 파일의 그림, 파일에 대한 자료 서술 스펙(DDS) 및 샘플 자료가 포함되어 있습니다. 예 1에서는 각기 다른 상황에서(실제 파일의 자료가 다를 경우) 파일을 결합하는 방법을 설명하는 여러 가지 경우를 보여줍니다.

이 예에서는 이해를 돕기 위해 결합 논리 파일은 JLF 레이블로 표시하며, 실제 파일은 PF1, PF2, PF3 레이블 등으로 표시합니다.

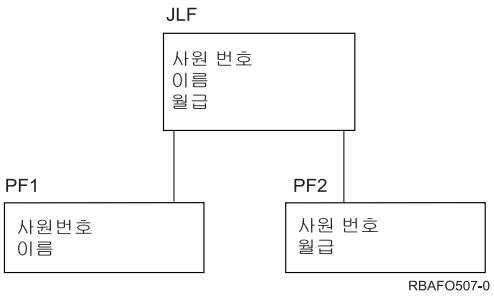
관련 개념

여러 표에 있는 자료 결합

두 개의 실제 파일 결합에 대한 기본 개념(예 1):

*결합 논리 파일*은 둘 이상의 실제 파일의 필드를 결합(한 레코드 형식으로)하는 논리 파일입니다. 이는 레코드 형식에서 모든 필드가 모든 실제 파일에 반드시 존재할 필요가 없기 때문입니다.

다음 예에서는 두 개의 실제 파일을 결합하는 결합 논리 파일을 보여주고 있습니다. 이 예는 예 1에서 다루는 다섯 가지 경우에 사용됩니다.



이 예에서 결합 논리 파일(JLF)에는 *Employee number*, *Name* 및 *Salary* 필드가 있습니다. 실제 파일 1(PF1)에는 *Employee number* 및 *Name*이 있고, 실제 파일 2(PF2)에는 *Employee number* 및 *Salary*가 있습니다. *Employee number*는 실제 파일 모두(PF1 및 PF2)에 공통되지만 *Name*은 PF1에만 있으며, *Salary*는 PF2에만 있습니다.

결합 논리 파일을 사용하면 어플리케이션 프로그램이 한 번의 읽기를 수행하고(결합 논리 파일의 레코드 형식에 대해) 양쪽의 실제 파일로부터 필요한 모든 자료를 얻습니다. 결합 스펙이 없으면, 하나는 PF1, 다른 하나는 PF2를 기초로 하는 두 개의 레코드 형식이 논리 파일에 포함되고, 어플리케이션 프로그램이 두 개의 실제 파일로부터 필요한 자료를 얻기 위해서 두 번의 읽기를 수행해야 합니다. 따라서 결합은 데이터베이스를 설계하는 데 있어 보다 많은 융통성을 제공합니다.

그러나 결합 논리 파일에는 다음과 같은 몇 가지 제한이 있습니다.

- 결합 논리 파일을 통해 실제 파일을 변경할 수 없습니다. 갱신, 삭제 또는 쓰기(추가) 조작을 수행하려면 두 번째의 복수 형식 논리 파일을 작성하여 실제 파일을 변경하는 데 사용해야 합니다. 실제 파일을 직접 사용하여 변경 조작을 수행할 수도 있습니다.
- 자료 파일 유틸리티(DFU)를 사용하여 결합 논리 파일을 표시할 수 없습니다.
- 결합 논리 파일에는 한 개의 레코드 형식만 지정할 수 있습니다.
- 결합 논리 파일의 레코드 형식은 공유될 수 없습니다.
- 결합 논리 파일은 다른 파일의 레코드 형식을 공유할 수 없습니다.
- 키 필드는 결합 레코드 형식에 정의된 필드여야 하며 JFILE 키워드에서 지정된 첫 번째 파일(1차 파일이라고 함)의 필드여야 합니다.
- 선택/생략 필드는 결합 레코드 형식에 정의된 필드여야 하지만 어떤 실제 파일에서 오든지 상관없습니다.
- 확약 제어는 결합 논리 파일과 함께 사용할 수 없습니다.

다음 예는 예 1에 대한 자료 서술 스펙(DDS)을 표시합니다.

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(NBR NBR)
          A            NBR                       JREF(PF1)
          A            NAME
          A            SALARY
          A          K NBR
     A

PF1
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            NBR           10
          A            NAME          20
          A          K NBR
     A

PF2
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            NBR           10
          A            SALARY         7  2
          A          K NBR
     A
```

다음 리스트는 예 1에 있는 결합 논리 파일에 대한 DDS를 설명합니다.

레코드 레벨 스펙은 결합 논리 파일에 사용되는 레코드 형식명을 식별합니다.

R 레코드 형식을 나타냅니다. 결합 논리 파일에는 한 개의 레코드 형식만 가능합니다.

JFILE

단순 및 복수 형식 논리 파일에서 사용되는 PFILE 키워드를 대체합니다. 적어도 두 개의 실제 파일을

지정해야 합니다. JFILE 키워드에 지정된 첫 번째 파일을 1차 파일(Primary File)이라고 합니다. JFILE 키워드에 지정된 기타 파일을 2차 파일(Secondary File)이라고 합니다.

결합 스펙은 실제 파일이 쌍으로 결합되는 방법을 서술합니다. 쌍의 두 번째 파일은 항상 2차 파일이 되며, 2차 파일 각각에 대해서 결합 스펙이 하나씩 있어야 합니다.

J 결합 스펙의 시작을 나타냅니다. 결합 논리 파일에는 적어도 한 개의 결합 스펙을 지정해야 합니다. 결합 스펙은 19열에서 28열 사이에 지정되는 첫 번째 필드명에서 끝나거나 17열에 지정되는 다음 번 J에서 끝납니다.

JOIN 결합 스펙에 의해 결합되는 두 개의 파일을 나타냅니다. 결합 논리 파일에 의해 두 개의 실제 파일만 결합하는 경우 JOIN 키워드는 선택적입니다.

JFLD JOIN 키워드에 지정된 실제 파일의 레코드를 결합하는 결합 필드를 식별합니다. JFLD는 각 결합 스펙에 대하여 적어도 한 번은 지정해야 합니다. 결합 필드는 모든 실제 파일에서 공통되는 필드입니다. 첫 번째 결합 필드는 JOIN 키워드에 지정된 첫 번째 파일의 필드이며, 두 번째 결합 필드는 JOIN 키워드에 지정된 두 번째 파일의 필드입니다.

문자 유형 필드를 제외한 결합 필드는 같은 속성(자료 형태, 길이 및 소수 자릿수)을 가지고 있어야 합니다. 문자 유형 필드이면 길이가 같지 않아도 됩니다. 같은 속성을 갖지 않은 실제 파일 필드를 결합하려는 경우 결합 논리 파일에서 사용하기 위해 그 필드를 다시 정의할 수 있습니다.

필드 레벨 스펙은 결합 논리 파일에 포함되는 필드를 식별합니다.

필드명 어플리케이션 프로그램에서 사용될 필드들을 지정합니다(이 예에서는 *Nbr*, *Name* 및 *Salary*).최소한 한 개의 필드명이 필요합니다. 논리 파일이 사용하는 실제 파일의 어떤 필드명이라도 지정할 수 있습니다. 또한 단순 및 복수 형식 논리 파일에서 처럼 RENAME, CONTACT 또는 SST 등과 같은 키워드를 사용할 수도 있습니다.

JREF 레코드 형식(결합 스펙 레벨 다음에 오며 키 필드 레벨이 있는 경우 앞에 옴)에서는 필드명이 어떤 실제 파일에서 온 것인지를 식별해야 합니다. 이 예에서 *Nbr* 필드는 PF1과 PF2 양쪽에 있습니다. 따라서 JREF 키워드는 *Nbr* 필드 설명이 어떤 파일로부터 온 것인지를 식별하는 데 필요합니다.

키 필드 레벨 스펙은 선택적이며 결합 논리 파일의 키 필드명을 포함합니다.

K 키 필드 스펙을 나타냅니다. K는 17열에 나옵니다. 키 필드 스펙은 선택적입니다.

키 필드명

키 필드명(이 예에서는 *Nbr*이 유일한 키 필드임)은 선택적이며, 이를 지정하면 논리 파일을 색인화(키 순) 파일로 만듭니다. 키 필드가 없으면 결합 논리 파일은 입력순 파일입니다. 결합 논리 파일에서 키 필드는 1차 파일의 필드여야 하며, 키 필드명은 논리 파일 레코드 형식의 19열에서 28열 사이에 지정되어야 합니다.

선택/생략 필드 레벨 스펙은 선택적이며, 결합 논리 파일의 선택/생략 필드명을 포함합니다.

S 또는 O

선택 또는 생략 스펙을 나타냅니다. S 또는 O는 17열에 나옵니다. 선택/생략 스펙은 선택적입니다.

선택/생략 필드명

선택/생략 값에 부합되는 레코드만 논리 파일을 사용하는 프로그램으로 리턴됩니다. 선택/생략 필드는 논리 파일 레코드 형식의 19열에서 28열 사이에 지정되어야 합니다.

관련 개념

DDS 개념

관련 참조

85 페이지의 『세 개 이상의 실제 파일 결합(예 7)』
결합 논리 파일을 사용하여 실제 파일을 최대 32개까지 결합할 수 있습니다. 이들 파일은 JFILE 키워드에 지정되어야 합니다. JFILE 키워드에 지정된 첫 번째 파일을 1차 파일이라 하고, 다른 파일들은 모두 2차 파일이라 합니다.

81 페이지의 『속성이 다른 결합 필드 사용(예 4)』
결합 필드로 사용하는 실제 파일의 필드에는 일반적으로 동일한 속성(길이, 자료 형식 및 소수 자릿수)이 있습니다. 그러나 경우에 따라 결합 필드에 다른 속성이 있을 수 있습니다.

결합 논리 파일 읽기: 다음은 67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』에 있는 결합 논리 파일이 어플리케이션 프로그램에 레코드를 제시하는 방법을 설명하고 있습니다.

JFILE 키워드에 PF1 파일이 맨 처음 지정되며 따라서 이 파일이 1차 파일입니다. 어플리케이션 프로그램이 레코드를 요구할 경우 시스템은 다음을 수행합니다.

1. 1차 파일의 첫 번째 결합 필드 값을 사용합니다(PF1의 *Nbr* 필드).
2. 결합 필드와 일치하는 2차 파일의 첫 번째 레코드를 찾습니다. (PF2의 *Nbr* 필드가 PF1의 *Nbr* 필드에 대응합니다.)
3. 일치될 때마다 실제 파일의 필드들을 한 개의 레코드로 결합하고 이 레코드를 프로그램에 제공합니다. 실제 파일에 있는 레코드 수에 따라 다음 조건 중 하나가 발생할 수 있습니다.
   a. 1차 파일의 모든 레코드에 대해 2차 파일에서 일치하는 레코드는 한 개만 존재합니다. 결합 논리 파일의 결과에는 1차 파일의 각 레코드에 대해 한 개의 레코드만 들어갈 수 있게 됩니다. 71 페이지의 『1차 및 2차 파일의 레코드 대응(케이스 1)』을 참조하십시오
   b. 1차 파일의 일부 레코드에 대해 2차 파일에서 일치하는 레코드는 존재하지 않습니다.

JDFTVAL 키워드를 지정하는 경우:

- 시스템이 2차 파일과 일치하는 레코드가 있는 1차 파일의 레코드를 2차 파일 또는 복수의 2차 파일에 결합시킵니다. 그 결과로 1차 파일의 각 레코드에 대해 하나 이상의 레코드가 존재하게 됩니다.
- 2차 파일과 일치하는 레코드가 없는 1차 파일 레코드에 대해 시스템은 2차 파일의 디폴트 필드 값을 추가하여 결합시킵니다. 디폴트 값을 정의하기 위해서 실제 파일에 DFT 키워드를 사용할 수 있습니다. 71 페이지의 『2차 파일에 레코드 누락: JDFTVAL 키워드 지정되지 않음(케이스 2A)』 및 72 페이지의 『2차 파일에 레코드 누락: JDFTVAL 키워드 지정됨(케이스 2B)』을 참조하십시오.

  주: 2차 파일에 DFT 키워드가 지정된 경우 DFT 키워드에 지정된 값이 결합에 사용됩니다. 그 결과로 1차 레코드 각각에 대해 최소한 한 개의 결합 레코드가 존재하게 됩니다.

- 2차 파일에 레코드가 있으나 1차 파일에는 일치되는 값이 없을 경우 레코드가 프로그램에 리턴되지 않습니다. 일치되는 1차 파일 레코드 없이 2차 파일 레코드가 존재하는지를 판별하기 위해서는 1차와 2차 파일의 순서를 뒤바꾸는 두 번째 결합 논리 파일을 사용할 수 있습니다.

JDFTVAL 키워드를 지정하지 않는 경우:

- 2차 파일 내에 일치하는 레코드가 있으면, 시스템은 2차 또는 복수의 2차 파일에 결합시킵니다. 그 결과로 1차 파일의 각 레코드에 대해 하나 이상의 레코드가 존재하게 됩니다.
- 2차 파일에 일치하는 레코드가 없을 경우 시스템은 레코드를 리턴시키지 않습니다.

주: JDFTVAL이 지정되지 않은 경우 시스템은 1차 파일의 레코드에 대해 일치되는 모든 2차 파일 레코드를 리턴합니다.

다음의 케이스 1에서 4까지는 순차 읽기 수행을 설명하고 있으며 케이스 5는 키에 의한 읽기를 설명하고 있습니다.

1차 및 2차 파일의 레코드 대응(케이스 1):

다음 두 표에 표시된 것처럼 67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정되고, PF1 및 PF2에 레코드 네 개가 있는 것으로 가정하십시오.

표 8. 실제 파일 1(PF1)

사원 번호	이름
235	Anne
440	Doug
500	Mark
729	Sue

표 9. 실제 파일 2(PF2)

사원 번호	급여
235	1700.00
440	950.50
500	2100.00
729	1400.90

프로그램이 4번의 읽기를 수행하여 다음과 같은 레코드를 얻습니다.

표 10. 결합 논리 파일(JLF)

사원 번호	이름	급여
235	Anne	1700.00
440	Doug	950.50
500	Mark	2100.00
729	Sue	1400.90

2차 파일에 레코드 누락: JDFTVAL 키워드 지정되지 않음(케이스 2A):

다음 두 표에 표시된 것처럼 67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정되고, PF1 및 PF2에 레코드 네 개가 있는 것으로 가정하십시오.

표 11. 실제 파일 1(PF1)

사원 번호	이름
235	Anne
440	Doug
500	Mark
729	Sue

표 12. 실제 파일 2(PF2)

사원 번호	급여
235	1700.00
440	950.50
729	1400.90

PF2에서 번호 500에는 레코드가 없습니다.

프로그램은 결합 논리 파일을 읽고 다음과 같은 레코드를 얻습니다.

표 13. 결합 논리 파일(JLF)

사원 번호	이름	급여
235	Anne	1700.00
440	Doug	950.50
729	Sue	1400.90

JDFTVAL 키워드를 지정하지 않고 2차 파일의 결합 필드에 대응이 존재하지 않는 경우 그 레코드는 결합 논리 파일에 포함되지 않습니다.

2차 파일에 레코드 누락: JDFTVAL 키워드 지정됨(케이스 2B):

67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정되고 다음의 DDS처럼 JDFTVAL 키워드가 지정된 것으로 가정하십시오.

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A                                     JDFTVAL
          A          R JOINREC                  JFILE(PF1 PF2)
          A          J                          JOIN(PF1 PF2)
          A                                     JFLD(NBR NBR)
          A            NBR                      JREF(PF1)
          A            NAME
          A            SALARY
          A          K NBR
     A
```

프로그램은 결합 논리 파일을 읽고 다음과 같은 레코드를 얻습니다.

표 14. 결합 논리 파일(JLF)

사원 번호	이름	급여
235	Anne	1700.00
440	Doug	950.50
500	Mark	0000.00
729	Sue	1400.90

JDFTVAL이 지정되면 시스템은 500에 대한 레코드가 비록 PF2에 없더라도 레코드를 리턴시킵니다. 이 레코드가 없으면 일부 필드 값이 결합 레코드에서 누락될 수 있습니다. 여기에서는 *Salary* 필드가 누락되었습니다. JDFTVAL이 지정되면 일반적으로 누락 문자 필드는 공백을 사용하며, 누락 숫자 필드는 0을 사용합니다. 따라서 이 경우에는 결합 레코드에서 누락된 레코드의 값이 0입니다. 그러나 실제 파일의 필드에 DFT 키워드가 지정되면 DFT 키워드에 지정된 디폴트 값이 사용됩니다.

2차 파일에 1차 파일의 레코드와 대응되는 레코드가 두 개 이상 있음(케이스 3):

다음 두 표에 표시된 것처럼 67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정되고, PF1에 레코드가 네 개 있고 PF2에 레코드가 다섯 개 있는 것으로 가정하십시오.

표 15. 실제 파일 1(PF1)

사원 번호	이름
235	Anne
440	Doug
500	Mark
729	Sue

표 16. 실제 파일 2(PF2)

사원 번호	급여
235	1700.00
235	1500.00
440	950.50
500	2100.00
729	1400.90

PF2에는 235에 대해 레코드가 중복되었습니다.

프로그램은 5개의 레코드를 얻습니다.

표 17. 결합 논리 파일(JLF)

사원 번호	이름	급여
235	Anne	1700.00
235	Anne	1500.00
440	Doug	950.50

표 17. 결합 논리 파일(JLF) (계속)

사원 번호	이름	급여
500	Mark	0000.00
729	Sue	1400.90

결합 레코드에는 235에 대해 레코드가 중복되었습니다. JDUPSQ 키워드를 사용하지 않는 한 중복 레코드에 대해 수신되는 레코드 순서를 예측할 수 없습니다.

관련 참조

79 페이지의 『2차 파일의 중복 레코드 읽기(예 3)』
때로는 2차 파일에 대한 결합 조작이 2차 파일로부터 2개 이상의 레코드를 생성합니다. 이런 경우 결합 스펙에 JDUPSEQ 키워드를 지정해 줌으로써 2차 파일의 해당 필드에서 중복된 레코드의 순서를 결정하도록 시스템에 알립니다.

2차 파일의 여분 레코드(케이스 4):

67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정되고, PF1에 레코드 4개 PF2에 레코드 5개가 있는 것으로 가정하십시오.

301에 대한 레코드는 PF2에만 존재합니다.

프로그램은 결합 논리 파일을 읽고 다음의 네 개 레코드만 얻습니다. 301에 대해서는 레코드가 나오지 않습니다.

표 18. 결합 논리 파일(JLF)

사원 번호	이름	급여
235	Anne	1700.00
440	Doug	950.50
500	Mark	2100.00
729	Sue	1400.90

이러한 결과는 JDFTVAL 키워드를 지정할 경우에도 같은데 결합 레코드를 수신하기 위해서는 반드시 1차 파일에 레코드가 항상 있어야 하기 때문입니다.

임의 액세스(케이스 5):

67 페이지의 『두 개의 실제 파일 결합에 대한 기본 개념(예 1)』과 같이 결합 논리 파일이 지정된 것으로 가정하십시오결합 논리 파일에 키 필드가 정의되어 있는 것에 주의하십시오. 이 경우는 결합 논리 파일을 사용한 임의 액세스 읽기 수행으로 어떤 레코드가 리턴되는지를 보여줍니다.

PF1과 PF2에 다음과 같은 레코드가 있는 것으로 가정하십시오.

표 19. 실제 파일 1(PF1)

사원 번호	이름
235	Anne

표 19. 실제 파일 1(PF1) (계속)

사원 번호	이름
440	Doug
500	Mark
729	Sue
997	Tim

표 20. 실제 파일 2(PF2)

사원 번호	급여
235	1700.00
440	950.50
729	1400.90
984	878.25
997	331.00
997	555.00

PF2에는 레코드 500에 대해 발견되는 레코드가 없으며 레코드 984만 PF2에 있고 997에 대해서는 중복 레코드가 발견되었습니다.

프로그램은 다음과 같은 레코드를 얻을 수 있습니다.

논리 파일의 *Nbr* 필드에 대해 프로그램으로부터 235라는 값이 주어지면 시스템은 다음과 같은 레코드를 제공합니다.

사원 번호	이름	급여
235	Anne	1700.00

논리 파일의 *Nbr* 필드에 대해 프로그램으로부터 500이라는 값이 주어지고 JDFTVAL 키워드가 지정되면 시스템은 다음과 같은 레코드를 제공합니다.

사원 번호	이름	급여
500	Mark	0000.00

주: JDFTVAL 키워드가 결합 논리 파일에 지정되지 않은 경우 2차 파일에 대응하는 레코드가 없기 때문에 500이라는 값에 대한 레코드는 찾을 수 없습니다.

논리 파일의 *Nbr* 필드에 대해 프로그램으로부터 984라는 값이 주어지면, 1차 파일에 레코드 984가 없기 때문에 시스템은 레코드를 제공하지 않으며 레코드 없음 예외가 발생합니다.

논리 파일의 *Nbr* 필드에 대해 997이라는 값이 주어지면, 시스템은 다음 레코드 중 하나를 리턴합니다.

사원 번호	이름	급여
997	Tim	331.00

또는

사원 번호	이름	급여
997	Tim	555.00

어떤 레코드가 프로그램으로 리턴될 것인지를 예측할 수 없습니다. 리턴될 레코드를 지정하려면 결합 논리 파일에 JDUPSEQ 키워드를 지정하십시오.

주:

1. 임의 액세스 사용 시 어플리케이션 프로그래머는 PF2에 중복 키가 있을 수 있음을 인식하고, 프로그램이 중복 키가 있는 레코드에 대해 읽기를 두 번 이상 수행하도록 해야 합니다. 프로그램이 순차 액세스를 사용할 경우에는 두 번째 읽기에서 두 번째 레코드가 얻어집니다.
2. JDUPSEQ 키워드를 지정할 경우 시스템은 결합 논리 파일에 대해 별도의 액세스 경로를 작성할 수 있습니다. (시스템이 공유할 수 있는 기존 액세스 경로를 찾아낼 가능성이 적기 때문입니다.) JDUPSEQ 키워드를 생략할 경우 시스템은 다른 파일의 액세스 경로를 공유할 수 있습니다. (이 경우 시스템은 PF2의 액세스 경로를 공유할 수 있습니다.)

관련 참조

79 페이지의 『2차 파일의 중복 레코드 읽기(예 3)』
때로는 2차 파일에 대한 결합 조작이 2차 파일로부터 2개 이상의 레코드를 생성합니다. 이런 경우 결합 스펙에 JDUPSEQ 키워드를 지정해 줌으로써 2차 파일의 해당 필드에서 중복된 레코드의 순서를 결정하도록 시스템에 알립니다.

결합 논리 파일 설정:

이 주제에서는 결합 논리 파일 설정 방법을 표시합니다.

논리 파일을 설정하려면 다음 단계를 수행하십시오.

1. 논리 파일 레코드 형식에서 포함시킬 모든 실제 파일 필드의 필드명을 알아내십시오. (DSPFFD(파일 필드 설명 표시) 명령을 사용하여 파일에 포함된 필드를 표시할 수 있습니다.)
2. 레코드 형식에 필드를 서술하십시오. 필드명을 수직 리스트에 기록하십시오. 이것이 결합 논리 파일에 대한 레코드 형식의 시작입니다.

   주: 필드명은 어떤 순서로든 지정할 수 있습니다. 서로 다른 실제 파일에 동일한 필드명이 나오는 경우 이들 필드에 대해 JREF 키워드로 실제 파일명을 지정하십시오. RENAME 키워드를 사용하여 필드를 재명명할 수 있으며, CONCAT 키워드를 사용하여 동일한 실제 파일에 들어 있는 필드들을 연결할 수 있습니다. 또한 SST 키워드를 사용하여 기존의 문자 필드, 16진 필드 또는 존 십진 필드의 서브세트를 정의할 수 있습니다. 문자 또는 존 십진 필드의 서브스트링은 문자 필드이며 16진 필드의 서브스트링도 16진 필드입니다. 필드의 자료 형태, 길이 또는 소수 자릿수를 변경하여 필드를 다시 정의할 수 있습니다.

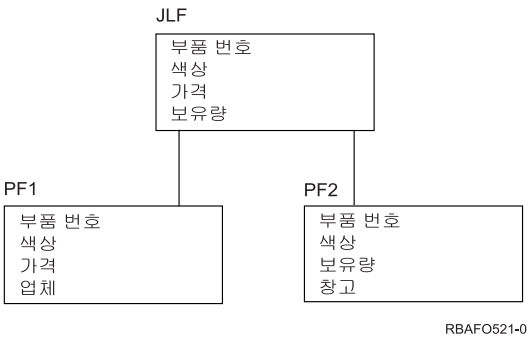
3. 실제 파일명을 JFILE 키워드의 매개변수 값으로 지정하십시오. 첫 번째로 지정하는 이름은 1차 파일입니다. 나머지는 모두 2차 파일이 됩니다. 최고의 성능을 위해 1차 파일 다음에 가장 적은 레코드 수를 가진 2차 파일을 먼저 지정하십시오.
4. 각 2차 파일에 대한 결합 스펙을 코딩하십시오. 각 결합 스펙에서 어떤 파일 쌍이 결합(JOIN 키워드, 2차 파일이 하나일 경우 선택사항)되는지 확인하고, 결합하는 데 어떤 필드가 사용(JFLD 키워드 사용, 각 결합 스펙에 적어도 하나가 필요함)되었는지 확인하십시오.
5. **옵션:** 다음 항목을 지정하십시오.
   a. JDFTVAL 키워드. 2차 파일과 대응하는 레코드가 존재하지 않더라도 1차 파일의 각 레코드를 리턴하려면 이를 지정하십시오.
   b. JDUPSEQ 키워드. 2차 파일에서 중복값을 가지고 있을지도 모르는 필드에 대해 이를 지정하십시오. JDUPSEQ는 어떤 필드로 중복값을 분류할 것인지(결합 필드 제외)를 지정하며, 사용될 순서를 지정합니다.
   c. 키 필드. 키 필드는 2차 파일로부터 올 수 없습니다. 키 필드를 생략할 경우 레코드는 1차 파일에 있는 대로 입력순으로 리턴됩니다.
   d. 선택/생략 필드. 경우에 따라서는 파일 레벨에서 동적 선택(DYNSLT) 키워드를 지정해야 합니다.
   e. 비입/출력용 필드.

관련 참조

82 페이지의『레코드 형식에 나오지 않는 필드 서술(예 5)』
비입출력용으로 결합 논리 파일에 사용될 수 있는(38열에 N이 지정됨) 두 필드 모두 레코드 형식에 포함될 수 없습니다. 이 예는 레코드 형식에 나오지 않는 필드 서술 방법을 표시합니다.

두 개 이상의 필드를 사용하여 파일 결합(예 2):

한 쌍의 파일을 결합하기 위해 두 개 이상의 결합 필드를 지정할 수 있습니다. 이 예는 논리 파일과 두 개의 실제 파일에 있는 필드를 나타냅니다.



결합 논리 파일(JLF)에는 *Part number*, *Color*, *Price* 및 *Quantity on hand* 필드가 있습니다. 실제 파일 1(PF1)에는 *Part number*, *Color*, *Price* 및 *Vendor*가 있고, 실제 파일 2(PF2)에는 *Part number*, *Color*, *Quantity on hand* 및 *Warehouse*가 있습니다. 이러한 파일의 자료 서술 스펙(DDS)은 다음과 같이 표시됩니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(PTNBR PTNBR)
          A                                      JFLD(COLOR COLOR)
          A            PTNBR                     JREF(PF1)
          A            COLOR                     JREF(PF1)
          A            PRICE
          A            QUANTOH
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            PTNBR          4
          A            COLOR         20
          A            PRICE          7  2
          A            VENDOR        40
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            PTNBR          4
          A            COLOR         20
          A            QUANTOH        5  0
          A            WAREHSE       30
     A
```

실제 파일에 다음과 같은 레코드가 있는 것으로 가정하십시오.

표 21. 실제 파일 1(PF1)

Part number	Color	Price	Vendor
100	검정색	22.50	ABC Corp.
100	흰색	20.00	Ajax Inc.
120	노란색	3.75	ABC Corp.
187	녹색	**110.95**	ABC Corp.
187	빨간색	**110.50**	ABC Corp.
190	청색	40.00	Ajax Inc.

표 22. 실제 파일 2(PF2)

Part number	Color	Quantity on hand	Warehouse
100	검정색	23	ABC Corp.
100	흰색	15	Ajax Inc.
120	노란색	102	ABC Corp.
187	녹색	**0**	ABC Corp.
187	빨간색	**2**	ABC Corp.
190	청색	**2**	Ajax Inc.

파일이 순차적으로 처리되는 경우 프로그램은 다음과 같은 레코드를 얻게 됩니다.

표 23. 결합 논리 파일(JLF)

Part number	Color	Price	Quantity on hand
100	검정색	22.50	23
100	흰색	20.00	15
120	노란색	3.75	102
187	녹색	**110.95**	**0**
187	빨간색	**110.50**	**2**

주: 2차 파일에서 두 필드에 대응하는 값이 없기 때문에 부품 번호 190, 색상 blue에 대한 레코드는 프로그램에서 사용할 수 없습니다. JDFTVAL이 지정되지 않았기 때문에 레코드는 리턴되지 않습니다.

2차 파일의 중복 레코드 읽기(예 3):

때로는 2차 파일에 대한 결합 조작이 2차 파일로부터 2개 이상의 레코드를 생성합니다. 이런 경우 결합 스펙에 JDUPSEQ 키워드를 지정해 줌으로써 2차 파일의 해당 필드에서 중복된 레코드의 순서를 결정하도록 시스템에 알립니다.

실제 파일 및 결합 논리 파일에 대한 자료 서술 스펙(DDS)이 다음과 같이 표시됩니다.

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JREC                      JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(NAME1 NAME2)
          A                                      JDUPSEQ(TELEPHONE)
          A            NAME1
          A            ADDR
          A            TELEPHONE
     A

PF1
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            NAME1         10
          A            ADDR          20
     A

PF2
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            NAME2         10
          A            TELEPHONE      8
     A
```

실제 파일에는 다음 레코드가 포함됩니다.

표 24. 실제 파일 1(PF1)

이름	주소
Anne	120 1st St.

표 24. 실제 파일 1(PF1) (계속)

이름	주소
Doug	40 Pillsbury
Mark	2 Lakeside Dr.

표 25. 실제 파일 2(PF2)

이름	전화
Anne	555–1111
Anne	555–6666
Anne	555–2222
Doug	555–5555

결합 논리 파일은 다음 레코드를 리턴합니다.

표 26. 결합 논리 파일(JLF)

이름	주소	전화
Anne	120 1st St.	555–1111
Anne	120 1st St.	555–2222
Anne	120 1st St.	555–6666
Doug	40 Pillsbury	555–5555

프로그램이 Anne, Doug, Mark순으로 사용 가능한 모든 레코드를 읽습니다. Anne의 주소는 한 개이나 전화번호는 세 개입니다. 따라서 Anne에 대해 리턴되는 레코드는 세 개입니다.

키워드 매개변수로서 *DESCEND를 지정하지 않으면 JDUPSEQ 키워드는 오름차순으로 분류하기 때문에 Anne에 대한 레코드는 전화번호에 의한 오름차순으로 분류됩니다. 다음 예는 DDS에서의 *DESCEND 사용법을 보여주고 있습니다.

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JREC                     JFILE(PF1 PF2)
          A          J                          JOIN(PF1 PF2)
          A                                     JFLD(NAME1 NAME2)
          A                                     JDUPSEQ(TELEPHONE *DESCEND)
          A            NAME1
          A            ADDR
          A            TELEPHONE
     A
```

*DESCEND가 있는 JDUPSEQ를 지정할 경우 레코드가 다음과 같이 리턴됩니다.

표 27. 결합 논리 파일(JLF)

이름	주소	전화
Anne	120 1st St.	555–6666
Anne	120 1st St.	555–2222
Anne	120 1st St.	555–1111

표 27. 결합 논리 파일(JLF) (계속)

이름	주소	전화
Doug	40 Pillsbury	555–5555

주: JDUPSEQ 키워드는 그것이 지정되어 있는 결합 스펙에만 적용됩니다.

관련 참조

90 페이지의 『복잡한 결합 논리 파일(예 10)』
이 예는 보다 복잡한 논리 파일을 보여주고 있습니다.

속성이 다른 결합 필드 사용(예 4):

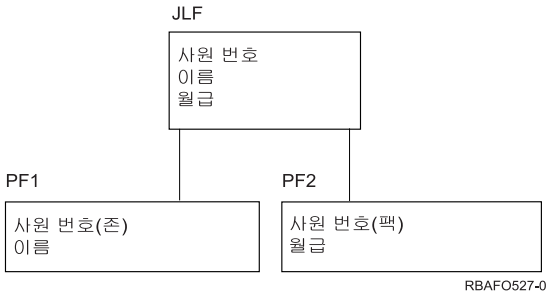
결합 필드로 사용하는 실제 파일의 필드에는 일반적으로 동일한 속성(길이, 자료 형식 및 소수 자릿수)이 있습니다. 그러나 경우에 따라 결합 필드에 다른 속성이 있을 수 있습니다.

예를 들어, 79 페이지의 『2차 파일의 중복 레코드 읽기(예 3)』에서와 같이 *Name1* 필드는 실제 파일 PF1에서 길이가 10자인 문자 필드이고, 이것은 실제 파일 PF2에 있는 길이 10자의 문자 필드인 *Name2* 필드에 결합될 수 있습니다. *Name1* 필드와 *Name2* 필드는 동일한 특징을 가지고 있으며 따라서 결합 필드로 사용할 수 있습니다.

필드를 다시 정의하지 않고도 결합 필드로서 다른 길이를 가진 문자 유형 필드를 사용할 수 있습니다. 예를 들어, PF1의 NAME1 필드 길이가 10자이고, PF2의 NAME2 필드가 15자이면 이 필드 중 하나를 다시 정의하지 않고도 결합 필드로 사용할 수 있습니다.

다음 예는 동일한 속성을 포함하지 않는 결합 필드를 표시합니다. 실제 파일 모두 사원번호용 필드가 있습니다. 실제 파일 PF1의 *Nbr* 필드와 실제 파일 PF2의 *Nbr* 필드는 34열에 길이 3이 지정되었으나, PF1 파일의 필드는 존(35열에 S)인 반면에 PF2 파일의 필드는 팩(35열에 P)입니다이 필드들을 결합 필드로 사용하여 두 파일을 결합하려면 두 필드의 한쪽 또는 양쪽을 다시 정의하여 동일한 속성을 갖도록 해야 합니다.

다음 예는 논리 파일 및 실제 파일의 필드를 나타냅니다.



결합 논리 파일(JLF)에는 *Employee number*, *Name* 및 *Salary* 필드가 있습니다. 실제 파일 1(PF1)에는 *Employee number*(존)과 *Name*이 있습니다. 실제 파일 2(PF2)에는 *Employee number*(팩) 및 *Salary*가 있습니다. 이러한 파일의 자료 서술 스펙(DDS)은 다음과 같이 표시됩니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(NBR NBR)
          A            NBR             S         JREF(2)
          A            NAME
          A            SALARY
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            NBR            3S 0 <-Zoned
          A            NAME          20
          A          K NBR
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            NBR            3P 0 <-Packed
          A            SALARY         7  2
          A          K NBR
     A
```

주: 이 예에서 논리 파일의 *Nbr* 필드는 PF2에서 온 것이며, 그 이유는 JREF(2)가 지정되었기 때문입니다실제 파일명을 지정하는 대신 JREF 키워드에 상대 파일 번호를 지정할 수 있습니다. 이 예에서 2는 PF2를 나타냅니다.

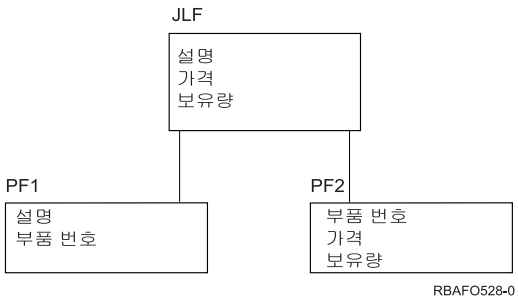
PF1 파일과 PF2 파일의 *Nbr* 필드가 결합 필드로서 사용되었기 때문에 이들은 같은 속성을 가져야 합니다. 그런데 이 예에서는 그렇지 않습니다. 따라서 동일한 속성을 갖도록 두 필드 중 한쪽이나 양쪽을 다시 정의해야 합니다. 이 예에서 두 개의 사원 번호 필드의 속성이 다른 것을 해결하려면 JLF(PF2 파일에서 온 것임)의 *Nbr* 필드가 존(JLF의 35열에 S)으로 다시 정의되어야 합니다.

레코드 형식에 나오지 않는 필드 서술(예 5):

비입출력용으로 결합 논리 파일에 사용될 수 있는(38열에 N이 지정됨) 두 필드 모두 레코드 형식에 포함될 수 없습니다. 이 예는 레코드 형식에 나오지 않는 필드 서술 방법을 표시합니다.

결합 논리 파일을 사용하는 프로그램은 비입출력용 필드를 보거나 읽을 수 없습니다. 비입/출력용 필드는 레코드 형식에 포함되지 않습니다. 비입출력용 필드는 키 필드가 될 수 없으며 결합 파일의 선택/생략문에 사용할 수 없습니다. 비입/출력용 필드는 결합 조작을 위해 레코드 레벨에서 다시 정의되는 결합 필드(JFLD 키워드에서 결합 스펙 레벨에 지정됨)로 사용할 수 있지만 프로그램에서는 필요하지 않습니다.

다음 예에서 프로그램은 재고 부품의 설명, 가격 및 재고량을 읽습니다. 부품 번호 자체는 부품에 대한 레코드를 함께 가져올 때를 제외하고는 필요하지 않습니다. 그러나 부품 번호가 서로 다른 속성을 가지고 있기 때문에 적어도 한 개는 다시 정의되어야 합니다.



결합 논리 파일(JLF)에는 *Description*, *Price* 및 *Quantity on hand* 필드가 있습니다. 실제 파일 1(PF1)에는 *Description* 및 *Part number*가 있고, 실제 파일 2(PF2)에는 *Part number*, *Price* 및 *Quantity on hand*가 있습니다. 이러한 파일의 자료 서술 스펙(DDS)은 다음과 같이 표시됩니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(PRTNBR PRTNBR)
          A            PRTNBR          S  N      JREF(1)
          A            DESC
          A            PRICE
          A            QUANT
          A          K DESC
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            DESC          30
          A            PRTNBR         6P 0
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            PRTNBR         6S 0
          A            PRICE          7  2
          A            QUANT          8  0
     A
```

PF1에서 *Prtnbr* 필드는 팩 십진 필드이고, PF2에서 *Prtnbr* 필드는 존(zone) 십진 필드입니다.결합 논리 파일에서 이 필드들은 결합 필드로 사용되며, PF1의 *Prtnbr* 필드는 필드 레벨에서 35열에 S를 지정하여 존 십진 필드로 다시 정의됩니다. JREF 키워드는 필드가 어떤 실제 파일로부터 온 것인지를 나타냅니다. 그러나 필드는 레코드 형식에 포함되지 않으므로 38열에 N이 지정되어 이를 비입/출용 필드로 만듭니다. 이 파일을 사용하는 프로그램은 그 필드를 볼 수 없게 됩니다.

이 예에서 판매 사무원이 부품에 대한 설명을 입력할 수 있습니다. 프로그램은 대응하는 레코드를 찾기 위해 결합 논리 파일을 읽고 한 개 이상의 부품을 표시하여 사용자로 하여금 설명, 가격 및 수량 등을 검사할 수 있도록 합니다. 이 어플리케이션에서는 고객의 주문을 받거나 창고에서 부품을 추가로 주문할 때 부품 번호는 필요없는 것으로 가정합니다.

결합 논리 파일에 키 필드 지정(예 6):

이 주제에서는 결합 논리 파일에 키 필드 지정 규칙에 관해 설명합니다.

결합 논리 파일에서 키 필드를 지정하려면 다음과 같은 규칙이 적용됩니다.

- 키 필드는 1차 실제 파일에 존재해야 합니다.
- 키 필드는 논리 파일 내의 결합 레코드 형식의 19열에서 28열 사이에 명명되어야 합니다.
- 논리 파일에서 비입출력용 필드(38열에 N이 지정됨)로 정의된 필드는 키 필드가 될 수 없습니다.

다음 예는 키 필드에 대한 규칙을 보여줍니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(NBR NUMBER)
          A                                      JFLD(FLD3 FLD31)
          A            FLD1                      RENAME(F1)
          A            FLD2                      JREF(2)
          A            FLD3          35   N
          A            NAME
          A            TELEPHONE                 CONCAT(AREA LOCAL)
          A          K FLD1
          A          K NAME
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            NBR            4
          A            F1            20
          A            FLD2           7  2
          A            FLD3          40
          A            NAME          20
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            NUMBER         4
          A            FLD2           7  2
          A            FLD31         35
          A            AREA           3
          A            LOCAL          7
     A
```

다음은 키 필드가 될 수 **없습니다**.

- *Nbr*(19-28열에 명명되지 않음)
- *Number*(19-28열에 명명되지 않음)
- *F1*(19-28열에 명명되지 않음)
- *Fld31*(2차 파일에서 옴)

- *Fld2*(2차 파일에서 옴)
- *Fld3*(비입출력용 필드임)
- *Area* 및 *Local*(19-28열에 명명되지 않음)
- *Telephone*(2차 파일의 필드를 기초로 함)

결합 논리 파일에 선택/생략문 지정:

이 주제에서는 결합 논리 파일에 선택/생략문 지정 규칙에 대해 설명합니다.

결합 논리 파일에 선택/생략문을 지정하려면 다음과 같은 규칙을 적용하십시오.

- 필드는 논리 파일이 사용하는 어떤 실제 파일(JFILE 키워드에 지정된)에서도 올 수 있습니다.
- 선택/생략문에서 지정하는 필드는 비입출력용 필드(38열에서 N이 지정됨)로 정의된 필드여서는 안됩니다.
- 경우에 따라서는 결합 논리 파일에 선택/생략문을 지정할 때 DYNSLT 키워드를 지정해야 합니다.

관련 개념

DDS 개념

관련 참조

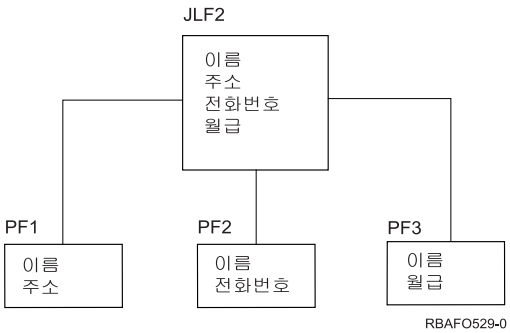
90 페이지의『복잡한 결합 논리 파일(예 10)』
이 예는 보다 복잡한 논리 파일을 보여주고 있습니다.

세 개 이상의 실제 파일 결합(예 7):

결합 논리 파일을 사용하여 실제 파일을 최대 32개까지 결합할 수 있습니다. 이들 파일은 JFILE 키워드에 지정되어야 합니다. JFILE 키워드에 지정된 첫 번째 파일을 1차 파일이라 하고, 다른 파일들은 모두 2차 파일이라 합니다.

실제 파일은 쌍으로 결합되어야 하며 각 쌍은 결합 스펙에 의하여 서술됩니다. 각 결합 스펙에는 한 개 이상의 결합 필드가 있어야 합니다.

다음 도표는 파일 내의 필드와 논리 파일 내의 모든 실제 파일에 공통되는 한 개의 필드를 나타냅니다.



결합 논리 파일(JLF2)에는 *Name*, *Address*, *Telephone*, *Salary*가 있습니다. 실제 파일 1(PF1)에는 *Name*과 *Address*, 실제 파일 2(PF2)에는 *Name*과 *Telephone*, 실제 파일 3(PF3)에는 *Name*과 *Salary*가 있습니다. 이 예에서는 *Name* 필드가 모든 실제 파일(PF1, PF2 및 PF3)에 공통되며 결합 필드 역할을 합니다.

다음 예는 실제 파일 및 논리 파일의 자료 서술 스펙(DDS)을 나타냅니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R JOINREC                   JFILE(PF1 PF2 P3)
          A          J                           JOIN(PF1 PF2)
          A                                      JFLD(NAME NAME)
          A          J                           JOIN(PF2 PF3)
          A                                      JFLD(NAME NAME)
          A            NAME                      JREF(PF1)
          A            ADDR
          A            TELEPHONE
          A            SALARY
          A          K NAME
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC1
          A            NAME          10
          A            ADDR          20
          A          K NAME
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC2
          A            NAME          10
          A            TELEPHONE      7
          A          K NAME
     A
```

PF3

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R REC3
          A            NAME          10
          A            SALARY         9  2
          A          K NAME
     A
```

실제 파일에 다음과 같은 레코드가 있는 것으로 가정하십시오.

표 28. 실제 파일 1(PF1)

이름	주소
Anne	120 1st St.
Doug	40 Pillsbury
Mark	2 Lakeside Dr.
Tom	335 Elm St.

표 29. 실제 파일 2(PF2)

이름	전화
Anne	555–1111
Doug	555–5555
Mark	555–0000
Sue	555–3210

표 30. 실제 파일 3(PF3)

이름	급여
Anne	1700.00
Doug	950.00
Mark	2100.00

프로그램은 다음과 같은 논리 파일 레코드를 읽습니다.

표 31. 결합 논리 파일(JLF)

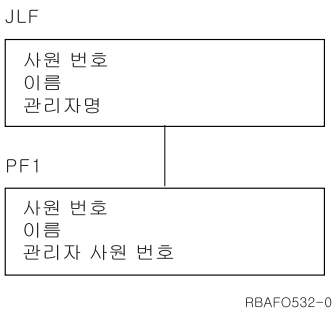
이름	주소	전화	급여
Anne	120 1st St.	555–1111	1700.00
Doug	40 Pillsbury	555–5555	950.00
Mark	2 Lakeside Dr..	555–0000	2100.00
Doug	40 Pillsbury	555–5555	

PF2와 PF3에 Tom에 대한 레코드가 없고 JDFTVAL 키워드가 지정되지 않았으므로 Tom에 대한 레코드는 리턴되지 않습니다. 1차 파일에 Sue에 대한 레코드가 없으므로 Sue에 대한 레코드는 리턴되지 않습니다.

실제 파일을 그 자체에 결합하는 방법(예 8):

실제 파일 그 자체로부터 2개 이상의 레코드를 결합하여 만들어지는 레코드를 읽기 위해 실제 파일을 그 자체에 결합시킬 수 있습니다.

다음 도표는 실제 파일을 자체에 결합하는 방법을 표시합니다.



결합 논리 파일(JLF)에는 *Employee number*, *Name* 및 *Manager's name*이 포함됩니다. 실제 파일(PF1)에는 *Employee number*, *Name* 및 *Manager's employee number*가 포함됩니다. 다음 예는 이러한 파일에 대한 자료 서술 스펙(DDS)을 표시합니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A                                     JDFTVAL
          A          R JOINREC                  JFILE(PF1 PF1)
          A          J                          JOIN(1 2)
          A                                     JFLD(MGRNBR NBR)
          A            NBR                      JREF(1)
          A            NAME                     JREF(1)
          A            MGRNAME                  RENAME(NAME)
          A                                     JREF(2)
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD1
          A            NBR             3
          A            NAME           10          DFT('none')
          A            MGRNBR          3
     A
```

주:

1. 상대 파일 번호가 JOIN 키워드에 지정되어야 하며 그 이유는 동일한 파일명이 JFILE 키워드에 두 번 지정되었기 때문입니다. 상대 파일 번호 1은 JFILE 키워드에 지정된 첫 번째 실제 파일을 참조하고, 2는 두 번째 실제 파일을 참조하며 이와 같이 번호와 파일의 관계가 계속됩니다.
2. JFILE 키워드에 동일한 실제 파일이 지정되는 경우 필드 레벨에서 지정된 각 필드에 대해 JREF 키워드가 필요합니다.

PF1에 다음과 같은 레코드가 있는 것으로 가정하십시오.

표 32. 실제 파일 1(PF1)

사원 번호	이름	관리자 사원 번호
235	Anne	440
440	Doug	729
500	Mark	440
729	Sue	888

프로그램은 다음과 같은 논리 파일 레코드를 읽습니다.

표 33. 결합 논리 파일(JLF)

사원 번호	이름	관리자 이름
235	Anne	Doug
440	Doug	Sue
500	Mark	Doug
729	Sue	none

주:

1. JDFTVAL 키워드가 지정되었으므로 관리자 이름 Sue에 대한 레코드가 리턴됩니다.
2. PF1 실제 파일의 *Name* 필드에 DFT 키워드가 지정되었으므로 none 값이 리턴됩니다.

2차 파일의 누락 레코드에 디폴트 자료 사용(예 9):

3개 이상의 파일을 결합하고 JDFTVAL 키워드를 지정하는 경우 2차 파일에서 누락된 결합 필드에 대해 시스템에서 제공하는 디폴트 값이 다른 2차 파일을 결합하는 데 사용됩니다.

2차 파일에 DFT 키워드가 지정되면 DFT 키워드에 지정된 값이 논리 파일에 사용됩니다.

파일의 자료 서술 스펙(DDS)이 다음과 같이 표시됩니다.

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A                                     JDFTVAL
          A          R JRCD                     JFILE(PF1 PF2 PF3)
          A          J                          JOIN(PF1 PF2)
          A                                     JFLD(NAME NAME)
          A          J                          JOIN(PF2 PF3)
          A                                     JFLD(TELEPHONE TELEPHONE)
          A            NAME                     JREF(PF1)
          A            ADDR
          A            TELEPHONE                JREF(PF2)
          A            LOC
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD1
          A            NAME          20
          A            ADDR          40
          A            COUNTRY       40
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD2
          A            NAME          20
          A            TELEPHONE      8          DFT('999-9999')
     A
```

PF3

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD3
          A            TELEPHONE      8
          A            LOC           30          DFT('No location assigned')
     A
```

PF1, PF2 및 PF3에 다음 레코드가 있는 것으로 가정하십시오.

표 34. 실제 파일 1(PF1)

이름	주소	국가
Anne	120 1st St.	USA

표 34. 실제 파일 1(PF1) (계속)

이름	주소	국가
Doug	40 Pillsbury	Canada
Mark	2 Lakeside Dr.	Canada
Sue	120 Broadway	USA

표 35. 실제 파일 2(PF2)

이름	전화
Anne	555–1234
Doug	555–2222
Sue	555–1144

표 36. 실제 파일 3(PF3)

전화	위치
555–1234	Room 312
555–2222	Main lobby
999–9999	전화 번호 없음

결합 논리 파일에 JDFTVAL을 지정하면 프로그램은 다음과 같은 논리 파일 레코드를 읽습니다.

표 37. 결합 논리 파일(JLF)

이름	주소	전화	위치
Anne	120 1st St.	555–1234	Room 312
Doug	40 Pillsbury	555–2222	Main lobby
Mark	2 Lakeside Dr.	999–9999	전화 번호 없음
Sue	120 Broadway	555–1144	할당된 위치 없음

이 예에서 Anne과 Doug에 대한 자료는 모두 찾을 수 있습니다. 그러나 Mark와 Sue에 대한 자료의 일부가 누락되었습니다.

- Mark는 전화번호가 없기 때문에 그에 대한 레코드가 PF2에서 누락되었습니다. PF2의 *Telephone* 필드에 대한 디폴트 값은 DFT 키워드를 사용하여 999-9999로 정의됩니다. 따라서 이 예에서 전화번호가 할당되지 않을 경우 되돌려지는 전화번호는 999-9999입니다. 결합 논리 파일에 지정된 JDFTVAL 키워드는 PF2의 *Telephone* 필드에 대한 디폴트 값(999-9999)을 PF3의 레코드와 대응시키는 데 사용됩니다. (PF3에는 전화번호 999-9999에 대한 설명을 보여주는 레코드가 포함되어 있습니다.) JDFTVAL 키워드가 없으면, Mark에 대한 레코드는 리턴되지 않습니다.
- 아직 Sue의 전화번호가 위치에 할당되지 않았으므로 PF3에는 555-1144에 대한 레코드가 없습니다. JDFTVAL 키워드가 지정되지 않으면 Sue에 대한 레코드는 리턴되지 않습니다. JDFTVAL이 지정되면 시스템은 PF3의 *Loc* 필드에 DFT 키워드에 지정된 디폴트 값(`No location assigned`)을 제공합니다.

복잡한 결합 논리 파일(예 10):

이 예는 보다 복잡한 논리 파일을 보여주고 있습니다.

자료가 다음과 같은 세 개의 실제 파일에 있는 것으로 가정하십시오.

공급자 마스터 파일**(PF1)**

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD1                      TEXT('VENDOR INFORMATION')
          A            VDRNBR         5          TEXT('VENDOR NUMBER')
          A            VDRNAM        25          TEXT('VENDOR NAME')
          A            STREET        15          TEXT('STREET ADDRESS')
          A            CITY          15          TEXT('CITY')
          A            STATE          2          TEXT('STATE')
          A            ZIPCODE        5          TEXT('ZIP CODE')
          A                                       DFT('00000')
          A            PAY            1          TEXT('PAY TERMS')
     A
```

주문 파일**(PF2)**

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD2                      TEXT('VENDORS ORDER')
          A            VDRNUM         5S 0       TEXT('VENDOR NUMBER')
          A            JOBNBR         6          TEXT('JOB NUMBER')
          A            PRTNBR         5S 0       TEXT('PART NUMBER')
          A                                       DFT(99999)
          A            QORDER         3S 0       TEXT('QUANTITY ORDERED')
          A            UNTPRC         6S 2       TEXT('PRICE')
     A
```

부분 파일**(PF3)**

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A          R RCD3                      TEXT('DESCRIPTION OF PARTS')
          A            PRTNBR         5S 0       TEXT('PART NUMBER')
          A                                       DFT(99999)
          A            DESCR         25          TEXT('DESCRIPTION')
          A            UNITPRICE      6S 2       TEXT('UNIT PRICE')
          A            WHSNBR         3          TEXT('WAREHOUSE NUMBER')
          A            PRTLOC         4          TEXT('LOCATION OF PART')
          A            QOHAND         5          TEXT('QUANTITY ON HAND')
     A
```

결합 논리 파일 레코드 형식에는 다음과 같은 필드가 있어야 합니다.

- *Vdrnam*(공급자명)
- *Street, City, State*, and *Zipcode*(공급자 주소)
- *Jobnbr*(작업 번호)
- *Prtnbr*(부품 번호)
- *Descr*(부품 설명)
- *Qorder*(주문량)
- *Untprc*(단가)
- *Whsnbr*(창고 번호)
- *Prtloc*(부품 위치)

이 결합 논리 파일의 자료 서술 스펙(DDS)이 다음과 같이 표시됩니다.

결합 논리 파일(JLF)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                  1  DYNSLT
     A                                  2 JDFTVAL
     A          R RECORD1                    JFILE(PF1 PF2 PF3)
     A     3  J                           JOIN(1 2)
     A                                      JFLD(VDRNBR VDRNUM)
     A                                  4  JDUPSEQ(JOBNBR)
     A     5  J                           JOIN(2 3)
     A                                  6  JFLD(PRTNBR PRTNBR)
     A                                      JFLD(UNTPRC UNITPRICE)
     A       7  VDRNUM         5A  N       TEXT('CHANGED ZONED TO CHAR')
     A            VDRNAM
     A            ADDRESS               8  CONCAT(STREET CITY STATE +
     A                                         ZIPCODE)
     A            JOBNBR
     A            PRTNBR                9  JREF(2)
     A            DESCR
     A            QORDER
     A            UNTPRC
     A            WHSNBR
     A            PRTLOC
     A    10  S VDRNAM                     COMP(EQ 'SEWING COMPANY')
     A          S QORDER                     COMP(GT 5)
     A
```

1 JDFTVAL 키워드와 선택 필드가 지정되었으므로 DYNSLT 키워드는 필수적입니다.

2 JDFTVAL 키워드는 실제 파일의 디폴트 값을 가져오기 위해 지정됩니다.

3 첫 번째 결합 스펙.

4 중복되는 공급자 번호가 PF2에서 발생하므로 JDUPSEQ 키워드가 지정됩니다.

5 두 번째 결합 스펙.

6 PF2 파일과 PF3 파일에서 정확한 레코드가 결합되도록 보장하기 위해 두 개의 JFLD 키워드가 지정됩니다.

7 *Vdrnum* 필드는 존 십진에서 문자로 다시 정의됩니다.

8 CONCAT 키워드는 동일한 실제 파일에 있는 필드 4개를 한 개의 필드로 연결합니다.

9 *Prtnbr* 필드가 두 개의 실제 파일에 존재하고 PF2에 있는 필드를 사용하려 하므로 JREF 키워드가 지정되어야 합니다.

10 선택/생략 필드는 *Vdrnam*와 *Qorder*입니다.

주: (이들은 두 개의 서로 다른 파일에서 옵니다).

결합 논리 파일 고려사항:

이 주제에서는 결합 논리 파일에 대한 성능 및 무결성 고려사항에 대해 설명합니다. 또한 결합 논리 파일에 대한 규칙 요약이 포함됩니다.

성능 고려사항:

이 주제에는 결합 논리 파일의 최상의 성능에 대한 추가 정보가 포함됩니다.

결합 논리 파일의 성능을 향상시키기 위해서 다음을 수행할 수 있습니다.

- 결합하는 실제 파일의 레코드 수가 서로 다를 경우 가장 적은 레코드를 가진 실제 파일을 먼저 지정하십시오(JOIN 키워드 다음의 첫 번째 매개변수).
- DYNSLT 키워드를 사용하도록 하십시오.
- 결합 논리 파일이 기존의 액세스 경로를 자동으로 공유하도록 결합 논리 파일을 서술하십시오.

주: 결합 논리 파일은 항상 JFLD 키워드에 지정된 필드 쌍 중 두 번째 필드를 사용하는 액세스 경로를 가집니다. 이 필드는 단순 논리 파일의 키 필드와 같은 역할을 합니다. 액세스 경로가 존재하지 않을 경우 액세스 경로는 즉시 유지보수를 통해 내재적으로 작성됩니다.

관련 개념

63 페이지의『동적 선택/생략』

동적 선택/생략 조작의 경우 프로그램이 파일에서 레코드를 읽으면 시스템은 선택/생략 값을 충족시키는 레코드만 리턴합니다. 즉 선택/생략의 실제적인 처리는 레코드가 추가되거나 변경될 때가 아닌, 레코드가 프로그램에 의해 읽힐 때 수행됩니다.

64 페이지의『기존 액세스 경로 사용』

두 개 이상의 파일이 동일한 실제 파일에 기초하고 동일한 키 필드가 동일한 순서를 가진 경우 이 파일들은 동일한 키순 액세스 경로를 자동적으로 공유하게 됩니다. 액세스 경로 공유 시, 액세스 경로 유지에 필요한 시스템 활동량과 파일에 의해 사용되는 보조 기억장치는 줄어듭니다.

자료 무결성 고려사항:

결합 논리 파일을 사용할 경우 자료 무결성을 고려해야 합니다.

결합 논리 파일에 사용하는 실제 파일을 잠그지 않으면 다음과 같은 상황이 발생할 수 있습니다.

- 사용자 프로그램이 2차 파일에 두 개 이상의 레코드가 있는 레코드를 읽습니다. 시스템이 한 개의 레코드를 사용자 프로그램에 제공합니다.
- 사용자 프로그램이 방금 읽은 1차 파일의 레코드를 다른 프로그램이 갱신하여 결합 필드를 변경시킵니다.
- 사용자 프로그램이 또 하나의 읽기를 수행합니다. 시스템은 1차 파일 내의 결합 필드의 현재(새로운) 값에 따라 다음 레코드를 제공합니다.

이와 똑같은 고려사항이 2차 파일에도 적용됩니다.

규칙 요약:

결합 데이터베이스 파일에 대한 규칙의 요약입니다.

필요 조건:

다음은 결합 논리 파일에 대한 기본 필요 조건입니다.

- 각 결합 논리 파일은 다음과 같아야 합니다.

- 한 개의 레코드 형식만 존재하면 JFILE 키워드로 지정합니다.
- JFILE 키워드에 실제 파일명이 최소한 두 개는 지정되어야 합니다. (JFILE 키워드에 지정되는 실제 파일명이 서로 다를 필요는 없습니다.)
- 적어도 한 개의 결합 스펙을 가져야 합니다(JFLD 키워드가 지정된 17열에 J가 있음).
- 최대 255개의 2차 파일
- 필드 레벨에서 N(비입/출력용)이 아닌 다른 용도의 필드명이 적어도 한 개는 있어야 합니다.

- JFILE 키워드에 실제 파일이 두 개만 지정된 경우 JOIN 키워드는 필요하지 않습니다. 결합 스펙이 하나만 있어도 되며 그것으로 두 개의 실제 파일이 결합됩니다.
- JFILE 키워드에 두 개 이상의 실제 파일이 지정되는 경우 다음과 같은 규칙이 적용됩니다.
  - 1차 파일은 첫 번째 JOIN 키워드에 지정된 파일의 쌍 중 첫 번째 파일이어야 합니다. (1차 파일은 다른 JOIN 키워드에 지정된 파일의 쌍 중 첫 번째 파일이 될 수도 있습니다.)

    주: JFILE 키워드에 동일한 파일명이 두 번 지정되었으면, JOIN 키워드 및 JREF 키워드에 상대 파일 번호가 지정되어야 합니다.
  - 모든 2차 파일은 JOIN 키워드의 파일의 쌍 중 두 번째 파일로서 한 번만 지정되어야 합니다. 즉 JFILE 키워드의 모든 2차 파일은 한 개의 결합 스펙에 포함되어야 합니다. (두 개의 2차 파일은 두 개의 결합 스펙을 뜻하며, 세 개의 2차 파일은 세 개의 결합 스펙을 뜻합니다.)
  - 2차 파일이 결합 스펙에 나오는 순서는 JFILE 키워드에 지정된 순서와 같아야 합니다.

결합 필드:

다음은 결합 필드에 대한 규칙입니다.

- 결합하는 모든 실제 파일은 하나 이상의 결합 필드에 의해 다른 실제 파일과 결합되어야 합니다. 결합 필드는 결합 스펙에서 JFLD 키워드의 매개변수 값으로 지정된 필드입니다.
- 결합 필드(JFLD 키워로 지정)는 동일한 속성(길이, 자료 유형, 소수 자릿수)을 가지거나 동일한 속성을 갖도록 결합 논리 파일의 레코드 형식에 다시 정의되어야 합니다. 결합 필드가 문자 유형이면 필드 길이가 다를 수 있습니다.
- 결합 필드를 결합 논리 파일의 레코드 형식에서 지정할 필요는 없습니다(결합 필드의 속성이 같도록 하나 또는 양쪽을 다시 정의해야 하는 경우는 제외).
- 결합 필드를 다시 정의하는 경우 결합 논리 파일을 사용하는 프로그램이 다시 정의된 필드를 사용하지 못하도록 38열에 N을 지정(비입/출력용 필드로 지정)할 수 있습니다.
- 결합 중인 실제 파일에 사용되는 최대 필드 길이는 실제 및 논리 파일의 최대 키 크기와 동일합니다.

관련 참조

9 페이지의『데이터베이스 파일 크기』

iSeries 서버에서 파일을 설계할 경우 데이터베이스 파일 크기를 고려해야 합니다.

결합 논리 파일의 필드:

다음은 결합 논리 파일의 필드에 대한 규칙입니다.

- 결합 논리 파일의 레코드 형식에 있는 필드는 논리 파일이 사용하는 실제 파일들 중 하나에 있어야 하며 CONCAT, RENAME, TRNTBL 또는 SST가 필드에 대해 지정된 경우 이 실제 파일들 중 하나의 필드의 결과여야 합니다.
- CONCAT 키워드에서 매개변수 값으로 지정된 필드는 동일한 실제 파일로부터 온 것이어야 합니다. CONCAT 키워드에서 지정된 첫 번째 필드명이 실제 파일에서 고유하지 않다면, 이 필드에 JREF 키워드를 지정하여 사용하려는 필드 설명이 들어 있는 파일을 식별해야 합니다.
- 결합 논리 파일의 레코드 형식에 필드명이 두 개 이상의 실제 파일에서 지정되는 경우 필드가 어떤 파일로부터 오는지를 JREF 키워드에서 고유하게 지정해 주어야 합니다.
- 키 필드(지정된 경우)는 1차 파일로부터 와야 합니다. 결합 논리 파일의 키 필드가 1차 파일의 키 필드일 필요는 없습니다.
- 선택/생략 필드는 결합 논리 파일이 사용하는 어떤 실제 파일로부터도 올 수 있습니다. 그러나 어떤 환경에서는 DYNSLT 키워드가 필요합니다.
- 지정된 경우 키 필드 및 선택/생략 필드는 레코드 형식에서 정의되어야 합니다.
- 실제 파일명이 JFILE 키워드에서 두 번 이상 지정되는 경우 JOIN 및 JREF 키워드에 상대 파일 번호를 사용해야 합니다.

기타 규칙:

다음은 결합 논리 파일 사용에 대한 기타 규칙입니다.

다음과 같은 규칙이 있습니다.

- 결합 논리 파일은 읽기 전용 파일입니다.
- 결합 레코드 형식은 공유될 수 없으며 다른 레코드 형식을 공유할 수도 없습니다.
- 다음 항목은 결합 논리 파일에 허용되지 않습니다.
  - REFACCPTH 및 FORMAT 키워드
  - 입/출력 필드(38열에 B가 지정됨)

데이터베이스 파일에 대한 액세스 경로 설명

이 주제에서는 데이터베이스 파일에 대한 액세스 경로를 설명하는 여러 가지 방법을 설명합니다.

액세스 경로는 레코드를 검색하는 순서를 서술합니다. 실제 파일 또는 논리 파일의 레코드는 도달순 액세스 경로나 키순 액세스 경로를 사용하여 검색할 수 있습니다. 논리 파일의 경우 각 레코드에서 한 개 이상의 필드 값을 근거로 레코드를 선택 또는 생략할 수 있습니다. 키 필드는 파일 멤버 내에서 특정 유형의 레코드를 배열하는 데 사용되는 필드입니다.

관련 개념

8 페이지의『액세스 경로 설명』

시스템에 데이터베이스 파일을 서술할 때 그 파일의 중요한 두 부분 즉, 레코드 형식과 액세스 경로를 서술하게 됩니다. 액세스 경로는 레코드를 검색하는 순서를 서술합니다. 액세스 경로를 서술하는 경우 키순 액세스 경로인지 도달순 액세스 경로인지를 서술해야 합니다.

데이터베이스 파일에 대한 도달순 액세스 경로 사용

데이터베이스 파일에 대한 도달순 액세스 경로는 레코드가 파일에 도착하여 저장된 순서가 기준이 됩니다. 이 주제에서는 도달순 액세스 경로를 사용할 수 있는 파일의 종류와 이 경로를 사용하여 파일을 설명하는 방법에 대해 설명합니다.

읽기 또는 갱신의 경우 레코드는 다음과 같이 액세스될 수 있습니다.

- 순차적으로 액세스됩니다. 이때 각 레코드는 파일 내의 순차적인 다음 실제 위치로부터 선택됩니다.
- 상대 레코드 번호에 따라 직접 액세스됩니다. 이때 레코드는 파일의 시작으로부터의 위치에 의해 식별됩니다.

외부 서술 파일은 파일의 키 필드가 지정되지 않은 경우 도달순 액세스 경로를 가집니다.

도달순 액세스 경로는 다음 파일의 경우에만 유효합니다.

- 실제 파일
- 논리 파일의 각 멤버가 오직 한 개의 실제 파일 멤버에만 기초를 둔 논리 파일
- 결합 논리 파일
- 뷰

다음과 같은 방법으로 도달순 액세스 경로를 사용할 수 있습니다.

- 입력순은 프로그램이 삭제 레코드가 이미 점유하고 있던 기억장치에 다른 레코드를 위치시켜, 그 기억장치를 사용할 수 있도록 하는 유일한 처리 방법입니다. 이 방법은 사용자가 제공하는 지정된 상대 레코드 번호로 레코드의 명시 삽입이 필요합니다. 시스템이 레코드 삭제에 의해 만들어진 공간을 관리하는 또 다른 방법은 실제 파일에 대해 지정할 수 있는 삭제 레코드 재사용 속성입니다.
- 고급 언어(HLL), DSPPFM(실제 파일 멤버 표시) 명령 및 CPYF(파일 복사) 명령을 통해, 키순 파일을 도착순으로 처리할 수 있습니다. 실제 파일, 하나의 실제 파일 멤버에 근거한 단순 논리 파일 또는 결합 논리 파일에 이와 같은 기능을 사용할 수 있습니다.
- 고급 언어를 사용하여 키순 파일을 상대 레코드 번호에 의해 직접 처리할 수도 있습니다. 실제 파일, 하나의 실제 파일 멤버에 근거한 단순 논리 파일 또는 결합 논리 파일에 이와 같은 기능을 사용할 수 있습니다.
- 도달순 액세스 경로는 기억장치를 추가로 사용하지 않으며 항상 파일과 함께 저장되거나 복원됩니다. (도달순 액세스 경로는 자료가 저장된 실제 순서에 불과하므로 자료를 저장할 때 도달순 액세스 경로가 저장됩니다.)

관련 개념

115 페이지의『삭제 레코드 재사용』
데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

229 페이지의『데이터베이스 레코드 삭제』
삭제 조작을 사용하면 기존의 데이터베이스 레코드를 삭제할 수 있습니다.

데이터베이스 파일에 대한 키순 액세스 경로 사용

데이터베이스 파일에 대한 키순 액세스 경로는 자료 서술 스펙(DDS)에 정의된 키 필드 내용이 기준이 됩니다. 이 주제에서는 파일에 키 필드를 배열하는 방법을 설명합니다.

이러한 유형의 액세스 경로는 레코드가 추가되거나 삭제될 때 또는 레코드가 갱신되고 키 필드의 내용이 변경될 때마다 갱신됩니다. 키순 액세스 경로는 실제 파일과 논리 파일에 모두 유효합니다. 파일의 레코드 순서는 파일이 작성될 때 DDS에서 정의되며 시스템에 의해 자동적으로 관리됩니다.

문자 필드로 정의된 키 필드는 EBCDIC 문자의 정의된 순서에 따라 배열됩니다. 해당 필드에 UNSIGNED(부호 없는 값) 또는 ABSVAL(절대값) DDS 키워드가 지정되지 않은 한, 숫자 필드로 정의된 키 필드는 대수 값을 기초로 배열됩니다. DBCS로 정의된 키 필드는 허용되나 그 비트 표시에 근거하여 한 바이트로만 배열됩니다.

대체 배열 순서를 사용하여 키 필드 배열:

문자 필드로 정의된 키 필드는 EBCDIC 문자순으로 배열되거나 대체 배열 순서에 근거해서 배열될 수도 있습니다.

다음 레코드를 참조하십시오.

레코드	Empname	Deptnbr	Empnbr
1	Jones, Mary	45	23318
2	Smith, Ron	45	41321
3	JOHNSON, JOHN	53	41322
4	Smith, ROBERT	27	56218
5	JONES, MARTIN	53	62213

*Empname*이 키 필드이고 문자 필드이면 EBCDIC 문자순을 사용하여 레코드가 다음과 같이 배열될 수 있습니다.

레코드	Empname	Deptnbr	Empnbr
1	Jones, Mary	45	23318
3	JOHNSON, JOHN	53	41322
5	JONES, MARTIN	53	62213
2	Smith, Ron	45	41321
4	Smith, ROBERT	27	56218

EBCDIC 순서는 대문자에 앞서 소문자를 분류하기 때문에 예기치 않은 분류 순서가 나오는 것에 유의하십시오. 즉 `Smith, ROBERT` 전에 `Smith, Ron`이 정렬되어 있습니다. 대체 배열 순서는 아래와 같이 대/소문자를 사용하여 레코드가 입력될 때 레코드 분류에 사용할 수 있습니다.

레코드	Empname	Deptnbr	Empnbr
3	JOHNSON, JOHN	53	41322
5	JONES, MARTIN	53	62213
1	Jones, Mary	45	23318
4	Smith, ROBERT	27	56218

레코드	Empname	Deptnbr	Empnbr
2	Smith, Ron	45	41321

문자 키 필드를 위한 대체 배열 순서를 사용하기 위해서는 ALTSEQ DDS 키워드와 대체 배열 순서를 포함하고 있는 표 이름을 지정하십시오. 표를 볼 때 표의 각각의 2바이트 자리는 한 문자와 일치하고 있습니다. 문자가 정렬되는 순서를 바꾸려면 해당 2자릿수 값을 정렬될 문자와 같은 값으로 바꾸십시오. 대/소문자 분류에 대해 자세히 알려면 라이브러리 QUSRSYS 안에 있는 QCASE256 표를 참조하십시오.

SRTSEQ 매개변수를 사용하여 키 필드 배열:

SRTSEQ 매개변수로, 가능한 몇 가지 분류 순서에 따라서 문자 자료가 들어 있는 키 필드를 배열할 수 있습니다.

다음 레코드를 참조하십시오.

레코드	Empname	Deptnbr	Empnbr
1	Jones, Marilyn	45	23318
2	Smith, Ron	45	41321
3	JOHNSON, JOHN	53	41322
4	Smith, ROBERT	27	56218
5	JONES, MARTIN	53	62213
6	Jones, Martin	08	29231

Empname 필드가 키 필드이고 문자 필드일 경우 레코드는 *HEX 순서(EBCDIC 순서)에 따라 다음과 같이 배열됩니다.

레코드	Empname	Deptnbr	Empnbr
1	Jones, Marilyn	45	23318
6	Jones, Martin	08	29231
3	JOHNSON, JOHN	53	41322
5	JONES, MARTIN	53	62213
2	Smith, Ron	45	41321
4	Smith, ROBERT	27	56218

*HEX 순서는 모든 소문자가 대문자 앞에 오도록 분류한다는 것에 유의하십시오. 따라서 Smith, Ron은 Smith, ROBERT 앞에 오도록 정렬하고 JOHNSON, JOHN은 대/소문자 Jones 사이에서 정렬합니다. 대/소문자를 섞어 사용한 레코드가 입력될 경우 *LAN GIDSHR 분류 순서를 사용하여 분류할 수도 있습니다. *LANGIDSHR 순서는 소문자와 대문자에 대해 동일한 배열 가중치를 사용하며, 그 결과는 다음과 같습니다.

레코드	Empname	Deptnbr	Empnbr
3	JOHNSON, JOHN	53	41322
1	Jones, Marilyn	45	23318
5	JONES, MARTIN	53	62213
6	Jones, Martin	08	29231
4	Smith, ROBERT	27	56218
2	Smith, Ron	45	41321

*LANGIDSHR 순서는 대문자와 소문자를 동일하게 처리한다는 것에 유의하십시오. 따라서, JONES, MARTIN과 Jones, Martin은 동일하며 원래 파일에 있던 순서대로 정렬됩니다. 특별한 이상이 없는 한, 보고서상에서 모든 소문자 Jones는 대문자 JONES 앞에 오는 것이 보기에 좋습니다. 대소문자가 산발적으로 사용된 레코드가 들어올 경우 *LANGIDUNQ 정렬 순서를 사용하여 정렬할 수 있습니다. *LANGIDUNQ 순서는 소문자와 대문자에 대해 서로 다른 순차적인 배열 가중치를 사용하며, 그 결과는 다음과 같습니다.

레코드	Empname	Deptnbr	Empnbr
3	JOHNSON, JOHN	53	41322
1	Jones, Marilyn	45	23318
6	Jones, Martin	08	29231
5	JONES, MARTIN	53	62213
4	Smith, ROBERT	27	56218
2	Smith, Ron	45	41321

*LANGIDSHR과 *LANGIDUNQ 정렬 순서는 시스템에서 지원하는 모든 언어에 대해서 존재합니다. LANGID 매개변수는 어떤 *LANGIDSHR이나 *LANGIDUNQ 정렬 순서가 사용될지를 결정합니다. SRTSEQ 매개변수는 정렬 순서를 지정하는 데에 사용되고 LANGID 매개변수는 그 언어를 지정하는 데에 사용됩니다.

오름차순 또는 내림차순으로 키 필드 배열:

키 필드는 오름차순이나 내림차순으로 배열할 수 있습니다.

다음 레코드를 참조하십시오.

레코드	Empnbr	Clsnbr	Clsnam	Cpdate
1	56218	412	Welding I	032188
2	41322	412	Welding I	011388
3	64002	412	Welding I	011388
4	23318	412	Welding I	032188
5	41321	412	Welding I	051888
6	62213	412	Welding I	032188

Empnbr 필드가 키 필드일 경우 이 레코드들을 구성하는 방법에는 두 가지가 있을 수 있습니다.

- 오름차순으로, 액세스 경로의 레코드 순서는 다음과 같습니다.

레코드	Empnbr	Clsnbr	Clsnam	Cpdate
4	23318	412	Welding I	032188
5	41321	412	Welding I	051888
2	41322	412	Welding I	011388
1	56218	412	Welding I	032188
6	62213	412	Welding I	032188
3	64002	412	Welding I	011388

- 내림차순으로, 액세스 경로의 레코드 순서는 다음과 같습니다.

레코드	Empnbr	Clsnbr	Clsnam	Cpdate
3	64002	412	Welding I	011388
6	62213	412	Welding I	032188

레코드	Empnbr	Clsnbr	Clsnam	Cpdate
1	56218	412	Welding I	032188
2	41322	412	Welding I	011388
5	41321	412	Welding I	051888
4	23318	412	Welding I	032188

키 필드를 서술할 때 디폴트 값은 오름차순입니다. 그러나 DESCEND DDS 키워드를 사용하여 키 필드를 내림차순으로 배열하도록 지정할 수 있습니다.

둘 이상의 키 필드 사용:

두 개 이상의 키 필드를 사용하여 데이터베이스 파일의 레코드를 배열할 수 있습니다. 이 키 필드들을 같은 순서로 사용할 필요는 없습니다.

예를 들어 두 개의 키 필드를 사용하는 경우 하나의 필드는 오름차순으로, 다른 필드는 내림차순으로 사용할 수 있습니다. 다음 레코드를 참조하십시오.

레코드	Order	Ordate	Line	Item	Qtyord	Extens
1	52218	063088	01	88682	425	031875
2	41834	062888	03	42111	30	020550
3	41834	062888	02	61132	4	021700
4	52218	063088	02	40001	62	021700
5	41834	062888	01	00623	50	025000

액세스 경로로 *Order* 필드와 *Line* 필드를 오름차순의 키 필드로 사용하는 경우 액세스 경로의 레코드 순서는 다음과 같습니다.

레코드	Order	Ordate	Line	Item	Qtyord	Extens
5	41834	062888	01	00623	50	025000
3	41834	062888	02	61132	4	021700
2	41834	062888	03	42111	30	020550
1	52218	063088	01	88682	425	031875
4	52218	063088	02	40001	62	021700

액세스 경로로 *Order*를 오름차순의 키 필드로 사용하고, *Line* 필드를 내림차순으로 사용하는 경우 액세스 경로의 레코드 순서는 다음과 같습니다.

레코드	Order	Ordate	Line	Item	Qtyord	Extens
2	41834	062888	03	42111	30	020550
3	41834	062888	02	61132	4	021700
5	41834	062888	01	00623	50	025000
4	52218	063088	02	40001	62	021700
1	52218	063088	01	88682	425	031875

레코드가 동일한 파일 내의 다른 레코드의 키 필드와 같은 키 필드를 가진 경우 그 파일에는 중복 키 값을 가진 레코드가 있다고 말합니다. 그러나 중복 키 값으로 불리우려면 레코드 내의 모든 키 필드들이 중복되어야 합니다. 예를 들어 한 레코드 형식에 *Order*와 *Ordate*라는 두 개의 키 필드가 있는 경우 중복 키 값은 *Order*

필드와 *Ordate* 필드의 내용이 두 개 이상의 레코드에서 같을 때 발생합니다. 다음 레코드에는 중복 키 값이 있습니다.

Order	Ordate	Line	Item	Qtyord	Extens
41834	062888	03	42111	30	020550
41834	062888	02	61132	04	021700
41834	062888	01	00623	50	025000

중복 키를 없애기 위해서 세 번째 키 필드로 *Line* 필드를 파일에 정의합니다.

(첫 번째 키 필드)	(두 번째 키 필드)				
order	**ordate**	(세 번째 키 필드) **line**	**Item**	**Qtyord**	**Extens**
41834	062888	03	42111	30	020550
41834	062888	02	61132	04	021700
41834	062888	01	00623	50	025000

두 개 이상의 레코드 형식을 가진 논리 파일은 비록 그 레코드 형식이 서로 다른 실제 파일에 기초를 두고 있더라도 중복 키 값을 가진 레코드를 가질 수 있습니다. 다시 말해서 서로 다른 레코드 형식에서 온 값이라 해도 그 키 값은 중복 키 값으로 간주됩니다.

중복 키 값 방지:

iSeries용 DB2 Universal Database는 파일에 중복 키 값이 있는 레코드를 사용합니다. 그러나 일부 파일에서는 중복 키 값을 허용하지 않을 수도 있습니다.

예를 들어 고객 번호 필드를 키 필드로서 정의하는 파일을 작성하는 것으로 가정하십시오. 이 경우 시스템이 파일의 각 레코드에 고유한 고객 번호를 지정하도록 할 수 있습니다.

자료 서술 스펙(DDS)에 UNIQUE 키워드를 지정하면 파일 내에서 중복 키 값을 방지할 수 있습니다. UNIQUE 키워드가 지정되면 레코드의 키 값이 파일에 이미 존재하고 있는 다른 레코드의 키 값과 같을 경우 그 레코드는 파일에 입력되거나 복사될 수 없습니다. 고유 키의 무결성을 강화하기 위해 고유 제한조건을 사용할 수도 있습니다.

중복 키 값을 가진 레코드들이 이미 실제 파일에 있을 경우 연관된 논리 파일에는 UNIQUE 키워드를 지정할 수 없습니다. UNIQUE 키워드를 지정하여 논리 파일을 작성할 경우에 연관된 실제 파일에 중복 키 값이 있으면 논리 파일이 작성되지 않습니다. 시스템은 이 사실을 알리는 메세지와 중복 키 값이 들어 있는 레코드(최대 20개)를 나타내는 메세지를 보냅니다.

파일에 UNIQUE 키워드가 지정되는 경우 새 레코드를 추가하는 데 사용되는 파일이 무엇인지에 관계없이 파일에 추가되는 레코드는 기존 레코드의 키 값과 중복되는 키 값을 가질 수 없습니다. 예를 들어 2개의 논리 파일 LF1과 LF2가 실제 파일 PF1에 기초를 두고 있는 것으로 가정하십시오. 그리고 LF1에는 UNIQUE가 지정되어 있습니다. LF2를 사용하여 PF1에 레코드를 추가할 경우 LF1에 중복 키 값을 갖는다면 그 레코드는 추가될 수 없습니다.

키 필드 중에 널값을 허용하는 필드가 있을 경우 이 필드에 삽입되는 널값은 파일 작성 시 액세스 경로 정의 방법에 따라 중복을 일으킬 수도 있고 일으키지 않을 수도 있습니다. UNIQUE 키워드의 *INCNULL 매개변수는 고유 액세스 경로에 중복이 존재하는지 판별할 때 널값이 포함되어 있는 것을 나타냅니다. *EXCNULL 매개변수는 중복이 존재하는지 판별할 때 널값이 포함되지 않은 것을 나타냅니다.

다음은 고유한 키 값을 필요로 하는 논리 파일에 대한 DDS를 표시하는 예입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* ORDER TRANSACTION LOGICAL FILE (ORDFILL)
          A                                     UNIQUE
          A           R ORDHDR                  PFILE(ORDHDRP)
          A           K ORDER
     A
          A           R ORDDTL                  PFILE(ORDDTLP)
          A           K ORDER
          A           K LINE
     A
```

이 예에서 키 필드의 내용(ORDHDR 레코드 형식의 *Order* 필드, ORDDTL 레코드 형식의 *Order* 및 *Line* 필드)은 ORDHDRP 파일, ORDDTLP 파일 또는 여기에 정의된 논리 파일을 통해 레코드가 추가되었는지에 관계없이 고유해야 합니다ORDDTL 레코드 형식 내의 두 번째 키 필드가 *Line* 필드로 지정되어 있을 때 두 개의 실제 파일의 *Order* 키 필드에는 동일한 값이 존재할 수 있습니다. 실제 파일 ORDDTLP는 두 개의 키 필드를 가지고 있고 실제 파일 ORDHDRP는 한 개의 키 필드만 가지고 있기 때문에 두 파일의 키 값이 혼동되지 않습니다.

관련 개념

284 페이지의『제한조건으로 데이터베이스 무결성 제어』

*제한조건*은 파일에 지정된 제한 또는 한계를 의미하며 레코드 추가, 변경 및 제거 시 데이터베이스의 자료에 일관성이 있는지 확인할 수 있습니다. 이 주제에서는 자료 일관성 확인을 위해 제한조건을 사용하는 방법에 대해 설명합니다.

DDS 개념

중복 키 배열:

자료 서술 스펙(DDS)에 UNIQUE 키워드를 지정하지 않을 경우 시스템에서 중복 키 값이 있는 레코드를 저장하는 방법을 지정할 수 있습니다(중복 키 값이 있을 경우).

다음 중 한 방법으로 중복 키 값을 가진 레코드를 액세스 경로에 저장되도록 지정할 수 있습니다.

- LIFO(Last-in-first-out: 후입선출). LIFO 키워드를 지정하면(**1**) 중복 키 값이 있는 레코드가 실제 레코드 순서에 따라 LIFO순으로 검색됩니다. 다음은 LIFO 키워드를 사용한 DDS의 예입니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
           A* ORDERP2
           A                                1  LIFO           A          R ORDER2
          A            .
          A            .
          A            .
          A          K ORDER
     A
```

- FIFO(First-in-first-out: 선입선출). FIFO 키워드를 지정하면 중복 키 값이 있는 레코드는 레코드의 실제 순서에 따라 FIFO순으로 검색됩니다.
- FCFO(First-changed-first-out: 선변경선출). FCFO 키워드를 지정하면 중복 키 값이 있는 레코드는실제 키 순서에 따라 FCFO순으로 검색됩니다.
- 중복 키에 대해서는 특정 순서가 지정되지 않는 경우(디폴트 값). FIFO, FCFO 또는 LIFO 키워드를 지정하지 않으면 중복 키가 있는 레코드 검색 시 정해진 순서가 없습니다. 중복 키 필드에 특정한 순서가 없으면 더 많은 액세스 경로의 공유를 제공하여 성능을 향상시킬 수 있습니다.

단순 또는 복수 형식 논리 파일이 여러 실제 파일 멤버를 기반으로 할 경우, CRTLF(논리 파일 작성) 명령이나 ADDLFM(논리 파일 멤버 추가) 명령의 DTAMBRS 매개변수에 파일 및 멤버가 지정되는 순서대로 중복 키 값이 있는 레코드가 읽힙니다. 하나 이상의 레코드 형식이 있는 논리 파일의 예는 DDS 개념에서 볼 수 있습니다.

중복 키 값이 있는 레코드의 LIFO 또는 FIFO 순서는 키 필드의 내용에 대한 갱신 순서로 결정되는 것이 아니라 파일 멤버에 있는 레코드의 실제 순서에 의해서만 결정됩니다. 실제 파일에 FIFO 키워드가 지정되었다고 가정하고(중복 키가 있는 레코드는 FIFO 순서임) 다음 표에서 레코드가 파일에 추가되는 순서를 표시하는 것으로 가정하십시오.

파일에 레코드가 추가된 순서	키 값
1	A
2	B
3	C
4	C
5	D

액세스 경로 순서는 다음과 같습니다(FIFO, 오름차순 키).

레코드 번호	키 값
1	A
2	B
3	C
4	C
5	D

중복 키 값을 가진 레코드 3과 4는 FIFO 순서입니다. 즉, 레코드 3이 레코드 4보다 먼저 파일에 추가되었으므로 레코드 3이 레코드 4보다 먼저 읽힙니다. 이것은 내림차순으로 레코드를 읽는 경우 분명해집니다. 이는 이 실제 파일에 기초를 둔 논리 파일을 작성하고, 그 논리 파일에 DESCEND 키워드를 지정함으로써 가능합니다.

액세스 경로 순서는 다음과 같습니다(FIFO, 내림차순 키).

레코드 번호	키 값
5	D
3	C
4	C

레코드 번호	키 값
2	B
1	A

실제 레코드 1의 키 값을 C로 변경하면 실제 파일의 액세스 경로 순서는 다음과 같습니다(FIFO, 오름차순 키).

레코드 번호	키 값
2	B
1	C
3	C
4	C
5	D

마지막으로 내림차순으로 변경하는 경우 논리 파일에 대한 액세스 경로의 새로운 순서는 다음과 같습니다(FIFO, 내림차순 키).

레코드 번호	키 값
5	D
1	C
3	C
4	C
2	B

변경 이후, 키 필드의 내용이 레코드 4가 추가된 후 갱신되었음에도 불구하고 레코드 1은 레코드 4 뒤에 나오지 않습니다.

중복 키 값을 가진 FCFO 순서는 키 값의 내용이 갱신된 순서에 따라 결정됩니다. 위의 예에서 키 값이 C가 되도록 레코드 1을 변경하면 액세스 경로 순서는 다음과 같습니다(FCFO, 오름차순 키만 해당).

레코드 번호	키 값
2	B
3	C
4	C
1	C
5	D

FCFO의 경우 중복 키의 순서는 FCFO 액세스 경로가 재작성되거나 롤백 조작이 수행될 때 변경될 수 있습니다. 어떤 경우에는 키 필드는 변경되지만 실제 키는 변경되지 않을 수 있습니다. 이러한 경우 키 필드는 변경되어도 FCFO 순서는 변경되지 않습니다. 예를 들어 색인 순서가 키의 절대값에 따르도록 변경될 경우 FCFO 순서는 변경되지 않습니다. 키가 음수에서 양수로 바뀌더라도 실제 키 값은 변경되지 않습니다. 실제 키가 변경되지 않기 때문에 FCFO 순서는 변경되지 않습니다.

삭제 레코드 재사용 속성이 실제 파일에 지정되면 중복 키 순서는 디폴트 값을 허용하거나 FCFO여야 합니다. 파일의 키 순서가 FIFO 또는 LIFO로 설정되었거나 실제 파일 상에 정의된 논리 파일의 중복 키 순서가 FIFO 또는 LIFO일 경우 실제 파일에는 삭제 레코드 재사용 속성이 허용되지 않습니다.

관련 개념

64 페이지의『기존 액세스 경로 사용』

두 개 이상의 파일이 동일한 실제 파일에 기초하고 동일한 키 필드가 동일한 순서를 가진 경우 이 파일들은 동일한 키순 액세스 경로를 자동적으로 공유하게 됩니다. 액세스 경로 공유 시, 액세스 경로 유지에 필요한 시스템 활동량과 파일에 의해 사용되는 보조 기억장치는 줄어듭니다.

기존 액세스 경로 스펙 사용

자료 서술 스펙(DDS)의 REFACCPTH 키워드를 사용하면 다른 파일의 액세스 경로 스펙을 사용할 수 있습니다. 파일 작성 시, 시스템은 공유할 액세스 경로를 결정합니다.

REFACCPTH 키워드를 사용하는 파일은 REFACCPTH 키워드에 지정된 파일의 액세스 경로를 공유할 필요는 없습니다. REFACCPTH 키워드는 지정해야 할 DDS문의 수를 줄이기 위해서만 사용됩니다. 즉 파일에 대한 키 필드 스펙을 코딩하는 대신, REFACCPTH 키워드를 지정할 수 있습니다. 파일이 작성될 때 시스템이 REFACCPTH 키워드에서 지정된 파일로부터 선택/생략 스펙 및 키 필드를 작성중인 파일에 복사합니다.

데이터베이스 파일 액세스 경로에서 부동 소수점 필드 사용

키순 데이터베이스 파일의 레코드 배열 순서는 자료 서술 스펙(DDS)의 SIGNED, UNSIGNED 및 ABSVAL 키워드에 의해 결정됩니다. 부동 소수점 필드의 경우 가장 왼쪽 비트는 부호, 그 다음은 지수, 그리고 유효부는 맨 끝에 오게 됩니다.

UNSIGNED가 지정되었을 때의 배열 순서는 다음과 같습니다.

- 양의 실수--양으로 무한대
- 음의 실수--음으로 무한대

DDS에서 SIGNED 키워드가 지정되었거나 디폴트 값으로 지정된 부동 소수점 키 필드에는 대수 순서(algebraic numeric sequence)가 지정됩니다. 배열 순서는 음으로 무한대-실수-양으로 무한대입니다.

DDS에서 ABSVAL 키워드가 지정된 부동 소수점 키 필드에는 절대값 순서가 지정됩니다.

다음과 같은 부동 소수점 배열 순서가 지켜집니다.

- 0(양 또는 음)은 다른 양/음의 실수와 같은 방법으로 배열됩니다.
- SIGNED 순서에서 −0은 +0 전에 배열됩니다.
- ABSVAL 순서에서 −0과 +0은 같은 배열순입니다.

키 필드에서는 비숫자(*NAN)값을 사용할 수 없습니다. 이를 시도하여 파일 작성 중 키 필드에서 이러한 *NAN 값이 감지될 경우 그 파일은 작성되지 않습니다.

데이터베이스 보안

이 주제에서는 데이터베이스 보안을 위해 취할 수 있는 조치에 대해 설명합니다.

관련 개념

보안 참조서 PDF

파일 및 자료 권한 부여

파일 및 자료 권한을 부여할 한 가지 방법을 선택하십시오.

- iSeries Navigator를 사용하여 사용자나 그룹에 권한부여할 수 있습니다.
- GRTOBJAUT(오브젝트 권한 부여) 명령을 사용하여 사용자가 데이터베이스 파일의 자료에 액세스할 수 있는 권한을 지정할 수 있습니다.
- SQL GRANT 명령문을 사용할 수 있습니다.

  관련 참조

  GRTOBJAUT(오브젝트 권한부여) 명령

  SQL GRANT

iSeries Navigator를 사용하여 사용자나 그룹에 권한 부여:

일부 사용자는 공용 권한에서 허용하는 권한이 아닌 오브젝트에 대한 다른 권한을 요구할 수 있습니다. 이 주제에서는 iSeries Navigator를 사용하여 사용자나 그룹에 권한을 부여하는 방법을 설명합니다.

오브젝트에 대해 사용자나 그룹의 권한을 부여하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. 허용 권한을 편집하려는 오브젝트가 보일 때까지 검색하십시오.
3. 허용 권한을 추가할 오브젝트를 마우스 오른쪽 버튼으로 클릭하고 **허용 권한**을 클릭하십시오.
4. 허용 권한 창에서 **추가**를 클릭하십시오.
5. 추가 창에서 사용자와 그룹을 하나 이상 선택하거나 사용자나 그룹 이름 필드에 사용자나 그룹 이름을 입력하십시오.
6. **확인**을 클릭하십시오. 이렇게 하면 리스트 맨 위에 사용자나 그룹이 추가됩니다.

주: 오브젝트에 대한 디폴트 권한이 사용자나 그룹에 부여됩니다. 시스템이 정의한 유형 중 하나로 사용자 권한을 변경하거나 권한을 사용자 정의할 수 있습니다.

iSeries Navigator를 사용하여 권한을 제거하고 사용자 정의할 수도 있습니다.

오브젝트 권한 유형:

다음은 데이터베이스 파일에 사용자 액세스를 허용하는 오브젝트 권한 유형 목록입니다.

오브젝트 조작 권한

다음을 수행하려면 오브젝트 조작 권한이 필요합니다.

- 처리를 위한 파일 열기(적어도 하나의 자료 권한이 있어야 합니다.)
- 파일 설명을 사용하는 프로그램의 컴파일
- 파일의 활동 멤버에 관한 서술적 정보 표시

- 조회 처리를 위한 파일 열기. 예를 들어 OPNQRYF(조회 파일 열기) 명령은 조회 처리를 위해 파일을 엽니다.

주: 열기 조작에서 지정 옵션이 요구하는 적절한 자료 권한도 있어야 합니다.

오브젝트 존재 권한

다음을 수행하려면 오브젝트 존재 권한이 필요합니다.

- 파일 삭제
- 파일 기억장치의 저장, 복원 및 해제. 사용자에게 오브젝트 존재 권한이 명시적으로 부여되지 않은 경우 *SAVSYS 특수 사용자 권한이 사용자가 파일의 기억장치를 저장, 복원 및 해제할 수 있도록 합니다. *SAVSYS는 오브젝트 존재 권한과는 다릅니다.
- 파일로부터의 멤버 제거
- 파일 소유권 이전

주: 저장/복원을 제외한 위의 모든 기능에는 파일에 대한 오브젝트 조작 권한도 필요합니다.

오브젝트 관리 권한

다음을 수행하려면 오브젝트 관리 권한이 필요합니다.

- 키순 액세스 경로를 가진 논리 파일 작성(논리 파일에 의해 참조된 실제 파일에 대해 오브젝트 관리 권한이 필요합니다.)
- 권한 부여 및 취소. 이미 가지고 있는 권한만 부여하거나 취소할 수 있습니다. (파일에 대한 오브젝트 조작 권한도 있어야 합니다.)
- 파일 변경
- 파일에 멤버 추가(파일의 소유자가 새로운 멤버의 소유자가 됩니다.)
- 파일 멤버 변경
- 파일 이동
- 파일 재명명
- 파일 멤버 재명명
- 파일 멤버 지우기(자료 삭제 권한도 필요합니다.)
- 파일 멤버 초기화(또한 디폴트 레코드로 초기화하려면 자료 추가 권한이 필요하고, 삭제 레코드로 초기화하려면 자료 삭제 권한이 필요합니다.)
- 파일 멤버 재구성(모든 자료 권한이 필요합니다.)

오브젝트 변경 권한

오브젝트 관리 권한과 같은 여러 조작(앞 페이지 참조)의 경우 오브젝트 변경 권한이 필요합니다. 오브젝트 변경 권한은 오브젝트 관리 권한의 대체 권한입니다.

오브젝트 참조 권한

오브젝트 참조 권한은 오브젝트에서의 조작이 참조 오브젝트에 의해 제한될 수 있는 다른 오브젝트에서 해당 오브젝트를 참조하는 데 사용합니다.

실제 파일 참조 제한조건을 추가하면 상위 파일에 대해 오브젝트 관리 권한이나 오브젝트 참조 권한이 검사됩니다.

관련 개념

284 페이지의 『제한조건으로 데이터베이스 무결성 제어』

*제한조건*은 파일에 지정된 제한 또는 한계를 의미하며 레코드 추가, 변경 및 제거 시 데이터베이스의 자료에 일관성이 있는지 확인할 수 있습니다. 이 주제에서는 자료 일관성 확인을 위해 제한조건을 사용하는 방법에 대해 설명합니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』

이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

자료 권한 유형:

다음은 실제 및 논리 파일에 사용자 액세스를 허용하는 데 사용할 수 있는 자료 권한 또는 허용 목록입니다.

읽기 권한

파일의 레코드를 읽을 수 있습니다.

추가 권한

파일에 새로운 레코드를 추가할 수 있습니다.

갱신 권한

기존의 레코드를 갱신할 수 있습니다. (갱신하기 위해 레코드를 읽으려면 읽기 권한도 있어야 합니다.)

삭제 권한

기존의 레코드를 삭제할 수 있습니다. (삭제하기 위해 레코드를 읽으려면 읽기 권한도 있어야 합니다.)

실행 권한

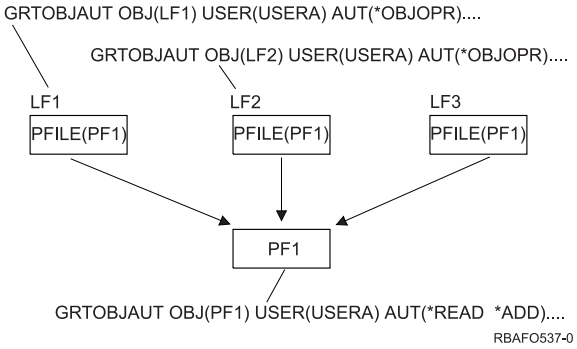
실행 권한을 사용하여 라이브러리에 대해 작업하고 프로그램을 시작할 수 있습니다. 예를 들어, 트리거와 연관된 파일을 변경하려면 트리거 프로그램에 대한 실행 권한이 있어야 합니다. 실행 권한이 없으면, 시스템이 트리거 프로그램을 시작하지 않습니다.

파일의 자료에 대한 권한은 보통 입출력 조작을 실제로 수행하기 전까지는 검증되지 않습니다. 그러나 OPNQRYF(조회 파일 열기) 명령 및 OPNDBF(데이터베이스 파일 열기) 명령은 파일이 열릴 때 자료 권한을 검증합니다.

파일 사용자에게 오브젝트 조작 권한이 허용되지 않을 경우 사용자가 파일을 열 수 없습니다.

다음 예는 논리 파일에 대해 부여된 권한과 그 논리 파일이 사용하는 실제 파일에 대해 부여된 권한 사이의 관계를 보여줍니다. 논리 파일 LF1, LF2 및 LF3은 실제 파일 PF1을 기초로 하고 있습니다. USERA가 PF1의 자료에 대해서는 읽기(*READ) 권한 및 추가(*ADD) 권한을, LF1과 LF2에 대해서는 오브젝트 조작(*OBJOPR) 권한, 읽기(*READ) 권한 및 추가(*ADD) 권한을 갖고 있습니다. 이는 USERA가 어떤 방법으로도 PF1을 열거나 그 자료를 직접 사용할 수 없음을 의미합니다. 왜냐하면 사용자가 PF1에 대해 오브젝트 조작 권한(*OBJOPR)을 갖고 있지 *않기 때문입니다.* USERA는 LF1을 열 수 있으며 LF1과 LF2를 통해 PF1로부터 레코드를 읽고 PF1에 레코드를 추가할 수 있습니다.

주: 사용자에게 LF3에 대한 권한이 허용되지 않았으므로 이를 사용할 수 없습니다.



관련 개념

306 페이지의 『데이터베이스의 자동 이벤트 트리거』

*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

공용 권한 지정

파일 작성 시 공용 권한을 지정하고 부여할 수 있습니다. 공용 권한에 대해 지정할 수 있는 값과 이를 부여하는 방법에 대해 읽어 보십시오.

CRTPF(실제 파일 작성) 또는 CRTSRCPF(소스 실제 파일 작성) 명령에서 AUT 매개변수를 통해 공용 권한을 지정할 수 있습니다. *공용 권한*은 파일에 대해 특정 권한을 갖고 있지 않은 사용자나 파일에 대해 특정 권한을 갖고 있는 그룹의 멤버가 아닌 사용자가 사용할 수 있는 권한입니다. 공용 권한은 마지막으로 검사되는 권한입니다. 즉 사용자가 파일에 대해 특정 권한을 갖고 있거나 사용자가 특정 권한을 가진 그룹의 한 멤버일 경우 공용 권한은 검사되지 않습니다. 공용 권한은 다음과 같이 지정할 수 있습니다.

- *LIBCRTAUT. 파일이 작성되는 라이브러리는 파일이 작성될 때 파일의 공용 권한을 결정하기 위해 검사됩니다. 권한은 각 라이브러리와 연관되어 있습니다. 이 권한은 라이브러리가 작성될 때 지정되고, 라이브러리에 작성된 모든 파일은 파일 작성(CRTLF, CRTPF 및 CRTSRCPF) 명령의 AUT 매개변수에 대해 *LIBCRTAUT 값이 지정될 때 이 공용 권한이 주어집니다. *LIBCRTAUT 값이 디폴트 공용 권한입니다.
- *CHANGE. 파일에 대해 특정 사용자 권한이나 그룹 권한을 갖고 있지 않은 모든 사용자가 파일의 자료를 변경할 수 있는 권한을 가집니다.

- *USE. 파일에 대해 특정한 사용자 권한이나 그룹 권한을 갖고 있지 않은 모든 사용자가 파일의 자료를 읽을 수 있는 권한을 가집니다.
- *EXCLUDE. 소유자, 보안 담당자, 특정 권한을 가진 사용자 또는 특정 권한을 가진 그룹의 멤버인 사용자만이 파일을 사용할 수 있습니다.
- *ALL. 파일에 대해 특정 사용자 권한이나 그룹 권한을 갖고 있지 않은 모든 사용자가 오브젝트 조작 권한, 오브젝트 관리 권한 및 오브젝트 존재 권한과 함께 모든 자료 권한을 가집니다.
- 권한부여 리스트명. 권한부여 리스트는 사용자와 그들이 가진 권한이 나열된 리스트입니다. 이 리스트로 사용자와 각자 가지고 있는 서로 다른 권한들을 함께 그룹화시킬 수 있습니다.

주: 논리 파일 작성 시 자료 권한은 부여되지 않습니다. 그 결과 *CHANGE는 *USE와 같게 되며 *ALL은 어떤 자료 권한도 부여하지 않습니다.

다음과 같은 방법으로 공용 권한을 부여할 수 있습니다.

- iSeries Navigator를 사용하여 공용 권한을 정의하십시오.
- EDTOBJAUT(오브젝트 권한 편집), GRTOBJAUT(오브젝트 권한 부여) 또는 RVKOBJAUT(오브젝트 권한 철회) 명령을 사용하여 파일의 공용 권한을 부여하거나 철회하십시오.

iSeries Navigatorr를 사용하여 새 파일에 디폴트 공용 권한을 설정할 수도 있습니다.

관련 참조

실제 파일 작성(CRTPF) 명령

소스 실제 파일 작성(CRTSRCPF) 명령

오브젝트 권한 편집(EDTOBJAUT) 명령

GRTOBJAUT(오브젝트 권한부여) 명령

RVKOBJAUT(오브젝트 권한 호출) 명령

iSeries Navigator를 사용하여 파일에 대한 공용 권한 정의:

공용 권한은 사용자가 오브젝트에 대해 특정 액세스가 없을 경우 이 사용자가 오브젝트에 대해 갖는 액세스 유형을 설명하기 위해 시스템의 모든 오브젝트에 대해 정의됩니다. 이 주제에서는 iSeries Navigator를 사용하여 파일에 대한 공용 권한을 정의하는 방법을 표시합니다.

공용 권한을 정의하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. 허용 권한을 편집하려는 오브젝트가 보일 때까지 검색하십시오.
3. 허용 권한을 추가할 오브젝트를 마우스 오른쪽 버튼으로 클릭하고 **허용 권한**을 클릭하십시오.
4. 허용 권한 창의 그룹 리스트에서 **공용**을 선택하십시오.
5. 세부 허용 권한을 구현하려면 **세부사항** 버튼을 클릭하십시오해당 상자를 확인하여 공용 권한에 필요한 허용 권한을 적용하십시오.
6. **확인**을 클릭하십시오.

iSeries Navigator를 사용하여 새 파일에 디폴트 공용 권한 설정:

디폴트 공용 권한을 설정하면 라이브러리에서 새로운 파일이 작성될 때 모든 새 오브젝트에 할당된 일반적인 권한을 가지도록 허용합니다. 여러 가지 보안 레벨을 요구하는 각 오브젝트에 대한 허용을 편집할 수 있습니다. iSeries Navigator를 사용하여 새 파일에 디폴트 공용 권한을 설정하려면 다음 단계를 수행하십시오.

디폴트 공용 권한을 설정하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 라이브러리를 확장하십시오.
4. 공용 권한을 설정하려는 라이브러리를 마우스 오른쪽 버튼으로 클릭하고 **허용 권한**을 클릭하십시오.
5. 허용 권한 창에서 **새 오브젝트**를 클릭하십시오.
6. 새 오브젝트 창에서 디폴트 공용 권한을 선택하십시오.
7. 권한부여 리스트를 할당하기 위해 권한부여 리스트의 이름을 입력하거나 **찾아보기**를 사용할 수 있습니다. 권한부여 리스트 등록정보를 보려면 **열기**를 클릭하십시오.
8. **확인**을 클릭하십시오.

시작 시스템 값
새 오브젝트의 디폴트 공용 권한에 대해 시스템 값을 사용하도록 지정합니다.

사용
오브젝트 속성에 대한 액세스와 오브젝트의 사용을 허용합니다. 공용은 오브젝트를 볼 수 있지만 변경할 수 없습니다.

변경
오브젝트의 내용을(예외 있음) 변경시킵니다.

모두
소유자에게 제한된 조작을 제외하고, 오브젝트에 대한 모든 조작을 허용합니다. 사용자나 그룹이 오브젝트 존재를 제어하고, 오브젝트에 대한 보안을 지정하고, 오브젝트를 변경하고, 오브젝트에 대한 기본 기능을 수행할 수 있습니다. 또한 사용자나 그룹은 오브젝트의 소유권을 변경할 수 있습니다.

제외
오브젝트에 대한 모든 조작이 금지됩니다. 이 허용을 가지는 사용자나 그룹의 어떠한 액세스나 조작도 오브젝트에 대해 허용되지 않습니다. 공용은 오브젝트를 사용하도록 허용되지 않음을 지정합니다.

권한부여 리스트 사용
오브젝트를 보안하기 위해 사용할 권한부여 리스트를 지정하도록 허용합니다.

데이터베이스 파일 기능을 사용하여 I/O 조작 제어

실제 파일을 작성할 경우 파일 기능을 지정하여 데이터베이스 파일 권한과는 독립적으로 데이터베이스 파일에 허용되는 입/출력(I/O) 조작을 제어할 수 있습니다.

CRTPF(실제 파일 작성) 및 CRTSRCPF(소스 실제 파일 작성) 명령에서 ALWUPD 및 ALWDLT 매개변수를 사용하여 파일의 갱신 가능 및 삭제 가능 여부를 지정할 수 있습니다. 갱신 및 삭제가 가능하지 않은 파일을 작성하면 자료가 기록된 후 자료가 파일에서 변경되거나 삭제될 수 없는 환경을 효과적으로 강제할 수 있습니다.

파일 기능은 논리 파일에 대해 명시적으로 설정될 수 없습니다. 논리 파일의 파일 기능은 기초가 되는 실제 파일의 파일 기능에 의해 결정됩니다.

파일이 작성된 후에는 파일 기능을 변경할 수 없습니다. 파일을 삭제한 다음 원하는 기능이 있는 파일로 다시 작성해야 합니다. DSPFD(파일 설명 표시) 명령을 사용하여 파일 기능을 판별할 수 있습니다.

관련 참조

실제 파일 작성(CRTPF) 명령

소스 실제 파일 작성(CRTSRCPF) 명령

파일 설명 표시(DSPFD) 명령

데이터베이스 파일의 특정 필드에 대한 액세스 제한

실제 데이터베이스 파일의 특정 필드에 대해 사용자 갱신 및 읽기 요청을 제한하는 방법 중 하나를 선택하십시오.

- 사용자에게 액세스를 허용할 필드만 포함하는 실제 파일의 논리 뷰를 작성합니다.
- SQL GRANT문을 사용하여 SQL(Structured Query Language) 표의 특정 열에 갱신 권한을 부여합니다.

관련 개념

『자료 보안을 위해 논리 파일 사용』

이 주제에서는 논리 파일을 사용하여 자료를 보안하는 방법에 대해 설명합니다.

SQL 프로그래밍

자료 보안을 위해 논리 파일 사용

이 주제에서는 논리 파일을 사용하여 자료를 보안하는 방법에 대해 설명합니다.

논리 파일을 사용하여 실제 파일의 필드가 보이지 않게 할 수 있습니다. 이는 사용자에게 보여주지 않을 필드는 포함시키지 않도록 논리 파일 레코드 형식을 서술함으로써 가능합니다.

논리 파일을 사용하여 보호하려는 필드에 대해 DDS 코딩 양식의 38열에 I(입력 전용)를 지정하면 실제 파일의 하나 이상의 필드들이 변경되는 것을 방지할 수 있습니다.

논리 파일을 사용하여 한 개 이상의 필드 내용에 있는 실제 파일의 레코드를 보안할 수 있습니다. 필드 내용을 근거로 레코드를 보안하려면 논리 파일 서술 시 선택 및 생략 키워드를 사용하십시오.

관련 개념

52 페이지의『논리 파일 레코드 형식 설명』

자료 서술 스펙(DDS)으로 서술되는 각 논리 파일 레코드 형식의 경우 레코드 형식명과 PFILE 키워드(단순 및 복수 형식 논리 파일의 경우) 또는 JFILE 키워드(결합 논리 파일의 경우)를 지정해야 합니다.

54 페이지의 『논리 파일의 필드 사용 서술』
논리 파일의 필드를 입력 전용, 입/출력용 또는 비입/출력용으로 지정할 수 있습니다.

59 페이지의 『논리 파일을 사용하여 레코드 선택 및 생략』
시스템은 논리 파일 사용 시 레코드를 선택하고 생략할 수 있습니다. 이로 인해 처리상의 편의나 보안을 위해 파일 내의 레코드를 제외시킬 수 있습니다.

데이터베이스 파일 처리

이 주제에서는 i5/OS 오퍼레이팅 시스템에서 보다 효율적으로 데이터베이스 파일을 처리하는 방법, 데이터베이스 파일을 열고, 조작하고, 닫는 방법 및 데이터베이스 파일과 관련된 오류 메세지를 모니터링하고 관리하는 방법에 대해 설명합니다.

데이터베이스 파일 처리: 런타임 고려사항

이 주제에서는 데이터베이스 파일 처리에 영향을 줄 수 있는 고려사항이나 기타 방법 및 파일 처리 매개변수에 대해 설명합니다.

처리하기 위해 파일을 열기 전에 프로그램 및 작업에서 파일을 어떻게 사용할 것인지를 고려해야 합니다. 런타임 파일 처리 매개변수에 대해 잘 알고 있으면 예기치 않은 결과를 피할 수 있고 프로그램 성능을 향상시킬 수 있습니다. 매개변수 값은 고급 언어 프로그램, 파일 속성 및 고급 언어 프로그램 호출 이전에 처리된 열기 또는 대체 명령 등에 의해 결정됩니다.

파일이 열릴 때 데이터베이스 파일 설명의 속성은 프로그램의 매개변수와 병합됩니다. 일반적으로, 프로그램이 파일을 열고 처리하는 데 있어 시스템이 필요로 하는 정보의 대부분은 파일 속성과 어플리케이션 프로그램 자체에서 발견됩니다.

그러나 때로는 파일 및 프로그램의 처리 매개변수를 대체할 필요가 있습니다. 예를 들어, 첫 번째 멤버가 아닌 파일 멤버를 처리하려면 처리할 멤버를 사용하도록 시스템에 지시해야 합니다. OVRDBF(데이터베이스 파일 대체) 명령을 수행하십시오. OVRDBF 명령을 사용하면 작업의 성능을 향상시킬 수 있는 처리 매개변수를 지정할 수 있으나, 파일 속성이나 프로그램에서는 이를 지정할 수 없습니다. OVRDBF 명령 매개변수가 파일과 프로그램 속성보다 우선합니다.

관련 개념

ILE 개념 PDF

제어 언어(CL)

관련 참조

실제 파일 작성(CRTPF) 명령

논리 파일 작성(CRTLF) 명령

소스 실제 파일 작성(CRTSRCPF) 명령

실제 파일 멤버 추가(ADDPFM) 명령

논리 파일 멤버 추가(ADDLFM) 명령

CHGPF(실제 파일 변경) 명령

실제 파일 멤버 변경(CHGPFM) 명령

CHGLF(논리 파일 변경) 명령

논리 파일 멤버 변경(CHGLFM) 명령

CHGSRCPF(소스 실제 파일 변경) 명령

데이터베이스 파일 대체(OVRDBF) 명령

데이터베이스 파일 열기(OPNDBF) 명령

조회 파일 열기(OPNQRYF) 명령

CLOF(파일 닫기) 명령

파일 및 멤버명

데이터베이스 파일의 자료를 처리하기 전에 FILE 및 MBR 매개변수와 함께 사용할 파일 및 멤버를 확인해야 합니다.

일반적으로 고급 언어 프로그램에서는 파일명은 지정하지만 멤버명을 반드시 지정할 필요는 없습니다. 그러면, 프로그램에서 파일을 열 때 시스템은 이 이름을 사용합니다. 프로그램에서 지정된 파일명을 대체하고 다른 파일을 열려면 OVRDBF(데이터베이스 파일 대체) 명령의 TOFILE 매개변수를 사용하면 됩니다. 프로그램에서 멤버명을 지정하지 않으면 파일의 첫 번째 멤버가 처리됩니다(작성 날짜 및 시간에 따라).

고급 언어 프로그램에서 멤버명을 지정할 수 없는 경우(일부 고급 언어는 멤버명을 허용하지 않음) 또는 첫 번째 멤버가 아닌 다른 멤버를 원하는 경우 OVRDBF 명령이나 열기(OPNDBF 또는 OPNQRYF) 명령을 사용하여 처리할 파일과 멤버를 지정할 수 있습니다(FILE 및 MBR 매개변수 사용).

파일 내의 모든 멤버를 처리하려면 MBR(*ALL) 매개변수가 지정된 OVRDBF 명령을 사용하십시오. 예를 들어 FILEX에 세 개의 멤버가 있고 이 멤버들을 모두 처리하려면 다음과 같이 지정합니다.

```
OVRDBF   FILE(FILEX)  MBR(*ALL)
```

OVRDBF 명령에서 MBR(*ALL)을 지정하면 프로그램은 멤버를 작성 순서대로 읽습니다. 각 멤버에 대해서 프로그램은 파일이 입력순 파일인지 또는 키순 파일인지에 따라 키순이나 입력순으로 레코드를 읽습니다.

파일 처리 옵션

이 주제에서는 몇 가지 런타임 처리 옵션에 대해 설명합니다.

처리 유형 지정:

프로그램에서 파일을 사용할 경우 해당 파일에 대해 어떤 유형의 조작을 수행할 것인지 시스템에서 알아야 합니다. 처리 유형은 OPTION 매개변수로 지정할 수 있습니다.

예를 들어 파일의 자료를 읽기만 하려는지, 아니면 자료를 읽고 갱신하려는지를 시스템이 알아야 합니다. 유효한 조작 옵션은 입력, 출력, 갱신 및 삭제 등입니다. 시스템은 사용하는 옵션이 고급 언어 프로그램에서 지정하는 정보인지 OPNDBF(데이터베이스 파일 열기) 명령 및 OPNQRYF(조회 파일 열기) 명령의 OPTION 매개변수로부터의 옵션인지를 결정합니다.

시스템은 옵션을 사용하여 프로그램에서 허용되는 조작을 결정합니다. 예를 들어 입력 전용으로만 파일을 열었는데 프로그램에서 출력 조작을 시도할 경우 프로그램에 오류가 전달됩니다.

일반적으로, 시스템은 사용자가 프로그램에서 입출력을 수행할 때 필요한 자료 권한을 갖고 있는지를 확인합니다. 그러나 OPNQRYF 또는 OPNDBF 명령을 사용할 경우 시스템은 사용자가 OPTION 매개변수에 지정된 조작을 수행할 수 있는 자료 권한을 갖고 있는지를 파일이 열릴 당시에 확인합니다.

또한 시스템은 이 옵션을 사용하여 프로그램에서 처리되고 있는 파일과 레코드의 자료 무결성을 보호하기 위해서 잠금을 결정합니다.

관련 개념

108 페이지의 『자료 권한 유형』
다음은 실제 및 논리 파일에 사용자 액세스를 허용하는 데 사용할 수 있는 자료 권한 또는 허용 목록입니다.

120 페이지의 『공유 자료 잠그기』
기본적으로 모든 데이터베이스 파일은 동시에 많은 사용자들에 의해 사용될 수 있습니다. 그러나 일부 조작은 파일, 멤버 또는 멤버 내의 자료 레코드를 잠금으로써 작업간의 공유를 방지할 수 있습니다.

초기 파일 위치 지정:

파일이 열린 후 시스템은 파일 처리가 시작될 위치를 알아야 합니다. POSITION 매개변수로 초기 파일 위치를 지정할 수 있습니다.

디폴트는 파일의 첫 번째 레코드 바로 앞에서 시작하는 것입니다(첫 번째 순차 읽기는 첫 번째 레코드를 읽음). 그러나 OVRDBF(데이터베이스 파일 대체) 명령을 사용하면 파일 끝이나 파일 중간의 특정 레코드에서 시작하도록 시스템에 지시할 수도 있습니다. 또한 프로그램에서 파일의 위치를 동적으로 설정할 수도 있습니다.

관련 개념

217 페이지의 『파일에서의 위치 설정』
작업에 의하여 파일이 열린 후, 시스템은 그 작업을 위해 파일에서의 위치를 유지보수합니다. 파일 위치는 파일을 처리하는 데 사용됩니다.

삭제 레코드 재사용:

데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

CRTPF(실제 파일 작성) 명령 또는 CHGPF(실제 파일 변경) 명령에 REUSEDLT(*YES)를 지정하면 다음과 같은 조작은 서로 다르게 작업될 수 있습니다.

- 삭제 레코드 공간을 재사용하는 파일에서는 입력 순서는 무의미합니다. 레코드가 파일의 끝에 추가되지 않을 수 있습니다.
- 삭제 레코드 공간을 재사용하는 파일에는 파일 끝(end-of-file) 지연을 적용할 수 없습니다.
- 삭제 레코드 공간의 백 퍼센트 재사용은 보증되지 않습니다. 파일에 삭제 레코드 공간이 계속 존재하더라도 *파일 가득 참(file full)* 조건이 충족되거나 파일이 확장될 수 있습니다.

주: 시스템에서 삭제 레코드 공간을 재사용하는 방식으로 인해, 삭제 레코드 공간을 재사용하도록 다음 파일 유형을 작성하거나 변경하지 않아야 합니다.

- 상대 레코드 번호를 사용하여 처리된 파일 및 다른 파일의 키로 사용된 상대 레코드 번호를 판별하도록 어플리케이션에서 사용하는 파일
- 대기행렬로 사용된 파일
- 파일 끝에 새로운 레코드를 삽입한다고 가정하는 어플리케이션에서 사용하는 모든 파일
- DB2 UDB 대칭형 멀티프로세싱 설치 시, 열을 갱신, 삽입 또는 삭제할 때 병렬 색인 유지보수를 수행하는 파일

사용자가 삭제 레코드 공간을 재사용하기 위해 기존 실제 파일을 변경하기로 결정하고, 이 실제 파일에 선입선출(FIFO) 또는 후입선출(FIFO) 중복 키 순서의 액세스 경로를 가진 논리 파일이 있는 경우 FIFO 또는 LIFO 속성이 없는 논리 파일을 다시 작성할 수 있으며 다음과 같은 방법으로 기존 액세스 경로의 재구성을 방지할 수 있습니다.

1. FIFO 또는 LIFO 속성을 가진 기존 논리 파일을 재명명하십시오.
2. 중복 키 순서가 파일에 지정되지 않는 것을 제외하고는 재명명된 파일과 동일한 2차 논리 파일을 작성하십시오. 새로운 파일에 원래 파일명을 제공하십시오. 새로운 파일은 재명명된 파일의 액세스 경로를 공유합니다.
3. 재명명된 파일을 삭제하십시오.

키순 액세스 경로 무시:

키 필드를 파일에 정의하면 시스템은 자동으로 키순 액세스 경로를 사용합니다. 그러나 때로는 ACCPTH 매개변수를 사용하여 키순 액세스 경로를 무시하면 성능이 향상될 수 있습니다.

ACCPTH 매개변수를 사용하여 키순 액세스 경로를 무시하고 파일을 도달순으로 처리할 수 있습니다. 일부 고급 언어나 OPNDBF(데이터베이스 파일 열기) 명령에서 키순 액세스 경로를 무시하도록 시스템에 지시할 수 있습니다.

키순 액세스 경로를 무시할 경우 키에 의한 자료 읽기는 허용되지 않습니다. 도달순 액세스 경로를 따라 순차적으로 수행됩니다. (선택/생략 값이 정의된 논리 파일에 이 옵션을 지정하면 도달순 액세스 경로가 사용되고 선택/생략 값에 부합되는 레코드만 프로그램에 리턴됩니다. (DYNSLT 키워드가 파일에 지정된 것처럼 처리됩니다.)

주: 둘 이상의 실제 파일 멤버를 기초로 한 논리 파일 멤버의 키순 액세스 경로는 무시될 수 없습니다.

파일 끝 처리 지연:

프로그램이 데이터베이스 파일의 자료 끝에 도달하면 일반적으로 시스템은 프로그램에 더 이상 읽을 자료가 없다는 신호를 보냅니다. 파일에 더 많은 자료가 도착할 때까지 시스템에서 프로그램을 보유하기 원할 경우 EOFDLY 매개변수를 사용하여 파일 끝 처리를 지연시킬 수 있습니다.

OVRDBF(데이터베이스 파일 대체) 명령에 EOFDLY 매개변수를 사용할 경우 파일에 자료가 도착하면 프로그램은 새로 도착한 레코드를 읽게 됩니다.

주: 파일 끝 지연은 삭제된 레코드를 재사용하는 파일에 사용할 수 없습니다.

관련 개념

222 페이지의『파일 끝에 도달했을 때의 추가 레코드 대기』

파일 끝 지연(end-of-file delay)은 파일 끝 상태가 발생한 후에도 데이터베이스 파일(논리 파일 또는 실제 파일)에서 계속 순차적으로 읽어나가는 방법입니다.

레코드 길이 지정:

옵션으로 고급 언어 프로그램에서 레코드 길이를 지정할 수 있습니다.

시스템은 프로그램에 처리할 레코드 길이를 알아야 하지만 사용자가 프로그램에서 레코드 길이를 지정할 필요는 없습니다. 시스템은 프로그램에서 지정한 파일의 속성 및 설명에 의해 이 정보를 자동으로 알아냅니다.

열린 파일에 프로그램에서 지정한 레코드 길이보다 긴 레코드가 있을 경우 시스템은 파일 멤버의 레코드 길이와 일치하는 기억장치를 할당하며 이 옵션은 무시됩니다. 이 경우 전체 레코드가 프로그램에 전달됩니다. (그러나 일부 고급 언어는 프로그램에서 지정한 레코드의 길이에 의해 정의된 레코드 부분만 액세스할 수 있도록 허용합니다.) 열린 파일에 프로그램에서 지정한 레코드 길이보다 짧은 레코드가 있을 경우 시스템은 프로그램에서 지정한 레코드 길이에 대한 기억장치를 할당합니다. 프로그램은 여분의 기억장치를 사용할 수 있으나, 입출력 조작에는 파일 멤버에 정의된 레코드 길이만 사용될 수 있습니다.

레코드 형식 무시:

복수 형식 논리 파일을 사용할 경우 시스템은 사용자가 해당 파일에 정의된 모든 형식을 사용하는 것으로 가정합니다. 그러나 형식을 모두 사용하지 않을 경우 사용하려는 형식과 무시하려는 형식을 지정할 수 있습니다.

이 옵션으로 형식을 무시하지 않으면 파일에 정의된 모든 형식을 프로그램에서 처리할 수 있습니다. 이 처리 옵션에 대한 자세한 정보는 해당 고급 언어 주제 콜렉션을 참조하십시오.

중복 키 존재 유무 판별:

키가 중복 키인지를 판별하는 데 사용하는 키순 액세스 경로 세트는 수행되는 입/출력(I/O) 조작에 따라 달라집니다. DUPKEYCHK 매개변수를 사용하여 중복 키 존재 여부를 판별할 수 있습니다.

입력(읽기)의 경우 사용되는 키순 액세스 경로는 파일이 열릴 때 사용된 액세스 경로입니다. 실제 파일에 대해 존재할 수 있는 다른 키순 액세스 경로는 고려되지 않습니다. 또한 키가 중복되었는지를 결정할 때 선택/생략 스펙에 의해 생략된 키순 액세스 경로의 레코드는 고려되지 않습니다.

출력(쓰기) 및 갱신의 경우 출력 또는 갱신 조작에서 키가 중복되었는지 결정할 때 실제 파일에 대해 존재하는 *IMMED 유지보수의 모든 고유하지 않은 키순 액세스 경로가 탐색됩니다. 피드백 시 파일에 대해 액세스 경로가 활동적으로 열려 있는 경우 *REBLD 및 *DLY 유지보수를 포함하는 키순 액세스 경로만 고려됩니다.

COBOL 프로그램으로 키순 파일을 처리하는 경우, COBOL 언어나 OPNDBF(데이터베이스 파일 열기) 명령 또는 OPNQRYF(조회 파일 열기) 명령을 통해 중복 키 피드백이 사용자의 프로그램으로 리턴되도록 지정할 수 있습니다. 그러나 COBOL에서 리턴된 중복 키 피드백이 있으면 성능이 저하될 수 있습니다.

자료 회복 및 무결성

이 주제에는 보다 효과적인 파일 처리를 위한 자료 무결성 런타임 고려사항이 들어 있습니다.

저널링 및 확약 제어를 사용하여 파일 보호:

저널링 및 확약 제어는 iSeries 시스템에서 자료 및 처리를 회복하는 데 있어 우선되는 방법입니다. COMMIT 매개변수를 사용하여 저널링 및 확약 제어로 사용 중인 파일을 보호할 수 있습니다.

데이터베이스 파일 저널링은 파일에 대해 STRJRNPF(실제 파일 저널 시작) 명령을 시작하여 실행됩니다. 액세스 경로 저널링은 파일에 대해 STRJRNAP(액세스 경로 저널 시작) 명령을 실행하거나 SMAPP(시스템 관리 액세스 경로 보호) 명령을 사용하여 시작됩니다. STRCMTCTL(확약 제어 시작) 명령 및 고급 언어 스펙을 통해 파일이 확약 제어하에서 수행되도록 할 수 있습니다. 또한 OPNDBF(데이터베이스 파일 열기) 명령 및 OPNQRYF(조회 파일 열기) 명령에서 확약 제어(COMMIT) 매개변수를 지정해도 됩니다.

참조 제한조건과 연관되어 있는 파일에서 삽입, 갱신 또는 삭제를 수행 중이나 삭제 규칙, 갱신 규칙 또는 모두 RESTRICT가 아닐 경우에는 저널링을 사용해야 합니다.

관련 개념

257 페이지의 『저널 관리』
저널 관리를 사용하면 하나 이상의 데이터베이스 파일에서 발생하는 자료 변경 내용을 모두 기록할 수 있습니다. 그런 다음 저널을 사용해 회복할 수 있습니다.

265 페이지의 『확약 제어로 자료 무결성 확인』
확약 제어는 데이터베이스 파일에 대한 일련의 변경 내용을 하나의 단위(트랜잭션)로 정의하여 처리할 수 있습니다. 확약 제어는 작업이나 시스템이 종료해도 복잡한 어플리케이션 트랜잭션이 논리적으로 동기화되는지 확인합니다. 2단계 확약 제어는 데이터베이스 파일과 같이 다중 시스템에서 확약될 수 있는 자원이 동기화 상태로 남아 있도록 보장해줍니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』
이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

보조 기억장치에 자료 및 액세스 경로 기록:

일반적으로 iSeries용 DB2 Universal Database는 변경된 자료가 주 기억장치에서 보조 기억장치로 기록되는 시점을 결정합니다. 그러나 데이터베이스 변경 내용이 보조 기억장치로 기록되는 시점을 제어할 수 있습니다.

변경된 자료가 주 기억장치에서 보조 기억장치로 기록되는 시점을 제어하려면 FRCRATIO(강제 기록 수) 매개변수를 데이터베이스 파일 작성, 변경 또는 대체 명령에 사용하고, 강제 액세스 경로(FRCACCPTH) 매개변수를 데이터베이스 파일 작성 및 변경 명령에 사용합니다. FRCRATIO 및 FRCACCPTH 매개변수를 사용하면 시스템의 성능 및 회복에 영향을 미칩니다.

관련 개념

257 페이지의『데이터베이스 회복 및 복원』

이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

레코드 형식 설명에 대한 변경 검사:

파일을 열 때 시스템은 프로그램이 컴파일된 이후 프로그램이 파일을 처리할 수 없을 정도로 사용 중인 레코드 형식의 설명이 변경되었는지를 검사합니다.

시스템은 보통 프로그램에 레벨 확인 상태를 통지합니다. 파일 작성 또는 변경 명령을 사용하는 경우 레벨 검사를 원한다고 지정할 수 있습니다. 또한 OVRDBF(데이터베이스 파일 대체) 명령에 LVLCHK 매개변수를 사용하여 파일에 정의된 레벨 검사 속성을 대체할 수도 있습니다.

관련 개념

252 페이지의『파일 설명에서 필드 변경의 영향』

이때 레벨 ID를 결정하기 위해 레코드 형식 설명의 정보가 사용됩니다. 파일 설명에서 필드를 변경하면 레벨 ID가 변경됩니다. 키 필드나 선택/생략 필드의 내용을 변경하면 새 액세스 경로를 사용하는 프로그램에 예상치 못한 결과를 가져올 수 있습니다.

파일 만기일 검사:

시스템은 지정한 파일이 현재 유효한지를 확인할 수 있습니다. 그러나 EXPDATE 및 EXPCHK 매개변수를 사용하여 파일에 만료일을 지정할 수 있습니다.

파일 작성 및 변경 명령에 EXPDATE 매개변수를 사용하여 파일이나 멤버에 만기일을 지정할 수 있습니다. 시스템이 OVRDBF(데이터베이스 파일 대체) 명령에 EXPCHK 매개변수를 사용하여 날짜를 확인할 것인지 여부를 지정할 수 있습니다. 만기일을 검사하여 현재 날짜가 만기일을 지난 경우에는 파일이 열릴 때 시스템 오퍼레이터에게 메세지가 전달됩니다.

관련 개념

41 페이지의『만기일』

EXPDATE 매개변수는 파일의 각 멤버에 대한 만기일을 지정합니다(ADDPFM, CHGPFM, CRTPF, CHGPF, CRTSRCPF 및 CHGSRCPF 명령).

작업에 의한 파일 자료 변경 방지:

프로그램을 테스트하지만 테스트에 사용되는 자료를 실제로 변경하고 싶지 않을 경우 프로그램이 파일에 시도하는 어떠한 변경도 기록되지 않도록 시스템에 지시할 수 있습니다. 이 조작의 경우 INHWRT 매개변수를 사용할 수 있습니다.

파일에 대한 변경을 금지하려면 OVRDBF(데이터베이스 파일 대체) 명령에 INHWRT(*YES)를 지정하십시오.

공유 자료 잠그기

기본적으로 모든 데이터베이스 파일은 동시에 많은 사용자들에 의해 사용될 수 있습니다. 그러나 일부 조작은 파일, 멤버 또는 멤버 내의 자료 레코드를 잠금으로써 작업간의 공유를 방지할 수 있습니다.

파일, 멤버 또는 레코드가 잠기면 다른 작업이 동일한 자료를 읽어 갱신할 수 없으므로 다른 작업이 첫 번째 작업의 갱신을 실수로 삭제할 수 없게 됩니다.

표를 열고 잠그려는 행을 편집하여 iSeries Navigator에서 행을 잠그거나 SQL LOCK TABLE문을 사용할 수 있습니다. 또는 다음 조작으로 파일, 멤버 또는 자료 레코드를 잠글 수 있으며, DSPRCDLCK(레코드 잠금 표시) 명령이나 iSeries Navigator를 사용하여 잠긴 레코드를 표시할 수 있습니다.

관련 참조

LOCK TABLE

레코드 잠금:

iSeries용 DB2 Universal Database에는 레코드에 대한 내장 무결성이 있습니다. 예를 들어, PGMA가 갱신 조작을 위해 레코드를 읽으면 그 레코드가 잠깁니다. PGMA가 그 레코드를 해제할 때까지는 다른 프로그램이 갱신 조작을 위해 동일한 레코드를 읽을 수 없지만 조회용으로는 읽을 수 있습니다.

이 같은 방법으로 시스템은 데이터베이스가 무결성을 가지도록 합니다. 그러나 WAITRCD 매개변수를 사용하여 잠긴 레코드가 해제될 대기 시간을 설정할 수 있습니다.

시스템은 프로그램에서 지정한 파일 처리 유형 및 요구된 조작에 따라 잠금 조건을 결정합니다. 예를 들어 열기 옵션에 갱신이나 삭제가 포함되는 경우 각 레코드 읽기가 잠기고 여러 사용자들이 동시에 그 레코드를 읽을 수는 있으나 한 명의 사용자만이 레코드를 갱신할 수 있게 됩니다.

시스템이 사용자가 요구하는 레코드를 사용할 수 없다는 내용의 메세지를 프로그램에 보내기 전에 잠겨진 레코드가 해제될 때까지 보통 특정한 시간(초) 동안 대기합니다. 디폴트 레코드 대기 시간은 60초입니다. 그러나 파일 작성 및 변경 명령과 데이터베이스 파일 대체 명령에 WAITRCD 매개변수를 사용하여 사용자의 대기 시간을 설정할 수 있습니다. 프로그램이 원하는 레코드가 다른 조작에 의해 잠겨 있다는 통지를 받으면 프로그램에서 적절한 조치를 취하도록 설정할 수 있습니다. (예를 들어 요구된 레코드가 현재 사용 불가능하다는 내용의 메세지를 오퍼레이터에게 보낼 수 있습니다.)

참조 무결성 CASCADE DELETE, SET NULL 또는 SET DEFAULT 삭제 규칙의 결과로서 레코드 잠금을 내재적으로 확보 중인 경우 잠금 대기 시간은 30초로 제한됩니다.

시스템은 잠겨진 레코드가 갱신 또는 삭제될 때 잠금을 자동적으로 해제합니다. 그러나 레코드를 갱신하지 않고 레코드 잠금을 해제할 수도 있습니다. 레코드 잠금 해제 방법에 대한 자세한 정보는 고급 언어 주제 콜렉션을 참조하십시오.

주: 확약 제어 사용 시에는 레코드 잠금 규칙이 변경됩니다.

DSPRCDLCK(레코드 잠금 표시) 명령이나 iSeries Navigator를 사용하여 잠긴 레코드를 표시할 수 있습니다.

관련 개념

확약 제어

124 페이지의 『DSPRCDLCK(레코드 잠금 표시) 명령을 사용하여 잠긴 레코드 표시』
또한 DSPRCDLCK(레코드 잠금 표시) 명령을 사용하여 실제 파일 멤버에 대한 레코드의 현재 잠금 상태(대기 또는 보류)를 표시할 수 있습니다.

관련 태스크

123 페이지의 『iSeries Navigator를 사용하여 잠긴 행 표시』
이 프로시듀어를 사용하여 iSeries Navigator로 잠긴 행을 표시할 수 있습니다.

파일 잠그기:

일부 파일 조작은 처리 시간 동안 파일을 독점적으로 할당합니다. 파일이 배타적으로 할당된 경우 파일을 열려는 프로그램은 파일이 릴리스될 때까지 기다려야 합니다. 그러나 WAITFILE 매개변수를 사용하여 파일을 사용할 수 있도록 대기 시간을 설정할 수 있습니다.

파일 작성 및 변경 명령과 데이터베이스 파일 대체 명령의 WAITFILE 매개변수에 대기 시간을 지정함으로써 파일이 사용 가능해질 때까지 프로그램이 대기해야 하는 시간을 제어할 수 있습니다. 대기 시간을 지정하지 않으면 시스템은 파일 대기 시간의 디폴트 값을 0초로 합니다.

파일 속성이 변경되는 조작 수행시에는 그 파일이 독점적으로 할당됩니다. 이러한 조작(이동, 재명명, 권한부여 또는 취소, 소유자 변경 또는 삭제 등)은 동일한 파일이나 해당 파일의 멤버를 사용하는 또 다른 조작과 동시에 수행될 수 없습니다. 그 외의 파일 조작(오브젝트의 표시, 열기, 덤프 또는 검사 등)은 파일 정의만 사용하므로 파일을 덜 독점적으로 잠급니다. 이러한 조작들은 서로 동시에 수행될 수 있으며 하나의 멤버에 대한 입출력 조작과도 동시에 수행될 수 있습니다.

멤버 잠금:

멤버 조작(추가 및 제거 등)은 파일을 독점적으로 자동 할당함으로써 다른 파일 조작이 동시에 이루어지는 것을 금지합니다. 같은 멤버에 대한 입출력 조작은 동시에 수행될 수 없으나 같은 파일의 서로 다른 멤버에 대한 입출력 조작은 동시에 수행될 수 있습니다.

레코드 형식 자료 잠금:

레코드 형식과 연관된 전체 레코드 세트를 잠가야 할 경우가 있습니다(예: 실제 파일 내의 모든 레코드). 이 경우 OVRDBF(데이터베이스 파일 대체) 명령에 RCDFMTLCK 매개변수를 사용할 수 있습니다.

데이터베이스 잠금 고려사항:

일반적으로 사용되는 데이터베이스 기능 및 데이터베이스 파일에 지정되는 잠금 유형을 요약합니다.

표 38은 가장 일반적으로 사용되는 데이터베이스 기능 및 데이터베이스 파일에 행해지는 잠금 유형을 요약합니다. 잠금 유형은 다음 페이지에서 설명됩니다.

표 38. 데이터베이스 기능 및 잠금

기능	명령	파일 잠금	멤버/자료 잠금	액세스 경로 잠금
멤버 추가	ADDPFM, ADDLFM	*EXCLRD		*EXCLRD
파일 속성 변경	CHGPF, CHGLF	*EXCL	*EXCLRD	*EXCLRD
멤버 속성 변경	CHGPFM, CHGLFM	*SHRRD	*EXCLRD	
오브젝트 소유자 변경	CHGOBJOWN	*EXCL		
오브젝트 검사	CHKOBJ	*SHRNUPD		
실제 파일 멤버 지우기	CLRPFM	*SHRRD	*EXCLRD[3]	
중복 오브젝트 작성	CRTDUPOBJ	*EXCL(새 오브젝트) *SHRNUPD(오브젝트)		
파일 작성	CRTPF, CRTLF, CRTSRCPF	*EXCL		
파일 삭제	DLTF	*EXCL		*EXCLRD
권한부여/취소	GRTOBJAUT, RVKOBJAUT	*EXCL		
실제 파일 멤버 초기화	INZPFM	*SHRRD	*EXCLRD	
오브젝트 이동	MOVOBJ	*EXCL		
파일 열기	OPNDBF, OPNQRYF	*SHRRD	*SHRRD	*EXCLRD
액세스 경로 재구성	EDTRBDAP, OPNDBF	*SHRRD	*SHRRD	*EXCLRD
멤버 제거	RMVM	*EXCLRD	*EXCL	*EXCLRD
파일 재명명	RNMOBJ	*EXCL	*EXCL	*EXCL
멤버 재명명	RNMM	*EXCLRD	*EXCL	*EXCL
실제 파일 멤버 재구성	RGZPFM	*SHRRD	*EXCL[4]	
파일 복원	RSTLIB, RSTOBJ	*EXCL		
파일 저장	SAVLIB, SAVOBJ, SAVCHGOBJ	*SHRNUPD[1]	*SHRNUPD[2]	

[1] 활동 중 저장의 경우 초기 파일 잠금은 *SHRUPD이고 그 다음에는 *SHRRD로 축소됩니다. 저장 명령의 활동 중 저장 잠금에 대한 내용은 백업 및 회복  을 참조하십시오.

[2] 활동 중 저장의 경우 멤버/자료 잠금은 *SHRRD입니다.

[3] 멤버가 이 프로세스 또는 다른 프로세스에서 개방되어 있으면 지원지지 않습니다.

[4] ALWCANCEL(*YES)이 지정되면 LOCK 키워드는 *SHRUPD 또는 *EXCLRD 잠금을 대신 지정할 수 있습니다.

다음 표는 유효한 잠금 조합을 나타냅니다.

잠금	***EXCL**	***EXCLRD**	***SHRUPD**	***SHRNUPD**	***SHRRD**
*EXCL[1]					
*EXCLRD[2]					X
*SHRUPD[3]			X		X
*SHRNUPD[4]				X	X
*SHRRD[5]		X	X	X	X

잠금	*EXCL	*EXCLRD	*SHRUPD	*SHRNUPD	*SHRRD

[1] 배타적 잠금(*EXCL). 오브젝트가 요구중인 작업에만 배타적으로 사용되도록 할당되어 다른 작업이 오브젝트를 사용할 수 없습니다.

[2] 읽기 허용 배타적 잠금(*EXCLRD). 오브젝트가 이를 요구한 작업에 할당되나 다른 작업이 이 오브젝트를 읽을 수 있습니다.

[3] 읽기 및 갱신 허용 공유 잠금(*SHRUPD). 오브젝트가 다른 작업과 공유되면 읽히거나 변경될 수 있습니다.

[4] 읽기 전용 공유 잠금(*SHRNUPD). 오브젝트가 다른 작업과 공유되면서 읽힐 수 있습니다.

[5] 공유 잠금(*SHRRD). 작업이 오브젝트의 배타적 사용을 요구하지 않으면 오브젝트가 다른 작업과 공유될 수 있습니다.

표 39는 제한조건이 상위 파일(PAR)과 연관되어 있는지 아니면 종속 파일(DEP)과 연관되어 있는지에 따라 데이터베이스 파일의 제한조건에 대한 데이터베이스 잠금을 보여줍니다.

표 39. 데이터베이스 제한조건 잠금. 괄호 안의 숫자는 표의 끝에 있는 주를 나타냅니다.

기능 유형	파일 유형	파일[5]	멤버[5]	기타 파일	기타 멤버
ADDPFM[1]	DEP	*EXCL	*EXCL	*EXCL	*EXCL
ADDPFM[1]	PAR	*EXCL	*EXCL	*EXCL	*EXCL
ADDPFCST[7]	*REFCST	*EXCL	*EXCL	*EXCL	*EXCL
ADDPFCST[6]	*UNQCST *PRIKEY	*EXCL	*EXCL	*EXCL	*EXCL
ADDPFCST	*UNIQUE *PRIKEY	*EXCL	*EXCL		
RMVM[2]	DEP	*EXCL	*EXCL	*EXCL	*EXCL
RMVM[2]	PAR	*EXCL	*EXCL	*EXCL	*EXCL
DLTF[3]	DEP	*EXCL	*EXCL	*EXCL	*EXCL
DLTF[3]	PAR	*EXCL	*EXCL	*EXCL	*EXCL
RMVPFCST[7]	*REFCST	*EXCL	*EXCL	*EXCL[4]	*EXCL
RMVPFCST[6]	*UNQCST *PRIKEY	*EXCL	*EXCL	*EXCL	*EXCL
RMVPFCST	*UNIQUE *PRIKEY	*EXCL	*EXCL		
CHGPFCST		*EXCL	*EXCL	*SHRRD	*EXCL

[1] 실제 파일 멤버를 추가하여 참조 제한조건이 설정되는 경우.

[2] 실제 파일 멤버를 제거하여 설정된 참조 제한조건이 정의되는 경우.

[3] 파일에 대해 설정되거나 정의된 제한조건이 있는 종속 또는 상위 파일을 삭제하는 경우.

[4] 설정되거나 정의된 제한조건이 있는 상위 파일에 대해 RMVPFCST(실제 파일 제한조건 제거) 명령이 호출되면 상위 파일 및 상위 파일 이상의 어떠한 논리 파일도 모두 잠깁니다(*EXCL).

[5] 참조 제한조건의 경우 열은 종속 파일 또는 종속 멤버를 의미합니다.

[6] 고유 제한조건 또는 1차 키 제한조건은 다른 파일이 종속 파일인 참조 제한조건의 상위 키입니다.

[7] 기타 파일이 상위 파일입니다.

iSeries Navigator를 사용하여 잠긴 행 표시:

이 프로시듀어를 사용하여 iSeries Navigator로 잠긴 행을 표시할 수 있습니다.

iSeries Navigator를 사용하여 잠긴 행을 표시하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스** → **라이브러리**를 확장하십시오.

3. 잠긴 행 정보를 표시하려는 표가 들어 있는 라이브러리를 클릭하십시오.
4. 표를 마우스 오른쪽 버튼으로 클릭하고 **잠긴 행**을 클릭하십시오. 잠긴 행 창에서는 잠긴 행을 표시합니다.

DSPRCDLCK(레코드 잠금 표시) 명령을 사용하여 잠긴 레코드 표시:

또한 DSPRCDLCK(레코드 잠금 표시) 명령을 사용하여 실제 파일 멤버에 대한 레코드의 현재 잠금 상태(대기 또는 보류)를 표시할 수 있습니다.

DSPRCDLCK 명령은 현재 보류 상태에 있는 잠금 유형도 표시합니다. 지정하는 매개변수에 따라 이 명령은 특정한 레코드의 잠금 상태를 표시하거나 멤버 내의 모든 레코드 잠금 상태를 표시합니다. 또한 WRKJOB(작업(job)에 대한 작업) 화면에서 레코드 잠금을 표시할 수도 있습니다.

관련 개념

제어 언어(CL)

백업 및 회복

관련 참조

레코드 잠금 표시(DSPRCDLCK) 명령

동일한 작업 또는 활성 그룹에서 데이터베이스 파일 공유

기본적으로 데이터베이스 관리 시스템에서는 하나의 파일이 동시에 많은 사용자에 의해 읽히고 변경되도록 되어 있습니다. 그러나 SHARE 매개변수를 통해 동일한 작업이나 활성 그룹의 데이터베이스 파일을 공유할 수 있습니다.

데이터베이스 파일을 열어 동일한 작업이나 활성 그룹 내에서 한 파일을 공유할 수 있습니다.

- 동일한 프로그램에서 한 번 이상
- 동일한 작업이나 활성 그룹 내의 서로 다른 프로그램에서

주: 통합 언어 환경®에서 열기 공유에 대한 자세한 정보는 ILE 개념  서적을 참조하십시오.

파일 작성, 파일 변경 및 데이터베이스 파일 대체 명령에서 SHARE 매개변수를 사용하면 파일, 파일 상태, 파일 위치 및 파일의 기억장치 등이 한 작업이나 활성 그룹 내에서 공유됩니다. 작업이나 활성 그룹에서의 파일 공유는 작업이 필요로 하는 주 기억장치를 줄여 주고 파일을 열고 닫는 데 소요되는 시간을 줄임으로써 성능을 향상시키게 됩니다.

SHARE(*YES) 매개변수를 사용하면 동일한 작업 또는 활성 그룹에서 수행되는 둘 이상의 프로그램간에 열린 자료 경로(ODP:Open Data Path)가 공유될 수 있습니다. 열린 자료 경로란 해당 파일에 대한 모든 입출력 조작이 수행되는 경로입니다. 어떤 점에서 이것은 프로그램을 파일에 연결시켜 주는 것이라고 볼 수 있습니다. SHARE(*YES) 매개변수가 지정되지 않으면 파일이 열릴 때마다 새로운 개방 자료가 작성됩니다. 활동 파일이 동일한 작업이나 활성 그룹 내에서 한 번 이상 열릴 경우 파일의 현재 열기에 대하여 활동 중인 ODP를 사용할 수 있습니다. 새로운 열린 자료 경로는 작성될 필요가 없습니다.

이것은 첫 번째 열기 이후에 파일을 여는 데 걸리는 시간을 줄여주며, 작업이나 활성 그룹에 필요한 주기억장치의 양도 줄여줍니다. ODP를 공유하려면 같은 파일의 첫 번째 열기 및 기타 열기에 SHARE(*YES)를 지정해야 합니다. 잘 설계된(성능면에서) 어플리케이션은 일반적으로 같은 작업이나 활성 그룹 내의 복수 프로그램에서 열리는 파일과 ODP를 공유합니다.

SHARE(*NO)를 지정하면 시스템은 파일의 ODP를 공유하지 않습니다. 일반적으로 이것은 거의 사용하지 않으며 특정한 프로그램에서 고유한 처리가 필요한 파일에 대해서만 지정합니다.

주: 고급 언어 프로그램은 파일이 공유되고 있지 않은 것처럼 열기 또는 닫기 조작을 처리합니다. 사용자는 해당 파일이 공유될 것임을 고급 언어 프로그램에서 지정하지 않습니다. SHARE 매개변수를 통해 파일이 동일 작업이나 활성 그룹에서 공유될 것임을 표시하게 됩니다. SHARE 매개변수는 데이터베이스 파일 작성, 변경 및 대체 명령 등에서만 지정됩니다.

동일한 작업이나 활성 그룹 내에서 공유 파일 열기 고려사항:

여기 나열된 내용은 같은 작업이나 활성 그룹 내에서 공유되는 데이터베이스 파일을 열 때 고려해야 할 사항입니다.

- 작업이나 활성 그룹에서 공유된 파일이 처음 열릴 때 지정된 파일의 후속 열기에 필요한 열기 옵션이 모두 지정되었는지 확인하십시오. 공유 파일의 후속 열기에 지정된 열기 옵션이 공유 파일의 첫 번째 열기 옵션과 일치하지 않을 경우 오류 메세지가 나옵니다. (프로그램이나 OPNDBF(데이터베이스 파일 열기) 또는 OPNQRYF(조회 파일 열기) 명령 매개변수의 내용을 변경하여 이를 정정하고 일치하지 않는 옵션을 제거할 수 있습니다.)

  예를 들어, PGMA는 작업이나 활성 그룹 내에서 FILE1을 여는 첫 번째 프로그램이며 파일을 읽기만 합니다. 그러나 PGMA가 같은 공유 파일로부터 레코드를 삭제하는 PGMB를 호출합니다. PGMB가 공유 파일로부터 레코드를 삭제하게 되므로 PGMA는 마치 정확한 레코드를 삭제하려고 한 것처럼 파일을 열어야 합니다. 이는 고급 언어에서 올바른 스펙을 사용함으로써 가능하게 됩니다(일부 고급 언어에서 이를 처리하려면 한 번도 수행되지 않은 파일 조작문을 사용해야 합니다. 자세한 내용은 고급 언어 주제 콜렉션을 참조하십시오.) OPNDBF 및 OPNQRYF 명령의 OPTION 매개변수에 파일 처리 옵션을 지정할 수도 있습니다.
- 때로는 한 작업이나 활성 그룹 내에서 파일을 공유하는 것이 바람직하지 않을 때가 있습니다. 예를 들어, 한 프로그램에서는 도달순으로 정렬된 파일의 레코드가 필요하고 다른 프로그램에서는 키순으로 정렬된 레코드가 필요할 수 있습니다. 이런 경우에는 열린 자료 경로(ODP)를 공유할 수 없습니다. OVRDBF(데이터베이스 파일 대체) 명령에 SHARE(*NO)를 지정하여 작업이나 활성 그룹 내에서 해당 파일이 공유되지 않도록 해야 합니다.
- 제품 라이브러리에서 공유된 파일이 맨 처음 열린 후, 디버그 모드가 UPDPROD(*NO)로 시작된 경우 해당 파일의 후속 공유 열기는 원래의 ODP를 공유하고 해당 파일의 변경을 허용합니다. 이를 방지하려면 수정되는 파일을 열기 전에 OVRDBF 명령에 SHARE(*NO)를 지정해야 합니다.
- 공유 파일의 첫 번째 열기에 확약 제어를 사용하면 모든 후속 공유 열기에도 확약 제어를 사용해야 합니다.
- 파일의 후속 공유 열기에 키 피드백, 삽입 키 피드백, 중복 키 피드백을 원한다면 이들을 전체 열기로 지정되어야 합니다.

- 프로그램이나 OVRDBF 명령에 라이브러리명을 지정하지 않는 경우(*LIBL이 사용됨), 시스템은 *LIBL이 지정된 동일한 공유 파일의 최종 열기 이후 라이브러리 리스트가 변경되지 않은 것으로 가정합니다. 라이브러리 리스트가 변경되었다면 사용자는 OVRDBF 명령에 라이브러리명을 지정하여 정확한 파일이 열리도록 해야 합니다.
- 파일의 공유 열기에 더 긴 레코드 길이 값이 지정되었어도 전체 열기에서 지정된 레코드 길이가 후속 공유 열기에 사용되는 레코드 길이가 됩니다.
- 공유 파일의 첫 번째 열기에서 지정된 대체 속성 및 프로그램 스펙이 처리됩니다. 후속 열기에서 지정된 대체 속성 및 프로그램 스펙은 OVRDBF 명령의 SHARE 또는 LVLCHK 매개변수에 지정된 파일명이나 값을 변경하는 것을 제외하고는 무시됩니다.
- OPNQRYF 명령을 사용하여 첫 번째 열기에 지정된 대체 속성은 OPNQRYF 명령으로 처리해야 하는 파일, 라이브러리 및 멤버 등의 이름을 변경하는 데 사용될 수 있습니다. TOFILE, MBR, LVLCHK 및 SEQONLY를 제외한 OVRDBF 명령에 지정된 매개변수 값은 OPNQRYF 명령에 의해 무시됩니다.
- OPNDBF 및 OPNQRYF 명령은 다음 규칙에 따라 ODP를 OPNSCOPE(열기 범위) 매개변수로 지정된 범위까지 가능하게 만듭니다.
  - 시스템은 먼저 활성 그룹 내의 공유 열기를 탐색한 다음 해당 작업의 공유 열기를 탐색합니다.
  - 활성 그룹에서 작용하는 공유 열기는 활성 그룹 간에 공유될 수 없습니다.
  - 해당 작업에서의 공유 열기는 그 당시 활성 그룹의 수와 관계없이 작업 전체에서 공유될 수 있습니다.

CPF4123 진단 메세지는 전체 열기와 후속 공유 열기 사이에서 일치되지 않는 내용들을 나열합니다. 이러한 일치되지 않는 내용으로 인해 공유 열기가 실패하지는 않습니다.

주: OPNQRYF 명령은 해당 작업이나 활성 그룹 내에서 기존 공유 OCP를 공유하지 않습니다. OPNQRYF 명령에 지정한 이름과 동일한 파일명, 라이브러리명 및 멤버명이 있는 작업이나 활성 그룹에 공유 ODP가 이미 있을 경우 시스템이 오류 메세지를 보내고 조회 파일은 열리지 않게 됩니다.

동일한 작업이나 활성 그룹 내에서 공유 파일 입/출력 고려사항:

여기 나열된 내용은 같은 작업이나 활성 그룹 내에서 공유되는 데이터베이스 파일 처리 시 고려해야 할 사항입니다.

- 공유 파일에는 열린 자료 경로가 한 개만 허용되므로 파일을 공유하고 있는 작업이나 활성 그룹 내의 모든 프로그램에서 유지되는 레코드의 위치는 하나뿐입니다. 프로그램이 읽기 또는 갱신용 읽기를 사용하여 레코드의 위치를 설정하고 그 공유 파일을 사용하는 다른 프로그램을 호출하면 호출된 프로그램이 호출하는 프로그램으로 리턴할 때 레코드 위치가 이동되거나 레코드 잠금이 해제될 수 있습니다. 이렇게 되면 예기치 않은 레코드 위치나 잠금 상태로 인하여 호출하는 프로그램에 오류가 발생할 수도 있습니다. 파일 공유시, 레코드 위치 및 레코드 잠금을 다시 설정함으로써 레코드 위치와 레코드 잠금 고려사항을 관리하는 것은 사용자의 책임입니다.
- 갱신 조작을 위해 공유 파일을 처음으로 열 때 이 파일을 공유하는 모든 후속 프로그램이 반드시 레코드 잠금을 요구하는 것은 아닙니다. 시스템이 파일을 사용하는 각 프로그램에 필요한 레코드 잠금 유형을 결정합니다. 시스템은 잠금 경합을 최소로 하는 반면 여전히 자료 무결성(DATA INTEGRITY)을 유지하고자 노력합니다.

예를 들어, PGMA는 작업이나 활성 그룹 내에서 공유된 파일을 여는 첫 번째 프로그램입니다. PGMA가 파일의 레코드를 갱신하려고 합니다. 따라서 프로그램이 레코드를 갱신하려고 읽으면 해당 레코드가 잠깁니다. 그 다음, PGMA가 PGMB를 호출합니다. PGMB도 공유 파일을 사용하나 PGMB는 파일의 레코드를 갱신하는 것이 아니라 단지 읽기만 합니다. PGMA는 원래 공유 파일을 갱신 가능한 파일로 열었으나 PGMB 자체의 처리 스펙 때문에 읽는 레코드를 잠그지 않습니다. 따라서 시스템은 자료 무결성을 보장하는 한편, 레코드 잠금 경합을 최소화합니다.

동일한 작업이나 활성 그룹 내에서 공유 파일 닫기 고려사항:

여기 나열된 내용은 같은 작업이나 활성 그룹 내에서 공유되는 데이터베이스 파일을 닫을 때 고려해야 할 사항입니다.

- (파일, 멤버 및 레코드 잠금의 해제, 보조 기억장치에 변경사항 강제 및 열린 자료 경로의 파손 등과 같은) 닫기 조작 처리는 공유 열린 자료 경로를 마지막으로 연 프로그램이 그 경로를 닫았을 때에만 완전히 수행됩니다.
- 파일이 OPNDBF(데이터베이스 파일 열기) 명령이나 OPNQRYF(조회 파일 열기) 명령으로 열린 경우 그 파일을 닫으려면 CLOF(파일 닫기) 명령을 사용하십시오. RCLRSC(자원 재생) 명령은 다음 매개변수 중 하나가 지정된 경우 OPNQRYF 명령으로 열린 파일을 닫을 경우에도 사용할 수 있습니다.
  - OPNSCOPE(*ACTGRPDFN) 및 디폴트 활성 그룹에 의해 열림이 요청될 경우
  - TYPE(*NORMAL)

  다음 매개변수 중 하나가 지정될 경우 RCLRSC(자원 재생) 명령이 실행될 때에도 파일은 열린 상태로 있습니다.
  - OPNSCOPE(*ACTGRPDFN) 및 디폴트가 아닌 활성 그룹으로부터 열림 요청을 받은 경우.
  - OPNSCOPE(*ACTGRP)
  - OPNSCOPE(*JOB)
  - TYPE(*PERM)

예 1: 유사한 처리 옵션이 있는 단일 파일 세트:

이 예에서 사용자는 사인 온하고 사용하는 대부분의 프로그램은 동일한 파일 세트를 처리하게 됩니다.

제어 언어(CL) 프로그램(PGMA)이 처음으로 사용되는 프로그램입니다(공유 파일의 대체 및 열기를 포함하는 어플리케이션을 설정하는). 그 다음, PGMA는 PGMB로 제어를 이전하며 PGMB는 어플리케이션 메뉴를 표시합니다. 이 예에서는 파일 A, B 및 C가 사용되고, 그 중 파일 A와 B가 공유되는 것으로 가정하십시오. 파일 A과 B는 SHARE(*NO)로 작성되었으므로 SHARE(*YES) 옵션을 지정하려면 OVRDBF(데이터베이스 파일로 대체) 명령이 각 OPNDBF(데이터베이스 파일 열기) 명령 앞에 있어야 합니다. 파일 C는 SHARE(*NO)로 작성되었으며, 이 예에서는 공유되지 않습니다.

주: 해당 코드 예제를 사용하는 것은 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```
PGMA:   PGM      /* PGMA - Initial program  */
                OVRDBF   FILE(A) SHARE(*YES)
                OVRDBF   FILE(B) SHARE(*YES)
                OPNDBF   FILE(A) OPTION(*ALL) ....
                OPNDBF   FILE(B) OPTION(*INP) ...
                TFRCTL   PGMB
                ENDPGM



PGMB:   PGM      /* PGMB - Menu program */
                DCLF     FILE(DISPLAY)
BEGIN:  SNDRCVF  RCDFMT(MENU)
                IF       (&RESPONSE *EQ '1') CALL PGM11
                IF       (&RESPONSE *EQ '2') CALL PGM12
        .
        .
                IF       (&RESPONSE *EQ '90') SIGNOFF
                GOTO     BEGIN
                ENDPGM
```

PGMA에서 열린 파일은 해당 작업 또는 동일한 활성 그룹 내에서 수행되는 PGMA, PGM11 및 PGM12까지 영향을 미치며, 파일 열기는 해당 활성 그룹으로까지 영향을 줍니다.

이 예에서는 다음과 같이 가정됩니다.

- PGM11은 파일 A과 B를 엽니다. 이러한 파일은 PGMA에서 OPNDBF 명령에 의해 공유 상태로 열렸으므로 열리는 시간이 단축됩니다. 공유 열린 자료 경로가 닫힐 때에도 닫히는 시간이 단축됩니다. 제어가 PGMB로 이전되었어도(PGMA에서 TFRCTL(제어 이전) 명령으로) OVRDBF 명령은 계속 유효합니다.
- PGM12은 파일 A, B 및 C를 엽니다. 파일 A와 B는 이미 공유 상태로 열렸으며 열리는 시간이 단축됩니다. 파일 C는 이 프로그램에서만 사용되므로 공유되는 상태로 열리지는 않습니다.

이 예에서는 한 파일 세트만 필요하기 때문에 CLOF(파일 닫기)는 사용되지 않았습니다. 오퍼레이터가 사인 오프할 때 파일들은 자동적으로 닫힙니다. PGMA(초기 프로그램)는 작업 시작시에만 호출되는 것으로 가정한 것입니다. 통합 언어 자원에서 자원 재생 방법에 대한 자세한 정보는 ILE 개념 서적을 참조하십시오.

주: PGMB의 화면 파일(DISPLAY)도 공유 파일로 지정될 수 있으며 이는 추후에 이를 사용하는 프로그램에서 해당 파일을 여는 데 있어 성능을 향상시킵니다.

예 1에서 OPNDBF 명령을 별개의 프로그램(PGMA)에 사용하여 작업 내의 다른 처리 프로그램이 가능한 한 효율적으로 수행되도록 합니다. 즉 작업 내의 다른 프로그램에서 사용되는 중요한 파일들은 PGMA에서 열립니다. PGMA에 의해 파일이 열리고 난 후 주 처리 프로그램(PGMB, PGM11 및 PGM12)이 파일을 공유할 수 있습니다. 따라서, 이들 프로그램의 열기 및 닫기 요구가 보다 빠르게 처리됩니다. 그 외에도 PGMB보다 PGMA에서 OPNDBF 명령을 사용하므로써 PGMB에 사용되는 주 기억장치가 줄어듭니다.

대체 및 열기는 초기 프로그램(PGMA)에 지정될 수 있습니다. 그런 다음 해당 프로그램은 작업에서 제거될 수 있습니다(예를 들면, 외부로 프로그램을 이전함으로써). 그러나 프로그램이 파일을 열 때 작성한 열린 자료 경로는 계속 존재하며 작업 내의 다른 프로그램에 의해 사용될 수 있습니다.

주: 대체는 파일이 열리기 전에 지정되어야 합니다. OVRDBF 명령의 매개변수 중 일부는 OPNDBF 명령에도 존재합니다. 두 명령이 모순될 경우 OVRDBF 값이 사용됩니다.

관련 개념

ILE 개념 PDF

예 2: 유사한 처리 옵션이 있는 복수 파일 세트:

이 예에서는 각 어플리케이션에 서로 다른 파일이 사용됩니다. 사용자는 일반적으로 상당한 시간 동안 한 어플리케이션을 처리한 후 새로운 어플리이션을 선택합니다.

OPNDBF 명령을 사용하여 요구된 파일을 여는 어플리케이션 프로그램(예: 미수금이나 미지급금)을 메뉴에서 오퍼레이터가 지정할 수 있는 것으로 가정하십시오. 어플리케이션이 종료될 때 CLOF 명령으로 파일이 닫힙니다. CLOF 명령은 작업에 필요한 주 기억장치를 줄이는 데 사용됩니다.

미수금 프로그램의 예는 다음과 같습니다.

주: 해당 코드 예제를 사용하여 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```
PGMC:   PGM      /* PGMC PROGRAM */
                DCLF     FILE(DISPLAY)
BEGIN:  SNDRCVF  RCDFMT(TOPMENU)
                IF       (&RESPONSE *EQ '1') CALL ACCRECV
                IF       (&RESPONSE *EQ '2') CALL ACCPAY
        .
        .
                IF       (&RESPONSE *EQ '90') SIGNOFF
                GOTO     BEGIN
                ENDPGM



ACCREC: PGM      /* ACCREC PROGRAM */
                DCLF     FILE(DISPLAY)
                OVRDBF   FILE(A) SHARE(*YES)
                OVRDBF   FILE(B) SHARE(*YES)
                OPNDBF   FILE(A) OPTION(*ALL) ....
                OPNDBF   FILE(B) OPTIONS(*INP) ...
BEGIN:  SNDRCVF  RCDFMT(ACCRMENU)
                IF       (&RESPONSE *EQ '1') CALL PGM21
                IF       (&RESPONSE *EQ '2') CALL PGM22
        .
        .
                IF       (&RESPONSE *EQ '88') DO /* Return */
                                 CLOF FILE(A)
                                 CLOF FILE(B)
                                 RETURN
                               ENDDO
                GOTO     BEGIN
                ENDPGM
```

미지급금 메뉴에 대한 프로그램도 이와 유사하나 다른 세트의 OPNDBF 명령 및 CLOF 명령을 사용합니다.

이 예에서 파일 A와 B는 SHARE(*NO)로 작성되었습니다. 따라서 OVRDBF 명령이 반드시 OPNDBF 명령 앞에 사용되어야 합니다. 예 1에서와 같이 별도의 프로그램에 OPNDBF 명령을 사용한 후, 이를 호출함으로써 각 작업에 의해 사용되는 주 기억장치를 줄일 수 있습니다. CLOF 명령에 대해서도 별도의 프로그램이 작성될 수 있습니다. OPNDBF 명령은 메뉴에서 호출되는 어플리케이션 설정 프로그램에 사용할 수 있으며, 이러한 프로그램은 특정 어플리케이션 프로그램 메뉴로 제어를 이전시킵니다. (이 설정 프로그램에서 지정된 모든 대체 속성은 그대로 유지됩니다.) 그러나 이러한 기능을 위해 별도의 프로그램을 호출하는 것도 다른 메뉴가 사용되는 빈도에 따라 시스템 자원을 사용하면 이 예에서 보는 바와 같이 각 어플리케이션 프로그램 메뉴에 OPNDBF 명령 및 CLOF 명령을 포함시키는 것이 더 효과적일 수 있습니다.

또 다른 방법은 CLOF 명령을 사용하는 대신 PGMC(설정 프로그램)에서 RCLRSC(자원 재생) 명령을 사용하는 것입니다. RCLRSC 명령은 모든 파일을 닫으며 호출된 또는 호출한 프로그램에 남아 있는 파일 및 프로그램과 연관된 잔여 기억장치를 모두 해제합니다. 그러나 RCLRSC는 OPNDBF나 OPNQRYF(조회 파일 열기) 명령에 지정된 다음 매개변수로 열린 파일은 닫지 *않습니다*.

- OPNSCOPE(*ACTGRPDFN) 및 디폴트가 아닌 활성 그룹으로부터 열림 요청을 받은 경우
- OPNSCOPE(*ACTGRP)이 열려있는 활성 그룹 번호보다 작은 번호의 활성 그룹으로부터 RCLRSC 명령이 발행되도록 요구하는 경우.
- OPNSCOPE(*JOB)
- TYPE(*PERM)

다음의 예에서는 파일을 닫는데 사용되는 RCLRSC 명령을 보여줍니다.

```
 .
 .
IF            (&RESPONSE *EQ '1') DO
                          CALL ACCRECV
                          RCLRSC
                          ENDDO
IF            (&RESPONSE *EQ '2') DO
                          CALL ACCPAY
                          RCLRSC
                          ENDDO
 .
 .
```

예 3: 여러 가지 처리 요구사항이 있는 단일 파일 세트:

이 예에서는 일부 프로그램에 읽기 전용 파일 처리가 필요하고 다른 프로그램에 옵션(입력/갱신/추가/삭제)의 일부 또는 전부가 필요한 경우 사용할 수 있는 방법을 표시합니다.

파일이 어떤 프로그램에서는 특정한 명령 매개변수로 처리되어야 하는 반면, 다른 프로그램에서는 그렇지 않을 경우에도 똑같은 방법이 적용됩니다. (예를 들어, 때로는 확약 옵션이 사용되어야 합니다.)

단일 OPNDBF(데이터베이스 파일 열기) 명령을 사용하여 OPTION(*ALL)을 지정할 수 있으며 열린 자료 경로는 공유 상태로 열리게 됩니다(예를 들어, 이전의 OVRDBF(데이터베이스 파일로 대체) 명령이 SHARE(*YES)를 지정하는 데 사용되었을 경우). 그러면 각 프로그램이 옵션의 서브세트를 열 수 있습니다. 프로그램은 프로그램의 스펙에 따라 열기 유형을 요구합니다. 프로그램은 더 이상의 고려사항을 요구하지 않는데, 그 이유는

입력 전용으로 열기를 지정하는 프로그램은 공유 열기를 수행하지 않았을 때와 유사하게 작동하기 때문입니다. (예를 들어, 레코드가 읽힐 때 레코드 잠금은 추가로 발생하지 않습니다.)

그러나 OPNDBF 명령에서 지정된 일부 옵션은 프로그램이 어떻게 작동할 것인가에 영향을 줍니다. 예를 들어 SEQONLY(*NO)가 프로그램에서 파일 열기 명령에 지정된 것으로 가정하십시오. OPNDBF 명령에 SEQONLY(*YES)를 사용하고 프로그램을 순차 전용 처리로 수행하지 않으면 오류가 발생하게 됩니다.

ACCPTH 매개변수도 프로그램이 액세스 경로를 사용하는 방법(입력순 또는 키순)과 일관성이 있어야 합니다.

OPNDBF 명령에 COMMIT(*YES)이 지정되고 STRCMTCTL(확약 제어 시작) 명령이 LCKLVL(*ALL) 또는 LCKLVL(*CS)을 지정하면 레코드의 읽기 조작으로 인해 해당 레코드가 잠깁니다(확약 제어 레코드 잠금 규칙을 따름). 이로 인해 레코드가 예기치 않게 잠기게 되며 프로그램에 오류가 발생합니다.

동일한 자료에 OPNDBF 명령 두 개를 사용할 수 있습니다(예를 들어, 하나는 OPTION(*ALL)을 지정하고 다른 하나는 OPTION(*INP)을 지정). 두 번째의 사용은 같은 실제 파일을 가리키는 논리 파일이어야 합니다. 그러면 이 논리 파일은 SHARE(*YES)로 열릴 수 있으며 같은 작업중에 여러 번 사용될 수 있습니다.

데이터베이스 파일의 순차 전용 처리

프로그램이 입력 전용 또는 출력 전용으로 데이터베이스 파일을 순차적으로 처리하는 경우 OVRDBF(데이터베이스 파일 대체) 명령 및 OPNDBF(데이터베이스 파일 열기) 명령에서 순차 전용 처리(SEQONLY) 매개변수를 사용하여 성능을 향상시킬 수 있습니다.

SEQONLY 처리를 사용하려면 파일이 입력 전용 또는 출력 전용으로 열려 있어야 합니다. NBRRCDS 매개변수는 어떤 열기 옵션의 조합에도 사용될 수 있습니다. (OPNQRYF(조회 파일 열기) 명령은 가능한 한 순차 전용 처리를 사용합니다.) 고급 언어 스펙에 따라 고급 언어도 순차 전용 처리를 디폴트 값으로 사용할 수 있습니다. 예를 들어 파일을 읽기 전용으로 열고, 고급 언어 프로그램에 지정된 유일한 파일 조작이 순차 읽기 조작인 경우 고급 언어는 자동적으로 순차 전용 처리를 요구합니다.

주: 파일 위치지정 조작은 순차 읽기 조작이라고 할 수 없습니다. 따라서 위치지정 조작이 들어 있는 고급 언어 프로그램은 자동으로 순차 전용 처리를 요구하지 *않습니다*(RPG/400 언어의 STELL 조작 및 COBOL/400 언어의 START 조작은 파일 위치지정 조작의 예입니다.) 고급 언어 프로그램이 순차 전용 처리를 자동으로 요구할 수 없더라도 사용자가 OVRDBF 명령의 SEQONLY 매개변수를 사용하여 이를 요구할 수 있습니다.

순차 전용 처리를 지정할 경우 시스템 데이터베이스 주 기억장치 영역과 작업의 내부 자료 주 기억장치 영역간에 단위로 이동시킬 레코드 수를 지정할 수도 있습니다. 이동시킬 순차 전용 레코드 수를 지정하지 않으면 시스템은 4096바이트 버퍼에 맞는 레코드 수를 기초로 계산합니다.

또한 시스템은 보조 기억장치와 주 기억장치간에 하나의 단위로 이동되는 레코드 수를 제어하는 방법을 제공합니다. 자료가 실제로 저장된 순서대로 파일 내의 자료를 읽는 경우 OVRDBF 명령의 NBRRCDS 매개변수를 사용함으로써 작업의 성능을 향상시킬 수 있습니다.

주: 순차 전용 처리는 실제 파일이 액세스 경로와 동일한 순서인 경우를 제외하고는 키순 액세스 경로 파일과 같이 사용해서는 안 됩니다. 실제 자료가 액세스 경로의 순서로 재구성될 때까지는 SEQONLY(*YES) 처리로 인해 어플리케이션 성능이 저하될 수 있습니다.

순차 전용 처리의 열기 고려사항:

순차 전용 처리가 지정된 경우 파일을 여는 데 적용할 고려사항입니다. 시스템 순차 전용 처리가 허용되지 않는다고 판단하면 순차 전용 처리 요구가 받아들여지지 않는다는 메세지를 프로그램에 전달합니다. 그러나 해당 파일은 처리를 위해 여전히 열려 있습니다.

- 프로그램이 출력 전용으로 멤버를 열였고 SEQONLY(*YES)가 지정되었으며(레코드 수가 지정되지 않음) 열려진 멤버가 논리 멤버이거나 고유 키순 실제 멤버이거나 실제 멤버에 대한 다른 액세스 경로가 있는 경우 SEQONLY(*YES)가 SEQONLY(*NO)로 변경되어 출력 조작 시 프로그램이 가능한 오류(예를 들어, 중복 키, 변환 맵핑 및 선택/생략 오류 등)를 처리할 수 있도록 합니다. 시스템이 순차 전용 처리를 수행하도록 하려면 SEQONLY 매개변수를 *YES 값과 레코드 스펙 수 모두를 포함하도록 변경하십시오.
- 순차 전용 처리는 입력 전용(읽기) 조작이나 출력 전용(추가) 조작에만 지정할 수 있습니다. 프로그램이 갱신 조작이나 삭제 조작을 지정하면 순차 전용 처리는 시스템에서 허용되지 않습니다.
- 출력을 위해 파일이 열리는 경우 그 파일은 실제 파일이거나 하나의 실제 파일 멤버를 기초로 한 논리 파일이어야 합니다.
- 멤버가 출력 전용으로 열리는 경우에만, 순차 전용 처리는 확약 제어와 함께 지정될 수 있습니다.
- 확약 제어에 의해 열린 파일에 대해 순차 전용 처리가 사용되고 있으며 작업에 대해 롤백 조작이 수행되었을 경우 롤백 조작 시 작업의 기억장치 영역에 있는 레코드는 시스템 기억장치 영역에 기록되지 않으며 확약 제어 트랜잭션 저널에 표시되지 않습니다. 특정 확약 제어 처리에 대해 수행 중인 롤백 이전에 시스템 기억장치에 레코드가 전혀 기록되지 않았으면, 전체 확약 제어 트랜잭션은 저널에 반영되지 않습니다.
- 출력 전용의 경우 하나의 단위로 이동되도록 지정한 레코드 수와 강제율(force ratio)가 비교되어 필요한 만큼 자동적으로 조정됩니다. 레코드 수가 강제율보다 클 경우 레코드 수는 강제율과 같도록 줄어듭니다. 그 반대의 경우에는 강제율이 레코드 수와 같도록 줄어듭니다.
- 프로그램이 멤버를 출력 전용으로 열고 SEQONLY(*YES)가 지정되고(레코드 수는 지정되지 않음) 중복 키 또는 삽입 키 피드백이 요청되었으면, 레코드가 파일에 삽입될 때 레코드 별로 피드백이 제공되도록 SEQONLY(*YES)가 SEQONLY(*NO)로 변경됩니다.
- 다음 조건에 모두 해당되는 경우 블록 안의 레코드 수는 1로 변경됩니다.
  - 멤버가 출력 전용 처리로 열렸습니다.
  - 순차 전용 처리를 지정한 대체 조작이 유효하지 않습니다.
  - 열려진 파일이 레코드의 증분 수가 0으로 설정되었기 때문에 확장될 수 없습니다.
  - 파일 안의 사용 가능한 바이트 수가 레코드 블록에 맞는 바이트 수보다 적습니다.

순차 전용 처리가 지정되지 않고, OPNQRYF(조회 파일 열기) 명령을 사용하여 파일이 열릴 때 적용되는 고려사항은 다음과 같습니다. 이 조건이 충족되면 순차 전용 처리가 수행되고 조회 파일이 열렸음을 알리는 메세지가 전달됩니다.

- OPNQRYF 명령이 그룹 필드(GRPFLD) 매개변수에 한 개 이상의 필드명을 지정하거나 OPNQRYF이 그룹 처리를 필요로 하는 경우.
- OPNQRYF 명령이 한 개 이상의 필드를 지정하거나 UNIQUEKEY 매개변수에 *ALL을 지정하는 경우.
- SQL SELECT문에서 DISTINCT 옵션과 함께 뷰가 사용되면 SEQONLY(*YES)가 자동적으로 수행됩니다.

관련 개념

OPNQRYF(조회 파일 열기) 명령 사용
OPNQRYF(조회 파일 열기) 명령은 데이터베이스 파일에서 여러 자료 처리 기능을 수행할 수 있도록 하는 제어 언어(CL) 명령입니다. 이 주제에서는 OPNQRYF 명령을 사용한 조회 작성 방법, 주요 기능에 매개변수 지정 방법 및 고급 언어 프로그램에서 이를 사용하는 방법에 대해 설명합니다.

순차 전용 처리의 입/출력 고려사항:

순차 전용 처리가 지정될 때 파일의 입출력 조작에 대해 고려해야 할 사항입니다.

- 입력의 경우 프로그램은 입력 버퍼로부터 한 번에 한 개의 레코드를 받습니다. 입력 버퍼 내의 모든 레코드가 처리되면 시스템은 자동적으로 레코드의 다음 세트를 읽습니다.

  주: 레코드가 입력 버퍼에서 읽힌 후에 생긴 변경사항은 입력 버퍼에 반영되지 않습니다.
- 출력의 경우 프로그램은 출력 버퍼에 한 번에 한 개의 레코드를 이동시켜야 합니다. 출력 버퍼가 가득 차면 시스템은 자동적으로 레코드들을 데이터베이스에 추가합니다.

  주: 저널을 사용하는 경우 항목들이 모두 논리적으로 함께 발생한 것처럼 버퍼 전체가 한 번에 저널에 기록됩니다. 이 저널 처리는 레코드가 데이터베이스에 추가되기 전에 발생합니다.

  순차 전용 처리를 출력에 사용할 경우 파일에 대한 변경이 발생할 때마다 사용자가 변경사항 모두를 볼 수 없을 경우가 있습니다. 예를 들어 PGMA에 의해 사용되는 파일에 순차 전용이 지정되고 PGMA가 파일에 새 레코드를 추가하는 한편, SEQONLY 매개변수에 버퍼 내의 레코드 수 5를 지정된 경우 버퍼가 가득 찰 때에만 새로이 추가된 레코드가 데이터베이스로 보내집니다. 이 예에서 다섯 번째 레코드가 추가될 때 처음 다섯 개의 레코드가 데이터베이스로 보내지며 시스템 내의 다른 작업이 이들을 처리할 수 있게 됩니다.

  또한 순차 전용 처리를 출력에 사용할 경우 버퍼에서 데이터베이스로 레코드 이동 시 발생할 수 있는 오류를 사용자가 처리하지 않으면 데이터베이스에 일부 레코드가 추가되지 않을 수도 있습니다. 예를 들어 버퍼가 다섯 개의 레코드를 보유하고 있으며 버퍼 내의 세 번째 레코드에 파일의 다른 레코드와 중복되는 키가 있고 그 파일이 고유 키 파일로 정의된 것으로 가정하십시오. 이 경우 시스템이 버퍼에서 데이터베이스로 보낼 때 맨 처음의 레코드 두 개를 추가한 후, 세 번째에서 중복 키 오류가 발생하게 됩니다. 이 오류로 인해 버퍼 내의 세 번째, 네 번째 및 다섯 번째 레코드는 데이터베이스에 추가되지 *않습니다*.
- 자료 끝 강제 기능은 출력 조작에서 버퍼 내의 모든 레코드를 데이터베이스로 강제 기록시키는 데 사용할 수 있습니다(앞에서 설명한 것과 같이 고유 키를 갖는 파일에 중복 키를 유발시키는 레코드는 제외됨). 자료 끝 강제 기능은 특정 고급 언어에서만 사용할 수 있습니다.
- 다음 조건에 모두 해당되는 경우 블록 안의 레코드 수는 1로 변경됩니다.
  - 멤버가 출력 전용 처리 또는 순차 전용 처리로 열려 있습니다.

- 순차 전용 처리를 지정한 대체 조작이 유효하지 않습니다.
- 열려진 파일은 레코드의 증분 수가 0으로 설정되었으므로 확장될 것입니다.
- 파일 안의 사용 가능한 바이트 수가 레코드 블록에 맞는 바이트 수보다 적습니다.

순차 전용 처리의 닫기 고려사항:

이 고려사항은 순차 전용 처리가 지정된 경우 파일 닫기에 적용됩니다.

순차 전용 처리가 지정된 파일을 닫을 때 출력 버퍼 내에 있는 모든 레코드는 데이터베이스에 추가됩니다. 그러나 레코드에 오류가 발생할 경우 오류가 있는 레코드 다음에 오는 레코드들은 데이터베이스에 추가되지 않습니다.

동일 작업 내의 복수 프로그램이 순차 전용 출력 파일을 공유하고 있으면 출력 버퍼는 최종 닫기가 발생할 때까지 비워지지 않습니다. 그 결과 닫기(작업에서의 마지막 닫기는 제외)는 이 작업이나 다른 작업에 대해 아직 버퍼 내에 있는 레코드를 데이터베이스에 표시하지 않습니다.

데이터베이스 파일 처리를 위한 런타임 고려사항 요약

이 주제는 프로그램의 데이터베이스 파일 멤버 사용을 제어하는 매개변수와 이러한 매개변수를 지정할 수 있는 위치를 표시합니다.

두 곳 이상에서 지정될 수 있는 매개변수는 시스템이 그 값을 병합합니다. OVRDBF(데이터베이스 파일 대체) 명령 매개변수는 프로그램 매개변수보다 우선하며, OPNDBF(데이터베이스 파일 열기) 또는 OPNQRYF(조회 파일 열기) 명령 매개변수는 파일 작성 매개변수나 파일 변경 매개변수보다 우선합니다.

주: TOFILE, MBR, LVLCHK, SEQONLY, SHARE, WAITRCD 및 INHWRT 등을 제외한 다른 대체 매개변수는 OPNQRYF 명령에 의해 무시됩니다.

다음 표는 제어 언어(CL) 명령에 지정된 데이터베이스 처리 옵션을 표시한 것입니다.

표 40. CL 명령에 지정된 데이터베이스 처리 옵션

설명	매개변수	명령				
		CRTPF, CRTLF	**CHGPF, CHGLF**	**OPNDBF**	**OPNQRYF**	**OVRDBF**
파일명	파일	X	X[1]	X	X	X
라이브러리명		X	X[2]	X	X	X
멤버명	MBR	X		X	X	X
멤버 처리 옵션	OPTION			X	X	
레코드 형식 잠금 상태	RCDFMTLCK					X
열기 후 파일 시작 위치	POSITION					X
프로그램이 순차 처리만 수행함	SEQONLY			X	X	X
키순 액세스 경로 무시	ACCPTH			X		

표 40. CL 명령에 지정된 데이터베이스 처리 옵션 (계속)

설명	매개변수	명령				
		CRTPF, CRTLF	**CHGPF, CHGLF**	**OPNDBF**	**OPNQRYF**	**OVRDBF**
파일 잠금 대기 시간	WAITFILE	X	X			X
레코드 잠금 대기 시간	WAITRCD	X	X			X
대체 방지	SECURE					X
보조 기억장치에서 주 기억장치로 보내지는 레코드 수	NBRRCDS					X
다른 프로그램과 열린 자료 경로 공유	SHARE	X	X			X
형식 선택 프로그램	FMTSLR	X[3]	X[3]			X
강제율	FRCRATIO	X	X			X
기록 금지	INHWRT					X
레벨 검사 레코드 형식	LVLCHK	X	X			X
만기일 검사	EXPCHK					X
만기일	EXPDATE	X[4]	X[4]			X
강제 액세스 경로	FRCACCPTH	X	X			
확약 제어	COMMIT			X	X	
파일 끝 지연	EOFDLY					X
중복 키 검사	DUPKEYCHK			X	X	
삭제 레코드 공간의 재사용	REUSEDLT	X[4]	X[4]			
코드화 문자 세트 ID	CCSID	X[4]	X[4]			
정렬 순서	SRTSEQ	X	X		X	
언어 ID	LANGID	X	X		X	

[1] 파일명: CHGPF 및 CHGLF 명령은 식별용으로만 파일명을 사용합니다. 파일명은 변경할 수 없습니다.

[2] 라이브러리명: CHGPF 및 CHGLF 명령은 식별용으로만 파일명을 사용합니다. 라이브러리명은 변경할 수 없습니다.

[3] 형식 선택 프로그램: CRTLF 및 CHGLF 명령에서만 사용됩니다.

[4] 만기일, 삭제 레코드 재사용 및 코드화 문자 세트 ID: CRTPF 및 CHGPF 명령에서만 사용됩니다.

다음 표는 프로그램에 지정된 데이터베이스 처리 옵션입니다.

표 41. 프로그램에 지정된 데이터베이스 처리 옵션

설명	RPG/400 언어	COBOL/400 언어	iSeries BASIC	iSeries PL/I	iSeries Pascal
파일명	X	X	X	X	X
라이브러리명			X	X	X
멤버명			X	X	X
프로그램 레코드 길이	X	X	X	X	X
멤버 처리 옵션	X	X	X	X	X
레코드 형식 잠금 상태			X	X	
프로그램이 사용할 레코드 형식	X		X		
레코드의 실제 파일 멤버 지우기		X	X		X
프로그램이 순차 처리만 수행함	X	X		X	X
키순 액세스 경로 무시	X	X	X	X	X
다른 프로그램과 열린 자료 경로 공유				X	X
레벨 검사 레코드 형식	X	X	X	X	
확약 제어	X	X		X	
중복 키 검사		X			
주: 제어 언어(CL) 프로그램에서도 이러한 매개변수들을 지정할 수 있습니다. CL 명령에 지정될 수 있는 데이터베이스 처리 옵션에 대해 자세히 알려면 134 페이지의 표 40을 참조하십시오.					

데이터베이스 성능에 대한 기억장치 풀 페이징 옵션 효과

공유된 풀의 페이징 옵션은 데이터베이스 파일을 읽고 변경하는 성능에 중요한 영향을 줄 수 있습니다.

- *FIXED의 페이징 옵션으로 프로그램은 다음과 같이 사용하는 기억영역을 최소화합니다.
  - 보조 기억장치에서 주 기억장치로 보다 작은 블록을 사용하여 자료 이동
  - 보조 기억장치로의 빈번한 파일 변경 기록(기존 레코드의 갱신이나 새로운 레코드의 추가)

  이 옵션으로 시스템은 페이징 옵션이 추가되기 전과 마찬가지로 많은 것을 수행합니다.
- *CALC 페이징 옵션은 데이터베이스 파일을 읽고 갱신하는 프로그램의 성능을 향상시켜 줍니다. 공유 풀 내에서 사용 가능한 기억영역이 충분한 경우 프로그램은 다음을 수행합니다.
  - 주 기억장치에서 보조 기억장치로 보다 큰 블럭을 사용하여 자료 이동
  - 보조 기억장치로의 빈번하지 않은 파일 변경 기록

  데이터베이스 파일의 페이징 작업은 파일 사용과 기억영역의 가용성에 따라서 매우 동적으로 이루어 집니다. 자주 참조되는 파일은 가끔 참조되는 파일보다는 상주해 있는 편이 더 좋습니다. 기억장치는 일반적인 자료에 대해서는 캐시와 유사하게 사용됩니다. *CALC 페이징 옵션을 사용하면 I/O 조작의 총 횟수가 줄어들게 됩니다.

관련 개념

성능

데이터베이스 파일 열기

이 주제에서는 프로그램에서 OPNQRYF(조회 파일 열기) 및 OPNDBF(데이터베이스 파일 열기) 명령을 사용하여 데이터베이스 파일 멤버를 여는 방법에 대해 설명합니다. 고급 언어 프로그램 작성에 관한 예, 성능 고려사항 및 안내서가 포함됩니다. 또한 발생 가능한 일반적인 오류에 대해 설명합니다.

데이터베이스 파일 멤버 열기

고급 언어 프로그램에서는 명령문으로 데이터베이스 파일을 열 수 있습니다. 또한 OPNDBF(데이터베이스 파일 열기) 및 OPNQRYF(조회 파일 열기) 등의 제어 언어(CL) 열기 명령을 사용할 수 있습니다.

프로그램에서 데이터베이스 파일을 사용하려면 프로그램이 데이터베이스 파일에 대해 열기 조작을 수행해야 합니다. 일부 프로그래밍 언어에서는 열기 조작을 지정하지 않을 경우 자동적으로 파일을 열어줍니다. 프로그램이나 OVRDBF(데이터베이스 파일 대체) 명령에서 멤버명을 지정하지 않은 경우에는 파일의 첫 번째 멤버(작성 날짜 및 시간을 근거로 하여)가 사용됩니다.

멤버명을 지정하면 정확한 파일명은 가지고 있으나 멤버명이 없는 파일은 무시됩니다. 여러 라이브러리에 FILEA로 명명된 여러 개의 데이터베이스 파일이 있을 경우 열리는 멤버는 라이브러리 리스트의 첫 번째 위치에 있는 것입니다. 예를 들어, 라이브러리 리스트에 LIB1, LIB2, LIB3이 들어 있고 이 세 라이브러리에 FILEA라는 이름의 파일이 포함되어 있으며, LIB3에 있는 FILEA만 MBRA라는 멤버를 가지고 있는 것으로 가정하십시오. 이 경우 LIB3에 있는 FILEA의 MRBA라는 멤버만 열리고 다른 FILEA들은 무시됩니다.

시스템은 멤버를 찾은 후, 프로그램을 데이터베이스 파일에 연결시킵니다. 이로써 프로그램은 해당 파일에 대한 입출력 조작을 수행할 수 있게 됩니다. 고급 언어 프로그램에서 파일 열기에 대한 자세한 내용은 해당 고급 언어 주제 콜렉션을 참조하십시오.

또한 OPNDBF 명령과 OPNQRYF 명령으로 데이터베이스 파일 멤버를 열 수 있습니다. OPNDBF 명령은 공유 파일을 열기 위한 작업의 초기 프로그램에 유용합니다. OPNQRYF 명령은 프로그램 밖에서 레코드를 선택하고 배열하는 데 매우 효과적입니다. 그러면 프로그램은 OPNQRYF 명령에 의해 제공되는 정보를 사용하여 필요한 자료만 처리할 수 있습니다.

관련 개념

제어 언어(CL)

관련 참조

OPNDF(데이터베이스 파일 열기) 명령

조회 파일 열기(OPNQRYF) 명령

OPNDBF(데이터베이스 파일 열기) 명령 사용

일반적으로 OPNDBF(데이터베이스 파일 열기) 명령을 사용할 경우 명령 매개변수 값에 디폴트 값을 사용할 수 있습니다. 경우에 따라서 디폴트 값을 사용하는 대신 특정값을 지정할 수도 있습니다. 다음은 사용자가 지정할 수 있는 매개변수를 나열한 것입니다.

ACCPTH 매개변수

파일이 키순 액세스 경로를 가지며 (1) 열기 옵션이 *OUT이거나 (2) 열기 옵션이 *INP 또는 *ALL인데 반하여 프로그램에서 키순 액세스 경로를 사용하지 않는 경우 사용자는 OPNDBF 매개변수에 ACCPTH(*ARRIVAL)을 지정할 수 있습니다. 키순 액세스 경로를 무시하면 작업 성능을 향상시킬 수 있습니다.

COMMIT 매개변수

어플리케이션 프로그램이 확약 제어를 사용하는 경우 *YES를 지정하십시오. *YES를 지정하는 경우 사용자의 작업은 확약 제어 환경에서 수행 중이어야 하며(STRCMTCTL(확약 제어 시작) 명령이 처리됨), 그렇지 않으면 OPNDBF 명령이 실패합니다. 어플리케이션 프로그램이 확약 제어를 사용하지 않을 경우에는 디폴트 값 *NO를 사용하십시오.

DUPKEYCHK 매개변수

중복 키 피드백 필요 여부를 지정합니다. *YES를 지정하면 중복 키 피드백이 I/O 조작에 리턴됩니다. *NO를 지정하면 중복 키 피드백이 I/O 조작에 리턴되지 않습니다. 어플리케이션 프로그램이 COBOL/400 언어 또는 C/400® 언어로 작성되지 않았을 경우 또는 COBOL 또는 C 프로그램이 리턴된 중복 키 피드백 정보를 사용하지 않을 경우 디폴트 값(*NO)를 사용하십시오.

MBR 매개변수

파일의 첫 번째 멤버가 아닌 다른 멤버를 열려고 하는 경우 그 멤버의 이름을 지정하거나 OPNDBF 명령 이전에 OVRDBF(데이터베이스 파일 대체) 명령을 발행해야 합니다.

주: 후속 프로그램에서의 열기에서(첫 번째 멤버가 아닌) 다른 멤버를 사용하려면 OVRDBF 명령에 멤버명을 지정해야 합니다.

OPNID 매개변수

파일명이 아닌 ID를 사용하려면 이를 지정해야 합니다. 열기 ID는 다른 제어 언어(CL) 명령에서 파일을 처리하는 데 사용할 수 있습니다. 예를 들어 CLOF(파일 닫기) 명령은 ID를 사용하여 닫혀질 파일을 지정합니다.

OPNSCOPE 매개변수

이 매개변수는 열린 자료 경로(ODP)의 유효 범위를 지정합니다. 디폴트 활성 그룹으로부터 요구가 있었으며, ODP가 명령을 발행한 프로그램의 호출 레벨까지 확장된 경우 *ACTGRPDFN를 지정하십시오. 또 다른 활성 그룹으로부터 요구가 있으면 ODP는 그 활성 그룹에까지 확장됩니다. ODP가 명령을 발행하는 프로그램의 활성 그룹으로까지 확장되게 하려면 *ACTGRP를 지정하십시오. ODP가 해당 작업까지 확장되게 하려면 *JOB을 지정하십시오. 이 매개변수와 TYPE 매개변수를 지정하면 오류 메세지가 발생됩니다.

OPTION 매개변수

어플리케이션 프로그램이 입력 전용 처리(레코드를 갱신하지 않고 읽기만 하는 것)를 사용하는 경우 *INP 옵션을 지정하십시오. 그러면 시스템이 가능한 갱신에 대한 각 레코드의 잠금을 시도하지 않고도 레코드를 읽을 수 있습니다. 어플리케이션 프로그램이 출력 전용 처리(파일에 레코드를 기록하지만 기존의 레코드를 읽거나 갱신하지 않음)를 사용하는 경우 *OUT 옵션을 지정하십시오.

주: 프로그램이 활동 레코드에 대해 직접 출력 조작을 수행하는 경우(상대 레코드 번호로 갱신), *OUT 대신 *ALL을 지정해야 합니다. 프로그램이 삭제 레코드에 대해서만 직접 출력 조작을 수행할 경우 *OUT을 지정해야 합니다.

SEQONLY 매개변수

후속 어플리케이션 프로그램이 파일을 순차적으로 처리할 경우 *YES를 지정하십시오. 이 매개변수는 시스템 자료 버퍼와 프로그램 자료 버퍼간에 전송되어야 할 레코드 수를 지정하는 데 사용되기도 합니다. SEQONLY(*YES)는 OPNDBF(데이터베이스 파일 열기) 명령에 OPTION(*INP) 또는 OPTION(*OUT)을 지정하지 않는 한 허용되지 않습니다. 순차 전용 처리는 실제 파일 자료가 액세스 경로순인 경우를 제외하고는 키순 액세스 경로 파일과 함께 사용해서는 안 됩니다.

TYPE 매개변수

어플리케이션 프로그램에서 모니터링되지 않은 예외가 발생할 때 수행되기를 원하는 작업을 지정합니다. *NORMAL을 지정하면 다음 조작 중 하나가 발생합니다.

- 호출 스택의 상위 단계에서 열린 파일을 닫기 위해 RCLRSC(자원 재생) 명령을 사용할 수 있습니다.
- 사용자가 사용하는 고급 언어에 의해 닫기 작업이 수행될 수 있습니다.

파일을 다시 열지 않고 어플리케이션을 계속하려면 *PERM을 지정하십시오. 다음 두 가지 모두 발생할 경우 TYPE(*NORMAL)은 파일을 닫습니다.

- 프로그램에서 오류 메세지를 수신할 때
- 호출 스택의 상위 단계에 있는 파일이 열려 있을 때

TYPE(*PERM)을 사용하면 오류 메세지가 나와도 파일이 계속 열려 있도록 합니다. OPNSCOPE 매개변수를 지정한 경우 이 매개변수를 지정해서는 안 됩니다.

OPNQRYF(조회 파일 열기) 명령 사용

OPNQRYF(조회 파일 열기) 명령은 데이터베이스 파일에서 여러 자료 처리 기능을 수행할 수 있도록 하는 제어 언어(CL) 명령입니다. 이 주제에서는 OPNQRYF 명령을 사용한 조회 작성 방법, 주요 기능에 매개변수 지정 방법 및 고급 언어 프로그램에서 이를 사용하는 방법에 대해 설명합니다.

기본적으로 OPNQRYF 명령은 처리 프로그램과 데이터베이스 레코드간의 필터 역할을 합니다. 데이터베이스 파일은 실제 파일이나 논리 파일이 될 수 있습니다. CRTPF(실제 파일 작성) 또는 CRTLF(논리 파일 작성) 명령과 달리 OPNQRYF 명령은 자료 처리를 위해 임시 파일만 작성하며 영구 파일은 작성하지 않습니다.

OPNQRYF 명령에는 자료 서술 스펙(DDS), CRTPF 및 CRTLF 명령의 기능과 유사한 기능이 있습니다. DDS는 파일을 작성하기 위해 소스문과 별개의 절차를 필요로 합니다. OPNQRYF 명령을 사용하면 DDS를 사용하지 않고 동적 정의를 사용할 수 있습니다. OPNQRYF 명령은 DDS 기능을 전부 지원하지는 않으나 DDS 기능으로는 할 수 없는 중요한 기능을 지원합니다. 또한, iSeries용 Query를 사용하여 OPNQRYF 명령이 수행하는 기능의 일부를 수행할 수 있습니다. 그러나 OPNQRYF 명령은 프로그래머의 툴로서 더 유용합니다.

OPNQRYF 명령 매개변수는 SQL SELECT문과 유사한 여러 가지 기능을 갖고 있습니다. 예를 들면, FILE 매개변수는 SQL FROM문과 유사하고, QRYSLT 매개변수는 SQL WHERE문과 유사하며, GRPFLD 매개변수는 SQL GROUP BY문과 유사하고, GRPSLT 매개변수는 SQL HAVING문과 유사합니다.

다음은 OPNQRYF 명령에서 제공하는 주요 기능 리스트입니다.

- 동적 레코드 선택
- 동적 키순 액세스 경로
- 결합의 동적 키순 액세스 경로
- 동적 결합
- 2차 파일에서 누락 레코드 처리
- 고유 키 처리
- 맵핑된 필드 정의
- 그룹 처리
- 최종 합계 전용 처리
- 성능 향상
- 개방 조회 식별자(ID)
- 정렬 순서 처리

OPNQRYF 명령을 이해하려면 파일에서의 형식 사용 및 다른 형식을 가진 파일의 사용 등 두 가지 처리 방식을 잘 알고 있어야 합니다. OPNQRYF 명령의 일반적인 용도는 자료의 선택, 배열 및 형식화이며, 이렇게 함으로써 고급 언어 프로그램에서 자료를 순차적으로 읽을 수 있습니다.

관련 개념

SQL 프로그래밍

제어 언어(CL)

관련 참조

조회 파일 열기(OPNQRYF) 명령

OPNQRYF(조회 파일 열기) 명령으로 조회 작성:

조회를 작성하려면 OPNQRYF(조회 파일 열기) 명령을 사용할 수 있습니다. 또는, iSeries Navigator에서 SQL 스크립트 실행을 사용하여 조회를 작성할 수 있습니다.

관련 개념

SQL 스크립트 실행 인터페이스를 사용하여 데이터베이스 조회

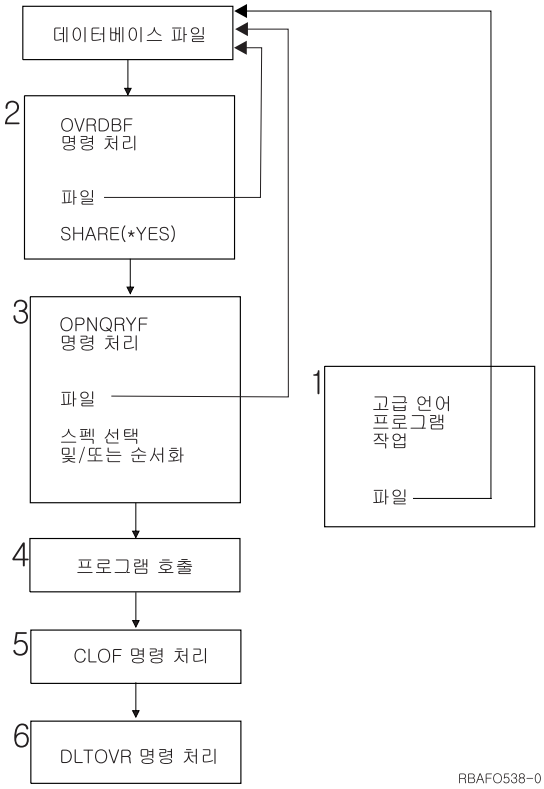
관련 참조

조회 파일 열기(OPNQRYF) 명령

파일의 기존 레코드 형식 사용:

OPNQRYF(조회 파일 열기) 명령은 레코드를 선택하고 프로그램에서는 그 선택값에 부합되는 레코드만 처리합니다. 이러한 방법을 사용하여 레코드 세트를 선택하거나 레코드를 저장 순서와는 다른 순서로 리턴시키거나 모두를 수행할 수 있습니다.

Code 필드가 D인 레코드만 프로그램에서 처리하는 것으로 가정하십시오. *Code* 필드가 D인 레코드만 있는 것처럼 프로그램을 작성합니다. 즉 프로그램에 선택 조작을 코딩하지 않습니다. 그런 다음 사용자는 OPNQRYF 명령을 수행하고 *Code* 필드가 D인 레코드만 프로그램으로 리턴되도록 지정합니다. 다음 도표는 OPNQRYF 명령을 사용하여 레코드를 선택하고 순서를 설정하는 한 예입니다.



1 외부 서술 자료를 사용하는 일반 프로그램처럼 데이터베이스 파일을 처리하는 고급 언어 프로그램을 작성하십시오. 한 개의 형식만 사용할 수 있으며 그 형식이 파일 안에 존재해야 합니다.

2 OVRDBF(데이터베이스 파일 대체) 명령을 실행하여 처리된 파일 및 멤버와 SHARE(*YES)를 지정하십시오. (멤버가 영구적으로 SHARE(*YES)로 변경되고 사용할 멤버가 첫 번째 멤버이거나 유일한 멤버일 경우 이 단계는 필요하지 않습니다.)

OPNQRYF 명령에 지정된 파일명을 대체하지 않는 한, OVRDBF 명령은 OPNQRYF 명령 다음에 수행될 수 있습니다. 여기 다루어진 예에서는 OVRDBF 명령이 먼저 나옵니다.

OPNQRYF 명령과 함께 OVRDBF 명령을 사용하는 데에는 몇 가지 제한이 있습니다. 예를 들면 MBR(*ALL)은 오류 메세지를 유발시키며 파일이 열리지 않게 합니다.

3 OPNQRYF 명령을 수행하십시오. 이 명령은 데이터베이스 파일, 멤버, 형식명, 모든 선택 옵션, 모든 순서 옵션, 그리고 열린 파일의 유효 범위 등을 지정합니다.

4 1단계에서 작성된 고급 언어(HLL) 프로그램을 호출합니다. 고급 언어 이외에 CPYFRMQRYF(조회 파일에서 복사) 명령도 OPNQRYF 명령에 의해 작성된 파일을 처리하기 위해 사용할 수 있습니다. 기타 제어 언어(CL) 명령(예를 들어, CPYF(파일 복사) 및 DSPPFM(실제 파일 멤버 표시) 명령)과 유틸리티(예: Query)는 OPNQRYF 명령으로 작성한 파일에 대해서는 작동하지 않습니다.

5 사용자가 파일을 계속 열어두려 하지 않는 한, 3단계에서 열었던 파일을 닫으십시오. CLOF(파일 닫기) 명령으로 파일을 닫을 수 있습니다.

6 DLTOVR(대체 속성 삭제) 명령으로 2단계에서 지정된 대체 속성을 삭제하십시오. 대체 속성을 매번 삭제할 필요는 없으나 일관성을 위해 모든 예에서 명령이 표시됩니다.

관련 개념

191 페이지의『작업에서 공유된 파일』

어플리케이션 프로그램이 OPNQRYF(조회 파일 열기) 명령으로 작성된 열린 자료 경로를 사용하기 위해서는 프로그램이 반드시 조회 파일을 공유해야 합니다. 프로그램에서 조회 파일을 공유 상태로 열지 않으면 컴파일 시 원래 사용하도록 지정되었던 파일을 완전히 열게 됩니다(OPNQRYF 명령에 의하여 작성된 조회 열린 자료 경로가 아님).

다른 레코드 형식의 파일 사용:

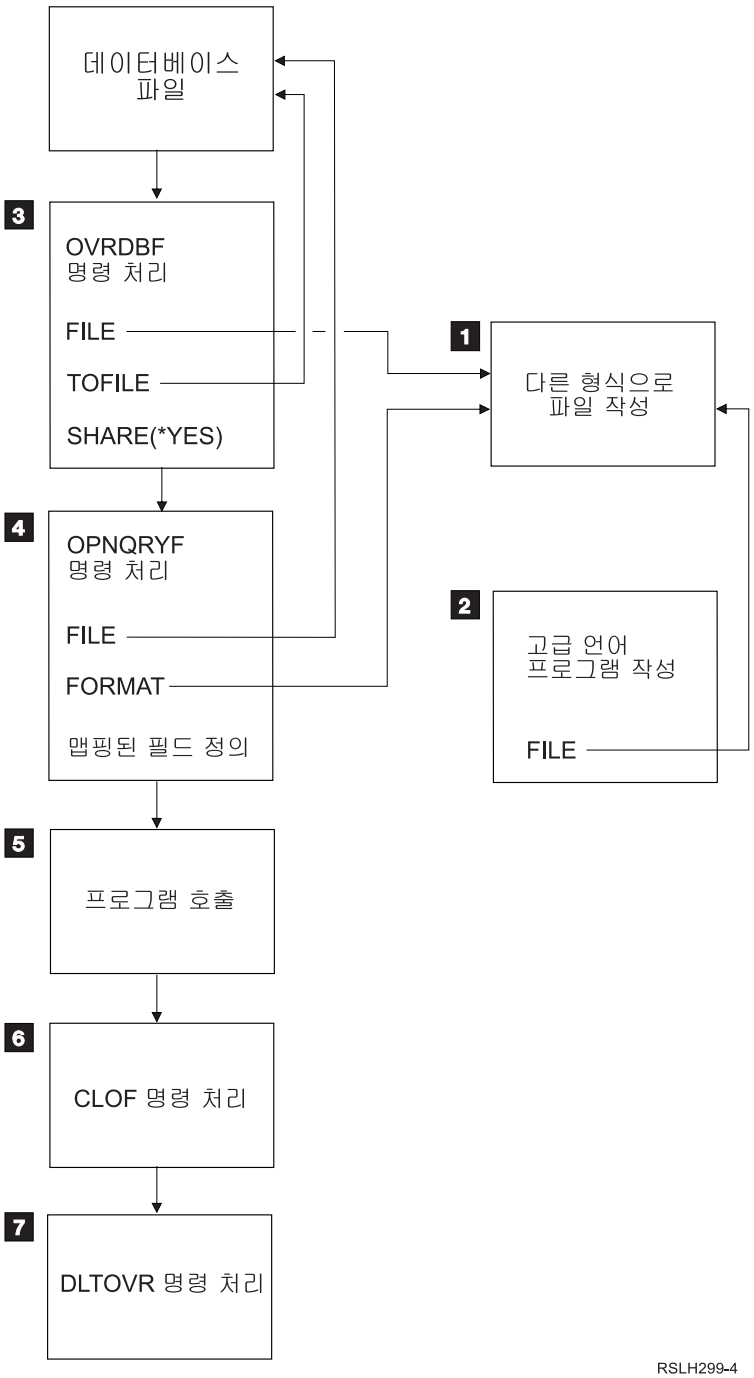
OPNQRYF(조회 파일 열기) 명령의 보다 확장된 기능을 사용하기 위해(예를 들어, 다른 파일로부터 동적으로 레코드를 결합하는 것) 다른 레코드 형식이 들어 있는 새로운 파일을 정의해야 합니다.

이 새 파일은 사용자가 처리하려는 파일과는 별개이며, OPNQRYF 명령으로 작성하려는 필드가 들어 있습니다. 이 강력한 기능을 사용하여 데이터베이스 레코드에는 현재 존재하지 않으나 이들 레코드들로부터 파생될 수 있는 필드를 정의할 수 있습니다.

고급 언어 프로그램을 코딩할 때 다른 형식을 가진 파일명을 지정함으로써 기존 필드와 파생된 필드 모두의 외부 서술 필드 정의가 프로그램에 의해 처리될 수 있도록 합니다.

고급 언어 프로그램을 호출하기 전에 OVRDBF(데이터베이스 파일 대체) 명령을 지정하여 프로그램 파일명을 개방 조회 파일로 보내야 합니다. OPNQRYF 명령에서는 고급 언어 프로그램에서 사용될 데이터베이스 파일 및 특수 형식을 가진 새로운 파일을 모두 지정하십시오. 조회하려는 파일에 SHARE(*YES)가 지정되지 않은 경우 OVRDBF 명령에 SHARE(*YES)를 지정해야 합니다.

다음 도표는 프로세스의 흐름을 나타낸 것입니다.



RSLH299-4

1 다른 레코드 형식을 가진 파일에 대해 자료 서술 스펙(DDS)을 지정하고 파일을 작성하십시오. 이 파일에는 고급 언어 프로그램으로 처리하려는 필드가 들어 있습니다. 일반적으로, 이 파일에는 자료가 들어 있지 않으며 멤버를 필요로 하지 않습니다. 사용자는 주로 이 파일을 키가 없는 실제 파일로서 작성합니다. 필드 참조 파일을 사용하여 필드를 서술할 수 있습니다. 레코드 형식명은 지정된 데이터베이스 파일의 레코드 형식명과 다를 수 있습니다. 이러한 기능을 위해 임의의 데이터베이스 파일이나 DDS 파일을 사용할 수 있습니다. 파일은 논리 파일일 수도 있고 색인화 파일일 수도 있습니다. 이 파일에는 멤버가 하나 이상일 수 있으며 자료가 있거나 없을 수도 있습니다.

2 1단계에서 작성한 레코드 형식을 가진 파일을 처리할 고급 언어 프로그램을 작성하십시오. 이 프로그램에서 자료가 들어 있는 데이터베이스 파일의 이름은 지정하지 마십시오.

3 OVRDBF 명령을 실행하십시오. FILE 매개변수에 다른(새로운) 레코드 형식을 가진 파일명을 지정하십시오. TOFILE 매개변수에 조회하려는 데이터베이스 파일명을 지정하십시오. MBR 매개변수에 멤버명을 지정할 수도 있습니다. 조회하려는 데이터베이스 파일 멤버에 SHARE(*YES)가 지정되지 않은 경우 OVRDBF 명령에 SHARE(*YES)를 지정해야 합니다.

4 OPNQRYF 명령을 실행하십시오. FILE 매개변수에 조회될 데이터베이스 파일을 지정하고 FORMAT 매개변수에 1단계에서 작성된 다른(새로운) 형식을 가진 파일명을 지정하십시오. OPNQRYF 명령에서는 데이터베이스 파일의 자료를 1단계에서 작성된 형식으로 맵핑하는 법을 기술하기 위해 맵핑된 필드 정의가 필요할 수 있습니다. 또한 선택 옵션, 순서 옵션, 열린 파일의 영향 범위를 지정할 수 있습니다.

5 2단계에서 작성한 고급 언어 프로그램을 호출하십시오.

6 4단계에서 FILE 매개변수에 지정한 첫 번째 파일은 SHARE(*YES)로서 OPNQRYF에 의해 열렸으며, 아직도 열려 있는 상태이므로 그 파일을 닫아야 합니다. CLOF(파일 닫기) 명령을 사용하여 파일을 닫을 수 있습니다.

7 3단계에서 지정된 대체 속성을 삭제하십시오.

앞에서 설명한 단계는 외부 서술 자료를 사용하는 정상적인 흐름을 보여준 것입니다. 각각의 OPNQRYF 명령에 대해 고유한 DDS 및 레코드 형식을 반드시 작성할 필요는 없습니다. 기존의 레코드 형식을 다시 사용할 수도 있으나 레코드 형식 내의 모든 필드는 실제로 존재하는 데이터베이스 파일 내의 실제 필드이거나 맵핑된 필드로 정의되어야 합니다. 프로그램 서술 자료를 사용하는 경우 프로그램을 언제나 작성할 수 있습니다.

OPNQRYF 명령에 의해 작성된 자료를 보유하기 위해 1단계에서 작성된 파일을 사용할 수 있습니다. 예를 들어, 다른 형식의 파일에 자료를 복사하는 고급 언어 처리 프로그램으로 5단계를 대신하거나 CPYFRMQRYF(조회 파일에서 복사) 명령을 사용할 수 있습니다. CPYF(파일 복사) 명령은 사용할 수 없습니다. 5단계 다음에 CPYF 명령이나 조회를 수행할 수 있습니다.

OPNQRYF(조회 파일 열기) 명령으로 CL 프로그램 코딩:

OPNQRYF(조회 파일 열기) 명령에는 코딩 오류를 방지할 수 있는 다음과 같은 기본 규칙이 있습니다.

1. 앰퍼샌드(&) 없이 데이터베이스 파일로부터 선택 필드를 지정하십시오. 제어 언어(CL) 프로그램에서 DCL 또는 DCLF로 선언된 필드는 앰퍼샌드를 필요로 합니다.
2. CL 프로그램에서 DCL 또는 DCLF로 정의된 필드를 작은 따옴표로 묶으십시오(예: '&testfld').
3. 모든 매개변수 비교는 문자 필드에 비교될 때는 큰 따옴표로, 숫자 필드에 비교될 때는 작은 따옴표로 묶으십시오.

다음 예에서 필드 INVCUS 및 INVPRD는 문자 자료로서 정의됩니다.

```
QRYSLT('INVCUS *EQ "' *CAT &K1CUST *CAT '" *AND +
                INVPRD *GE "' *CAT &LPRD *CAT '" *AND +
                INVPRD *LE "' *CAT &HPRD *CAT '"')
```

필드를 숫자 자료로 정의한 경우 QRYSLT 매개변수는 다음과 같이 표시됩니다.

```
QRYSLT('INVCUS *EQ ' *CAT &K1CUST *CAT ' *AND +
                INVPRD *GE ' *CAT &LPRD *CAT ' *AND +
                INVPRD *LE ' *CAT &HPRD *CAT ' ')
```

관련 개념

『OPNQRYF(조회 파일 열기) 명령에 주 사용』

사용법 주의사항에서는 OPNQRYF(조회 파일 열기) 명령의 주요 기능에 매개변수를 지정하는 방법과 고급 언어 프로그램으로 OPNQRYF 명령을 사용하는 방법에 대해 설명합니다.

길이 0인 리터럴과 포함(*CT) 함수:

OPNQRYF(조회 파일 열기) 명령에서 길이 0인 리터럴은 인용부호 사이에 아무것도 없는(공백조차도 없는) 인용 스트링(″″)으로 표시됩니다. 길이 0인 리터럴 지원은 포함(*CT) 함수의 비교 인수로 사용될 경우 비교 결과를 변경합니다.

버전 2, 릴리스 1, 수정 1에서 길이 0의 리터럴 개념이 도입되었습니다.

다음 명령문을 참조하십시오.

```
QRYSLT('field *CT " "')
```

길이 0인 리터럴 지원으로 명령문은 자료가 들어 있는 레코드를 리턴합니다. 이는 임의의 문자 수 뒤에 임의의 문자 수가 위치하는 와일드 카드 비교입니다. 그것은 다음과 같습니다.

```
'field = %WLDCRD("* *")'
```

길이 0인 리터럴이 지원되기 전(버전 2, 릴리스 1, 개정 1 이전)에는 인수(″″)가 1바이트 공백으로 해석되었습니다. 이 명령문은 필드의 어느 곳엔가 1바이트 공백을 포함하는 레코드를 리턴시킵니다. 기본적으로 이는 임의의 문자 수 뒤에 공백, 그 뒤에 임의의 문자 수가 위치하는 와일드 카드 비교입니다. 그것은 다음과 같습니다.

```
'field = %WLDCRD("* *")'
```

포함 함수를 사용하여 버전 2, 릴리스 1, 수정 1이전과 같은 결과를 얻으려면 명시적으로 공백을 찾도록 QRYSLT를 코딩해야 합니다.

```
QRYSLT('field *CT " "')
```

관련 개념

『OPNQRYF(조회 파일 열기) 명령에 주 사용』

사용법 주의사항에서는 OPNQRYF(조회 파일 열기) 명령의 주요 기능에 매개변수를 지정하는 방법과 고급 언어 프로그램으로 OPNQRYF 명령을 사용하는 방법에 대해 설명합니다.

OPNQRYF(조회 파일 열기) 명령에 주 사용:

사용법 주의사항에서는 OPNQRYF(조회 파일 열기) 명령의 주요 기능에 매개변수를 지정하는 방법과 고급 언어 프로그램으로 OPNQRYF 명령을 사용하는 방법에 대해 설명합니다.

주:

1. 명령 입력 행에서 OPNSCOPE(*ACTGRPDFN)나 TYPE(*NORMAL) 매개변수와 함께 OPNQRYF 명령을 수행할 경우 OPNQRYF 명령이 성공적으로 수행된 이후에 발생하는 오류 메세지로 인해 파일이 닫히지 않습니다. TYPE(*NORMAL)을 사용했을 때 버전 2 릴리스 3 이전에서는 그러한 메세지에 의해 파일이 닫혔었습니다. 시스템은 오류 메세지가 발생할 경우 RCLRSC(자원 재생) 명령을 자동적으로 수행합니다. (단 시스템이 그 명령에서 오류를 감지할 때 나오는 메세지 CPF0001은 제외됩니다.) 그러나 RCLRSC 명령은 RCLRSC 명령이 수행되었던 레벨보다 호출 스택의 상위 레벨에 있는 디폴트 활성 그룹으로부터 열려진 파일만을 닫습니다.
2. OPNQRYF 명령으로 순차 처리를 수행하는 프로그램을 실행한 다음 파일 위치는 보통 파일 끝에 오게 됩니다. 같은 프로그램을 수행하거나 같은 파일을 사용하는 다른 프로그램을 수행하려면 파일을 위치시키거나 동일한 OPNQRYF 명령으로 파일을 닫은 후 열어야 합니다. POSDBF(데이터베이스 파일 위치 지정) 명령을 사용하여 파일을 위치시킬 수 있습니다. 어떤 경우에는 고급 언어 프로그램문을 사용할 수도 있습니다.

관련 개념

145 페이지의 『길이 0인 리터럴과 포함(*CT) 함수』
OPNQRYF(조회 파일 열기) 명령에서 길이 0인 리터럴은 인용부호 사이에 아무것도 없는(공백조차도 없는) 인용 스트링(″″)으로 표시됩니다. 길이 0인 리터럴 지원은 포함(*CT) 함수의 비교 인수로 사용될 경우 비교 결과를 변경합니다.

『DDS를 사용하지 않고 레코드 선택』
동적 레코드 선택을 사용하면 자료 서술 스펙(DDS)을 사용하지 않고도 파일에서 레코드의 서브세트를 요청할 수 있습니다.

관련 태스크

144 페이지의 『OPNQRYF(조회 파일 열기) 명령으로 CL 프로그램 코딩』
OPNQRYF(조회 파일 열기) 명령에는 코딩 오류를 방지할 수 있는 다음과 같은 기본 규칙이 있습니다.

DDS를 사용하지 않고 레코드 선택:

동적 레코드 선택을 사용하면 자료 서술 스펙(DDS)을 사용하지 않고도 파일에서 레코드의 서브세트를 요청할 수 있습니다.

예를 들어 특정값이나 특정 범위 내의 값을 가진 레코드(예를 들어, 1000에서 1050 사이의 모든 고객 번호)를 선택할 수 있습니다. OPNQRYF(조회 파일 열기) 명령은 이러한 기능 및 그 외의 선택 기능을 조합시켜 강력한 레코드 선택 기능을 수행할 수 있습니다.

관련 개념

145 페이지의 『OPNQRYF(조회 파일 열기) 명령에 주 사용』
사용법 주의사항에서는 OPNQRYF(조회 파일 열기) 명령의 주요 기능에 매개변수를 지정하는 방법과 고급 언어 프로그램으로 OPNQRYF 명령을 사용하는 방법에 대해 설명합니다.

OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이러한 주제에서는 OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드를 선택하는 예를 보여 줍니다.

다음의 모든 예에서는 단일 형식 데이터베이스 파일(실제 또는 논리)을 처리 중인 것으로 가정한 것입니다. (파일이 복수 형식 논리 파일일 경우 OPNQRYF 명령의 FILE 매개변수를 사용하여 레코드 형식명을 지정할 수 있습니다.)

관련 개념

제어 언어(CL)

예 1: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 사용하여 특정값이 있는 레코드를 선택하는 방법을 표시합니다.

Code 필드의 값이 D인 FILEA에서 모든 레코드를 선택하는 것으로 가정하십시오. 처리 프로그램은 PGMB입니다. PGMB는 선택값에 맞는 레코드만을 보여줍니다. (프로그램에서 이를 테스트할 필요는 없습니다.)

주: OPNQRYF 명령에 프롬트 기능을 사용함으로써 매개변수를 보다 쉽게 지정할 수 있습니다. 예를 들어, 시스템에서 작은 따옴표를 추가하기 때문에 QRYSLT 매개변수에 표현식을 분리문자로 묶지 않고 지정할 수 있습니다.

다음 매개변수를 지정하십시오.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('CODE *EQ "D" ')
CALL      PGM(PGMB)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

주:

1. QRYSLT 매개변수의 표현식 전체를 작은 따옴표(‘ ’)로 묶어야 합니다.
2. OPNQRYF 명령에 필드명을 지정할 때 레코드 내의 이름은 따옴표로 묶지 마십시오.
3. 문자 리터럴은 따옴표(″ ″)로 묶어야 합니다. (예에서는 따옴표가 사용됩니다.) 데이터베이스에서 찾으려는 값과 일치시키기 위하여 문자를 대문자 또는 소문자로 하여 큰 따옴표 안에 위치시키는 것이 중요합니다. (예에서는 모두 대문자로 표시되어 있습니다.)
4. 숫자 상수에 대해 선택을 요구하려면 다음과 같이 지정하십시오.

   ```
   OPNQRYF    FILE(FILEA) QRYSLT('AMT *GT 1000.00')
   ```

   주:

   숫자 상수는 따옴표로 묶지 *않습니다.*
5. 필드 값을 제어 언어(CL) 변수와 비교할 경우 다음과 같이 어포스트로피를 사용하십시오(문자 CL 변수만 사용 가능).
   - 문자, 날짜, 시간 또는 시간소인 필드에 대해 선택하려면 다음과 같이 지정하거나

     ```
     OPNQRYF    FILE(FILEA) QRYSLT('"' *CAT &CHAR *CAT '" *EQ FIELDA')
     ```

또는 다음과 같이 역순으로 지정하십시오.

```
OPNQRYF    FILE(FILEA) QRYSLT('FIELDA *EQ "' *CAT &CHAR *CAT '"')
```

주:

작은 따옴표(‘ ’)와 큰 따옴표(″ ″)는 CL 변수와 *CAT 연산자를 묶습니다.

- 숫자 필드에 대해 선택하려면 다음과 같이 지정하거나

```
OPNQRYF    FILE(FILEA) QRYSLT(&CHARNUM *CAT ' *EQ NUM')
```

또는 다음과 같이 역순으로 지정하십시오.

```
OPNQRYF    FILE(FILEA) QRYSLT('NUM *EQ ' *CAT &CHARNUM);
```

주:

작은 따옴표는 필드와 연산자만 묶습니다.

두 개의 필드 또는 상수를 비교할 때는 그 자료 유형이 호환 가능해야 합니다. 다음 표에서는 유효한 비교를 나타내고 있습니다.

표 42. OPNQRYF 명령에 유효한 자료 유형 비교

	모든 숫자	문자	날짜[1]	시간[1]	시간소인[1]
모든 숫자	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
문자	유효하지 않음	유효함	유효함[2]	유효함[2]	유효함[2]
날짜[1]	유효하지 않음	유효함[2]	유효함	유효하지 않음	유효하지 않음
시간[1]	유효하지 않음	유효함[2]	유효하지 않음	유효함	유효하지 않음
시간소인[1]	유효하지 않음	유효함[2]	유효하지 않음	유효하지 않음	유효함

[1] 날짜, 시간 및 시간소인 자료 유형은 필드와 표현식으로는 표시될 수 있으나, 상수로는 표시될 수 없습니다. 그러나 문자 상수는 날짜, 시간 또는 시간소인값을 표시할 수 있습니다.

[2] 문자 필드나 상수는 날짜 자료 유형과 비교 시는 유효한 날짜값을, 시간 자료 유형과 비교 시는 유효한 시간값을, 그리고 시간소인 자료 유형과 비교 시는 유효한 시간소인값을 표시해야 합니다.

시스템의 파일이 키순 액세스 경로 내에서 선택된 필드를 사용할 경우 레코드 선택의 성능이 크게 향상될 수 있습니다. 이것은 시스템이 선택값에 맞는 레코드에만 신속하게 액세스할 수 있게 합니다. 해당 액세스 경로가 없는 경우 시스템은 선택값이 맞는지를 확인하기 위하여 모든 레코드를 읽어야 합니다.

선택하려는 필드에 액세스 경로가 존재해도 시스템에서 그 액세스 경로를 사용하지 않을 수 있습니다. 예를 들어 입력순으로 자료를 처리하는 것이 보다 빠르면 시스템은 그 방법을 선택하게 됩니다.

관련 개념

337 페이지의『2바이트 문자 세트 고려사항』

이 주제에서는 iSeries 시스템에 데이터베이스를 적용할 경우 2바이트 문자 세트(DBCS) 고려사항을 설명합니다.

186 페이지의『OPNQRYF(조회 파일 열기) 명령: 성능 고려사항』

다음은 OPNQRYF(조회 파일 열기) 명령 사용 시 성능 최적화를 위한 추가 정보 및 기술입니다.

예 2: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 사용하여 특정 날짜값이 있는 레코드를 선택하는 방법을 표시합니다.

레코드의 *Date* 필드가 현재 날짜와 동일한 레코드를 모두 처리하는 것으로 가정하십시오. 또한 *Date* 필드가 시스템 날짜와 같은 형식이라고 가정하십시오. 제어 언어(CL) 프로그램에서 다음을 지정할 수 있습니다.

```
DCL        VAR(&CURDAT);  TYPE(*CHAR)  LEN(6)
RTVSYSVAL  SYSVAL(QDATE) RTNVAR(&CURDAT);
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('"' *CAT &CURDAT *CAT '" *EQ DATE')
CALL       PGM(PGMB)
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

CL 변수가 선행 앰퍼샌드(&)로 할당되며 작은 따옴표로 묶이지 않습니다. 전체 표현식이 작은 따옴표로 묶입니다. CAT 연산자와 CL 변수는 작은 따옴표와 따옴표 모두로 묶입니다.

데이터베이스 내의 자료가 문자, 날짜, 시간, 시간소인 또는 숫자로 정의되는지를 알아야 합니다. 이 예에서 *Date* 필드는 문자로 가정됩니다.

DATE 필드가 날짜 자료 유형으로 정의되면 앞의 예는 다음과 같이 지정될 수 있습니다.

```
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA) QRYSLT('%CURDATE *EQ DATE')
CALL       PGM(PGMB)
CLOF OPENID(FILEA)
DLTOVR     FILE(FILEA)
```

주: 날짜 필드가 시스템 날짜와 동일한 형식을 가질 필요는 없습니다.

이 예를 다음과 같이 지정할 수도 있습니다.

```
DCL VAR(&CVTDAT); TYPE(*CHAR) LEN(6)
DCL VAR(&CURDAT); TYPE(*CHAR) LEN(8)
RTVSYSVAL SYSVAL(QDATE) RTNVAR(&CVTDAT);
CVTDAT DATE(&CVTDAT); TOVAR(&CURDAT); TOSEP(/)
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA)
              QRYSLT('"' *CAT &CURDAT *CAT '" *EQ DATE')
CALL       PGM(PGMB)
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

여기서, *DATE*는 FILEA의 날짜 자료 유형을 가지고 있으며, 작업 디폴트 날짜 형식은 MMDDYY이고, 작업 디폴트 날짜 분리자는 슬래시(*/*)입니다.

주: 다음 형식 즉, MMDDYY, DDMMYY, YYMMDD 또는 줄리안 형식에서 날짜의 문자 표현의 경우 작업 디폴트 날짜 형식과 분리자는 인식되는 것과 동일해야 합니다.

대신 상수를 사용 중이면, QRYSLT는 다음과 같이 지정됩니다.

```
QRYSLT('"12/31/87" *EQ DATE')
```

작업 디폴트 날짜 형식은 MMDDYY이고 작업 디폴트 분리자는 슬래시(/)이어야 합니다.

데이터베이스에 숫자 필드가 있고 그것을 변수와 비교하려는 경우 문자 변수만이 사용될 수 있습니다. 예를 들어, 팩 *Date* 필드가 변수보다 큰 레코드를 모두 선택하려면 그 변수가 문자 형식이어야 합니다. 일반적으로 이것은 OPNQRYF 명령 이전에 CHGVAR(변수 변경) 명령을 사용하여 변수를 십진 필드에서 문자 필드로 변경해야 한다는 의미입니다. CHGVAR 명령은 다음과 같이 지정됩니다.

```
CHGVAR VAR(&CHARVAR);  VALUE('123188')
```

QRYSLT 매개변수는 다음과 같이 지정됩니다. (앞의 예와의 차이점을 보십시오.)

```
QRYSLT(&CHARVAR  *CAT ' *GT DATE')
```

대신 상수를 사용 중이면, QRYSLT는 다음과 같이 지정됩니다.

```
QRYSLT('123187 *GT DATE')
```

예 3: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 사용하여 값 범위 내의 레코드를 선택하는 방법을 표시합니다.

Date 필드가 문자 형식 YYMMDD로 정의되고 『.』가 있는 것으로 가정하십시오. 1988년도의 레코드를 모두 처리하는 것으로 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('DATE *EQ %RANGE("88.01.01" +
                              "88.12.31") ')
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

또한 *DATE* 필드에 날짜 자료 유형이 있고, 작업 디폴트 날짜 형식이 YYMMDD이고, 작업 디폴트 날짜 분리자가 마침표(.)일 경우 이 예가 유효합니다.

주: 다음 형식 즉, MMDDYY, DDMMYY, YYMMDD 또는 줄리안 형식에서 날짜의 문자 표현의 경우 작업 디폴트 날짜 형식과 분리자는 인식되는 것과 동일해야 합니다.

범위가 문자 자료 유형으로 정의된 변수이고 DATE 필드가 문자 자료 유형으로 정의된 경우 QRYSLT 매개변수를 다음과 같이 지정하십시오.

```
QRYSLT('DATE *EQ %RANGE("' *CAT &LORNG *CAT '"' *BCAT '"'  +
      *CAT &HIRNG *CAT '")')
```

그러나 DATE 필드가 숫자 자료 유형으로 정의될 경우 QRYSLT 매개변수는 다음과 같이 지정됩니다.

```
QRYSLT('DATE *EQ %RANGE(' *CAT &LORNG *BCAT &HIRNG *CAT ')')
```

주: QRYSLT 매개변수가 제어 언어(CL) 프로그램 내에 있는 경우 *BCAT를 사용할 수 있으나 대화식 명령에서는 사용할 수 없습니다.

예 4: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령의 포함 기능을 사용하여 레코드를 선택하는 방법을 표시합니다.

Addr 필드에 BROADWAY라는 거리가 포함되어 있는 레코드를 모두 처리하는 것으로 가정하십시오. 포함(*CT) 기능은 지정된 필드의 어디에 문자가 나오는지를 결정합니다. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('ADDR *CT "BROADWAY" ')
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

이 예에서 자료는 데이터베이스 레코드에서 대문자로 되어 있는 것으로 가정하십시오. 자료가 소문자 또는 대소문자로 혼합되어 있는 경우 비교하기 전에 소문자나 대소문자 혼합형 자료를 대문자로 변환하기 위한 변환 기능을 지정할 수 있습니다. 시스템이 제공하는 표 QSYSTRNTBL은 a에서 z까지의 모든 문자를 대문자로 변환합니다. (변환에는 어떤 변환표든지 사용할 수 있습니다.) 따라서 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('%XLATE(ADDR QSYSTRNTBL) *CT +
                                 "BROADWAY" ')
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

QRYSLT문에서 %XLATE 기능이 사용되는 경우 고급 언어 프로그램에 전달되는 필드 값은 데이터베이스에 있는 것처럼 나옵니다. MAPFLD 매개변수에 %XLATE 기능을 사용하여 필드를 강제로 대문자로 표시할 수 있습니다.

예 5: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령의 여러 필드를 사용하여 레코드를 선택하는 방법을 표시합니다.

Amt 필드가 0이거나 *Lstdat* 필드(문자 형식의 YYMMDD 순서) 88-12-31이하인 모든 레코드를 처리하는 것으로 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('AMT *EQ 0 *OR LSTDAT +
                                 *LE "88-12-31" ')
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

이 예는 *LSTDAT* 필드가 날짜 자료 유형인 경우에도 해당됩니다. *LSTDAT* 필드는 유효한 모든 날짜 형식일 수 있으나, 작업 디폴트 날짜 형식은 YYMMDD여야 하고, 작업 디폴트 날짜 분리자는 대시(-)여야 합니다.

주: 다음 형식 즉, MMDDYY, DDMMYY, YYMMDD 또는 줄리안 형식에서 날짜의 문자 표현의 경우 작업 디폴트 날짜 형식과 분리자는 인식되는 것과 동일해야 합니다.

변수 사용 시 QRYSLT 매개변수는 다음과 같이 입력되거나

```
QRYSLT('AMT *EQ ' *CAT &VARAMT *CAT ' *OR +
      LSTDAT *LE "' *CAT &VARDAT *CAT '"')
```

또는 다음과 같이 역순으로 지정하십시오.

```
QRYSLT('"' *CAT &VARDAT *CAT '" *GT LSTDAT *OR ' +
      *CAT &VARAMT *CAT ' *EQ AMT')
```

주:

&VARAMT 변수는 문자 유형으로 정의해야 합니다. 변수가 숫자 유형으로 제어 언어(CL) 프로그램에 전달되는 경우 이를 문자 유형으로 변환해야 연결이 가능합니다. 이러한 변환을 수행하기 위해서는 CHGVAR(변수 변경) 명령을 사용할 수 있습니다.

예 6: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 한 프로그램에서 OPNQRYF(조회 파일 열기) 명령을 여러 번 사용하는 방법을 표시합니다.

고급 언어 프로그램에서는 OPNQRYF 명령을 여러 번 사용할 수 있습니다. 예를 들어 특정 선택값을 프롬트하여 한 페이지 이상의 레코드를 표시하는 것으로 가정하십시오. 첫 번째 레코드 요구가 끝나면 사용자는 다른 선택값을 지정하여 그 레코드를 표시할 수 있습니다. 이 경우 다음과 같이 작업을 수행할 수 있습니다.

1. 고급 언어 프로그램을 호출하기 전에 OVRDBF(데이터베이스 파일 대체) 명령을 사용하여 SHARE(*YES)를 지정하십시오.
2. 고급 언어 프로그램에서 사용자에게 선택값을 프롬트하십시오.
3. 선택값을 OPNQRYF 명령을 발행하는 제어 언어(CL) 프로그램으로 전달하십시오(또는 프로그램 QCMDEXC를 호출하여 명령을 수행하십시오). 프로그램이 OPNQRYF 명령을 처리하기 전에 파일을 닫아야 합니다. 일반적으로는 CLOF(파일 닫기) 명령을 사용하며 아직 열리지 않은 파일을 모니터링합니다.
4. 고급 언어 프로그램으로 리턴하십시오.
5. 고급 언어 프로그램에서 파일을 여십시오.
6. 레코드를 처리하십시오.
7. 프로그램에서 파일을 닫으십시오.
8. 2단계로 리턴하십시오.

프로그램이 완료되면 CLOF 명령이나 RCLRSC(자원 재생) 명령을 수행하여 파일을 닫은 후, 1단계에서 지정한 OVRDBF 명령을 삭제하십시오.

주: 호출된 CL 프로그램 내의 대체 명령은 주 프로그램에서의 열기에 영향을 미치지 않습니다. 모든 대체속성은 프로그램이 종료될 때 암시적으로 삭제됩니다. (그러나 필요한 경우 고급 언어 프로그램에서 프로그램 QCMDEXC를 호출하여 대체 속성을 지정할 수 있습니다.)

예 7: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 사용하여 팩 숫자 자료 필드에 필드를 맵핑하는 방법을 표시합니다.

MMDDYY 형식의 팩 십진 *Date* 필드를 가지고 있고 1988년에 해당되는 모든 레코드를 선택하는 것으로 가정하십시오. 팩 십진 필드의 일부분에서 레코드를 직접 선택할 수는 없으나 OPNQRYF 명령에 MAPFLD 매개변수를 사용하여 새로운 필드를 작성함으로써 필드의 일부분을 선택할 수 있습니다.

각 맵핑된 필드 정의 형식은 다음과 같습니다.

(결과 필드 '표현식' 속성)

위에서

결과 필드
결과 필드명

표현식 결과 필드가 파생되는 방법. 표현식에는 서브스트링, 기타 내장 기능 또는 수학적 명령문이 포함될 수 있습니다.

속성 결과 필드의 선택적 속성. 속성을 지정하지 않으면(또는 필드가 파일에서 정의되지 않으면), OPNQRYF 명령이 표현식 필드에 따라 결정된 필드 속성을 계산합니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) QRYSLT('YEAR *EQ "88" ') +
                         MAPFLD((CHAR6 '%DIGITS(DATE)')  +
                             (YEAR '%SST(CHAR6 5 2)' *CHAR 2))
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

이 예에서 DATE가 날짜 자료 유형이었다면 다음과 같이 지정해야 합니다.

```
OPNQRYF    FILE(FILEA) +
QRYSLT ('YEAR *EQ 88') +
MAPFLD((YEAR '%YEAR(DATE)'))
```

맨 처음의 맵핑된 필드 정의에서는 *Char6* 필드가 팩 십진 *Date* 필드로부터 작성되도록 지정되어 있습니다 %DIGITS 기능은 팩 십진에서 문자로 변환시키며 모든 십진 정의를 무시합니다. (즉 1234.56은 '123456'으로 변환됩니다.) *Char6* 필드 정의가 지정되지 않았으므로 시스템에서 길이 6을 할당합니다. 두 번째 맵핑된 필드는 *Year* 필드를 유형 *CHAR(문자) 및 길이 2로 정의합니다. 이 식은 서브스트링 함수를 사용하여 *Char6* 필드의 마지막 2자를 *Year* 필드로 맵핑합니다.

맵핑된 필드 정의는 지정된 순서대로 처리됨에 유의하십시오. 이 예에서 *Date* 필드는 문자로 변환되었으며 *Char6* 필드에 할당되었습니다. 그 다음, *Char6* 필드의 끝 두자리(연도)는 *Year* 필드에 할당되었습니다. 이 순서를 변경시키면 잘못된 결과를 얻게 됩니다.

주: 맵핑된 필드 정의는 항상 QRYSLT 매개변수가 평가되기 이전에 처리됩니다.

다음과 같이 QRYSLT 매개변수에 서브스트링을 지정하고 맵핑된 필드 정의 중 하나를 삭제함으로써 같은 결과를 얻을 수 있습니다.

```
OPNQRYF    FILE(FILEA) +
                         QRYSLT('%SST(CHAR6 5 2) *EQ "88" ') +
                         MAPFLD((CHAR6 '%DIGITS(DATE)'))
```

예 8: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령의 『와일드 카드』 기능을 표시합니다.

MMDDYY 형식으로 된 팩 십진 *Date* 필드가 있고 1988년 3월에 해당되는 레코드를 선택하는 것으로 가정하십시오. 이를 수행하기 위해서는 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) +
                         QRYSLT('%DIGITS(DATE) *EQ %WLDCRD("03__88")')
CALL      PGM(PGMC)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

%DIGITS 기능의 결과에 대해 데이터베이스 필드를 정의하는 데 MAPFLD 매개변수가 필요한 경우는 그 결과가 인수로서 단순 필드명(기능이나 표현식이 아님)을 지원하는 기능과 함께 사용되어야 할 경우 뿐입니다. %WLDCRD 조작에는 *EQ 연산자 앞에 표시되는 피연산자에 대해 그러한 제한이 없습니다.

데이터베이스의 필드가 숫자 형식이지만 그 정의를 *Char6* 필드와 동일하게 만들려면 문자를 따옴표로 묶어야 한다는 점에 유의하십시오. 와일드 카드 기능은 DATE, TIME 또는 TIMESTAMP 자료 유형을 지원하지 않습니다.

%WLDCRD 기능을 사용하면 선택값에 대응하는 레코드를 모두 선택할 수 있습니다. 밑줄(_)은 모든 단일 문자 값에 대응될 수 있습니다. 예 8의 두 개의 밑줄 문자는 3월 중의 모든 날짜가 선택될 수 있게 합니다. %WLDCRD 기능으로 와일드 카드 문자를 지정할 수도 있습니다(밑줄은 디폴트임).

와일드 카드 기능은 다음의 두 가지 형식을 지원합니다.

- 밑줄(또는 사용자가 지정한 문자)이 다음 예에서처럼 임의의 단일 문자와 일치하는 이전 예에 표시된 것과 같은 고정 위치 와일드 카드.

  ```
  QRYSLT('FLDA *EQ %WLDCRD("A_C")')
  ```

  ABC, ACC, ADC, AxC 등이 성공적으로 비교됩니다. 이 예에서 분석되는 필드는 필드의 길이가 정확하게 3자일 경우에만 비교됩니다. 만일 필드가 세 자보다 길 경우 와일드 카드 지원의 두 번째 형식이 필요합니다.
- 변수 위치 와일드 카드는 0이상의 문자 수에 대응됩니다. OPNQRYF(조회 파일 열기) 명령은 이러한 와일드 카드 가변문자 유형에 별표(*)를 사용하거나 사용자 나름대로 문자를 지정하도록 합니다. 이 예에서는 별표가 사용됩니다.

  ```
  QRYSLT('FLDB *EQ %WLDCRD("A*C*") ')
  ```

  AC, ABC, AxC, ABCD, AxxxxxxxC 등이 성공적으로 비교됩니다. 이 별표로 인해 존재할 수도 있는 모든 중간문자를 명령이 무시하게 됩니다. 이 예에서 별표는 필드에서 나중에 나올 수도 있는 문자의 앞뒤에 모두 지정되었음에 유의하십시오. 별표가 탐색 인수의 끝부분에서 생략될 경우 필드가 문자 C로 끝날 때에만 필드가 선택됩니다.

패턴의 나머지 부분이 필드의 아무곳에서나 시작하는 레코드를 선택하려면 와일드 카드 스트링의 맨 처음에 별표를 지정해야 합니다. 마찬가지로, 패턴의 나머지 부분이 필드의 어느 곳에서든지 끝나는 레코드를 선택할 경우에도 패턴 스트링은 별표로 끝나야 합니다.

예를 들어, 다음과 같이 지정할 수 있습니다.

```
QRYSLT('FLDB *EQ %WLDCRD("*ABC*DEF*") ')
```

ABCDEF, ABCxDEF, ABCxDEFx, ABCxxxxxxDEF, ABCxxxDEFxxx, xABCDEF, xABCxDEFx 등이 모두 대응됩니다.

이 예에서 보듯이 와일드 카드 기능 두 개를 결합시킬 수 있습니다.

```
QRYSLT('FLDB *EQ %WLDCRD("ABC_*DEF*") ')
```

ABCxDEF, ABCxxxxxxDEF, ABCxxxDEFxxx 등이 모두 대응됩니다. 밑줄은 ABC와 DEF 사이에 적어도 한 개의 문자가 나오는 경우에만 대응됩니다. (예를 들어, ABCDEF는 여기에 대응되지 않습니다.)

다음을 포함하는 *Name* 필드가 있다고 가정하십시오.

- JOHNS
- JOHNS SMITH
- JOHNSON
- JOHNSTON

다음을 지정하면 첫 번째 레코드만 가져올 수 있습니다.

```
QRYSLT('NAME *EQ "JOHNS"')
```

사용자가 지정한 값에 공백이 추가되면서 비교되기 때문에 그 외의 레코드는 선택되지 않습니다. 네 개의 이름을 모두 선택하는 방법은 다음과 같습니다.

```
QRYSLT('NAME *EQ %WLDCRD("JOHNS*")')
```

관련 개념

337 페이지의『2바이트 문자 세트 고려사항』

이 주제에서는 iSeries 시스템에 데이터베이스를 적용할 경우 2바이트 문자 세트(DBCS) 고려사항을 설명합니다.

제어 언어(CL)

예 9: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드를 선택할 경우 복합 선택문을 지정하는 방법을 표시합니다.

복합 선택문도 지정할 수 있습니다. 예를 들어, 다음과 같이 지정할 수 있습니다.

```
QRYSLT('DATE *EQ "880101" *AND AMT *GT 5000.00')

QRYSLT('DATE *EQ "880101" *OR AMT *GT 5000.00')
```

다음과 같이 지정할 수도 있습니다.

```
QRYSLT('CODE *EQ "A" *AND TYPE *EQ "X" *OR CODE *EQ "B")
```

오퍼레이터 처리 우선순위를 설정하는 규칙은 제어 언어(CL) 주제에서 설명됩니다.그 규칙의 일부는 다음과 같습니다.

- *AND 조작이 먼저 처리된 후, 다음과 같은 경우 레코드가 선택됩니다.

  Code 필드 = ″A″ 및 *Type* 필드 = ″X″
  또는
  Code 필드 = ″B″

- 이 예에서처럼 괄호를 사용하여 표현식 처리 방법을 제어할 수 있습니다.

```
QRYSLT('(CODE *EQ "A" *OR CODE *EQ "B") *AND TYPE *EQ "X" +
                  *OR CODE *EQ "C"')
```

  Code 필드 = ″A″ 및 *Type* 필드 = ″X″
  또는
  Code 필드 = ″B″ 및 *Type* 필드 = ″X″
  또는
  Code 필드 = ″C″

또한 다음 예에서와 같이 축약 형식 대신 제어 언어(CL) 주제에 설명된 기호를 사용할 수 있습니다(예를 들어, *EQ 대신 = 사용 가능).

```
QRYSLT('CODE = "A" & TYPE = "X" | AMT > 5000.00')
```

이 명령은 다음과 같은 조건의 모든 레코드를 선택합니다.

Code 필드 = ″A″ 및 *Type* 필드 = ″X″
또는
Amt 필드 > 5000.00

복합 선택문은 다음과 같이 작성할 수도 있습니다.

```
QRYSLT('CUSNBR = %RANGE("60000" "69999") & TYPE = "B" +
       & SALES>0 & ACCRCV / SALES>.3')
```

이 명령은 다음과 같은 조건의 모든 레코드를 선택합니다.

Cusnbr 필드가 범위 60000-69999 내에 있으며 *Type* 필드 = ″B″ 및
Sales 필드는 0보다 크고 *Accrcv*를 *Sales*로 나누면 30%보다 큼

예 10: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 실행하여 레코드를 선택할 경우 코드화 문자 세트 ID(CCSID) 사용을 표시합니다.

모든 데이터베이스 파일에 있는 각각의 문자와 DBCS 필드에는 CCSID가 표시됩니다. 이 CCSID로 필드의 비교, 결합 또는 표시가 의미있는 방법으로 수행되도록 파일에 저장되는 자료를 추가로 정의할 수 있습니다. 예를 들어, FILE1의 FIELD1(FIELD1의 CCSID는 37(미국))과 FILE2의 FIELD2(FIELD2의 CCSID는 273(오스트리아, 독일))를 비교할 경우 적절한 맵핑이 발생해야 비교가 가능해집니다.

```
OPNQRYF FILE(FILEA FILEB) FORMAT(RESULTF) +
               JFLD((FILEA/NAME FILEB/CUSTOMER))
```

NAME 필드의 CCSID가 37이고 CUSTOMER 필드의 CCSID가 273이면 NAME이나 CUSTOMER의 맵핑은 두 필드의 결합이 의미있는 결과를 제공하도록 OPNQRYF 명령 처리중에 수행됩니다.

일반적으로 MAPFLD, QRYSLT 및 GRPSLT 매개변수에 정의된 상수에는 현재 작업에서 정의된 CCSID가 표시됩니다. 이것은 다른 작업 CCSID를 갖는 두 사용자가 같은 OPNQRYF 명령(또는 OPNQRYF 명령이 들어 있는 프로그램)을 수행하고 OPNQRYF 명령에 자체적으로 정의된 상수가 있을 경우 상수에 표시된 CCSID가 다르게 처리되도록 하기 때문에 사용자는 다른 결과를 얻을 수 있습니다.

사용자는 MAPFLD 매개변수를 사용하여 특정 CCSID를 상수에 표시할 수 있습니다. 정의에 상수 하나만을 포함하는 MAPFLD를 지정한 다음 MAPFLD에 대한 CCSID를 지정하면 상수에는 MAPFLD 매개변수에 지정된 CCSID가 표시됩니다. 예를 들면,

```
OPNQRYF FILE(FILEA) FORMAT(RESULTF) QRYSLT('NAME *EQ MAP1') +
               MAPFLD((MAP1 '"Smith"' *CHAR 5 *N 37))
```

상수『Smith』에는 OPNQRYF 명령을 수행한 사용자의 작업 CCSID에 관계없이 CCSID 37이 표시됩니다. 이 예에서 모든 사용자는 같은 결과 레코드(결과 레코드가 사용자의 작업 CCSID로 맵핑되지만)를 얻게 됩니다. 역으로, 조회가 다음과 같이 지정되면

```
OPNQRYF FILE(FILEA) FORMAT(RESULTF) QRYSLT('NAME *EQ "Smith"')
```

조회의 결과는 OPNQRYF 명령을 발행한 사용자의 작업 CCSID에 따라 달라집니다.

관련 개념

i5/OS 국제화

예 11: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택:

이 예는 OPNQRYF(조회 파일 열기) 명령을 실행하여 레코드를 선택할 경우 정렬 순서 및 언어 ID 사용을 표시합니다.

정렬 순서 사용 방법을 참조하려면 표 43에 있는 STAFF 파일에 이 주제의 예를 실행하십시오.

표 43. STAFF 파일

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	0
20	Pernal	20	Sales	8	18171.25	612.45
30	Merenghi	38	MGR	5	17506.75	0
40	OBrien	38	Sales	6	18006.00	846.55
50	Hanes	15	Mgr	10	20659.80	0

표 43. STAFF 파일 (계속)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
60	Quigley	38	SALES	00	16808.30	650.25
70	Rothman	15	Sales	7	16502.83	1152.00
80	James	20	Clerk	0	13504.60	128.20
90	Koonitz	42	sales	6	18001.75	1386.70
100	Plotz	42	mgr	6	18352.80	0

이 예에서는 다음 각 정렬 순서를 사용하는 특정 명령문에 대해 결과가 표시됩니다.

- *HEX 정렬 순서
- 언어 식별자 ENU에 대한 공유 가중치 정렬 순서
- 언어 ID ENU에 대한 고유 가중치 정렬 순서

주: OPNQRYF 명령에서 SRTSEQ(*LANGIDUNQ)나 SRTSEQ(*LANGIDSHR), LANGID(ENU)를 지정하여, ENU가 언어 식별자로 선택되었습니다.

다음 명령어는 JOB 필드에서 MGR 값을 가진 레코드를 선택합니다.

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
```

표 44는 *HEX 정렬 순서를 사용한 레코드 선택 결과를 보여줍니다. JOB 필드에 대해서 레코드 선택 범주에 속하는 레코드들이 QRYSLT문에서 지정된 것과 똑같이 선택되는데, 오직 대문자 MGR만이 선택됩니다.

*표 44. *HEX 정렬 순서 사용.* OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"') SRTSEQ(*HEX)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
30	Merenghi	38	MGR	5	17506.75	0

표 45는 공유 가중치 정렬 순서를 사용한 레코드 선택 결과를 보여줍니다. JOB 필드에 대해서 레코드 선택 범주에 속하는 레코드들이 대소문자를 무시하고 모두 선택됩니다. 이 때의 정렬 순서는 mgr, Mgr 그리고 MGR 값입니다.

표 45. 공유 가중치 정렬 순서 사용. OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"') SRTSEQ(LANGIDSHR) LANGID(ENU)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	0
30	Merenghi	38	MGR	5	17506.75	0
50	Hanes	15	Mgr	10	20659.80	0
100	Plotz	42	mgr	6	18352.80	0

159 페이지의 표 46은 고유 가중치 순서를 사용한 레코드 선택 결과를 보여줍니다. JOB 필드에 대해서 레코드 선택 범주에 속하는 레코드가 대소문자를 구분하여 선택됩니다. 이 때의 정렬 순서는 mgr, Mgr, 그리고 MGR 값을 모두 다르게 처리합니다. MGR 값이 선택됩니다.

표 46. 고유 가중치 정렬 순서 사용. OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ ″MGR″') SRTSEQ(LANGIDUNQ) LANGID(ENU)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
30	Merenghi	38	MGR	5	17506.75	0

DDS를 사용하지 않고 키순 액세스 경로 지정:

동적 액세스 경로 기능을 사용하면 처리될 자료에 대해 키순 액세스 경로를 지정할 수 있습니다. 공유할 수 있는 액세스 경로가 이미 있는 경우에는 시스템이 이를 공유할 수 있습니다. 새로운 액세스 경로가 필요하면 레코드가 프로그램으로 전달되기 전에 액세스 경로가 작성됩니다.

예 1: DDS를 사용하지 않고 키순 액세스 경로 지정:

이 예는 키 필드 하나를 사용하여 레코드를 배열하는 방법을 표시합니다.

프로그램 PGMB로 *Cust* 필드 값에 의해 배열되는 FILEA의 레코드를 처리하는 것으로 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) KEYFLD(CUST)
CALL       PGM(PGMD)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

주: 조회 파일 열기(OPNQRYF) 명령의 FORMAT 매개변수를 사용할 필요가 없습니다. FILEA를 처리된 파일로 지정하여 PGMD가 작성되기 때문입니다. FILEA는 입력순 또는 키순 파일일 수 있습니다. FILEA가 키순 파일이면, 이 파일의 키 필드는 *Cust* 필드이거나 전혀 다른 필드가 될 수 있습니다.

예 2: DDS를 사용하지 않고 키순 액세스 경로 지정:

이 예는 복수 키 필드를 사용하여 레코드를 배열하는 방법을 표시합니다.

처음에는 *Cust*순으로 그 다음에는 *Cust* 내의 *Date*순으로 레코드를 처리하려면 다음과 같이 지정하십시오.

```
OPNQRYF    FILE(FILEA) KEYFLD(CUST DATE)
```

*Date*를 내림차순으로 표시하려면 다음과 같이 지정하십시오.

```
OPNQRYF    FILE(FILEA) KEYFLD((CUST) (DATE *DESCEND))
```

이 두 예에서 FORMAT 매개변수는 사용되지 않습니다. (다른 형식이 정의되면 모든 키 필드가 형식 내에 존재해야 합니다.)

예 3: DDS를 사용하지 않고 키순 액세스 경로 지정:

이 예는 고유 가중치 정렬 순서를 사용하여 레코드를 배열하는 방법을 표시합니다.

157 페이지의 표 43의 STAFF 파일을 사용하여 고유 가중치 정렬 순서의 JOB 필드 값으로 레코드를 처리하려면 다음을 지정하십시오.

```
OPNQRYF FILE(STAFF) KEYFLD(JOB) SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

JOB 필드에 있는 조회 결과는 다음 순서로 이루어집니다.

- Clerk
- mgr
- Mgr
- Mgr
- MGR
- sales
- Sales
- Sales
- Sales
- SALES

예 4: DDS를 사용하지 않고 키순 액세스 경로 지정:

이 예는 공유 가중치 정렬 순서를 사용하여 레코드를 배열하는 방법을 표시합니다.

157 페이지의 표 43의 STAFF 파일을 사용하여 고유 가중치 정렬 순서의 JOB 필드 값으로 레코드를 처리하려면 다음을 지정하십시오.

```
OPNQRYF FILE(STAFF) KEYFLD(JOB) SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

이 조회의 결과는 예 3의 결과와 유사합니다. *mgr* 및 *sales* 항목은 대소문자가 동일하게 처리되므로 어떤 순서로도 처리될 수 있습니다. 즉, 공유 가중치 정렬 순서는 mgr, Mgr, MGR을 동일한 값으로 처리합니다. 마찬가지로 sales, Sales, SALES도 모두 동일한 값으로 처리됩니다.

다른 파일로부터 키 필드 지정:

결합 논리 파일의 동적 키순 액세스 경로는 서로 다른 실제 파일 내에 있을 수 있는 키로 처리 순서를 지정할 수 있도록 합니다. (DDS는 키를 1차 파일에만 제한합니다.)

스펙(specification)은 이전 방법과 동일합니다. 액세스 경로는 필요한 키 필드는 어느 것이나 사용해서 지정할 수 있습니다. 키 필드가 어떤 실제 파일에 있는지에 대해서는 아무런 제한이 없습니다. 그러나 키 필드가 결합 스펙의 1차 파일 이외의 파일에 존재할 경우 시스템은 결합 레코드를 임시로 복사해 두어야 합니다. 또한 시스템은 조회 파일이 열리기 전에 복사된 레코드에 키순 액세스 경로를 빌드해야 합니다. 키 필드는 FORMAT 매개변수에서 식별된 형식 내에 존재해야 합니다.

예: 다른 파일로부터 키 필드 지정

이 예는 2차 파일의 필드를 키 필드로 사용하는 방법을 표시합니다.

JOINLF라는 결합 논리 파일이 이미 있는 것으로 가정하십시오. FILEX가 1차 파일로 지정되며 이는 FILEY에 결합됩니다. FILEY 내에 있는 *Descrp* 필드로 JOINLF의 레코드를 처리하려고 합니다.

파일 레코드 형식에 다음과 같은 필드가 들어 있는 것으로 가정하십시오.

FILEX	FILEY	JOINLF
Item	Item	Item
Qty	Descrp	Qty
		Descrp

다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(JOINLF) SHARE(*YES)
OPNQRYF   FILE(JOINLF) KEYFLD(DESCRP)
CALL      PGM(PGMC)
CLOF      OPNID(JOINLF)
DLTOVR    FILE(JOINLF)
```

Descrp 내의 *Qty*로(*Descrp*는 1차 키 필드이고, *Qty*는 2차 키 필드임) 레코드를 배열하려면 다음과 같이 지정하면 됩니다.

```
OPNQRYF   FILE(JOINLF) KEYFLD(DESCRP QTY)
```

DDS를 사용하지 않는 동적 결합 데이터베이스 파일:

동적 결합 기능을 사용하면 먼저 자료 서술 스펙(DDS)을 지정하고 결합 논리 파일을 작성하지 않고도 파일을 결합할 수 있습니다.

결합을 위한 레코드 형식을 지정하려면 OPNQRYF(조회 파일 열기) 명령에 FORMAT 매개변수를 사용해야 합니다. 결합 논리 파일 및 뷰를 포함하여 어떤 실제 파일이나 논리 파일도 결합이 가능합니다. (DDS에서는 논리 파일의 결합이 허용되지 않습니다.) 키순 액세스 경로 또는 도달순 액세스 경로를 지정할 수 있습니다. 키를 지정하는 경우 그 키는 결합에 포함되는 어떤 파일에서도 올 수 있습니다. (DDS는 키를 1차 파일에만 제한합니다.)

다음 예에서는 FORMAT 매개변수에 지정된 파일이 작성된 것으로 가정합니다. 외부 서술 자료 정의를 사용하기 위해 처리 프로그램을 작성하기 전에 파일을 먼저 작성하는 것이 일반적입니다.

다음 모든 예에서 결합 순서(JORDER) 매개변수에 대해 디폴트 값이 사용됩니다. JORDER 매개변수의 디폴트 값은 파일 결합 시 순서를 정해서 시스템에 알려주는 *ANY입니다. 즉 시스템은 1차 파일로 어떤 파일을 사용할 것인지와 2차 파일로 어떤 파일을 사용할 것인지 등을 결정합니다. 이로써 시스템은 결합 기능의 성능을 향상시킬 수 있습니다.

레코드 선택 범주와 마찬가지로, 결합 범주도 지정된 정렬 순서(SRTSEQ)와 언어 식별자(LANGID)의 영향을 받습니다.

관련 참조

157 페이지의 『예 11: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택』
이 예는 OPNQRYF(조회 파일 열기) 명령을 실행하여 레코드를 선택할 경우 정렬 순서 및 언어 ID 사용을 표시합니다.

예 1: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합:

이 예는 DDS를 사용하지 않는 데이터베이스 파일을 동적으로 결합하는 방법을 보여 줍니다.

FILEA와 FILEB를 결합하려고 하고, 이 파일에 다음 필드가 포함된다고 가정하십시오.

FILEA	FILEB	JOINAB
Cust	Cust	Cust
이름	Amt	이름
Addr		Amt

결합 필드는 두 개의 파일에 모두 존재하는 *Cust*입니다. 결합 파일에 대한 OPNQRYF(조회 파일 열기) 명령에 레코드 형식명을 지정할 수 있습니다. 파일에는 멤버가 필요치 않으며 레코드가 키순으로 있어야 할 필요는 없습니다.

다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(JOINAB) TOFILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA FILEB) FORMAT(JOINAB) +
               JFLD((FILEA/CUST FILEB/CUST)) +
             MAPFLD((CUST 'FILEA/CUST'))
CALL       PGM(PGME) /* Created using file JOINAB as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(JOINAB)
```

파일 JOINAB는 자료가 없는 실제 파일입니다. 이 파일에는 OPNQRYF 명령의 FORMAT 매개변수에 지정될 레코드 형식이 들어 있습니다.

OVRDBF(데이터베이스 파일 대체) 명령의 TOFILE 매개변수는 결합 조작의 1차 파일명(OPNQRYF 명령의 FILE 매개변수에 지정된 첫 번째 파일)을 지정함에 유의하십시오. 이 예에서 OPNQRYF 명령의 FILE 매개변수는 파일이 결합될 순서대로(A에서 B) 파일을 식별합니다. 파일의 형식은 파일 JOINAB에 있습니다.

JFLD 매개변수는 FILEA의 *Cust* 필드가 FILEB의 *Cust* 필드에 결합되는 것을 나타냅니다. *Cust* 필드는 결합된 모든 레코드 형식에서 고유하지 않기 때문에 이것은 JFLD 매개변수에서 규정되어야 합니다. 어떤 경우에는 OPNQRYF 명령에 JFLD 명령을 지정하지 않았더라도 시스템이 가장 효율적인 값을 결정하려고 합니다. 예를 들면 앞에서의 예를 사용하여 다음과 같이 지정하는 경우입니다.

```
OPNQRYF FILE(FILEA FILEB) FORMAT(JOINAB) +
                          QRYSLT('FILEA/CUST *EQ FILEB/CUST') +
             MAPFLD((CUST 'FILEA/CUST'))
```

시스템은 QRYSLT 매개변수에 지정된 값으로 인해 *Cust* 필드를 사용하여 FILEA와 FILEB를 결합합니다이 예에서 JFLD 매개변수가 명령에 지정되지 않았음에 유의하십시오. 그러나 JDFTVAL(*ONLYDFT) 또는 JDFTVAL(*YES)가 OPNQRYF 명령에 지정되는 경우 JFLD 매개변수가 지정되어야 합니다.

파일 JOINAB의 레코드 형식에 있는 *Cust* 필드에 어떤 파일이 사용되는지 서술하려면 OPNQRYF(조회 파일 열기) 명령에 MAPFLD 매개변수가 필요합니다. 필드가 MAPFLD 매개변수에 정의되면 이 필드의 규정화되지 않은 이름(이 경우 파일명이 없는 *Cust* 필드)은 OPNQRYF 명령의 어느 곳에서나 사용 가능합니다. *Cust* 필드가 MAPFLD 매개변수에 정의되므로 JFLD 매개변수의 첫 번째 값은 규정화될 필요가 없습니다. 예를 들어, 다음을 지정하여 동일한 결과를 얻을 수 있습니다.

```
             JFLD((CUST FILEB/CUST)) +
MAPFLD((CUST 'FILEA/CUST'))
```

MAPFLD 매개변수에 의해 정의된 파일이 아닌 다른 파일의 필드를 지정하기 위해 OPNQRYF 명령에서 같은 필드명을 사용할 경우 필드명은 파일명으로 규정화되어야 합니다.

KEYFLD 매개변수가 지정되지 않았으므로 레코드는 OPNQRYF 명령이 레코드를 선택하는 방법에 따라 어떤 순서로든지 표시될 수 있습니다. 1차 파일과 동일하게 레코드를 배열하도록 시스템에 지시할 수 있습니다. 이렇게 하려면 KEYFLD 매개변수에 *FILE을 지정하십시오. 1차 파일이 입력순으로 되어 있는 경우에라도 이렇게 지정할 수 있습니다.

2차 파일에서 레코드가 누락된 경우 시스템이 해야 할 일을 설명하기 위해 OPNQRYF 명령에 JDFTVAL 매개변수(DDS의 JDFTVAL 키워드와 유사함)를 지정할 수 있습니다. 이 예에서 JDFTVAL 매개변수가 지정되지 않았으므로 양쪽 파일에 다 존재하는 레코드만 선택됩니다.

조회 결과를 향상시키도록 시스템에 지시할 경우(OPNQRYF 명령의 매개변수를 통해), 시스템은 일반적으로 가장 적은 수의 레코드를 가진 파일을 1차 파일로 선택하게 됩니다. 그러나 시스템은 임시 파일을 빌드하지 않으려고도 할 것입니다.

JORDER(*FILE)를 지정함으로써 OPNQRYF 명령의 FILE 매개변수에 지정한 파일의 결합 순서를 시스템이 수행하도록 할 수 있습니다. JDFTVAL(*YES) 또는 JDFTVAL(*ONLYDFT)가 지정되면 다른 순서로 인해 결과가 달라질 수 있으므로 시스템은 결합 파일 순서를 절대로 변경하지 않게 됩니다.

예 2: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합:

이 예는 DDS를 사용하지 않은 데이터베이스 파일을 동적으로 결합할 경우 2차 파일 레코드가 있는 레코드만 읽는 방법을 표시합니다.

FILEAB, FILECD, FILEEF 파일을 결합하여 2차 파일에서 대응되는 레코드가 있는 레코드만 선택하는 것으로 가정하십시오. 파일 JOINF를 정의한 다음 사용해야 할 형식을 서술하십시오. 파일의 레코드 형식에 다음과 같은 필드가 들어 있는 것으로 가정하십시오.

FILEAB	**FILECD**	**FILEEF**	**JOINF**
Abitm	Cditm	Efitm	Abitm
Abord	Cddscp	Efcolr	Abord
Abdat	Cdcolr	Efqty	Cddscp
			Cdcolr
			Efqty

이 경우 결합 파일을 구성하는 파일 내의 모든 필드명은 두 개의 문자로 된 접두부로 시작하며(파일의 모든 필드에 대해 동일함), 모든 파일에 걸쳐 동일한 접미부로 끝납니다(예: *xxitm*). 이렇게 하면 모든 필드명이 고유하게 되므로 이들을 규정화할 필요가 없게 됩니다.

xxitm 필드로 FILEAB를 FILECD에 결합시킬 수 있습니다. *xxitm*과 *xxcolr* 두 개의 필드로 FILECD를 FILEEF에 결합시킬 수 있습니다. 이 파일에는 키순 액세스 경로가 존재할 필요가 없습니다. 그러나 키순 액세스 경로가 존재할 경우 성능이 상당히 향상되는데, 이것은 시스템이 가능할 경우 기존 액세스 경로를 사용하여 레코드를 배열하고 선택하기 때문입니다. 액세스 경로가 존재하지 않는 경우 시스템은 파일이 열려 있는 동안 자동적으로 액세스 경로를 작성 및 유지보수합니다.

```
OVRDBF     FILE(JOINF) TOFILE(FILEAB) SHARE(*YES)
OPNQRYF    FILE(FILEAB FILECD FILEEF) +
                           FORMAT(JOINF) +
                           JFLD((ABITM CDITM)(CDITM EFITM) +
                           (CDCOLR EFCOLR))
CALL       PGM(PGME) /* Created using file JOINF as input */
CLOF       OPNID(FILEAB)
DLTOVR     FILE(JOINF)
```

앞의 순서대로 각 결합 필드 쌍을 지정할 필요는 없습니다. 예를 들어 다음의 JFLD 매개변수 값으로 같은 결과를 얻게 됩니다.

```
JFLD((CDCOLR EFCOLR)(ABITM CDITM) (CDITM EFITM))
```

각 결합 필드 쌍의 속성이 같을 필요는 없습니다. 문자 필드의 일반적인 채움(padding)과 숫자 필드의 십진 배열은 자동적으로 이루어집니다.

JDFTVAL 매개변수가 지정되지 않았으므로 *NO가 가정되고 결합 레코드를 구성하는 데에 디폴트 값이 사용되지 않습니다. JDFTVAL(*YES)를 지정하고 파일 FILEAB의 레코드와 동일한 결합 필드 값을 가진 레코드가 파일 FILECD 내에 없으면, 파일 FILEEF로 결합되는 데 있어 *Cddscp* 및 *Cdcolr* 필드에 대해 디폴트 값이 사용됩니다이러한 디폴트를 사용하면 파일 FILEEF 내에서 대응되는 레코드를 찾을 수 있습니다(디폴트가 2차 파일의 레코드와 대응하는지에 따라). 그렇지 않을 경우에는 디폴트 값이 이들 파일과 *Efqty* 필드에 나옵니다.

예 3: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합:

이 예는 DDS를 사용하지 않은 데이터베이스 파일을 동적으로 결합할 경우 맵핑된 필드를 결합 필드로 사용하는 방법을 표시합니다.

결합 필드 쌍의 하나로서 MAPFLD 매개변수에 정의된 필드를 사용할 수 있습니다. 이는 2차 파일 키가 단일 필드로 정의되고(예: 6자의 날짜 필드) 1차 파일에 같은 정보(예:월, 일 및 년)가 들어 있는 별도의 필드들이 있는 경우에 유용합니다. FILEA에 문자 필드 *Year, Month* 및 *Day*가 있고, YYMMDD 형식의 *Date* 필드가 있는 FILEB와 결합하려는 것으로 가정하십시오. 또한 필수 형식으로 파일 JOINAB를 정의한 것으로 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(JOINAB) TOFILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA FILEB) FORMAT(JOINAB) +
                           JFLD((YYMMDD FILEB/DATE)) +
```

```
                          MAPFLD((YYMMDD 'YEAR *CAT MONTH *CAT DAY'))
CALL       PGM(PGME) /* Created using file JOINAB as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(JOINAB)
```

MAPFLD 매개변수는 *YYMMDD* 필드를 FILEA에 있는 여러 개의 필드를 연결한 것으로 정의합니다. 시스템이 표현식에서 속성을 결정하기 때문에 MAPFLD 매개변수에 *YYMMDD* 필드의 필드 속성(예: 길이나 유형)을 지정할 필요는 없습니다.

2차 결합 파일에서 누락 레코드 처리:

시스템은 2차 파일에서 누락 레코드에 디폴트 값을 허용할 것인지의 여부를 사용자가 제어할 수 있게 합니다(결합 논리 파일에 대한 JDFTVAL DDS 키워드와 유사함). 또한 디폴트 값을 가진 레코드만 처리하도록 지정할 수도 있습니다. 즉 사용자는 2차 파일에서 누락 레코드가 있는 레코드만 선택할 수 있습니다.

예: 2차 결합 파일에서 누락 레코드의 처리

이 예는 2차 파일에 레코드가 없는 1차 파일에서 레코드를 읽는 방법을 표시합니다.

162 페이지의 『예 1: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합』에서는 JDFTVAL 매개변수가 지정되지 않았으므로 FILEA에서 FILEB로 성공적으로 결합된 레코드만 읽힙니다. FILEB 내에 대응되는 레코드가 없는 FILEA 내의 레코드의 리스트를 원할 경우 다음 예에서와 같이 JDFTVAL 매개변수에 *ONLYDFT를 지정하면 됩니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA FILEB) FORMAT(FILEA) +
                          JFLD((CUST FILEB/CUST)) +
                          MAPFLD((CUST 'FILEA/CUST')) +
                          JDFTVAL(*ONLYDFT)
CALL       PGM(PGME) /* Created using file FILEA as input */
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

JDFTVAL(*ONLYDFT)는 2차 파일(FILEB) 내에 동일한 레코드가 없을 때에만 레코드가 프로그램으로 리턴되도록 합니다.

FILEB의 필드에 대해 결합 조작에서 리턴되는 값은 모두 디폴트 값이므로 FILEA의 형식만 사용하는 것이 일반적입니다. 나오는 레코드는 FILEB 내에 대응되는 레코드가 없는 레코드들입니다. FORMAT 매개변수는 FILE 매개변수에 하나 이상의 파일을 서술할 때에는 필요하나 지정되는 파일명은 FILE 매개변수에 지정된 파일 중 하나가 될 수 있습니다. 프로그램은 FILEA를 사용하여 작성됩니다.

그와 반대로, FILEA에 대응되는 레코드가 없는 FILEB 내의 레코드의 리스트를 얻을 수 있습니다. 이는 모든 스펙에서 2차 파일을 1차 파일로 만듦으로써 가능하게 됩니다. 다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(FILEB) SHARE(*YES)
OPNQRYF    FILE(FILEB FILEA) FORMAT(FILEB) JFLD((CUST FILEA/CUST)) +
                          MAPFLD((CUST 'FILEB/CUST')) JDFTVAL(*ONLYDFT)
CALL       PGM(PGMF) /* Created using file FILEB as input */
CLOF       OPNID(FILEB)
DLTOVR     FILE(FILEB)
```

주: OPNQRYF(조회 파일 열기) 명령의 FILE 매개변수에 첫 번째 파일을 지정해야 하므로 이 예의 OVRDBF(데이터베이스 파일 대체) 명령은 FILE(FILEB)을 사용합니다. CLOF(파일 닫기) 명령에도 FILEB를 지정합니다. JFLD 및 MAPFLD 매개변수도 변경됩니다. 프로그램은 FILEB를 사용하여 작성됩니다.

고유 키 처리:

고유 키 처리를 사용하여 그룹의 첫 번째 레코드만 처리할 수 있습니다. 그룹은 동일한 키 값을 가진 한 개 이상의 레코드로 정의됩니다. 첫 번째 레코드를 처리한다는 것은 레코드가 고유 키를 가진다는 의미입니다.

고유 키 처리 사용 시 파일은 순차적으로만 읽을 수 있습니다. 키 필드는 지정된 정렬 순서(SRTSEQ)와 언어 식별자(LANGID)에 따라서 분류됩니다.

고유 키 처리를 지정하고, 파일에 실제로 중복된 키가 있을 경우 동일한 키 값이 있는 각 레코드 그룹에 대해 레코드 하나만 수신합니다.

관련 참조

159 페이지의 『예 3: DDS를 사용하지 않고 키순 액세스 경로 지정』
이 예는 고유 가중치 정렬 순서를 사용하여 레코드를 배열하는 방법을 표시합니다.

160 페이지의 『예 4: DDS를 사용하지 않고 키순 액세스 경로 지정』
이 예는 공유 가중치 정렬 순서를 사용하여 레코드를 배열하는 방법을 표시합니다.

예 1: 고유 키 처리:

이 예는 고유 키 레코드만 읽는 방법을 표시합니다.

Cust 필드에 중복 키가 있는 레코드가 있는 FILEA를 처리하는 것으로 가정하십시오. 프로그램 PGMF에 의해서 *Cust* 필드의 각 고유한 값에 대해 첫 번째 레코드만 처리할 경우에는 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) KEYFLD(CUST) UNIQUEKEY(*ALL)
CALL       PGM(PGMF)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

예 2: 고유 키 처리:

이 예는 키 필드의 일부만 사용하여 레코드를 읽는 방법을 표시합니다.

Slsman, Cust, Date 등의 순서를 가진 동일한 파일을 처리하지만 각 *Slsman* 및 *Cust*에 한 개의 레코드만 원하는 것으로 가정하십시오. 또한 파일 내의 레코드가 다음과 같은 것으로 가정하십시오.

Slsman	Cust	날짜	Record #
01	5000	880109	1
01	5000	880115	2
01	4025	880103	3
01	4025	880101	4
02	3000	880101	5

첫 번째 키 필드부터 시작하여 고유한 키 필드 번호를 지정합니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) KEYFLD(SLSMAN CUST DATE) UNIQUEKEY(2)
CALL       PGM(PGMD)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

프로그램은 다음과 같은 레코드를 검색합니다.

Slsman	Cust	날짜	Record #
01	4025	880101	4
01	5000	880109	1
02	3000	880101	5

주: 널값은 동일한 것으로 처리되므로 첫 번째 널값만이 리턴됩니다.

기존 필드 정의에서 파생된 필드 정의:

다음은 기존 필드 정의에서 파생된 필드를 정의하여 허용된 연산입니다.

맵핑된 필드 정의는 다음을 허용합니다.

- 선택값을 지정하는 내부 필드를 작성할 수 있도록 합니다(152 페이지의 『예 7: OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 선택』 참조).
- 여러 개의 파일에 동일한 필드명이 있을 경우 혼동을 피할 수 있도록 해줍니다(162 페이지의 『예 1: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합』 1 참조).
- 처리될 형식에만 존재하며 데이터베이스 자체에는 존재하지 않는 필드를 작성할 수 있도록 합니다. 이를 사용하여 변환, 서브스트링, 연결 및 복잡한 수학 연산 등을 수행할 수 있습니다. 다음 예에서는 이러한 기능을 설명합니다.

예 1: 기존 필드 정의에서 파생된 필드 정의:

이 예는 파생된 필드 사용을 표시합니다.

레코드 형식에 *Price* 필드와 *Qty* 필드가 있는 것으로 가정하십시오. 파생된 *Exten* 필드를 작성하기 위해 OPNQRYF(조회 파일 열기) 명령을 사용하여 한 개의 필드에 다른 필드를 곱할 수 있습니다. FILEA를 처리하려고 하며 이미 FILEAA를 작성하였습니다. 파일의 레코드 형식에 다음과 같은 필드가 들어 있는 것으로 가정하십시오.

FILEA	FILEAA
Order	Order
Item	Item
Qty	Exten
Price	Brfdsc
Descrp	

Exten 필드는 맵핑된 필드입니다. 그 값은 *Qty*에 *Price*를 곱함으로써 결정됩니다. 새로운 형식에서는 *Qty*나 *Price* 필드가 반드시 있을 필요는 없지만 사용자가 원하면 이들은 새로운 형식으로 존재할 수도 있습니다. *Brfdsc* 필드는 *Descrp* 필드의 간단한 설명입니다(처음 10자를 사용함).

새로운 형식을 처리하기 위해 PGMF를 지정한 것으로 가정하십시오. 이 프로그램을 작성하기 위해 읽을 파일로서 FILEAA를 사용하십시오. 다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FILEAA) +
                           MAPFLD((EXTEN 'PRICE * QTY') +
                           (BRFDSC 'DESCRP'))
CALL       PGM(PGMF) /* Created using file FILEAA as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(FILEAA)
```

Exten 필드의 속성은 FILEAA의 레코드 형식에서 정의된 속성임에 유의하십시오. 필드에 대해 계산된 값이 너무 클 경우 예외가 프로그램에 전달됩니다.

필드의 시작으로부터 문자만을 원할 경우 *Brfdsc* 필드에 맵핑하기 위해 서브스트링 기능을 사용할 필요는 없습니다. *Brfdsc* 필드의 길이는 FILEAA 레코드 형식에 정의되어 있습니다.

FORMAT 매개변수에 지정된 형식의 모든 필드는 OPNQRYF 명령에서 서술되어야 합니다. 즉 출력 형식 내의 모든 필드는 FILE 매개변수에 지정된 파일의 레코드 형식 중 하나에 존재하거나 MAPFLD 매개변수에 정의되어야 합니다. FORMAT 매개변수의 형식에 프로그램에서 사용하지 않는 필드가 있는 경우 MAPFLD 매개변수를 사용하여 그 필드에 0이나 공백을 지정할 수 있습니다. 출력 형식에서 *Fldc* 필드가 문자 필드이고 *Fldn* 필드는 숫자 필드이며 프로그램에서 이 두 값 다 사용하지 않는다고 가정할 때 다음과 같이 지정하여 OPNQRYF 명령상의 오류를 피할 수 있습니다.

```
MAPFLD((FLDC ' " " ')(FLDN 0))
```

따옴표로 공백 값을 묶어야 함에 유의하십시오. 사용하지 않는 필드의 정의에 상수를 사용함으로써 OPNQRYF 명령을 사용할 때마다 고유한 형식을 작성하지 않아도 됩니다.

예 2: 기존 필드 정의에서 파생된 필드 정의:

이 예는 내장 함수 사용을 표시합니다.

FILEA에서 *Fldm* 필드의 사인(sine)인 수리적 함수를 계산하는 것으로 가정하십시오. 우선 다음과 같은 필드가 들어 있는 레코드 형식의 파일(FILEAA로 가정)을 작성하십시오.

FILEA	**FILEAA**
코드	코드
Fldm	Fldm
	Sinm

그런 다음 FILEAA를 입력용으로 사용하여 프로그램을 작성할 수 있으며(PGMF라고 가정), 다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FILEAA) +
                              MAPFLD((SINM '%SIN(FLDM)'))
CALL       PGM(PGMF) /* Created using file FILEAA as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(FILEAA)
```

내장 기능 %SIN은 인수로 지정된 필드의 sine을 계산합니다. *Sinm* 필드는 FORMAT 매개변수에서 지정된 형식으로 정의되므로 OPNQRYF(조회 파일 열기) 명령은 사인 값(부동 소수점 값)의 내부 정의를 *Sinm* 필드의 정의로 변환합니다. 이 기능을 사용하여 부동 소수점 필드 사용 시 특정 고급 언어에서 제한되는 사항을 피할 수 있습니다. 예를 들어, *Sinm* 필드를 팩 십진 필드로 정의한 경우 그 값이 부동 소수점 필드를 사용하여 작성된 것이라도 PGMF는 임의의 고급 언어를 사용하여 작성될 수 있습니다.

사인 이외에도 사용 가능한 여러 가지 기능이 있습니다. 내장 함수에 대한 전체 리스트를 보려면 제어 언어(CL) 주제에서 OPNQRYF 명령을 참조하십시오.

관련 개념

제어 언어(CL)

예 3: 기존 필드 정의에서 파생된 필드 정의:

이 예는 파생된 필드 및 내장 함수 사용을 표시합니다.

앞의 예에서 *Fldx*라는 필드도 FILEA에 존재하며 이 *Fldx* 필드는 *Fldm* 필드의 사인을 보유하는 데 사용되는 적절한 속성이 있는 것으로 가정하십시오. 또한 *Fldx* 필드의 내용을 사용하지 않는 것으로 가정하십시오. MAPFLD 매개변수를 사용하여 필드를 고급 언어 프로그램으로 전달하기 전에 그 필드의 내용을 변경할 수 있습니다. 예를 들어, 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) MAPFLD((FLDX '%SIN(FLDM)'))
CALL       PGM(PGMF) /* Created using file FILEA as input */
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

이 경우 FORMAT 매개변수에 다른 레코드 형식을 지정할 필요가 없습니다. (디폴트는 FILE 매개변수의 첫 번째 파일 형식을 사용합니다.) 따라서 프로그램은 FILEA를 사용하여 작성됩니다. 이 기능 사용 시, 다시 정의하는 필드가 계산된 값을 정확하게 처리할 수 있는 속성을 가지고 있는지 확인해야 합니다. 가장 쉬운 방법은 각 조회에서 처리하려고 하는 특정 필드를 가진 별도의 파일을 작성하는 것입니다.

또한 맵핑된 필드 정의 및 %XLATE 기능과 함께 이 기능을 사용하여 필드를 변환함으로써, 해당 필드가 데이터베이스에 존재할 때와는 다른 형태로 프로그램에 제공되도록 할 수 있습니다. 예를 들어 소문자 필드를 변환하여 프로그램이 대문자로만 보도록 할 수 있습니다.

정렬 순서와 언어 식별자는 %MIN과 %MAX 내장 기능의 결과에 영향을 줄 수 있습니다. 예를 들어, 문자의 대소문자 버전은 선택한 정렬 순서와 언어 식별자에 따라 같을 수도 있으며 다를 수도 있습니다.

주: 변환된 값은 최소값과 최대값을 결정하는 데 사용되지만 변환되지 않은 값은 결과 레코드로 리턴됩니다.

예에서는 FILEA를 입력 파일로 사용합니다. OPNQRYF(조회 파일 열기) 명령을 사용하여 자료를 갱신할 수도 있습니다. 그러나 맵핑된 필드 정의를 사용하여 필드를 변경하면 필드에 대한 갱신은 무시됩니다.

0으로 나눔 오류 처리:

0으로 나누는 것은 OPNQRYF(조회 파일 열기) 명령에 의해 오류로 간주됩니다. 이 주제에서는 답으로 0을 원하는 경우 0으로 나눔 오류를 처리하는 방법에 대해 설명합니다.

일반적으로, 레코드 선택은 필드 맵핑 오류가 발생하기 이전에 수행됩니다. 따라서 0으로 나누기 오류를 발생시키는 레코드는 생략할 수 있으며, 이 경우 OPNQRYF 명령에 의한 처리는 계속됩니다. 답으로 0을 원할 경우 다음은 일반적인 상업 자료에 대한 실제적인 솔루션입니다.

A를 B로 나누어 C를 구하는 것으로 가정하십시오(A/B = C로 기술함). 또한 B가 0이 될 수 있는 다음과 같은 정의를 가정하십시오.

필드	자릿수	소수 자릿수
A	6	2
B	3	0
C	6	2

다음과 같은 알고리즘이 사용될 수 있습니다.

```
(A * B) / %MAX((B * B) .nnnn1)
```

%MAX 기능은 B * B 또는 작은 값의 최대값을 리턴시킵니다. 작은 값은 충분한 선행 0을 가지고 있어서 B가 0이 아닌 이상 B * B로 계산되는 값보다 작아야 합니다. 이 예에서 B의 소수 자릿수는 0이므로 .1이 사용될 수 있습니다. 선행 0의 개수는 B의 소수 자릿수의 두 배가 되어야 합니다. 예를 들어, B의 소수 자릿수가 2이면 .00001을 사용해야 합니다.

다음과 같은 MAPFLD 정의를 지정하십시오.

```
MAPFLD((C '(A * B) / %MAX((B * B) .1)'))
```

첫 번째 곱셈은 B가 0일 경우 0인 피젯수를 만들기 위한 것입니다. 이렇게 하면 나눗셈 계산 시 결과가 0이 됩니다. B가 0일 경우에는 0에 의한 나눗셈이 일어나지 않으며 이는 .1 값이 젯수로 사용되기 때문입니다.

데이터베이스 파일 레코드의 자료 요약(그룹화):

그룹 처리 기능을 사용하여 기존의 데이터베이스 레코드에서 나온 자료를 요약할 수 있습니다.

다음과 같이 지정할 수 있습니다.

- 그룹화 필드
- 그룹화 전후의 선택값
- 새로운 레코드에 대한 키순 액세스 경로
- 각 그룹에서 레코드의 개수를 계산하는 것 이외에도 레코드의 합계, 평균, 표준 편차 및 분산 등과 같은 기능을 수행할 수 있게 하는 맵핑된 필드 정의

- 그룹화된 필드 값에 의해 가중치를 제공하는 정렬 순서와 언어 식별자

일반적으로 다음과 같은 유형의 필드만 포함하고 있는 레코드 형식으로 파일을 작성하여 시작합니다.

- 그룹화 필드. 그룹을 정의하는 GRPFLD 매개변수에서 지정됩니다. 각 그룹에는 모든 그룹화 필드에 대한 상수값 세트가 들어 있습니다. 그룹화 필드는 FORMAT 매개변수에서 지정한 레코드 형식에 나오지 않아도 됩니다.
- 총계 필드. 한 개 이상의 다음 내장 기능과 함께 MAPFLD 매개변수를 사용하여 정의됩니다.

  %COUNT
  그룹 내의 레코드 개수 계산

  %SUM
  그룹에서 필드 값의 합계

  %AVG
  그룹에서 필드의 산술 평균

  %MAX
  그룹에서 필드의 최대값

  %MIN
  그룹에서 필드의 최소값

  %STDDEV
  그룹에서 필드의 표준 편차

  %VAR
  그룹에서 필드의 분산
- 상수 필드. 필드 값에 상수가 들어가도록 합니다. OPNQRYF(조회 파일 열기) 명령이 출력 형식에 있는 모든 필드를 알아야 한다는 제한이 그룹화 기능에 대해서도 적용됩니다.

그룹 처리 사용 시 파일은 순차적으로만 읽힐 수 있습니다.

예: 데이터베이스 파일 레코드의 자료 요약(그룹화):

이 예는 그룹 처리 기능을 사용하여 기존 데이터베이스 레코드의 자료를 요약하는 방법을 표시합니다.

자료를 고객 번호별로 그룹화하고 amount 필드를 분석하는 것으로 가정하십시오. 데이터베이스 파일은 FILEA이며 다음과 같은 필드를 가진 레코드 형식의 FILEAA 파일을 작성합니다.

FILEA	**FILEAA**
Cust	Cust
유형	Count(고객당 레코드 수)
Amt	Amtsum(amount 필드의 합계)
	Amtavg(amount 필드의 평균)
	Amtmax(amount 필드의 최대값)

새 파일에서 필드를 정의할 때에는 그 필드가 결과가 들어가기에 충분한 크기인지 확인해야 합니다. 예를 들어, *Amt* 필드가 5자리로 정의되면 *Amtsum* 필드를 7자리로 정의해야 합니다. 연산 넘침(arithematic overflow)은 프로그램을 이상 종료시킵니다.

FILEA의 레코드에 다음과 같은 값이 있는 것으로 가정하십시오.

Cust	유형	Amt
001	A	500.00
001	B	700.00
004	A	100.00
002	A	1200.00
003	B	900.00
001	A	300.00
004	A	300.00
003	B	600.00

레코드를 인쇄하기 위해 FILEAA를 입력으로 사용하는 프로그램(PGMG)을 작성합니다.

```
OVRDBF     FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FILEAA) KEYFLD(CUST) +
                          GRPFLD(CUST) MAPFLD((COUNT '%COUNT') +
                              (AMTSUM '%SUM(AMT)') +
                              (AMTAVG '%AVG(AMT)') +
                              (AMTMAX '%MAX(AMT)'))
CALL       PGM(PGMG) /* Created using file FILEAA as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(FILEAA)
```

프로그램에 의해 검색된 레코드는 다음과 같습니다.

Cust	Count	Amtsum	Amtavg	Amtmax
001	3	1500.00	500.00	700.00
002	1	1200.00	1200.00	1200.00
003	2	1500.00	750.00	900.00
004	2	400.00	200.00	300.00

주: GRPFLD 매개변수를 지정하는 경우 그룹이 오름차순으로 표시되지 않을 수 있습니다. 특정 순서를 지정하려면 KEYFLD 매개변수를 지정해야 합니다.

이 예에서 *Amtsum* 값이 700.00을 초과하는 요약 레코드만 인쇄하는 것으로 가정하십시오. *Amtsum* 필드는 지정된 고객에 대한 총계 필드이므로 그룹화 이후 선택을 지정하기 위해 GRPSLT 매개변수를 사용합니다 GRPSLT 매개변수를 추가하십시오.

```
GRPSLT('AMTSUM *GT 700.00')
```

프로그램에 의해 검색된 레코드는 다음과 같습니다.

Cust	Count	Amtsum	Amtavg	Amtmax
001	3	1500.00	500.00	700.00
002	1	1200.00	1200.00	1200.00

Cust	Count	Amtsum	Amtavg	Amtmax
003	2	1500.00	750.00	900.00

OPNQRYF(조회 파일 열기) 명령은 그룹화 이전(QRYSLT 매개변수) 및 그룹화 이후(GRPSLT 매개변수)에 선택하는 것을 모두 지원합니다.

Type 필드가 A인 추가 고객 레코드를 선택한다고 가정해봅시다. *Type*은 파일 FILEA의 레코드 형식 필드이고 총계 필드가 아니기 때문에 다음과 같이 그룹화하기 전에 QRYSLT문을 추가하십시오.

```
QRYSLT('TYPE *EQ "A" ')
```

주: 선택에 사용되는 필드는 프로그램이 처리하는 형식으로 나오지 않아도 됩니다.

프로그램에 의해 검색된 레코드는 다음과 같습니다.

Cust	Count	Amtsum	Amtavg	Amtmax
001	2	800.00	400.00	500.00
002	1	1200.00	1200.00	1200.00

주: 그룹화하기 전에 선택이 이루어졌기 때문에 CUST 001의 값이 변경되었습니다.

이전의 QRYSLT 매개변수 값 이외에 출력을 *Amtavg* 필드별로 내림차순으로 배열하는 것으로 가정하십시오. 이는 OPNQRYF 명령의 KEYFLD 매개변수를 변경함으로써 가능합니다.

```
KEYFLD((AMTAVG *DESCEND))
```

프로그램에 의해 검색된 레코드는 다음과 같습니다.

Cust	Count	Amtsum	Amtavg	Amtmax
002	1	1200.00	1200.00	1200.00
001	2	800.00	400.00	500.00

최종 합계 전용 처리:

총계만 처리는 그룹화 필드를 지정하지 않는 특수 유형의 그룹화 작업으로서 한 개의 레코드만 출력됩니다. 출력은 레코드 하나뿐입니다. 그룹화에 대한 모든 특수 내장 기능이 지정될 수 있습니다. 또한 총계를 계산하는 데 사용될 레코드의 선택을 지정할 수 있습니다.

예 1: 총계만 처리:

이 예는 단순 총계 처리를 표시합니다.

데이터베이스 파일 FILEA가 있고 다음과 같이 최종 합계 레코드가 들어갈 파일 FINTOT를 작성하는 것으로 가정하십시오.

FILEA	FINTOT
코드	Count(선택된 모든 레코드의 개수)

FILEA	FINTOT
Amt	Totamt(amount 필드의 합계) Maxamt(amount 필드의 최대값)

FINTOT 파일은 최종 합계로 작성되는 단일 레코드를 보유하기 위해 특별히 작성됩니다. 다음과 같이 지정할 수 있습니다.

```
OVRDBF     FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) +
                           MAPFLD((COUNT '%COUNT') +
                               (TOTAMT '%SUM(AMT)') (MAXAMT '%MAX(AMT)'))
CALL       PGM(PGMG) /* Created using file FINTOT as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(FINTOT)
```

예 2: 총계만 처리:

이 예는 레코드를 선택하는 총계 처리를 표시합니다.

앞의 예를 *Code* 필드가 B인 레코드만 최종 합계에 포함되도록 변경하는 것으로 가정하십시오. 이 경우 다음과 같이 QRYSLT 매개변수를 추가하면 됩니다.

```
OVRDBF     FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) +
                           QRYSLT('CODE *EQ "B" ') MAPFLD((COUNT '%COUNT') +
                               (TOTAMT '%SUM(AMT)') (MAXAMT '%MAX(AMT)'))
CALL       PGM(PGMG) /* Created using file FINTOT as input */
CLOF      OPNID(FILEA)
DLTOVR     FILE(FINTOT)
```

총계 기능과 함께 GRPSLT 키워드를 사용할 수 있습니다. 지정한 GRPSLT 선택값은 최종 합계 레코드의 수신 여부를 결정합니다.

예 3: 총계만 처리:

이 예는 새 레코드 형식을 사용한 총계 처리를 표시합니다.

제어 언어(CL) 프로그램으로 새로운 파일/형식을 처리하는 것으로 가정하십시오. 파일을 읽고 최종 합계를 나타내는 메세지를 보내려 합니다. 다음과 같이 지정할 수 있습니다.

```
DCLF       FILE(FINTOT)
DCL        &COUNTA *CHAR LEN(7)
DCL        &TOTAMTA *CHAR LEN(9)
OVRDBF     FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) MAPFLD((COUNT '%COUNT') +
                               (TOTAMT '%SUM(AMT)'))
RCVF
CLOF      OPNID(FILEA)
CHGVAR     &COUNTA &COUNT
CHGVAR     &TOTAMTA &TOTAMT
SNDPGMMSG  MSG('COUNT=' *CAT &COUNTA *CAT +
                             ' Total amount=' *CAT &TOTAMTA);
DLTOVR     FILE(FINTOT)
```

숫자 필드를 즉시 메세지에 포함시키려면 문자 필드로 변환시켜야 합니다.

시스템에서 OPNQRYF(조회 파일 열기) 명령을 실행하는 방법 제어:

최적화 기능을 사용하여 조회의 결과를 어떻게 사용하는가를 지정할 수 있습니다.

OPNQRYF(조회 파일 열기) 명령을 사용할 경우 성능에 대해 고려해야 할 두 가지 단계가 있습니다. 첫 단계는 OPNQRYF 명령 자체가 실제로 처리될 때입니다. 이 단계에서는 OPNQRYF 명령이 조회 요구에 대해 기존의 액세스 경로를 사용할 것인지, 아니면 새로운 액세스 경로를 빌드할 것인지 결정합니다. 성능상의 고려사항이 적용되어야 하는 두 번째 단계는 어플리케이션 프로그램이 OPNQRYF 명령의 결과를 사용하여 자료를 처리할 때입니다.

대부분의 일괄처리 유형의 기능에서는 항상 위에서 언급한 두 가지 단계에 소요되는 총 시간에만 관심을 갖게 됩니다. 따라서 OPNQRYF 명령의 디폴트는 OPTIMIZE(*ALLIO)입니다. 이것은 OPNQRYF 명령이 위의 두 단계에 소요되는 총 시간을 고려한다는 의미입니다.

대화식 환경에서 OPNQRYF 명령을 사용할 경우 전체 파일 처리에는 관심을 두지 않습니다. 이 경우 사용자는 레코드로 가득 찬 첫 번째 화면이 가능한 한 신속하게 표시되기를 원할 것입니다. 그러므로 첫 번째 단계에서는 가능한 한 액세스 경로를 빌드하지 않으려 할 것입니다. 이 경우에는 OPTIMIZE(*FIRSTIO)를 지정하면 됩니다.

동일한 OPNQRYF 명령의 결과를 여러 프로그램에서 처리하려면 첫 번째 단계에서 효율적인 열린 자료 경로(ODP)가 작성되어야 합니다. 즉 사용자가 OPNQRYF 명령에 OPTIMIZE(*MINWAIT)를 지정함으로써 두 번째 단계에서 처리 프로그램이 읽어야 할 레코드 수를 최소화시킬 수 있습니다.

공유할 액세스 경로가 없을 경우 OPNQRYF 명령의 KEYFLD 또는 GRPFLD 매개변수가 액세스 경로를 작성하도록 요구하면 OPTIMIZE 항목과 관계없이 액세스 경로가 전체적으로 작성됩니다. 최적화는 주로 선택 처리에만 영향을 미칩니다.

관련 개념

데이터베이스 성능 및 조회 최적화

예 1: 시스템에서 OPNQRYF(조회 파일 열기) 명령을 실행하는 방법 제어:

이 예는 첫 번째 레코드 세트 최적화 방법을 표시합니다.

Code 필드가 B인 모든 레코드를 요구하는 대화식 작업이 있다고 가정합시다. 사용 중인 프로그램의 서브파일에는 화면당 레코드가 15개 포함됩니다. 결과의 첫 번째 화면을 가능한 한 신속하게 오퍼레이터에게 보내려고 한다면 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF   FILE(FILEA) QRYSLT('CODE = "B" ') +
                        SEQONLY(*YES 15) OPTIMIZE(*FIRSTIO)
CALL      PGM(PGMA)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

시스템은 *Code* 필드에 대한 액세스 경로가 이미 존재하는지에 관계없이 전체를 조회 완료하기 전에 조회 처리를 최적화시키고 첫 번째 버퍼에 레코드를 채웁니다.

예 2: 시스템에서 조회 파일 열기 명령을 실행하는 방법 제어:

이 예는 읽을 레코드 수를 최소화하도록 최적화하는 방법을 표시합니다.

OPNQRYF(조회 파일 열기) 명령에 의해 작성된 하나의 파일에 액세스할 프로그램이 여러 개 있다고 가정합니다. 이 경우 어플리케이션 프로그램이 원하는 자료만을 읽도록 성능을 향상시킬 수 있습니다. 이는 OPNQRYF가 가능한 한 효율적으로 선택을 해야함을 의미합니다. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF  FILE(FILEA) QRYSLT('CODE *EQ "B"') +
                  KEYFLD(CUST) OPTIMIZE(*MINWAIT)
CALL      PGM(PGMA)
POSDBF   OPNID(FILEA) POSITION(*START)
CALL      PGM(PGMB)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

파일 작성 및 FORMAT 매개변수 사용 시 고려사항:

이 주제에서는 파일 작성 및 FORMAT 매개변수 사용 시 고려사항에 대해 설명합니다.

FILE 매개변수에 여러 개의 파일을 지정하여 결합 처리를 요구할 경우 FORMAT 매개변수에 레코드 형식명을 지정해야 합니다(즉 FORMAT(*FILE)을 지정할 수 없음). 또한 레코드 형식명은 주로 그룹화 기능과 함께 지정되거나 파생된 필드를 정의하기 위해 MAPFLD 매개변수에 복합 표현식을 지정할 경우 지정됩니다. 다음 지침 및 규칙을 고려하십시오.

- 레코드 형식명은 사용자가 선택하는 이름입니다. 이 이름은 조회하려는 데이터베이스 파일 내의 형식명과 다를 수 있습니다.
- 필드명은 사용자가 선택하는 이름입니다. 필드명이 조회중인 데이터베이스 파일 내에서 고유할 경우 시스템은 필드의 값을 조회된 파일 레코드 형식(FILE 매개변수) 및 조회 결과 형식(FORMAT 매개변수)에 있는 같은 이름에 암시적으로 맵핑시킵니다.
- 필드명은 고유하나 FILE 매개변수에서 지정된 파일과 FORMAT 매개변수에서 지정된 파일의 속성이 서로 다르면, 자료는 암시적으로 맵핑됩니다.
- 파생 필드를 정의하기 위해 MAPFLD 매개변수를 사용할 경우 정확한 필드 속성이 사용되어야 합니다. 예를 들어 그룹화 %SUM 기능을 사용할 경우 합계가 들어가기에 충분한 크기의 필드를 정의해야 합니다. 그렇지 않으면 연산 넘침(arithematic overflow)이 발생하고 프로그램에 예외가 전달됩니다.
- FORMAT 매개변수에서 식별되는 레코드 형식에 맵핑되는 모든 필드 값은 십진 배열됩니다. 조회 결과의 레코드 형식에 소수 자릿수가 0인 5자리로 된 필드가 있으며 계산되었거나 그 필드에 맵핑되어야 할 값이 0.12345라고 가정합니다. 그러면 소수점의 오른쪽 자리들이 절단되기 때문에 필드에서 0이라는 결과를 얻게 됩니다.

관련 참조

162 페이지의『예 1: DDS를 사용하지 않은 데이터베이스 파일의 동적 결합』
이 예는 DDS를 사용하지 않는 데이터베이스 파일을 동적으로 결합하는 방법을 보여 줍니다.

레코드 배열 시 고려사항:

이 주제에서는 OPNQRYF(조회 파일 열기) 명령을 사용하여 레코드 배열 시 고려사항에 대해 설명합니다.

OPNQRYF 명령에 대한 디폴트 처리는 KEYFLD 매개변수에 지정된 순서와 충돌하지 않으면서 성능을 향상시키는 순서로 레코드를 배열시킵니다. 따라서 특정 키 필드의 이름을 지정하거나 KEYFLD(*FILE)를 지정하기 위해 KEYFLD 매개변수를 지정하지 않는 한, 프로그램에 리턴되는 레코드의 순서는 동일한 OPNQRYF 명령 실행 시마다 달라질 수 있습니다.

OPNQRYF 명령에 KEYFLD(*FILE) 매개변수 옵션을 지정하고, 작업 디폴트가 지정된 조회나 OPNQRYF SRTSEQ 매개변수에 *HEX 이외의 정렬 순서가 지정되면 실제 파일 순서를 반영하지 않는 순서로 레코드를 수신할 수 있습니다. 파일이 정해지면, 조회 정렬 순서가 파일의 키 필드에 적용되고 정보용 메세지 CPI431F가 나옵니다. 파일의 정렬 순서 및 대체 배열 순서표가 존재할 경우 순서 지정 시 무시됩니다. 이것은 전체 필드명에 대한 리스트가 업어도 사용자가 어떤 필드를 정렬 순서에 적용시킬지를 표시하게 합니다. 조회(예를 들어 *HEX)시 정렬 순서가 지정되지 않을 경우 순서화는 버전 2 릴리스 3 이전과 동일한 방법으로 수행됩니다.

분산 자료 관리 파일에 대한 고려사항:

OPNQRYF(조회 파일 열기) 명령은 분산 자료 관리(DDM) 파일을 처리할 수 있으나 몇 가지 고려사항이 있습니다.

FILE 매개변수에서 식별된 모든 DDM 파일은 동일한 IBM iSeries 시스템 또는 System/38 목표 시스템에 있어야 합니다. 그룹 처리를 지정하고 DDM 파일을 사용하는 OPNQRYF(조회 파일 열기) 명령은 소스 시스템과 목표 시스템이 동일한 유형이어야 합니다(모두 System/38이거나 모두 iSeries 시스템이어야 함).

고급 언어 프로그램 작성 시 고려사항:

고급 언어 프로그램 작성 시 몇 가지 고려사항이 있습니다.

FORMAT 매개변수를 생략할 경우 고급 언어 프로그램은 사용자가 데이터베이스 파일에 직접 액세스하는 것처럼 코딩됩니다. 선택 또는 순서 지정은 프로그램 외부에서 발생하며 프로그램은 사용자가 지정한 순서대로 선택한 레코드를 받게 됩니다. 프로그램은 사용자의 선택값에 의해 생략된 레코드는 받지 않습니다. 선택/생략값을 가진 논리 파일을 통해 처리할 경우에도 동일한 기능이 발생합니다.

FORMAT 매개변수를 사용하는 경우는 FORMAT 매개변수에 사용된 것과 동일한 파일명을 지정합니다. 프로그램은 이 파일에 실제로 자료가 들어 있는 것처럼 작성됩니다.

파일을 순차적으로 읽는 경우 고급 언어는 자동적으로 키 필드가 무시되도록 지정할 수 있습니다. 보통 프로그램이 도달순으로 레코드를 읽는 것처럼 작성합니다. OPNQRYF(조회 파일 열기) 명령에서 KEYFLD 매개변수가 사용되는 경우 사용자는 진단 메세지를 받게 되며 이 메세지는 무시될 수 있습니다.

파일을 키에 의해 임의로 처리할 경우 고급 언어는 키 스펙을 필요로 합니다. 선택값이 있으면, 이를 사용해 프로그램이 데이터베이스에 존재하는 레코드를 액세스하는 것을 방지할 수 있습니다. OPNQRYF 명령이 사용되었는지 또는 DDS 선택/생략 논리를 사용하여 작성된 논리 파일이 사용되었는지에 따라 임의 읽기에서 레코드 없음(record not found) 상태가 발생할 수 있습니다.

어떤 경우에는 연산 넘침 등과 같은 맵핑 오류로 인한 예외를 모니터링할 수 있습니다. 그러나 결과를 정확하게 처리하기 위해서는 모든 필드에 속성을 정의하는 것이 좋은 방법입니다.

관련 태스크

파일의 기존 레코드 형식 사용

OPNQRYF(조회 파일 열기) 명령은 레코드를 선택하고 프로그램에서는 그 선택값에 부합되는 레코드만 처리합니다. 이러한 방법을 사용하여 레코드 세트를 선택하거나 레코드를 저장 순서와는 다른 순서로 리턴시키거나 모두를 수행할 수 있습니다.

OPNQRYF(조회 파일 열기) 명령 수행 시 전달되는 메세지:

OPNQRYF(조회 파일 열기) 명령 수행 시 대화식 사용자에게 OPNQRYF 요구의 상태를 알려 주는 메세지가 전달됩니다. 예를 들어, 요구를 충족시키기 위해 OPNQRYF 명령으로 키순 액세스 경로가 작성될 경우 사용자에게 메세지가 전달됩니다.

OPNQRYF 명령 수행 중에는 다음과 같은 메세지가 전달될 수 있습니다.

메세지 ID	설명
CPI4301	조회 수행 중.
CPI4302	조회 수행 중
CPI4303	조회 수행 중
CPI4304	조회 수행 중
CPI4305	조회 수행 중
CPI4306	조회 수행 중파일에서 액세스 경로 작성 중...
CPI4307	조회 수행 중&2의 &1 파일에서 해쉬 표 작성 중.
CPI4011	조회 수행 중

이러한 상태 메세지가 표시되지 않게 하려면 제어 언어(CL) 주제에서 메세지 처리에 대한 설명을 참조하십시오.

작업이 디버그(STRDBG(디버그 시작) 명령 사용) 하에서 수행될 경우 또는 조회 옵션 파일 옵션 DEBUG_MESSAGES *YES로 요청된 경우 사용자의 작업 기록부로 메세지가 전달됩니다. 이 메세지는 프로세스 요구에 사용될 충족 메소드에 대해 설명합니다. 이 메세지들은 발생되는 최적화 처리에 대한 정보를 제공합니다. 최대 성능을 달성하기 위해 OPNQRYF 요구를 조정하기 위한 툴로서 메세지를 사용할 수 있습니다. 다음은 이러한 메세지를 나열한 것입니다.

CPI4321

파일에 대해 설정된 액세스 경로...

CPI4322

키순 파일에서 설정된 액세스 경로...

CPI4324

파일에서 설정된 임시 파일...

CPI4325

조회에 대해 설정된 임시 파일...

CPI4326

결합 위치에서 처리된 파일...

CPI4327

결합 위치 1에서 처리된 파일...

CPI4328

사용할 파일 액세스 경로

CPI4329

파일에 사용되는 도착 순서

CPI432A

시간종료된 조회 최적화 프로그램...

CPI432C

파일에 대해 고려된 모든 액세스 경로...

CPI432E

선택 필드가 다른 속성들과 맵핑됨

CPI432F

파일에 대한 액세스 경로 제안

CPI433B

옵션 파일 조회 갱신 불가능

CPI4330

파일 &1의 병행 &10 스캔에 사용되는 &6 타스크.

CPI4332

파일에 작성된 병행 색인에 &6 타스크가 사용됨

CPI4333

결합 처리에 사용되는 해싱 알고리즘.

CPI4338

파일의 비트맵 처리에 &1 액세스 경로가 사용됨

CPI4339

조회 옵션이 &1 라이브러리에 &2 파일을 회복함.

CPI4341

분산 조회 실행.

CPI4342

조회를 위해 분산 결합 실행.

CPI4345

임시 분배 결과 파일 &4 작성.

CPI4346

&2의 분산 조회 단계 &1에 대한 Optimizer 디버그 메세지가 따라옴.

CPI4347

조회가 복수 단계에서 처리되고 있습니다.

대부분의 메세지는 특별한 옵션이 수행되었어야 하는 이유를 제공합니다. 각 메세지에 대한 2차 레벨 텍스트는 옵션이 선택되는 이유에 대한 확장 설명을 제공합니다. 어떤 메세지는 OPNQRYF 요구의 성능 향상을 돕기 위한 방법을 제시합니다.

관련 개념

제어 언어(CL)

입력 작업 이상을 위해 OPNQRYF(조회 파일 열기) 명령 사용:

OPNQRYF(조회 파일 열기) 명령은 처리 유형을 결정하기 위해 OPTION 매개변수를 지원합니다. 디폴트는 OPTION(*INP)이며, 파일은 입력 전용으로만 열립니다. 또한 개방 조회 파일을 통해 레코드를 추가, 갱신 또는 삭제하기 위해 OPNQRYF 명령 및 고급 언어 프로그램에 기타 OPTION 값을 사용할 수 있습니다.

그러나 UNIQUEKEY, GRPFLD 또는 GRPSLT 매개변수를 지정하거나, 총계 함수 중 하나를 사용하거나 FILE 매개변수에 복수 파일을 지정하는 경우 파일을 입력 전용으로만 사용할 수 있습니다.

결합 논리 파일은 입력 전용 처리로 제한됩니다. group, join, union, distinct 또는 사용자 정의 표 기능이 뷰 정의에 지정되면 입력 전용 처리로만 뷰가 제한을 받습니다. 조회 Optimizer가 임시 파일을 작성하여 조회를 구현할 필요가 있는 경우 파일의 사용은 입력에만 제한됩니다.

필드 값을 현재 값에서 파일의 어떤 레코드에 있는 다른 값으로 변경하려면 OPNQRYF 명령과 특정 고급 언어 프로그램을 조합시켜서 사용할 수 있습니다. 예를 들어, *Flda* 필드가 ABC인 모든 레코드를 변경하여 *Flda* 필드를 XYZ로 만든다고 가정합시다. 다음과 같이 지정할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF   FILE(FILEA) OPTION(*ALL) QRYSLT('FLDA *EQ "ABC" ')
CALL      PGM(PGMA)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

프로그램 PGMA는 읽을 수 있는 모든 레코드를 처리합니다. 그러나 조회 선택은 *Flda* 필드가 ABC인 레코드만으로 제한합니다. 프로그램은 각 레코드 내의 필드 값을 XYZ로 변경한 후 레코드를 갱신합니다.

또한 OPNQRYF 명령을 사용하여 데이터베이스 파일 내의 레코드를 삭제할 수도 있습니다. 예를 들어, 레코드에 필드가 있을 경우 그 값이 X이면 해당 레코드를 삭제해야 한다고 가정합시다. 다음과 같이 프로그램은 읽힌 레코드를 모두 삭제하도록 작성되고 OPNQRYF 명령을 사용하여 삭제될 레코드를 선택할 수 있습니다.

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF   FILE(FILEA) OPTION(*ALL) QRYSLT('DLTCOD *EQ "X" ')
CALL      PGM(PGMB)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

OPNQRYF 명령을 사용하여 레코드를 추가할 수도 있습니다. 그러나 조회 스펙에 선택값이 포함되어 있으면 선택값으로 인해 추가된 레코드를 프로그램이 읽지 못하도록 할 수도 있습니다.

OPNQRYF(조회 파일 열기) 명령을 사용하여 날짜, 시간 및 시간소인 비교:

날짜, 시간 또는 시간소인값은 같은 자료 유형의 다른 값이나 해당 자료 유형의 스트링 표현과 비교될 수 있습니다.

모든 비교는 연대순으로 0001년 1월 1일에서 멀어질수록 그 시간값이 *커집니다*.

시간값과 시간값의 스트링 표시가 포함되는 비교에는 항상 초(second)가 포함됩니다. 스트링 표시에 초가 생략되면 0초를 의미합니다.

시간소인값에 관한 비교는 동일한 것으로 간주될 수도 있는 표시와 관계없이 시간순으로 이루어집니다. 따라서 다음 술부는 참(true)입니다.

TIMESTAMP ('1990-02-23-00.00.00') > '1990-02-22-24.00.00'

문자, DBCS 개방 또는 DBCS 선택 필드나 상수가 날짜, 시간 또는 시간소인으로 표시되면 다음 규칙이 적용됩니다.

날짜: 날짜 형식이 *ISO, *USA, *EUR, *JIS, *YMD, *MDY 또는 *DMY일 경우 필드나 리터럴 길이는 8 이상이어야 합니다. 날짜 형식이 *JUL(yyddd)이면 변수의 길이는 최소한 6(yy와 ddd 사이의 분리자를 포함해서)이어야 합니다. 필드나 리터럴은 공백으로 채울 수 있습니다.

시간: 모든 시간 형식(*USA, *ISO, *EUR, *JIS, *HMS)에 대해 필드나 리터럴의 길이는 4 이상이어야 합니다. 필드나 리터럴은 공백으로 채울 수 있습니다.

시간소인: 시간소인 형식(yyyy-mm-dd-hh.mm.ss.uuuuuu)의 경우 필드나 리터럴의 길이는 16 이상이어야 합니다. 필드나 리터럴은 공백으로 채울 수 있습니다.

OPNQRYF(조회 파일 열기) 명령을 사용하여 날짜, 시간 및 시간소인 산술 수행:

날짜, 시간 및 시간소인값은 가감될 수 있습니다. 이러한 연산에는 *기간(duration)*이라는 십진수가 포함될 수 있습니다. 이러한 주제에는 날짜, 시간 및 시간소인의 산술 연산 수행에 대한 규칙 스펙 및 기간 정의가 포함됩니다.

기간:

*기간*은 시간 간격을 표시한 숫자입니다. 이 주제에서는 기간의 여러 유형을 소개합니다.

날짜 기간

날짜 기간(date duration)은 DECIMAL(8,0) 숫자로 표현된 년, 월 및 일의 숫자를 나타냅니다. 올바로 해석되려면 숫자는 *yyyymmdd* 형식이어야 합니다. 여기서 *yyyy*는 년도를 나타내고 *mm*은 월, *dd*는 일을 나타냅니다. 표현식 HIREDATE - BRTHDATE에서 처럼 한 날짜값에서 다른 날짜값을 뺀 결과가 날짜 기간입니다.

레이블된 기간

레이블된 기간(labeled duration)은 %DURYEAR, %DURMONTH, %DURDAY, %DURHOUR, %DURMINUTE, %DURSEC 또는 %DURMICSEC 등 7개의 기간 내장 기능 중 하나에 대한 피연산자로 사용되는 숫자(표현식의 결과일 수 있음)로 표현된 특정 시간 단위를 나타냅니다. 그 연산은 각각 년, 월, 일, 시, 분, 초 및 마이크로초에 대한 것입니다. 할당된 숫자는 DECIMAL(15,0) 숫자에 지정된 것처럼 변환됩니다. 레이블된 기간은 다른 피연산자가 자료 유형 *DATE, *TIME 또는 *TIMESTP 값일 때 산술 연산자의 피연산자로만 사용될 수 있습니다. 따라서 표현식 HIREDATE + %DURMONTH(2) + %DURDAY(14)는 유효한 반면 HIREDATE + (%DURMONTH(2) + %DURDAY(14))는 유효하지 않습니다. 이 두 표현식 모두에서 레이블된 기간은 %DURMONTH(2)와 %DURDAY(14)입니다.

시간 기간

시간 기간(time duration)은 DECIMAL(6,0) 숫자로 표현된 시, 분 및 초의 숫자를 나타냅니다. 올바로 해석되려면 숫자는 *hhmmss* 형식이어야 하며 여기서 *hh*는 시간을 나타내고, *mm*은 분, *ss*는 초를 나타냅니다. 한 시간값에서 다른 시간값을 뺀 결과가 시간 기간입니다.

시간소인 기간

시간소인 기간(timestamp duration)은 DECIMAL(20,6)으로 표현된 년, 월, 일, 시, 분, 초 및 마이크로초의 숫자를 나타냅니다. 올바로 해석되려면 숫자는 *yyyymmddhhmmsszzzzzz* 형식이어야 합니다. 여기서 *yyyy, mm, dd, hh, mm, ss와 zzzzzz*는 각각 년, 월, 일, 시, 분, 초 및 마이크로초를 나타냅니다. 한 시간소인값에서 다른 시간소인값을 뺀 결과가 시간소인 기간입니다.

날짜, 시간 및 시간소인 산술 규칙:

날짜 및 시간값에 대해 수행할 수 있는 유일한 연산은 덧셈과 뺄셈입니다. 날짜나 시간값이 덧셈 피연산자이면 나머지 피연산자는 기간이어야 합니다. 이 주제에서는 날짜, 시간 및 시간소인 산술 수행에 대한 규칙을 설명합니다.

날짜와 시간값을 가진 덧셈 연산자 사용을 규제하는 특정 규칙은 다음과 같습니다.

- 한 피연산자가 날짜이면 나머지 피연산자는 날짜 기간 또는 년, 월 또는 일의 레이블된 기간이어야 합니다.
- 한 피연산자가 시간이면 나머지 피연산자는 시간 기간 또는 시, 분 또는 초의 레이블된 기간이어야 합니다.
- 한 피연산자가 시간소인이면 나머지 피연산자는 기간이어야 합니다. 모든 유형의 기간이 유효합니다.

날짜와 시간값에 대해 뺄셈 연산자를 사용하는 규칙은 덧셈의 경우와는 다릅니다. 날짜나 시간값은 기간에서 뺄 수 없는데 이는 두 개의 날짜와 시간값을 빼는 연산은 날짜나 시간값에서 기간을 빼는 연산과 같지 않기 때문입니다. 날짜와 시간값을 가진 뺄셈 연산자 사용에 대한 특정 규칙은 다음과 같습니다.

- 첫 번째 피연산자가 날짜이면 두 번째 피연산자는 날짜, 날짜 기간, 날짜의 스트링 표시 또는 년, 월, 일의 레이블된 기간이어야 합니다.
- 두 번째 피연산자가 날짜이면 첫 번째 피연산자는 날짜 혹은 날짜의 스트링 표시이어야 합니다.
- 첫 번째 피연산자가 시간이면 두 번째 피연산자는 시간, 시간 기간, 시간의 스트링 표시 또는 시, 분, 초의 레이블된 기간이어야 합니다.
- 두 번째 피연산자가 시간이면 첫 번째 피연산자는 시간 또는 시간의 스트링 표시이어야 합니다.
- 첫 번째 피연산자가 시간소인이면 두 번째 피연산자는 시간소인, 시간소인의 스트링 표시 또는 기간이어야 합니다.
- 두 번째 피연산자가 시간소인이면 첫 번째 피연산자는 시간소인 또는 시간소인의 스트링 표시이어야 합니다.

날짜 빼기:

한 날짜(DATE1)에서 다른 날짜(DATE2)를 뺀 결과가 두 날짜 사이의 년, 월 및 일 수를 지정하는 날짜 기간입니다.

결과의 자료 형태는 DECIMAL(8,0)입니다. DATE1이 DATE2보다 크거나 같으면 DATE1에서 DATE2를 뺍니다. 그러나 DATE1이 DATE2보다 작으면 DATE2에서 DATE1을 빼고 그 결과의 부호가 음부호로 됩니다. 다음의 단계적인 설명은 RESULT = DATE1 - DATE2 연산에 포함된 단계를 잘 나타내고 있습니다.

```
If %DAY(DATE2) <= %DAY(DATE1) ;
        then %DAY(RESULT) = %DAY(DATE1) - %DAY(DATE2).

If %DAY(DATE2) > %DAY(DATE1) ;
        then %DAY(RESULT) = N + %DAY(DATE1) - %DAY(DATE2) ;
        where N = the last day of %MONTH(DATE2). ;
      %MONTH(DATE2) is then incremented by 1.

If %MONTH(DATE2) <= %MONTH(DATE1) ;
         then %MONTH(RESULT) = %MONTH(DATE1) - %MONTH(DATE2).

If %MONTH(DATE2) > %MONTH(DATE1) ;
         then %MONTH(RESULT) = 12 + %MONTH(DATE1) - %MONTH(DATE2). ;
      %YEAR(DATE2) is then incremented by 1.

%YEAR(RESULT) = %YEAR(DATE1) - %YEAR(DATE2).
```

예를 들어, %DATE('3/15/2000') - '12/31/1999'의 결과는 215(또는 기간이 0년, 2개월 및 15일)입니다.

증가 및 감소 날짜:

날짜에 기간을 더하거나 날짜에서 기간을 뺀 결과는 날짜입니다.

(이러한 연산에 있어서 달은 캘린더 페이지와 동일함을 나타냅니다. 즉, 날짜에 월을 더하는 것은 그 날짜가 표시되는 페이지에서 시작해서 달력의 페이지를 넘기는 것과 같습니다.) 그 결과는 반드시 001년 1월 1일과 9999년 12월 31일 사이(9999년 12월 31일을 포함하여)의 날짜에 해당되어야 합니다. 연도의 기간을 더하거나 빼면 날짜의 연도 부분만 영향을 받습니다. 결과가 윤년이 아닌 해의 2월 29일인 경우(이 경우 일은 28일로 변경됨)를 제외하고 월과 일은 변경되지 않습니다.

마찬가지로 월의 기간을 더하거나 빼면 월과 필요한 경우 년도만 영향을 받습니다. 결과가 유효하지 않은 경우(예: 9월 31일)를 제외하고는 날짜의 일부분은 변경되지 않습니다. 결과가 유효하지 않은 경우에는 일이 월의 최종일로 설정됩니다.

물론 일(day)의 기간을 더하거나 빼면 날짜의 일(day) 부분과 잠재적으로는 월과 연도에도 영향을 미칩니다.

날짜 기간이 양수이든지 음수이든지 날짜에 더하거나 뺄 수 있습니다. 레이블된 기간의 경우 결과는 유효한 날짜를 나타냅니다.

양의 날짜 기간을 날짜에 더하거나 음의 날짜 기간을 날짜에서 빼면 그 날짜는 지정된 수의 년, 월 및 일수만큼 순서대로 증가합니다. 따라서, DATE1 + X(여기서, X는 양의 DECIMAL(8,0)(숫자)이며, 다음의 표현식과 같습니다. DATE1 + %DURYEAR(%YEAR(X)) + %DURMONTH(%MONTH(X)) + %DURDAY(%DAY(X))

양의 날짜 기간을 날짜에서 빼거나 음의 날짜 기간을 날짜에 더하면 날짜는 지정된 수의 일, 월 및 년수 만큼 순서대로 감소합니다. 따라서, DATE1 - X(여기서, X는 양의 DECIMAL(8,0)(숫자)이며, 다음의 표현식과 같습니다. DATE1 - %DURDAY(%DAY(X)) - %DURMONTH(%MONTH(X)) - %DURYEAR(%YEAR(X))

날짜에 기간 추가 시 주어진 날짜에 한 달을 추가하면 그 날짜가 한 달 후에 존재하지 않는 경우(이 경우 날짜는 나중 달의 최종일의 날짜로 설정됨)를 *제외하고*는 한 달 후의 같은 날짜가 제공됩니다. 예를 들어, 1월 28일에 한 달을 더하면 2월 28일이 되고 1월 29일, 30일 또는 31일에 한 달을 더하면 2월 28일 또는 윤년의 경우 2월 29일이 됩니다.

주: 주어진 날짜에 한 달 이상이 추가된 후 동일한 수의 달수를 결과에서 뺏을 때 최종 날짜가 반드시 원래 날짜와 같지는 않습니다.

시간 빼기:

한 시간(TIME1)에서 다른 시간(TIME2)을 뺀 결과가 두 시간 사이의 시, 분 및 초수를 지정하는 시간 기간입니다.

결과의 자료 형태는 DECIMAL(6,0)입니다. TIME1이 TIME2보다 크거나 같으면 TIME1에서 TIME2를 뺍니다. 그러나 TIME1이 TIME2보다 작으면 TIME2에서 TIME1을 빼고 그 결과의 부호는 음부호가 됩니다. 다음의 단계적인 설명은 RESULT = TIME1 - TIME2 연산에 관련된 단계를 잘 나타내고 있습니다.

If %SECOND(TIME2) <= %SECOND(TIME1) ;
then %SECOND(RESULT) = %SECOND(TIME1) - %SECOND(TIME2).

If %SECOND(TIME2) > %SECOND(TIME1) ;
then %SECOND(RESULT) = 60 + %SECOND(TIME1) - %SECOND(TIME2). ;
%MINUTE(TIME2) is then incremented by 1.

If %MINUTE(TIME2) <= %MINUTE(TIME1) ;
then %MINUTE(RESULT) = %MINUTE(TIME1) - %MINUTE(TIME2).

If %MINUTE(TIME2) > %MINUTE(TIME1) ;
then %MINUTE(RESULT) = 60 + %MINUTE(TIME1) - %MINUTE(TIME2). ;
%HOUR(TIME2) is then incremented by 1.

%HOUR(RESULT) = %HOUR(TIME1) - %HOUR(TIME2).

예를 들어, %TIME('11:02:26') - '00:32:56'의 결과는 102930(10시간, 29분, 30초)이 됩니다.

증가 및 감소 시간:

시간에 기간을 더하거나 시간에서 기간을 뺀 결과는 시간입니다. 시간의 넘침(overflow)이나 부족(underflow)은 무시되므로 결과는 항상 시간이 되도록 되어 있습니다.

시(hour) 기간을 더하거나 빼면 시간의 시 부분만 영향을 받습니다. 분과 초는 변경되지 않습니다.

마찬가지로 분의 기간을 더하거나 빼면 분과 필요한 경우 시(hour)만 영향을 받습니다. 시간의 초 부분은 변경되지 않습니다.

물론 초의 기간을 더하거나 빼면 시간의 초 부분과 잠재적으로는 분과 시에 영향을 미칩니다.

시간 기간은 양수이든지 음수이든지 시간에서 더하고 뺄 수 있습니다. 결과는 시, 분, 초 순서대로 지정된 숫자를 더하거나 뺀 시간입니다. TIME1 + X, 여기서 X는 DECIMAL(6,0) 숫자이며, 다음의 표현식과 같습니다.
TIME1 + %DURHOUR(%HOUR(X)) + %DURMINUTE(%MINUTE(X)) + %DURSEC(%SECOND(X))

시간소인 빼기:

한 시간소인(TS1)에서 다른 시간소인(TS2)을 뺀 결과는 두 시간소인간의 년, 월, 일, 시, 분, 초 및 마이크로초를 지정하는 시간소인 기간입니다.

결과의 자료 형태는 DECIMAL(20,6)입니다. TS1이 TS2보다 크거나 같으면 TS1에서 TS2를 뺍니다. 그러나 TS1이 TS2보다 작으면 TS2에서 TS1을 빼고 그 결과의 부호는 음부호로 됩니다. 다음의 단계적인 설명은 연산 RESULT = TS1 - TS2 연산에 관련된 단계를 잘 나타내고 있습니다.

```
If %MICSEC(TS2) <= %MICSEC(TS1) ;
        then %MICSEC(RESULT) = %MICSEC(TS1) - ;
        %MICSEC(TS2).

If %MICSEC(TS2) > %MICSEC(TS1) ;
        then %MICSEC(RESULT) = 1000000 + ;
        %MICSEC(TS1) - %MICSEC(TS2) ;
        and %SECOND(TS2) is incremented by 1.
```

시간소인의 초와 분 부분은 시간 감산 규칙에 지정된 것처럼 감산됩니다.

```
If %HOUR(TS2) <= %HOUR(TS1) ;
        then %HOUR(RESULT) = %HOUR(TS1) - %HOUR(TS2).

If %HOUR(TS2) > %HOUR(TS1) ;
        then %HOUR(RESULT) = 24 + %HOUR(TS1) - %HOUR(TS2) ;
        and %DAY(TS2) is incremented by 1.
```

시간소인의 날짜 부분은 날짜 감산 규칙에 지정된 대로 감산됩니다.

증가 및 감소 시간소인:

시간소인에 기간을 더하거나 시간소인에서 기간을 뺀 결과는 시간소인입니다.

시(hour)의 넘침(overflow) 또는 부족(underflow)이 결과의 날짜 부분에 영향을 미치는 경우(이 경우는 유효한 날짜 범위 내에 있어야 함)를 제외하고 날짜와 시간 연산은 사전에 정의된 대로 수행됩니다. 마이크로초는 초로 넘칩니다.

임의 처리에 OPNQRYF(조회 파일 열기) 명령 사용:

순차 처리에 OPNQRYF(조회 파일 열기) 명령을 사용할 수 있습니다. 임의 처리에도 OPNQRYF 명령을 사용할 수 있습니다.

OPNQRYF 명령을 임의 처리 조작(예를 들어, RPG/400 언어 조작 CHAIN 또는 COBOL/400 언어 조작 READ)에 사용할 수 있습니다. 그러나 그룹 기능 또는 고유 키 기능을 사용하는 경우에는 파일을 임의로 처리할 수 없습니다.

OPNQRYF(조회 파일 열기) 명령: 성능 고려사항:

다음은 OPNQRYF(조회 파일 열기) 명령 사용 시 성능 최적화를 위한 추가 정보 및 기술입니다.

OPNQRYF 명령이 기존의 키순 액세스 경로를 사용할 경우 성능을 최고로 향상시킬 수 있습니다. 예를 들어, 코드 필드가 B이고 코드 필드에 액세스 경로가 있는 모든 레코드를 선택하고자 할 경우 시스템은 그 레코드들을 읽고 실행 중에 선택(동적 선택)하기보다는 액세스 경로를 사용해서 선택(키 위치의 선택)할 수 있습니다.

다음 조건 중에 해당되는 것이 있으면 OPNQRYF 명령은 기존 색인을 사용할 수 없습니다.

- 액세스 경로의 키 필드가 서브스트링에서 만들어진 경우.
- 액세스 경로의 키 필드가 연결 기능에서 만들어진 경우.
- 다음 두 경우는 조회와 연관된(SRTSEQ 매개변수에 지정된) 정렬 순서표에 대한 사항입니다.
  - 정렬 순서표가 공유 가중치 순서표인 경우
  - 정렬 순서표가 액세스 경로와 연관된 순서표(정렬 순서표나 대체 배열 순서표)와 일치하지 않는 경우.
- 다음 두 경우는 조회와 연관된(SRTSEQ 매개변수에 지정된) 정렬 순서표에 대한 사항입니다.
  - 정렬 순서표가 고유 가중치 순서표인 경우.
  - 다음과 같이 정렬 순서표가 액세스 경로와 연관된 순서표(정렬 순서표나 대체 배열 순서표)와 일치하지 않는 경우.
    - 순서화가 지정됨(KEYFLD 매개변수).
    - 레코드 선택(QRYSLT 매개변수)이 *EQ, *NE, *CT, %WLDCRD 또는 %VALUES를 사용하지 않은 경우.
    - *EQ나 *NE 연산자를 사용하지 않은 결합 선택(JFLD 매개변수)이 존재함.

OPNQRYF 처리의 일부는 요구를 만족시키는 가장 신속한 방법이 무엇인가를 결정하는 것입니다. 만일 사용중인 파일이 크고 대부분의 레코드가 B인 *Code* 필드를 가지고 있는 경우 기존 키순 액세스 경로를 사용하는 것보다는 입력순으로 처리하는 것이 더 빠릅니다. 프로그램에서 표시하는 레코드는 동일합니다. 액세스 경로가 *Code* 필드에 있을 경우에만 OPNQRYF 처리가 이런 유형의 결정을 내릴 수 있습니다. 일반적으로 파일 내의 레코드 중 약 20% 이상을 요구할 경우 OPNQRYF 처리는 기존 액세스 경로를 무시하고 파일을 도달순으로 읽으려고 합니다.

Code 필드에 대해 액세스 경로가 없으면 프로그램은 파일의 모든 레코드를 읽고 선택된 레코드만 프로그램으로 전달합니다. 즉 파일이 입력순으로 처리됩니다.

시스템은 어플리케이션 프로그램보다 더 신속하게 선택을 수행할 수 있습니다. 만일 해당 키순 액세스 경로가 없을 경우에는 프로그램이나 시스템 중 한쪽이 처리할 레코드를 선택합니다. 시스템으로 하여금 선택 처리를 수행하게 하는 것이 어플리케이션 프로그램에 모든 레코드를 전달하는 것보다 훨씬 신속합니다.

이는 특히 갱신 조작을 위해 파일을 열 때 해당되는데, 그 이유는 개별적인 레코드가 프로그램에 전달되어야 하고 읽힌 각 레코드들이 잠겨지기 때문입니다(프로그램이 레코드를 갱신해야 할 경우). 시스템이 레코드 선택을 수행하므로 선택값을 만족시키는 레코드들만이 프로그램에 전달되고 잠겨집니다.

레코드를 특정 순서로 읽기 위해 KEYFLD 매개변수를 사용하는 경우 같은 키순 스펙을 사용하는 액세스 경로가 이미 존재하거나 사용자의 스펙과 유사한 키 액세스 경로(사용자가 지정한 모든 필드와 키의 끝에 추가의 필드를 포함하는 키와 같은)가 존재하면 가장 신속한 결과를 가져옵니다. 이는 GRPFLD 매개변수와 JFLD 매개변수의 to-field에서도 마찬가지입니다. 이러한 액세스 경로가 없을 경우 시스템은 액세스 경로를 빌드하고 파일이 작업에서 열려 있는 동안 이를 유지보수합니다.

정렬할 레코드 수(파일의 총 레코드 수일 필요는 없음)가 1000을 넘고 파일 레코드의 20%를 넘을 경우 존재하지 않는 액세스 경로에 의해 파일의 모든 레코드를 처리하는 것은 일반적으로 전체 레코드 배열을 사용하는 것만큼 효과적이지 않습니다. 일반적으로 정렬을 하는 것보다는 키순 액세스 경로를 구성하는 것이 신속하나 작업이 수행되는 총 시간을 고려해 볼 때 입력순 처리를 사용했을 때의 빠른 처리 때문에 자료 정렬을 선호하게 됩니다. 사용 가능한 액세스 경로가 이미 있을 경우 그 액세스 경로를 사용하는 것이 자료를 정렬하는 것보다 빠릅니다. 이것이 가장 빠른 레코드 처리 방법이라면 OPNQRYF(조회 파일 열기) 명령의 ALWCPYDTA(*OPTIMIZE) 매개변수를 사용하여 시스템이 전체 레코드 정렬을 할 수 있습니다.

모든 조회 레코드를 읽지 않을 경우 그리고 OPTIMIZE 매개변수가 *FIRSTIO 또는 *MINWAIT인 경우 검색하려는 레코드 수를 지정할 수 있습니다. 레코드 수가 리턴할 것으로 기대되는 총 조회 수보다 훨씬 적을 경우 시스템이 보다 빠른 액세스 방법을 선택할 수 있습니다.

그룹화 기능을 사용하는 경우 그룹화 이후 선택하는 것보다(QRYSLT 매개변수) 그룹화 이전에 선택을 지정(GRPSLT 매개변수)하는 것이 보다 빠른 결과를 가져옵니다. 총계 기능을 포함하는 비교에 대해서는 GRPSLT 매개변수만을 사용하십시오.

대부분의 OPNQRYF 명령 사용 시, 자료를 액세스하고 그 자료를 프로그램에 전달하는 데 새로운 또는 기존의 액세스 경로가 사용됩니다. OPNQRYF 명령의 어떤 경우에는 시스템이 임시 파일을 작성해야 합니다. 임시 파일 작성 시기에 대한 규칙은 복잡하지만 이러한 상황이 발생하는 전형적인 경우는 다음과 같습니다.

- 동적 결합을 지정하고 KEYFLD 매개변수가 서로 다른 실제 파일로부터의 키 필드를 서술하는 경우
- 동적 결합을 지정하고 GRPFLD 매개변수가 서로 다른 실제 파일로부터의 필드를 서술하는 경우
- GRPFLD와 KEYFLD 매개변수를 모두 지정하지만 이들이 같지 않은 경우
- KEYFLD 매개변수에 지정된 필드의 길이가 2000바이트보다 긴 경우
- 동적 결합과 OPTIMIZE 매개변수에 *MINWAIT를 지정하는 경우
- 결합 논리 파일을 사용하여 동적 결합을 지정하고, 결합 논리 파일의 결합 유형(JDFTVAL)이 동적 결합의 결합 유형과 대응되지 않는 경우
- 논리 파일을 지정하고 논리 파일 형식이 둘 이상의 실제 파일을 참조하는 경우
- SQL 뷰 지정 시 시스템에서 뷰의 결과를 포함할 임시 파일을 요구하는 경우
- ALWCPYDTA(*OPTIMIZE) 매개변수가 지정되고 임시 결과 사용 시 조회 성능이 향상되는 경우

동적 결합(JDFTVAL(*NO))이 발생할 경우 OPNQRYF 명령은 파일을 재배열하고 최소 선택 레코드를 가지고 있는 파일과 최대 레코드를 가지고 있는 파일을 결합함으로써 성능을 향상시킵니다. OPNQRYF 명령이 파일을 재배열하는 것을 방지하려면 JORDER(*FILE)를 지정하십시오. 이렇게 하면 OPNQRYF 명령이 파일을 FILE 매개변수에 지정된 순서대로 강제로 결합합니다.

관련 개념

데이터베이스 성능 및 조회 최적화

OPNQRYF(조회 파일 열기) 명령: 정렬 순서표에 대한 성능 고려사항:

정렬 순서표의 성능 최적화를 위한 추가 정보 및 기술에 대해 설명합니다.

그룹화, 결합 및 선택: OPNQRYF(조회 파일 열기) 명령 성능:

기존 색인을 사용할 때 Optimizer는 필드 선택과 결합 그리고 그룹화의 속성이 기존 색인에 있는 키의 속성과 일치하는지를 확인합니다.

또한 조회와 연관된 정렬 순서표는 기존 색인의 키 필드와 연관된 순서표(정렬 순서표 또는 대체 배열 순서)와 일치되어야 합니다. 순서표가 일치하지 않으면 기존 색인을 사용할 수 없습니다.

그러나 조회와 연관된 정렬 순서표가 고유 가중치 순서표(*HEX가 포함된)일 경우 몇 가지 최적화가 가능합니다. Optimizer는 다음과 같은 연산자나 기능을 사용하는 필드 그룹화, 필드 선택 및 결합 술어에 대해서 정렬 순서표가 지정되지 않은 것처럼 작용합니다.

- *EQ
- *NE
- *CT
- %WLDCRD
- %VALUES

Optimizer는 다음과 같은 경우 키가 필드 및 액세스 경로 모두와 일치하는 기존의 액세스 경로를 마음대로 사용할 수 있다는 장점을 갖게 됩니다.

- 순서표를 포함하지 않는 경우.
- 고유-가중치 순서표를 포함하는 경우(이 표는 조회와 연관된 고유-가중치 정렬 순서표와 일치하지 않아도 됩니다)

순서 지정: OPNQRYF(조회 파일 열기) 명령 수행:

Optimizer는 필드 순서화를 위해 마음대로 기존 액세스 경로를 사용할 수 없습니다. Optimizer가 순서화를 요구하는 정렬의 수행을 선택하지 않는 한, 색인과 조회에 연관된 정렬 순서표는 일치되어야 합니다.

정렬이 사용될 때 Optimizer는 선택 범주와 일치하는 기존 액세스 경로를 자유롭게 사용할 수 있으며 변환이 정렬중에 수행됩니다.

다른 데이터베이스 기능과의 성능 비교:

OPNQRYF(조회 파일 열기) 명령은 논리 파일 및 결합 논리 파일과 같은 데이터베이스 지원을 사용합니다. 따라서 키순 액세스 경로를 구성하거나 결합 조작 등과 같은 기능을 동일하게 수행합니다.

OPNQRYF 명령에 의해 수행되는 선택 기능(QRYSLT 및 GRPSLT 매개변수에 대하여)은 논리 파일의 선택/생략과 유사합니다. 중요한 차이점은 OPNQRYF 명령의 경우 시스템에서 사용 가능한 액세스 경로와 OPTIMIZE 매개변수에 어떤 값이 지정되었는지에 따라 액세스 경로 선택을 사용할 것인지 또는 동적 선택을 사용할 것인지(논리 파일에 대한 DDS에서 DYNSLT 키워드의 지정 여부와 유사함)를 시스템이 결정한다는 것입니다.

필드 용도:

그룹화 기능을 사용하는 경우 개방 조회 파일에 대한 레코드 형식(FORMAT 매개변수)의 모든 필드와 모든 키 필드(KEYFLD 매개변수)는 그룹화 필드(GRPFLD 매개변수에 지정됨) 또는 맵핑된 필드(MAPFLD 매개변수에 지정됨)여야 합니다.

맵핑된 필드는 그룹화 필드, 상수 및 총계 함수로만 정의됩니다.

총계 기능으로는 %AVG, %COUNT, %MAX(하나의 피연산자만 사용), %MIN(하나의 피연산자만 사용), %STDDEV, %SUM 및 %VAR 등이 있습니다. 다음 경우에 그룹 처리가 필요합니다.

- GRPFLD 매개변수에 필드명 그룹화를 지정할 때
- GRPSLT 매개변수에 그룹 선택값을 지정할 때
- MAPFLD 매개변수에 지정한 맵핑 필드가 그 정의에서 총계 함수를 사용할 때

큰 오브젝트 자료 유형 BLOB, CLOB 또는 DBCLOB이 지정된 필드는 CPYFRMQRYF(조회 파일에서 복사) 명령이나 SQL(Structured Query Language)을 통해서만 읽을 수 있습니다. 큰 오브젝트 필드 자료는 열린 조회 파일에서 곧바로 액세스할 수 없습니다. CPYFRMQRYF 명령은 열린 조회 파일에서 큰 오브젝트 필드에 액세스는 데 사용되어야 합니다. 큰 오브젝트 자료 유형(BLOB, CLOB 또는 DBCLOB)이 있는 필드는 OPNQRYF 매개변수 KEYFLD, UNIQUEKEY, JFLD 및 GRPFLD에 지정할 수 없습니다.

DATALINK 유형 필드는 선택, 그룹화, 순서화, 결합 시 표시되지 않습니다. DATALINK 필드가 그 형식에서 나오지 않으면 자료 공간에서 존재하기 때문에 프로세스되지 않는 형식으로 되돌아갈 것입니다.

레코드 형식에 포함되고 FILE 매개변수에서 식별되며, N이라는 용도(입력도 출력도 아님)로 정의(파일 작성에서 사용된 DDS에서)된 필드는 OPNQRYF 명령의 어떤 매개변수에서도 지정될 수 없습니다. I(입력 전용) 또는 B(입출력용)로 정의된 필드만이 지정 가능합니다. FORMAT 매개변수에서 식별된 레코드 형식에 N으로 정의된 용도를 가진 필드는 모두 OPNQRYF 명령에 의해 무시됩니다.

열린 조회 파일 레코드의 필드는 주로 FORMAT 매개변수에서 식별된 레코드 형식의 필드와 동일한 용도 속성(입력 전용 또는 입출력용)을 가지며, 예외는 아래에서 설명됩니다. 출력 또는 갱신 및 용도를 포함하는 옵션(OPTION 매개변수)에 대하여 파일이 열리는 경우 그리고 FORMAT 매개변수에서 식별된 레코드 형식의 B(입출력용) 필드가 개방 조회 파일 레코드 형식에서 I(입력 전용)로 변경된 경우 OPNQRYF 명령이 정보 메세지를 보냅니다.

결합 처리 또는 그룹 처리를 요구하는 경우 또는 UNIQUEKEY 처리를 지정하는 경우 조회 레코드 내의 모든 필드는 입력 전용으로 지정됩니다. 처리중인 파일(FILE 매개변수에서 식별된 파일)의 입력 전용 필드로부터의 맵핑은 개방 조회 파일 레코드 형식에서 입력 전용으로 주어집니다. MAPFLD 매개변수를 사용하여 정의된 필드는 일반적으로 개방 조회 파일에서 입력 전용으로 주어집니다. 다음 사항이 모두 해당할 경우 MAPFLD 매개변수에 정의된 필드에는 그 상수 필드 용도에 일치하는 값이 지정됩니다.

- 이 주제의 앞에서 설명한 조건으로 인해 입력 전용 필드를 필요하지 않습니다.
- MAPFLD 매개변수에 지정된 필드 정의 표현식이 필드명(연산자 또는 내장 함수가 없는)입니다.
- 필드 정의 표현식에 사용된 필드가 FILE 매개변수에 지정된 파일, 멤버, 레코드 형식 중 하나에 존재합니다(MAPFLD 매개변수를 사용하여 정의된 다른 필드가 아닌).

- 기본 필드와 맵핑 필드가 필드 유형에 있어서 서로 호환됩니다. (맵핑이 같은 길이의 존 필드와 문자 필드 사이에서 발생하지 않는 한 맵핑이 숫자와 문자 필드 유형을 혼합하지 않습니다.)
- 기본 필드가 0이 아닌 십진 정밀도의 2진 자료이면 맵핑된 필드 또한 반드시 2진 자료의 같은 정밀도를 사용해야 합니다.

작업에서 공유된 파일:

어플리케이션 프로그램이 OPNQRYF(조회 파일 열기) 명령으로 작성된 열린 자료 경로를 사용하기 위해서는 프로그램이 반드시 조회 파일을 공유해야 합니다. 프로그램에서 조회 파일을 공유 상태로 열지 않으면 컴파일 시 원래 사용하도록 지정되었던 파일을 완전히 열게 됩니다(OPNQRYF 명령에 의하여 작성된 조회 열린 자료 경로가 아님).

프로그램은 다음 조건에 따라 조회 열린 자료 경로를 공유하게 됩니다.

- 어플리케이션 프로그램은 공유된 대로 파일을 열어야 합니다. 프로그램은 조회된 첫 번째 또는 유일한 멤버(FILE 매개변수에 지정된 대로)가 SHARE(*YES) 속성을 갖고 있을 때 프로그램이 조건에 맞게 됩니다. 첫 번째 또는 유일한 멤버가 SHARE(*NO) 속성을 가지면 프로그램을 호출하기 전에 OVRDBF(데이터베이스 파일 대체) 명령에 SHARE(*YES)를 지정해야 합니다.
- 어플리케이션 프로그램에 의하여 열려진 파일은 OPNQRYF 명령에 의하여 열려진 파일과 동일한 이름을 가지고 있어야 합니다. 프로그램에 지정된 파일이 조회된 첫 번째 또는 유일한 멤버(FILE 매개변수에서 지정된 대로)와 같은 파일명 및 멤버명일 때 프로그램은 이 조건에 부합됩니다. 첫 번째 또는 유일한 멤버의 멤버명이 다를 때에는 OVRDBF(데이터베이스 파일 대체) 명령에 조회된 첫 번째 또는 유일한 멤버명 대신 프로그램이 컴파일되었을 당시의 파일명을 지정해야 합니다.
- 프로그램은 동일한 활성 그룹내에서 열린 자료 경로(ODP)의 조회로 유효하게 된 범위까지 실행되어야 합니다. ODP 조회가 해당 작업 범위까지 유효할 경우 프로그램은 작업 내 모든 활성 그룹에서 수행될 수 있습니다.

OPNQRYF 명령은 해당 작업이나 활성 그룹 내에서 기존 열린 자료 경로를 공유할 수 없습니다. 조회 파일을 여는 요청이 개방 자료 경로가 열기 요청에 있는 동일한 라이브러리, 파일 및 멤버명을 갖는 경우와 다음 중 하나가 참이면 오류 메세지로 실패합니다.

- OPNSCOPE(*ACTGRPDFN) 또는 OPNSCOPE(*ACTGRP)는 OPNQRYF 명령에 지정되며, 열린 자료 경로는 OPNQRYF 명령이 실행되는 동일한 활성 그룹 또는 작업의 범위를 갖습니다.
- OPNSCOPE(*ACTGRPDFN) 또는 OPNSCOPE(*ACTGRP)는 OPNQRYF 명령에 지정되며, 열리 자료 경로는 OPNQRYF 명령이 실행되는 동일한 활성 그룹 또는 작업의 범위를 갖습니다.

후속 공유 열기는 OPNQRYF 명령이 실행될 때 유효하던 같은 열기 옵션(예: SEQONLY)을 사용해야 합니다.

관련 개념

124 페이지의 『동일한 작업 또는 활성 그룹에서 데이터베이스 파일 공유』
기본적으로 데이터베이스 관리 시스템에서는 하나의 파일이 동시에 많은 사용자에 의해 읽히고 변경되도록 되어 있습니다. 그러나 SHARE 매개변수를 통해 동일한 작업이나 활성 그룹의 데이터베이스 파일을 공유할 수 있습니다.

레코드 형식 설명 변경 여부 검사:

레코드 형식 레벨 검사가 지정된 경우 프로그램이 컴파일되었던 당시의 레코드 형식에 대해 열린 조회 파일 레코드 형식의 형식 레벨 번호(FORMAT 매개변수에 식별됨)를 검사합니다. 이는 프로그램이 이전에 열렸던 조회 파일을 공유할 때 발생합니다.

다음 조건에 해당되는 경우 레코드 형식 레벨에 대한 프로그램의 공유 열기가 확인됩니다.

- 조회된 첫 번째 파일이나 유일한 파일(FILE 매개변수에 지정된 대로)에는 LVLCHK(*YES) 속성이 있어야 합니다.
- 조회된 첫 번째 파일이나 유일한 파일을 LVLCHK(*NO)로 대체해서는 안됩니다.

OPNQRYF(조회 파일 열기) 명령에 대한 기타 런타임 고려사항:

열린 조회 파일에서 복사 및 대체를 포함하여 OPNQRYF(조회 파일 열기) 명령에 대한 기타 런타임 고려사항을 읽어 보십시오.

대체 및 OPNQRYF(조회 파일 열기) 명령:

대체 속성은 개방 조회 파일에 의해 처리되어야 하는 파일명, 라이브러리명 및 멤버명을 변경할 수 있습니다. 그러나 OVRDBF(데이터베이스 파일 대체) 명령에 지정된 TOFILE, MBR, LVLCHK, INHWRT 또는 SEQONLY 이외의 매개변수 값은 OPNQRYF(조회 파일 열기) 명령에 의해 무시됩니다.

조회된 첫 번째 또는 유일한 멤버에 이름 변경 대체 속성이 적용되는 경우 추가 대체에는 새로운 이름을 지정해야 하며 OPNQRYF 명령의 FILE 매개변수에 지정된 이름을 지정해서는 안됩니다.

열린 조회 파일에서 복사:

CPYFRMQRYF(조회 파일로부터 복사) 명령은 열린 조회 파일에서 다른 파일로 복사하거나 레코드의 형식화된 리스트을 인쇄하는 데 사용됩니다.

입력, 갱신 또는 OPNQRYF(조회 파일 열기) 명령의 FILE 매개변수의 모든 조작 값을 사용하여 지정된 모든 개방 조회 파일(분산 자료 관리(DDM) 파일을 사용하는 것은 제외)은 CPYFRMQRYF 명령을 사용하여 복사할 수 있습니다. CPYFRMQRYF 명령은 논리 파일의 복사에는 사용될 수 없습니다.

비록 CPYFRMQRYF 명령이 열린 조회 파일의 개방 자료 경로를 사용할지라도 파일을 열지는 않습니다. 따라서 복사하고 있는 데이터베이스 파일에 SHARE(*YES)를 지정할 필요는 없습니다.

관련 개념

데이터베이스 파일 관리

예 1: 열린 조회 파일에서 복사:

이 예는 OPNQRYF(조회 파일 열기) 및 CPYFRMQRYF(조회 파일에서 복사) 명령을 사용하여 레코드 서브세트가 있는 파일 작성 방법을 표시합니다.

STATE 필드의 값이 Texas인 레코드만 들어 있는 CUSTOMER/ADDRESS 파일로부터 파일을 작성하려고 한다고 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OPNQRYF FILE(CUSTOMER/ADDRESS) QRYSLT('STATE *EQ "TEXAS"')
CPYFRMQRYF FROMOPNID(ADDRESS) TOFILE(TEXAS/ADDRESS) CRTFILE(*YES)
```

예 2: 열린 조회 파일에서 복사:

이 예는 OPNQRYF(조회 파일 열기) 및 CPYFRMQRYF(조회 파일에서 복사) 명령을 사용한 선택 사항을 기준으로 레코드를 인쇄하는 방법을 표시합니다.

CITY 필드 값이 Chicago인 FILEA에서 모든 레코드를 인쇄한다고 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OPNQRYF FILE(FILEA) QRYSLT('CITY *EQ "CHICAGO"')
CPYFRMQRYF FROMOPNID(FILEA) TOFILE(*PRINT)
```

예 3: 열린 조회 파일에서 복사:

이 예는 OPNQRYF(조회 파일 열기) 및 CPYFRMQRYF(조회 파일에서 복사) 명령을 사용하여 레코드 서브세트를 디스켓으로 복사하는 방법을 표시합니다.

FIELDB 값이 10인 FILEB에서 디스켓으로 모든 레코드를 복사한다고 가정하십시오. 다음과 같이 지정할 수 있습니다.

```
OPNQRYF FILE(FILEB) QRYSLT('FIELDB *EQ "10"') OPNID(MYID)
CPYFRMQRYF FROMOPNID(MYID) TOFILE(DISK1)
```

예 4: 열린 조회 파일에서 복사:

이 예는 OPNQRYF(조회 파일 열기) 및 CPYFRMQRYF(조회 파일에서 복사) 명령을 사용한 동적 결합 출력 사본 작성 방법을 표시합니다.

FILEA와 FILEB를 결합한 형식과 자료가 있는 실제 파일을 작성하려고 하고, 이 파일에 다음 필드가 포함된다고 가정하십시오.

```
FILEA        FILEB      JOINAB
Cust         Cust       Cust
Name         Amt        Name
Addr                    Amt
```

결합 필드는 Cust로, 양쪽 파일에 존재합니다. 파일을 결합하여 결과 사본을 새 실제 파일 MYLIB/FILEC에 저장하려면 다음과 같이 하십시오.

```
OPNQRYF FILE(FILEA FILEB) FORMAT(JOINAB) +
    JFLD((FILEA/CUST FILEB/CUST)) +
    MAPFLD((CUST 'FILEA/CUST')) OPNID(QRYFILE)
CPYFRMQRYF  FROMOPNID(QRYFILE) TOFILE(MYLIB/FILEC) CRTFILE(*YES)
```

CPYFRMQRYF 명령으로 MYLIB/FILEC 파일이 작성됩니다. 이 파일에는 FILEA의 파일 속성과 같은 파일 속성이 포함됩니다(일부 파일 속성은 변경될 수 있음). 파일 형식은 JOINAB와 같게 됩니다. 파일에는 Cust 필드를 사용하여 결합된 FILEA와 FILEB로부터 온 자료가 들어 있게 됩니다. MYLIB 라이브러리의 FILEC 파일은 제어 언어(CL) 명령(예: DSPPFM(실제 파일 멤버 표시) 명령) 및 유틸리티(예: 조회)가 있는 다른 실제 파일처럼 처리될 수 있습니다.

관련 개념

데이터베이스 파일 관리

OPNQRYF(조회 파일 열기) 명령 사용 시 일반적인 오류:

OPNQRYF(조회 파일 열기) 명령에 몇 가지 기능을 올바르게 지정해야 프로그램에서 올바른 경과를 얻을 수 있습니다.

문제가 발생하는 경우 DSPJOB(작업 표시) 명령을 가장 효율적인 툴로서 사용할 수 있으며, 이 명령은 파일 열기 옵션 및 파일 대체 옵션을 지원합니다. 문제가 있는 경우에는 이 두 옵션을 살펴보아야 합니다.

다음은 OPNQRYF 명령 사용 시 발생할 수 있는 가장 일반적인 문제점과 그 해결 방법을 나열한 것입니다.

- 공유 열린 자료 경로(ODP). OPNQRYF 명령은 공유 ODP를 통하여 조작됩니다. 파일이 정확하게 처리되려면 멤버가 공유 ODP에 대해 열려 있어야 합니다. 문제가 발생하는 경우 멤버가 열려 있는지 그리고 공유 ODP를 갖고 있는지를 확인하려면 DSPJOB 명령의 파일 열기 옵션을 사용하십시오.

  파일이 열리지 않는 이유는 주로 다음의 두 가지 경우입니다.

  – 처리될 멤버는 SHARE(*YES)여야 합니다. OVRDBF(데이터베이스 파일 대체) 명령을 사용하거나 멤버를 영구적으로 변경하십시오.

  – 파일이 닫혀 있습니다. 오류 메세지를 받거나 RCLRSC(자원 재생) 명령을 실행하는 프로그램보다 호출 스택에 있어 더 상위 레벨인 디폴트 활성 그룹에서 실행된 프로그램에서 OPNSCOPE(*ACTGRPDFN) 또는 TYPE(*NORMAL) 매개변수 옵션이 있는 OPNQRYF 명령을 수행했습니다. RCLRSC 명령을 수행한 프로그램보다 호출 스택에 있어 더 상위 단계에 있는 프로그램에서 열려 있었기 때문에 열린 조회 파일이 닫힙니다. 열린 조회 파일이 닫히면 OPNQRYF 명령을 다시 수행해야 합니다. 버전 2 릴리스 3 이전의 릴리즈에서는 TYPE(*NORMAL) 매개변수 옵션과 함께 OPNQRYF 명령을 사용할 경우 열린 조회 파일은 자원을 재생한 동일한 프로그램에서 열렸었다 하더라도 닫힙니다.

- 레벨 검사. 프로그램이 컴파일된 당시의 동일한 레코드 형식에 대해 사용자 프로그램이 수행되도록 하기 위해 레벨 검사가 사용됩니다. 레벨 검사에 문제가 있을 경우 보통 다음 중 하나가 원인이 됩니다.

  – 프로그램이 작성된 이후 레코드 형식이 변경되었습니다. 프로그램을 다시 작성하면 문제점이 정정됩니다.

  – 대체 속성으로 인해 프로그램에 틀린 파일이 제공되었습니다. 대체 속성이 정확하게 지정되었는지를 확인하기 위해 DSPJOB 명령에 파일 대체 옵션을 사용하십시오.

  – FORMAT 매개변수가 필요하나 이것이 지정되지 않았거나 잘못 지정되었습니다. 파일을 FORMAT 매개변수로 처리하는 경우 다음과 같은 점을 확인해야 합니다.

    - TOFILE 매개변수와 함께 사용된 OVRDBF 명령은 OPNQRYF 명령의 FILE 매개변수에 있는 첫 번째 파일을 서술합니다.

- FORMAT 매개변수가 프로그램을 작성하는 데 사용된 형식이 들어 있는 파일을 식별해야 합니다.

– FORMAT 매개변수가 다른 파일의 형식을 처리하는 데(예: 그룹 처리) 사용되었으나 OVRDBF 명령에서 SHARE(*YES)가 요구되지 않았습니다.

- 처리될 파일이 파일 끝에 도달했습니다. OPNQRYF 명령의 일반적인 사용은 파일을 한 번만 처리할 수 있도록 파일을 순차적으로 처리하는 것입니다. 이때 파일의 위치가 파일 끝에 도달했으므로 이를 다시 처리하려고 할 때 아무런 레코드도 받지 못하게 됩니다. 파일을 처음부터 다시 처리하려면 OPNQRYF 명령을 다시 수행하거나 처리하기 전에 파일을 다시 위치시켜야 합니다. POSDBF(데이터베이스 파일 위치 지정) 명령을 사용하거나 고급 언어 프로그램문을 사용하여 파일을 다시 위치시킬 수 있습니다.
- 레코드가 없습니다. 이는 FORMAT 키워드를 사용하였으나 OVRDBF 명령을 지정하지 않을 때 발생하게 됩니다.
- 구문 오류. OPNQRYF 명령의 스펙에서 지정될 때 시스템이 오류를 발견하였습니다.
- 조작이 유효하지 않습니다. 조회의 정의는 KEYFLD 매개변수를 포함하지 않으나 고급 언어 프로그램이 키 필드를 사용하여 조회 파일을 읽으려고 합니다.
- 확보(Get) 옵션이 유효하지 않습니다. 고급 언어 프로그램이 레코드를 읽거나 현재의 레코드 위치 앞에 레코드 위치를 설정하려고 시도하였으며, 조회 파일이 SQL문에서 그룹별(Group by) 옵션, 고유 키(Unique Key) 옵션, 또는 구별(Distinct) 옵션 중 하나를 사용하였습니다.

열린 자료 경로 고려사항:

이 주제에서는 열린 자료 경로(ODP)에 대한 고려사항을 요약합니다.

ODP에서 사용하는 파일, 라이브러리 및 파일 멤버 이름은 대체를 통해 다른 파일이나 파일 멤버 이름을 사용하도록 강요되는 경우를 제외하고는 FILE 매개변수에 지정된 첫 번째 파일 및 파일 멤버 이름과 동일합니다. 열린 조회 파일의 레코드 형식 이름은 FORMAT 매개변수에 지정된 것과 동일합니다.

OPNQRYF(조회 파일 열기) 명령은 SHARE(*YES)가 파일에 지정된 것처럼 공유된 ODP로 파일을 엽니다. 파일, 라이브러리 또는 파일 멤버 이름이 열린 조회 파일 이름과 다른 고급 언어(HLL) 프로그램으로 지정되면 대체 명령을 사용하여 올바른 파일, 라이브러리 및 멤버 이름을 지정해야 HHL 프로그램이 열린 조회 파일 ODP를 공유할 수 있습니다. 조회된 첫 번째 또는 유일한 멤버가 SHARE(*NO) 속성을 갖고 있으면 대체에 SHARE(*YES)를 지정해야 HLL 프로그램에서 조회 파일 ODP를 공유할 수 있습니다.

OPNQRYF 명령의 범위가 작업을 포함할 경우 동일한 파일의 후속 열기(조회 열기 제외)는 범위가 활성 그룹 또는 작업일 경우 ODP를 공유할 수 있습니다. OPNQRYF 명령의 범위가 활성 그룹을 포함할 경우 동일한 파일의 후속 열기(조회 열기 제외)는 동일한 활성 그룹 범위를 포함할 경우 ODP를 공유할 수 있습니다.

필드명:

다음은 OPNQRYF(조회 파일 열기) 명령 매개변수에 필드명을 지정하기 위한 요구사항을 나열한 것입니다.

MAPFLD 매개변수에 지정된 목록에서 요소의 첫 번째 부분으로 사용된 필드명은 단순한 이름이어야 하며, FORMAT 매개변수에서 식별된 레코드 형식의 필드명은 항상 단순한 이름으로 취급해야 합니다. OPNQRYF

명령 매개변수(QRYSLT, KEYFLD, JFLD, GRPFLD, GRPSLT 또는 MAPFLD 매개변수의 필드 정의 표현식 부분)에 지정된 다른 필드명은 완전한 필드명이며, 다음과 같이 지정됩니다.

필드명 MAPFLD 매개변수에 정의된 필드를 식별하거나 FILE 매개변수에 지정된 목록에 포함되는 모든 레코드 형식의 모든 필드명 중에서 고유한 필드명이 있는 단순 필드명을 지정하십시오. 지정된 필드명에 MAPFLD 매개변수 정의가 없고, 지정된 이름이 있는 필드를 포함하는 레코드 형식이 FILE 매개변수에 여러 개 있을 경우 FILE 매개변수의 목록에 동일한 파일 및 레코드 형식이 여러 번 지정되었다고 해도 이 형태가 허용되지 않습니다.

예를 들어, MAPFLD 매개변수에 AMOUNT라는 필드가 정의되면 AMOUNT가 유효합니다. FILE 매개변수에 지정된 임의의 레코드 형식의 AMOUNT라는 필드가 하나뿐이라면 MAPFLD 매개변수에 AMOUNT가 정의되지 않을 경우에도 역시 유효합니다.

파일명/필드명

단순 파일명이 FILE 매개변수에 지정된 모든 파일명 중에서 고유한 경우에만 레코드 형식이 필드를 포함하는 FILE 매개변수에 지정된 파일의 단순 이름으로 규정된 필드명을 지정하십시오. 동일한 단순 파일명이 FILE 매개변수에 지정된 리스트에서 여러 번 지정되면 다른 라이브러리, 멤버 또는 레코드 형식명을 사용하더라도 이 형식이 허용되지 않습니다.

예를 들어, WHS01 파일의 레코드 형식에 PARTNBR이라는 필드가 있고 파일명 WHS01이 FILE 매개변수에서 한 번만 지정된 경우 WHS01/PARTNBR이 유효합니다.

파일 번호/필드명

필드를 포함하는 레코드 형식에 대해 FILE 매개변수 리스트에서 몇 가지 요소로 규정되는 단순 필드명을 지정하십시오. 파일 번호 규정자는 선행 0 없이 지정해야 합니다. 이 형태는 FILE 매개변수에 지정된 리스트에서 동일한 단순 파일명이 두 번 이상 지정된 경우에만 필요합니다.

예를 들어, 2/BALDUE는 FILE 매개변수에 지정된 리스트에 있는 두 번째 파일 레코드 형식에 BALDUE라는 필드가 있을 경우에 유효합니다.

***MAPFLD/필드명**

필드가 MAPFLD 매개변수에 지정된 경우 특수 값 *MAPFLD로 규정되는 단순 필드명을 지정하십시오. 필드를 정의하면 이 형태는 단순 필드명을 규정자없이 지정하는 것과 같은 의미를 갖습니다. MAPFLD 매개변수에 필드를 정의하지 않으면 *MAPFLD를 지정할 수 없습니다.

예를 들어, AVGBAL 필드를 MAPFLD 매개변수에 지정된 맵핑된 필드 리스트 요소 중 하나의 첫 번째 부분으로 지정하면 *MAPFLD/AVGBAL이 유효합니다.

표현식:

이 주제에서는 OPNQRYF(조회 파일 열기) 명령에 지정된 표현식의 규약을 설명합니다.

QRYSLT, GRPSLT 및 MAPFLD 매개변수에 지정된 표현식은 다른 제어 언어(CL) 명령 매개변수에 지정된 표현식과 유사합니다. 필드 값 및 상수의 결합을 사용하여 논리, 관계, 숫자 및 스트링 연산을 수행합니다. 내장 함수뿐만 아니라 기호 및 명명된 연산자가 지원되며, 괄호를 사용하여 평가 순서를 제어합니다.

OPNQRYF 매개변수와 기타 CL 명령 매개변수에 지정된 표현식에는 차이점도 있습니다. QRYSLT, GRPSLT 및 MAPFLD 매개변수의 표현식과 일반 CL 표현식의 차이점은 다음과 같습니다.

- 표현식 스트링에 삽입된 공백이나 특수 문자가 있을 경우 이를 어포스트로피로 묶어야 합니다.
- 다음 차이점은 숫자 및 스트링 리터럴에 영향을 미칩니다.
  - 문자 스트링 상수는 작은 따옴표나 따옴표를 사용하여 인용됩니다.
  - 숫자 상수 앞과 뒤에 오는 0은 해당 속성의 중요한 부분입니다.
  - 부동 소수점 상수(특수 값 *INF 및 *NEGINF 포함)가 표현식에 사용됩니다.
- 다음 차이점은 CL 변수를 데이터베이스 필드와 대조합니다.
  - 접두부가 없는 앰퍼샌드(&)가 데이터베이스 필드명으로 사용됩니다.
  - 규정 필드명이 지원됩니다.
  - 데이터베이스 필드에 대한 '논리' 필드 유형이 없습니다.
  - 데이터베이스 필드에 대해 여러 가지 추가 자료 유형이 지원됩니다.
- 다음 CL 연산자는 OPNQRYF 명령에서 지원되지 않습니다.
  - *BCAT 또는 |>
  - *TCAT 또는 |<
- 다음 추가 연산자는 CL 지원 이상으로 지원됩니다.
  - //(나머지)
  - **(지수화)
  - *CT('포함' - 문자 스캔)
  - *XOR 또는 &&('논리적 예외 or')
- 다음 차이점은 내장 함수 지원에 영향을 미칩니다.
  - %SWITCH 내장 함수는 지원되지 않습니다.
  - 여러 가지 추가 내장 함수가 지원됩니다.
  - 일반적으로 내포된 내장 함수 및 내장 함수 인수용 표현식(예: '%LOG(%SIN(x))')이 허용됩니다.
  - 내장 함수 인수로서 표현식을 지원하려면 부호화된 숫자 값이나 표현식(예: '%MIN(3 (-2) x (y+4))')인 인수는 괄호로 묶어야 합니다.

다음 표는 QRYSLT, GRPSLT 또는 MAPFLD 매개변수의 표현식에 사용되는 모든 연산자의 우선순위를 표시합니다. *NOT 및 연산자를 제외하고 우선순위 1에서 5로 나열된 연산자만 MAPFLD 매개변수에 지정된 표현식에 허용됩니다.

우선순위	연산자
1	+, -(부호화된 숫자 값에 사용할 경우), *NOT, ¬
2	**
3	*, /,//(/ 앞에 및/또는 / 뒤에 공백이 있어야 함)
4	+, -(두 개의 피연산자 사이에 사용됨)

우선순위	연산자
5	*CAT, ||
6	*GT, *LT, *EQ, *GE, *LE, *NE, *NG, *NL, *CT, >, <, =, >=, <=, ¬=, ¬>, ¬<
7	*AND, &
8	*OR, *XOR, |, &&

¬ 및 *NOT 연산자를 제외하고, 우선순위가 1에서 4인 연산자는 숫자 연산자이므로 숫자 피연산자가 필요합니다. 우선순위가 5인 연산자는 스트링 연산자이므로 피연산자가 문자 또는 DBCS 스트링이어야 합니다. 우선순위가 6인 연산자는 관계 연산자라고 하며, 필드명이나 숫자 또는 스트링 표현식(상수가 아니라)인 피연산자가 하나 이상 있어야 합니다. 우선순위가 7 및 8인 연산자와 ¬ 및 *NOT 연산자(우선순위 1)는 논리 연산자입니다. 논리 표현식의 피연산자는 관계(관계 연산자를 해당 피연산자와 사용하여 구성된) 및 기타 논리 표현식입니다.

스트링 표현식의 피연산자(내장 함수용 스트링 피연산자 포함)는 문자 필드, DBCS 필드 및 상수입니다. 이런 표현식의 피연산자가 둘 다 DBCS 전용 필드이거나 상수일 경우 표현식 평가의 최종 결과는 DBCS 전용 필드 값입니다. 피연산자가 DBCS 또는 문자 필드 또는 상수의 조합일 경우 결과는 DBCS 개방 필드 값입니다. DBCS 필드가 연결되어 있을 경우 필드 사이의 외래 SI 및 SO 문자가 삭제됩니다.

- 부호가 앞에 붙은 연산자의 피연산자가 *BIN2가 아니라면(이 경우 결과는 *BIN4) + 또는 - 부호가 앞에 붙은 연산자로 산출한 결과에 피연산자와 동일한 속성이 있습니다. ** 연산자(지수화)의 결과는 배정밀도 부동 소수점 숫자(*FLT8)입니다. 두 개의 연산자를 필요로 하는 기타 숫자 연산자의 경우 두 연산자 중 하나가 부동 소수점 숫자이면 결과도 부동 소동 소수점 숫자(*FLT8)입니다. 두 연산자 모두 고정 소수점 숫자이면 시스템에서 다음 표의 정보를 사용하여 팩 십진(*DEC) 결과를 산출하는 데 필요한 전체 및 분수 자릿수를 판별합니다. 두 연산자 모두 정밀도가 0인 2진 필드 및/또는 정수 상수일 경우 연산자가 "/"가 아니면 결과는 *BIN4입니다. 이 경우 결과가 부동 소수점 결과와 같습니다. 필요한 전체 자릿수가 31을 넘을 경우 총 31자리의 결과 계산이 가능하도록 분수 자릿수가 줄어듭니다. 일부 분수 자릿수가 잘리고 마지막 계산 결과의 속성(해당 필드의 MAPFLD 매개변수에 지정된 속성)이 중간 결과의 정밀도보다 더 큰 정밀도를 요구할 경우 표현식 평가 중에 일부 정밀도가 손실되었음을 표시하는 경고 메세지가 전송됩니다.

연산	결과(총 자릿수)	결과(분수 자릿수)
+	MAX(d1-f1, d2-f2)+MAX(f1,f2)+1	MAX(f1, f2)
_	MAX (d1-f1, d2-f2)+MAX (f1,f2)+1	MAX(f1, f2)
*	d1+d2	f1+f2
/	31	31-(d1-f1+f2)
//	MIN(d1-f1,d2-f2)+MAX(f1,f2)	MAX(f1,f2)

범례:

d1 피연산자 1의 총 자릿수

f1 피연산자 1의 분수 자릿수

d2 피연산자 2의 총 자릿수

f2 피연산자 2의 분수 자릿수

숫자 또는 스트링 표현식이 MAPFLD 매개변수에 지정된 경우 최종 결과의 속성이 다음 두 방법 중 하나로 사용됩니다. 속성이 필드 값에 직접 사용되거나(*CALC 필드 유형이 지정되고 필드가 FORMAT 매개변수에서 식별된 원형 레코드 형식에 포함되지 않은 경우) 최종 결과가 MAPFLD 매개변수에 지정된 속성이나 FORMAT 매개변수로 식별된 레코드 형식의 필드 정의에 포함된 속성과 일치하도록 변경됩니다.

관계 연산자의 두 피연산자가 모두 상수일 수 있습니다. 관계 연산자의 왼쪽 및 오른쪽에 피연산자로 지정된 필드, 상수 또는 표현식은 동일한 유형(숫자 또는 스트링)이어야 합니다. 문자 필드가 DBCS 전용 필드와 연결될 수 없다는 점을 제외하고 문자 및 DBCS 필드 피연산자의 모든 조합이 허용됩니다.

DBCS 상수 유형에는 DBCS 전용 및 DBCS 개방 두 가지가 있습니다. DBCS 전용 상수에는 작은 따옴표 사이에 DBCS 자료만 포함됩니다. 이 자료는 SO/SI 문자로 묶여야 합니다. DBCS 개방 상수에는 DBCS 및 영숫자 자료 조합이 포함됩니다. SO 문자(HEX 0E)는 DBCS 문자 그룹의 시작을 표시하고, SI 문자(HEX 0F)는 이 그룹의 마지막 2바이트 문자 뒤에 옵니다.

숫자나 스트링 표현식이 QRYSLT 또는 GRPSLT 매개변수에 복합 선택 피연산자로 표시되면 선택 피연산자에 사용된 표현식의 최종 결과 속성이 다른 관계형 피연산자와 일치하도록 변경됩니다.

관계형 연산자의 피연산자에 반드시 동일한 속성이 있어야 하는 것은 아니지만 숫자 피연산자는 문자 피연산자와 혼용될 수 없습니다. 피연산자에 동일한 속성이 없을 경우 시스템은 연산을 수행하기 전에 이러한 속성을 동일한 속성으로 변경합니다(문자 스트링 피연산자의 길이가 다를 경우 *CT 연산자 제외). 이 변경은 두 피연산자가 고정 소수점 숫자 피연산자일 경우 팩 십진수 형식을 사용하고, 두 피연산자 중 하나가 부동 소수점 숫자일 경우 부동 소수점 형식을 사용합니다. 고정 소수점 숫자 피연산자를 변경하면 소수점이 배열되고 0으로 채워집니다. 숫자 유형을 변경하면 고정 소수점 숫자에 총 32자릿수 이상이 필요할 경우 분수 자릿수를 절단하거나 부동 소수점 숫자에 총 16자릿수 이상이 필요할 경우 중요도가 낮은 일부 자릿수를 드롭합니다. 길이가 짧은 피연산자를 공백으로 채우면 문자 피연산자가 변경됩니다.

*CT 연산자는 관계의 오른쪽으로 지정된 문자 스트링, 필드 또는 표현식 값이 있는지 판별하기 위해 관계의 왼쪽으로 지정되어야 하는 문자 필드 또는 스트링 표현식(단일 문자 스트링 리터럴로 구성된 표현식은 제외)을 스캔합니다. 두 번째 피연산자(검색 값)는 첫 번째 피연산자(기본 스트링)보다 길지 않아야 합니다.

스트링이 검색되면 관계가 충족되고 결과는 'true' 논리 값이 됩니다. 그렇지 않으면 결과가 'false' 논리 값이 됩니다. 다음 예는 이 프로세스를 표시합니다.

- BASEFLD 필드는 'THIS IS A TEST' 값을 포함합니다.
- TESTFLD 필드는 'TE' 값을 포함합니다.

표현식	결과
'BASEFLD *CT "IS A''''	True
'BASEFLD *CT TESTFLD	True
'BASEFLD *CT "X''''	False
'BASEFLD *CT TESTFLD || ''Z''''	False
'BASEFLD || "ABC'' *CT ''TAB''''	True

내장 함수:

여기 나열된 내장 함수는 QRYSLT 또는 GRPSLT 매개변수에 지정된 복합 선택 피연산자 또는 MAPFLD 매개변수에 파생된 필드를 정의하는 데 사용하는 표현식에 대해 지원됩니다.

숫자 인수는 숫자 필드, 숫자 상수 또는 숫자 표현식입니다. 스트링 인수는 숫자 필드, 문자 스트링 리터럴 또는 스트링 표현식입니다. 별도로 표시되지 않는 한 모든 내장 함수는 인수로 사용될 표현식(기타 내장 함수 포함)을 허용합니다.

FORMAT 매개변수로 식별되는 레코드 형식으로 표시되고, MAPFLD 매개변수의 표현식으로도 정의되는 필드의 경우 표현식은 다음 단락에 설명된 속성을 사용하여 계산됩니다. 그런 다음 결과 값이 필드 속성과 일치하도록 맵핑됩니다.

%ABSVAL(숫자 인수)

%ABSVAL은 숫자 인수를 허용하며 인수의 절대값을 리턴합니다. 인수가 *BIN2일 경우를 제외하고 (이 경우 리턴된 값은 *BIN4) 리턴된 값에는 인수와 동일한 속성이 포함됩니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 팩 십진수(*DEC)로 8자릿수 및 정밀도 0(날짜 기간), 6자릿수 및 정밀도 0(소요 시간) 또는 20자릿수 및 정밀도 6(시간소인 기간)입니다.

%ACOS(숫자 인수)

%ACOS는 숫자 인수를 허용하며 인수의 아크 코사인을 래디안으로 리턴합니다. %ACOS 및 %COS는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%AND(스트링 인수 ...)

%AND는 둘 이상의 문자 또는 16진 스트링 인수를 허용하며 인수의 bit-wise 'AND'(논리 and)인 스트링을 리턴합니다. 이 함수는 첫 번째 인수 스트링을 받아 다음 스트링을 AND로 연결하고 위의 결과에 표시된 각 후속 인수를 계속해서 AND로 연결합니다. 인수가 이전 결과보다 더 짧게 표시될 경우 오른쪽이 공백으로 채워집니다. 리턴된 값은 *HEX 유형의 스트링이며 길이는 가장 긴 인수 길이와 동일합니다. 인수 중 가변 길이 인수가 있을 경우 최대 길이가 인수 길이로 사용됩니다.

%ANTILOG(숫자 인수)

%ANTILOG는 숫자 인수를 허용하며 인수의 알고리즘(기본 10)을 리턴합니다. %ANTILOG 및 %LOG는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%ASIN(숫자 인수)

%ASIN는 숫자 인수를 허용하며 인수의 아크 코사인을 래디안으로 리턴합니다. %ASIN 및 %SIN은 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%ATAN(숫자 인수)

%ATAN은 숫자 인수를 허용하며 인수의 아크 탄젠트를 래디안으로 리턴합니다. %ATAN 및 %TAN은 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%ATANH(숫자 인수)

%ATANH는 숫자 인수를 허용하며 인수의 하이퍼볼릭 아크 탄젠트를 래디안으로 리턴합니다. %ATANH 및 %TANH는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%AVG(숫자 인수)

%AVG는 숫자 인수를 허용하며 GRPFLD 매개변수에 정의된 레코드 그룹에 대한 인수의 평균값을 리턴합니다. 이 인수는 필드명이거나 표현식(리터럴이 아님)이어야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 레코드를 선택하지 않을 경우 결과는 널(null) 값이 됩니다. 또는

- 인수가 고정 소수점일 경우 결과는 총 31자리이고 인수도 정수 자릿수가 같은 팩 십진수(*DEC)입니다.
- 인수가 부동 소수점일 경우 결과는 배정밀도 부동 소수점 숫자(*FLT8)입니다.
- 인수가 날짜 기간, 소요 시간 또는 시간소인 기간일 경우 리턴된 값은 팩 십집수(*DEC)로 31자릿수 및 정밀도 0(날짜 기간), 31자릿수 및 정밀도 0(소요 시간) 또는 31자릿수 및 정밀도 6(시간소인 기간)입니다.

%AVG는 그룹화 함수를 사용하는 조회에서 비그룹화 필드에 사용되는 총계 함수입니다.

%CHAR(날짜/시간 인수 날짜/시간 형식)

%CHAR는 날짜/시간 인수 및 날짜/시간 형식을 허용하며 지정된 형식을 사용하여 인수의 문자 표현을 리턴합니다. 날짜/시간 인수는 날짜, 시간 또는 시간소인 필드가 될 수 있습니다. 리턴된 값은 *CHAR 유형이며 현재 작업의 CCSID로 태그 처리됩니다.

날짜/시간 형식은 선택적입니다. 다음 형식 중 하나로 지정해야 합니다.

EUR 유럽 형식

ISO 국제 표준 기구(International Standards Organization) 형식

JIS 일본 공업 규격(Japanese Industrial Standard) 형식

USA 미국 형식

형식을 지정하지 않으면 작업 디폴트 형식이 사용됩니다.

예:

```
OPNQRYF
   FILE(library/file)
     GRPFLD(charfld)
     GRPSLT('charfld = %CHAR(timefld "USA")')
```

%COS(숫자 인수)

%COS는 숫자 인수를 허용하며 인수의 코사인을 리턴합니다. 인수는 래디안 단위로 지정해야 합니다. %COS 및 %ACOS는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%COSH(숫자 인수)

%COSH는 숫자 인수를 허용하며 인수의 하이퍼볼릭 코사인을 리턴합니다. 인수는 래디안 단위로 지정해야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%COT(숫자 인수)

%COT는 숫자 인수를 허용하며 인수의 코탄젠트를 리턴합니다. 인수는 래디안 단위로 지정해야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%COUNT

%COUNT는 인수를 지원하지 않으며 GRPFLD 매개변수에 정의된 레코드 그룹에 포함된 레코드 수를 리턴합니다. 리턴된 값은 총 10진 자릿수가 10이고 분수가 없는 4바이트 2진수(*BIN4)입니다. %COUNT는 그룹화 함수를 사용하는 조회에만 적용되는 총계 함수입니다.

%CURDATE

%CURDATE는 인수를 지원하지 않으며 현재 시간 읽기를 기준으로 하는 현재 날짜를 가져옵니다. 리턴된 값은 *DATE 유형입니다. 형식 및 분리자는 작업 속성에서 파생됩니다.

%CURSERVER

%CURSERVER는 인수를 지원하지 않습니다. 비분산 파일만 지정된 경우 로컬 시스템의 현재 서버명(또는 RDB 이름)을 가져옵니다. 분산 파일이 지정되면 COORDINATOR 노드의 현재 서버명(또는 RDB 이름)을 가져옵니다. 리턴된 값은 최대 길이가 18인 가변 길이 문자(*VCHAR) 유형입니다.

%CURTIME

%CURTIME은 인수를 지원하지 않으며 현재 시간 읽기를 기준으로 하는 현재 시간을 가져옵니다. 리턴된 값은 *TIME 유형입니다. 형식 및 분리자는 작업 속성에서 파생됩니다.

%CURTIMESTP

%CURTIMESTP는 인수를 지원하지 않으며 현재 시간 읽기를 기준으로 하는 현재 시간소인을 가져옵니다. 리턴된 값은 *TIMESTP 유형입니다. 형식 및 분리자는 작업 속성에서 파생됩니다.

%CURTIMEZONE

%CURTIMEZONE은 인수를 지원하지 않으며 현재 시간대를 가져옵니다. 리턴된 값은 팩 십진수(*DEC)로 6자릿수이고 정밀도는 0입니다.

%DATE(날짜/시간 인수)

%DATE는 날짜/시간 인수를 허용하며 날짜를 리턴합니다. 날짜/시간 인수는 범위가 1 - 3,652,059인 리터럴 값 또는 숫자 필드, 날짜 리터럴 또는 날짜 외부 형식을 포함하는 16진 필드 또는 문자, 날짜 또는 시간소인 필드가 될 수있습니다. 리턴된 값은 *DATE 유형입니다.

예:

```
 OPNQRYF
     FILE(library/file)
     QRYSLT(('%DATE(tstampfld) =
        "1989-10-23"'))
```

%DAY(날짜/시간 인수)

%DAY는 날짜/시간 인수를 허용하며 값의 요일 부분을 리턴합니다. 날짜/시간 인수는 날짜 또는 시간소인 필드, 날짜 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 날짜 기간의 경우 8자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 8자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

%DAYS(날짜/시간 인수)

%DAYS는 날짜/시간 인수를 허용하며 날짜의 정수 표현을 리턴합니다. 날짜/시간 인수는 날짜 또는 시간소인 필드, 문자 또는 날짜의 외부 형식을 포함하는 16진 필드 또는 날짜 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

%DIGITS(숫자 인수)

%DIGITS는 숫자 인수를 허용하며 숫자값의 문자 표현을 리턴합니다(부호 또는 소수점은 포함하지 않음). 결과는 현재 작업의 CCSID로 태그 처리됩니다. 예를 들어, %DIGITS(-1.5)는 문자 스트링 15를 리턴합니다. 숫자 인수는 부동 소수점 수가 아니여야 합니다.

%DURDAY(정수 인수)

%DURDAY는 정수 인수를 허용하며 레이블된 요일 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURHOUR(정수 인수)

%DURHOUR는 정수 인수를 허용하며 레이블된 시간 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURMICSEC(정수 인수)

%DURMICSEC는 인수 정수를 허용하며 레이블된 마이크로초 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURMINUTE(정수 인수)

%DURMINUTE은 정수 인수를 허용하며 레이블된 분 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURMONTH(정수 인수)

%DURMONTH는 정수 인수를 허용하며 레이블된 월 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURSEC(정수 인수)

%DURSEC는 정수 인수를 허용하며 레이블된 초 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의*에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

%DURYEAR(정수 인수)

%DURYEAR는 정수 인수를 허용하며 레이블된 년 기간을 리턴합니다. 이 함수의 정수 인수는 숫자 표현식, 필드 또는 리터럴이 될 수 있습니다.

이 내장 함수는 MAPFLD 매개변수의 *맵핑된 필드 정의* 값에서 자체 실행이 허용되며, QRYSLT, GRPSLT 또는 MAPFLD 매개변수에 날짜 또는 시간소인 필드가 있는 산술(더하기 또는 빼기) 표현식의 일부로 허용됩니다.

예:

```
OPNQRYF
   FILE((library/file))
   QRYSLT('startfld > %CURDATE + oneyear *AND
           endfld   < %CURDATE + %DURYEAR(2)')
   MAPFLD((oneyear '%DURYEAR(1)'))
```

%EXP(숫자 인수)

%EXP는 숫자 인수를 허용하며 인수에서 지정한 제곱수를 계산한 자연 로그의 기본이 되는 값을 리턴합니다. %EXP 및 %LN은 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%HASH(표현식 인수)

%HASH는 유효한 표현식을 허용하며 총 10진 자릿수가 10이고 분수가 없는 4바이트 2진수(*BIN4)를 리턴합니다. 리턴된 값은 선택한 레코드의 파티션 번호가 됩니다.

유효한 표현식은 %COUNT, %AVG, %MIN, %MAX, %SUM 및 %STDDEV 등의 총계 함수를 %HASH에 대한 피연산자로 포함할 수 없습니다.

분할 키가 EMPNO 및 LASTNAME으로 이루어진 경우 %HASH 함수를 사용하여 파티션 정의를 판별하십시오. 다음 예는 EMPLOYEE의 각 행에 대해 파티션 번호를 리턴합니다.

예:

```
OPNQRYF
     FILE((CORPDATA/EMPLOYEE))
     FORMAT(FNAME)
     MAPFLD((HASH '%HASH((1/EMPNO) (1/LN))'))
```

%HEX(16진 인수)

%HEX는 인수를 허용하며 인수값에 해당하는 16진수를 리턴합니다. 16진 인수의 유형에는 제한이 없습니다. 리턴된 값은 *CHAR 유형이며 현재 작업의 CCSID로 태그 처리됩니다.

%HOUR(날짜/시간 인수)

%HOUR는 날짜/시간 인수를 허용하며 값의 시간 부분을 리턴합니다. 날짜/시간 인수는 시간 또는 시간소인 필드, 시간 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 시간 기간의 경우 6자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 6자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

예:

```
Example:
  OPNQRYF
     FILE(library/file)
     QRYSLT(('%HOUR(timefld2) = 12'))
```

%LEN(길이 인수)

%LEN은 인수 한 개를 허용하며 값이 그래픽 필드 유형이 아닌 경우 값을 나타내는 데 사용되는 바이트 수를 리턴합니다. 값이 그래픽 필드 유형일 경우 그래픽 문자 수가 리턴됩니다. 길이 인수의 유형에는 제한이 없습니다. 리턴된 값은 *BIN4 유형입니다.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('%LEN(varlenfld) <= 30')
```

Argument Type	Result Length in Bytes
Character	1-32766
Hex	1-32766
DBCS-only	4-32766
DBCS-either	4-32766
DBCS-open	4-32766
DBCS-graphic	1-16383
Variable Character	0-32740
Variable Hex	0-32740
Variable DBCS-only	0-32740
Variable DBCS-either	0-32740
Variable DBCS-open	0-32740
Variable DBCS-graphic	0-16370
Date	4
Time	3
Timestamp	10
Binary *BIN4	2
Binary *BIN8	4
Floating point *FLT4	4
Floating point *FLT8	8
Packed decimal (p,s)	INTEGER(p/2)+1, (1-16)
Zoned decimal (p,s)	p (1-31)

p=precision, s=scale

스트링 참고: %LEN 함수가 자료 공간에 저장되면 값 길이를 리턴합니다.

- 고정 길이 필드의 경우 이 길이는 필드에 있는 실제 자료의 길이가 아니라 선언된 필드 크기와 항상 동일합니다.
- 가변 길이 필드의 경우 이 길이는 후행 공백을 포함하여 필드에 있는 실제 자료의 길이입니다.

예를 들어, FIXED10이 *CHAR(10) 필드이고 VAR10이 *VCHAR(10) 필드라고 가정합시다. 다음 예는 %LEN 함수의 결과를 표시합니다.

%LEN Statement	Field Data	Result
%LEN(fixed10)	'1234567890'	10
%LEN(fixed10)	'12345'	10
%LEN(var10)	'1234567890'	10
%LEN(var10)	'12345'	5
%LEN(var10)	'12345 '	7
%LEN(var10)	''	0

%LN(숫자 인수)

%LN은 숫자 인수를 허용하며 인수의 자연 로그를 리턴합니다. %LN 및 %EXP는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%LOG(숫자 인수)

%LOG는 숫자 인수를 허용하며 인수의 상용 로그(기본 10)를 리턴합니다. %LOG 및 %ANTILOG는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%MAX(숫자 또는 스트링 또는 날짜/시간 인수 ...)

%MAX는 하나 이상의 문자 스트링, DBCS 스트링, 숫자 또는 날짜/시간 인수를 허용하며 리스트에서 가장 큰 값을 리턴합니다. 날짜/시간 인수는 유형이 *DATE, *TIME 또는 *TIMESTP인 인수이거나 날짜 시간 또는 시간소인 기간인 인수입니다. 스트링 인수는 256바이트를 넘지 않아야 합니다.

인수가 하나만 지정된 경우 이 함수는 GRPFLD 매개변수에 정의된 레코드 그룹에 대한 최대 인수 값을 리턴하며 리턴된 값에는 인수와 같은 속성이 있습니다. 레코드를 선택하지 않을 경우 결과는 널(null) 값이 됩니다. 단일 인수가 날짜 기간, 소요 시간 또는 시간소인 기간일 경우 리턴된 값은 팩 십진수(*DEC)로 8자릿수 및 정밀도 0(날짜 기간), 6자릿수 및 정밀도 0(소요 시간) 또는 20자릿수 및 정밀도 6(시간소인 기간)입니다. 단일 인수를 사용할 경우 이 인수는 필드명이거나 표현식(리터럴이 아님)이어야 합니다. 인수가 하나뿐인 %MAX는 그룹화 함수를 사용하는 조회에서 비그룹화 필드에 사용되는 총계 함수입니다.

여러 인수가 지정될 경우 %MAX는 모든 인수의 최대 값을 리턴합니다. 모든 인수는 문자 스트링, DBCS 스트링, 숫자 또는 날짜/시간값 중 하나여야 합니다. 이 함수는 처음 두 인수의 최대값을 계산한 다음 이전 결과의 최대값과 다음 후속 인수를 판별합니다. 최종 결과는 다음 값 변환 규칙에 따라 판별됩니다.

인수에 이전 결과와 다른 속성이 있을 경우 두 값이 동일한 속성으로 변환되고 연산이 계속됩니다. 이 변환은 두 값이 고정 소수점 숫자 값일 경우 팩 십진수를 사용하고, 두 값 중 하나가 부동 소수점일 경우 부동 소수점을 사용합니다. 고정 소수점 숫자 값에 대한 변환은 소수점을 배열하고 값을 0으로 채웁니다. 숫자 유형을 변경하면 고정 소수점 숫자에 총 32자릿수 이상이 필요할 경우 분수 자릿수를 절단하거나 부동 소수점 숫자에 총 16자릿수 이상이 필요할 경우 중요도가 낮은 일부 자릿수를 드롭합니다. 길이가 짧은 필드를 공백으로 채우면 문자값이 변경됩니다.

%MICSEC(날짜/시간 인수)

%MICSEC는 날짜/시간 인수를 허용하며 값의 마이크로초 부분을 리턴합니다. 날짜/시간 인수는 시간소인(필드 또는 리터럴), 시간소인 기간(필드 또는 리터럴), 시간소인의 외부 형식을 포함하는 문자 필드 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다. 숫자 필드 인수는 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의되어야 합니다. 숫자 상수 인수에는 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

%MIN(숫자 또는 스트링 또는 날짜/시간 인수 ...)

%MIN은 하나 이상의 문자 스트링, DBCS 스트링, 숫자 또는 날짜/시간 인수를 허용하며 리스트에서 가장 작은 값을 리턴합니다. 날짜/시간 인수는 유형이 *DATE, *TIME 또는 *TIMESTP인 인수이거나 날짜 시간 또는 시간소인 기간인 인수입니다. 스트링 인수는 256바이트를 넘지 않아야 합니다.

인수가 하나만 지정된 경우 이 함수는 GRPFLD 매개변수에 정의된 레코드 그룹에 대한 최소 인수 값을 리턴하며 리턴된 값에는 인수와 같은 속성이 있습니다. 레코드를 선택하지 않을 경우 결과는 널(null) 값이 됩니다. 단일 인수가 날짜 기간, 소요 시간 또는 시간소인 기간일 경우 리턴된 값은 팩 십진수(*DEC)로 8자릿수 및 정밀도 0(날짜 기간), 6자릿수 및 정밀도 0(소요 시간) 또는 20자릿수 및 정밀도 6(시간소인 기간)입니다. 단일 인수를 사용할 경우 이 인수는 필드명이거나 표현식(리터럴이 아님)이어야 합니다. 인수가 하나뿐인 %MIN은 그룹화 함수를 사용하는 조회에서 비그룹화 필드에 사용되는 총계 함수입니다.

여러 인수가 지정될 경우 %MIN은 모든 인수의 최소 값을 리턴합니다. 모든 인수는 문자 스트링, DBCS 스트링, 숫자 또는 날짜/시간값 중 하나여야 합니다. 이 함수는 처음 두 인수의 최소값을 계산한 다음 이전 결과의 최소값과 다음 후속 인수를 판별합니다. 최종 결과는 아래 설명한 값 변경 규칙에 따라 판별됩니다.

인수에 이전 속성과 다른 속성이 있을 경우 두 값이 동일한 속성으로 변경되고 연산이 계속됩니다. 이 변경은 두 값이 고정 소수점 숫자 값일 경우 팩 십진수를 사용하고, 두 값 중 하나가 부동 소수점 숫자일 경우 부동 소수점 숫자를 사용합니다. 고정 소수점 숫자 값에 대한 변경은 소수점을 배열하고 0으로 채웁니다. 숫자 유형을 변경하면 고정 소수점 숫자에 총 32자릿수 이상이 필요할 경우 분수 자릿수를 절단하거나 부동 소수전 숫자에 총 15자릿수가 필요할 경우 중요도가 낮은 일부 자릿수를 드롭합니다. 길이가 짧은 필드를 공백으로 채우면 문자값이 변경됩니다.

%MINUTE(날짜/시간 인수)

%MINUTE은 날짜/시간 인수를 허용하며 값의 분 부분을 리턴합니다. 날짜/시간 인수는 시간 또는 시간소인 필드, 시간 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 시간 기간의 경우 6자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 6자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

%MONTH(날짜/시간 인수)

%MONTH는 날짜/시간 인수를 허용하며 값의 월 부분을 리턴합니다. 날짜/시간 인수는 날짜 또는 시간소인 필드, 날짜 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 날짜 기간의 경우 8자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 8자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

%NODENAME(정수 인수)

%NODENAME은 FILE 매개변수에 지정된 파일을 식별하는 데 사용하는 정수 인수를 허용합니다.

이 인수는 0보다 커야 하고 파일 매개변수에 지정된 파일 개수보다 작거나 같아야 합니다. %NODENAME 함수는 검색된 레코드의 RDB 이름을 리턴합니다. 리턴된 값은 길이가 18인 *VCHAR 유형입니다.

EMPLOYEE 표의 모든 레코드의 노드 이름을 찾으십시오.

예:

```
OPNQRYF
     FILE((CORPDATA/EMPLOYEE))
     FORMAT(FNAME)
     MAPFLD((NODENAME '%NODENAME(1)'))
```

EMPLOYEE 및 DEPARTMENT 표를 결합하고, 직원 수(EMPNO)를 선택하고, 각 레코드가 결합 시작에 관련된 노드를 판별하십시오.

예:

```
OPNQRYF
  FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
  FORMAT(FNAME)
  JFLD((EMPLOYEE/DEPTNO DEPARTMENT/DEPTNO *EQ))
  MAPFLD((EMPNO 'EMPLOYEE/EMPNO')
         (NODENAME1 '%NODENAME(1)')
         (NODENAME1 '%NODENAME(2)'))
```

EMPLOYEE 및 DEPARTMENT 표를 결합하고, 두 표의 레코드가 같은 노드에 있는 결과의 모든 레코드를 선택하십시오.

예:

```
OPNQRYF
  FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
  FORMAT(FNAME)
  JFLD((1/NODENAME1 2/NODENAME2 *EQ))
  MAPFLD((NODENAME1 '%NODENAME(1)')
         (NODENAME2 '%NODENAME(2)'))
```

%NODENUMBER(정수 인수)

%NODENUMBER는 FILE 매개변수에 지정된 파일을 식별하는 데 사용하는 정수 인수를 허용합니다. 이 인수는 0보다 커야 하고 파일 매개변수에 지정된 파일 개수보다 작거나 같아야 합니다. %NODENUMBER 함수는 총 10진 자릿수가 10이고 분수가 없는 4바이트 2진수(*BIN4)를 리턴합니다. 리턴된 값은 선택한 레코드의 노드 번호가 됩니다.

인수에서 비분산 파일을 식별하면 값 0이 리턴됩니다.

OPNQRYF의 경우 외부 또는 예외 결합이 수행된 2차 파일의 노드 번호가 리턴됩니다.

CORPDATA.EMPLOYEE가 분산 파일일 경우 각 레코드의 노드 번호와 사원 이름이 리턴됩니다.

예:

```
OPNQRYF
     FILE((CORPDATA/EMPLOYEE))
     FORMAT(FNAME)
     MAPFLD((NODENAME '%NODENUMBER(1)')
            (LNAME '1/LASTNAME'))
```

%NONNULL(인수 ...)

%NONNULL은 인수 두 개 이상의 리스트를 허용하며 리스트의 첫 번째 비 널(null)값을 리턴합니다. 인수 리스트의 항목은 필도 또는 모든 유형의 리터럴값이 될 수 있습니다. 리턴된 값의 유형은 목록에서 리스트에서 선택한 항목의 유형입니다.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('%NONNULL(fld1 fld2 0) > 0')
```

위 예는 FLD1 필드나 FLD2 필드에 0보다 큰 비 널(null)값이을 포함하는 파일에서 레코드를 선택합니다. FLD1 및 FLD2가 모두 널(null)이면 세 번째 인수로 전달된 상수 '0' 때문에 이 예에서 지정된 %NONNULL 함수가 '0'을 리턴합니다. DBCS 그래픽인 필드가 있으면 모든 필드가 DBCS 그래픽이어야 합니다.

%NOT(스트링 인수)

%NOT은 문자 또는 16진 스트링 인수를 허용하며 인수의 bit-wise 'NOT'(논리적 not)인 스트링을 리턴합니다. 리턴된 값은 *HEX 유형의 스트링이며 인수와 길이가 같습니다.

%OR(스트링 인수 ...)

%OR은 둘 이상의 문자 스트링 인수를 허용하며 인수의 bit-wise 'OR'(논리적 포괄 or)인 스트링을 리턴합니다. 이 함수는 첫 번째 인수 스트링을 받아 다음 스트링을 OR로 연결하고 위의 결과에 표시된 각 후속 인수를 계속해서 OR로 연결합니다. 인수가 이전 결과보다 더 짧게 표시될 경우 공백으로 채워집니다. 최종 결과는 가장 긴 인수와 길이가 같은 스트링입니다. 인수 중 가변 길이 인수가 있을 경우 최대 길이가 인수 길이로 사용됩니다.

%PARTITION(정수 인수)

%PARTITION은 FILE 매개변수에 지정된 파일을 식별하는 데 사용하는 정수 인수를 허용합니다. 이 인수는 0보다 커야 하고 파일 매개변수에 지정된 파일 개수보다 작거나 같아야 합니다. 파티션 함수는 총 10진 자릿수가 10이고 분수가 없는 4바이트 2진수(*BIN4)를 리턴합니다. 리턴된 값은 레코드의 파티션 번호가 됩니다.

인수에서 비분산 파일을 식별하면 값 0이 리턴됩니다.

EMPLOYEE 표의 모든 행의 PARTITION 번호를 찾으십시오. 이를 통해 자료 분포가 비대칭인지 판별할 수 있습니다.

예:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
     FORMAT(FNAME)
     MAPFLD((PART1 '%PARTITION(1)'))
```

파티션 번호가 100인 모든 레코드의 EMPLOYEE 표에서 사원 번호(EMPNO)를 선택하십시오.

예:

```
OPNQRYF
     FILE((EMPLOYEE))
     QRYSLT('%PARTITION(1) *EQ 100')
```

EMPLOYEE 및 DEPARTMENT 표를 결합하고 두 표의 레코드에 동일한 파티션 번호가 있는 결과의 모든 레코드를 선택하십시오.

예:

```
OPNQRYF
     FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
     FORMAT(FNAME)
     JFLD((1/PART1 2/PART2 *EQ))
     MAPFLD((PART1 '%PARTITION(1)')
     (PART2 '%PARTITION(2)'))
```

%SECOND(날짜/시간 인수)

%SECOND는 날짜 시간 인수를 허용하며 값의 두 번째 부분을 리턴합니다. 날짜/시간 인수는 시간 또는 시간소인 필드, 시간 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 시간 기간의 경우 6자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 6자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

%SIN(숫자 인수)

%SIN은 숫자 인수를 허용하며 인수의 사인(sine)을 리턴합니다. 인수는 래디안 단위로 지정해야 합니다. %SIN 및 %ASIN은 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%SINH(숫자 인수)

%SINH는 숫자 인수를 허용하며 인수의 하이퍼볼릭 사인(sine)을 리턴합니다. 인수는 래디안 단위로 지정해야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%SQRT(숫자 인수)

%SQRT는 숫자 인수를 허용하며 인수의 제곱근을 리턴합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%SST(스트링 인수 시작 위치 표현식 <길이 표현식>)

%SST 및 %SUBSTRING은 문자, 16진수, DBCS 또는 그래픽 스트링, 시작 위치 표현식 및 선택적

길이 표현식을 인수로 허용하며, 스트링 인수와 유형 및 CCSID가 동일하고 길이 표현식에서 지정한 값과 길이가 같은 스트링 인수의 서브스트링을 리턴합니다.

이러한 함수(%SST 및 %SUBSTRING)를 DBCS 자료에 사용하면 단일 바이트 서브스트링화가 수행됩니다. 예상치 못한 결과를 가져오는 SO 및 SI 문자는 손실됩니다. DBCS 서브스트링 조작의 결과는 DBCS 개방 유형입니다.

스트링 인수는 고정 또는 가변 길이 문자, 16진수, DBCS 또는 그래픽 필드 또는 고정 또는 가변 길이 문자, 16진수, DBCS 또는 그래픽 스트링을 산출하는 표현식이 될 수있습니다.

두 번째 및 세 번째 인수에 대한 표현식에서 파생된 값은 유효한 정수여야 합니다. 두 번째 인수에는 첫 번째 인수의 길이 속성(또는 가변 길이 필드의 최대 길이)과 1 사이의 값이 있어야 하고, 세 번째 인수에는 첫 번째 인수의 길이 속성(또는 가변 길이 필드의 최대 길이)과 1 사이의 값이 있어야 합니다.

한 인수가 DBCS 그래픽일 경우 두 번째 및 세 번째 인수도 DBCS 그래픽 문자(바이트가 아니라)로 지정해야 합니다.

두 번째나 세 번째 인수에 대한 표현식이 지정된 경우 이 표현식 양쪽에 괄호가 있어야 합니다.

표현식이 가변 길이 결과를 산출하면 이러한 표현식 범위의 유효성을 보장할 수 없고 입/출력 처리 동안 오류가 발생할 수 있습니다.

세 번째 인수에 허용된 최대값(길이)은 DBCS 그래픽(이 경우 16383)을 제외하고는 32766입니다. 그러나 세 번째 피연산자가 표현식으로 표시되면 결과는 가변 길이로 표시됩니다. 따라서 표현식 값은 DBCS 그래픽을 제외하고(이 경우 16370을 초과할 수 없음) 32740을 초과할 수 없습니다.

사용자가 세 번째 인수를 생략할 수 있습니다. 세 번째 인수를 지정하지 않고 첫 번째 인수가 다음인 경우:

- 고정 길이, 세 번째 인수의 디폴트값이 LENGTH(argument_1) - value_for_argument_2 + 1
- 가변 길이, 세 번째 인수의 디폴트값이 0의 최대값이고 LENGTH(argument_1) - value_for_argument_2 + 1
- 길이가 argument_2의 값보다 작은 가변 길이, 세 번째 인수의 세 번째 값이 0이고 결과는 빈 스트링.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('field1 =
     %SST(field2 (numfld1+3)
     (numfld1+numfld2))')
```

%STDDEV(숫자 인수)

%STRIP은 문자-, DBCS- 또는 그래픽 스트링 인수, 선택적 스트립 문자 및 선택적 스트립 함수를 인수로 허용하며, 스트립 함수에서 지정된 대로 스트링 인수에서 제거된 스트립 문자가 있는 결과 스트링을 리턴합니다.

스트링 인수는 리터럴, 고정 또는 가변 길이 문자, 16진수, DBCS 또는 그래픽 필드 또는 고정 또는 가변 길이 문자, 16진수, DBCS 또는 그래픽 스트링을 산출하는 표현식이 될 수있습니다.

스트립 문자는 자료 유형이 소스 스트링과 호환 가능하고 어포스트로피로 묶여 있는 단일 문자여야 합니다. 디폴트는 문자 자료, DBCS 개방 및 DBCS 선택 필드용 단일 SBCS 공간 , DBCS 전용 자료의 단일 DBCS 공간 및 그래픽 자료용 단일 그래픽 공간입니다.

스트립 함수는 다음 세 가지 함수 중 하나일 수 있습니다.

***LEAD**
선행 스트립 문자 제거

***TRAIL**
추적 스트립 문자 제거

***BOTH**
선행 및 추적 스트립 문자 모두 제거

디폴트 스트립 함수는 *BOTH입니다.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('%STRIP(fld '.' *TRAIL) = 'Mr')
```

%SUBSTRING(스트링 필드 이름 시작 위치 길이)

%SUBSTRING은 %SST와 같은 연산을 수행합니다. 자세한 정보는 %SST 설명을 참조하십시오.

%SUM(숫자 인수)

%SUM은 숫자 인수를 허용하며 GRPFLD 매개변수에 정의된 레코드 그룹의 인수에 대한 모든 값의 합을 리턴하고 괄호로 묶여 있어야 합니다. 인수는 필드명이나 표현식(리터럴이 아님)이어야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 레코드를 선택하지 않을 경우 결과는 널(null) 값이 됩니다. 또는

- 인수가 부동 소수점 숫자일 경우 리턴된 결과는 배정밀도 부동 소수점 숫자(*FLT8)입니다.
- 인수가 정밀도가 0인 2진 숫자일 경우 리턴된 값은 *BIN4입니다.
- 인수가 정밀도가 0이 아닌 2진 숫자이거나 고정 소수점 숫자일 경우 리턴된 값은 총 31자리이고 인수만큼의 분수 자릿수가 있는 팩 십진수(*DEC)입니다.
- 인수 유형이 날짜 기간, 소요 시간 또는 시간소인 기간일 경우 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%SUM은 그룹화 함수를 사용하는 조회에서 비그룹화 필드에 사용되는 총계 함수입니다.

%TAN(숫자 인수)

%TAN은 숫자 인수를 허용하며 인수의 탄젠트를 리턴합니다. 인수는 래디안 단위로 지정해야 합니다. %TAN 및 %ATAN은 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%TANH(숫자 인수)

%TAN은 숫자 인수를 허용하며 인수의 하이퍼볼릭 탄젠트를 리턴합니다. 인수는 래디안 단위로 지정해야 합니다. %TANH 및 %ATANH는 역연산입니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 이러한 유형의 인수는 필드값이나 리터럴값으로 지정될 수 있습니다. 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다.

%TIME(날짜/시간 인수)

%TIME은 날짜/시간 인수를 허용하며 시간을 리턴합니다. 날짜/시간 인수는 날짜 또는 시간소인 필드, 시간 외부 형식을 포함하는 문자 또는 16진 필드 또는 시간리터럴이 될 수 있습니다. 리터된 값은 *TIME 유형입니다.

%TIMESTP(날짜/시간 인수)

%TIMESTP는 날짜/시간 인수 한 개 또는 두 개를 허용하며 시간소인을 리턴합니다.

- 날짜/시간 인수가 하나만 지정된 경우 시간소인(필드 또는 리터럴) 또는 시간소인 외부 형식을 포함하는 문자 또는 16진 필드입니다.
- 인수가 둘 다 지정된 경우
  1. 첫 번째 날짜/시간 인수는 날짜(필드 또는 리터럴) 또는 날짜 외부 형식을 포함하는 문자 또는 16진 필드여야 합니다.
  2. 두 번째 날짜/시간 인수는 시간(필드 또는 리터럴) 또는 시간 외부 형식을 포함하는 문자 또는 16진 필드여야 합니다.

리턴된 값은 *TIMESTP 유형입니다.

%USER

%USER는 인수를 지원하지 않으며 조회를 실행 중인 작업의 사용자 프로필명을 리턴합니다. 리턴된 값은 최대 길이가 18인 가변 길이 문자(*VCHAR) 유형입니다.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('field = %USER')
```

%VAR(숫자 인수)

%VAR은 숫자 인수를 허용하며 GRPFLD 매개변수에 의해 정의된 레코드 그룹에 대한 인수의 변수를 리턴합니다. 이 인수는 필드명이거나 표현식(리터럴이 아님)이어야 합니다.

날짜 기간, 소요 시간 및 시간소인 기간 인수 유형은 숫자값으로 취급됩니다. 레코드를 선택하지 않을 경우 결과는 널(null) 값이 됩니다. 또는 리턴된 값은 배정밀도 부동 소수점 숫자(*FLT8)입니다. %VAR은 그룹화 함수를 사용하는 조회에서 비그룹화 필드에 사용되는 총계 함수입니다.

%XLATE(스트링 인수 검증 표)

%XLATE은 문자 스트링 인수 및 표 오브젝트 이름(*TBL)을 허용하며 표의 내용을 사용하여 변환된 첫 번째 인수값인 스트링을 리턴합니다. 리턴된 값은 첫 번째 인수와 길이 및 CCSID가 같은 스트링입니다.

두 번째 인수는 단순 또는 규정 표 오브젝트명이어야 합니다. 라이브러리명이 지정되지 않은 경우 *LIBL을 사용하여 표를 찾습니다.

%XOR(스트링 인수...)

%XOR은 둘 이상의 문자 스트링 인수를 허용하며 인수의 bit-wise 'XOR'(논리적 배타 or)인 스트링을 리턴합니다. 이 함수는 첫 번째 인수 스트링을 받아 다음 스트링을 XOR로 연결하고 위의 결과에 표시된 각 후속 인수를 계속해서 XOR로 연결합니다. 인수가 이전 결과보다 더 길게 표시될 경우 이전 결과가 XOR 연산 전에 공백으로 채워집니다. 인수 중 가변 길이 인수가 있을 경우 최대 길이가 인수 길이로 사용됩니다. 최종 결과는 *HEX 유형의 스트링이며 길이는 가장 긴 인수 길이와 동일합니다.

%YEAR

%YEAR는 날짜/시간 인수를 허용하며 값의 연도 부분을 리턴합니다. 날짜/시간 인수는 날짜 또는 시간소인 필드, 날짜 기간 또는 시간소인 기간(필드 또는 리터럴) 또는 숫자 필드 또는 리터럴이 될 수 있습니다. 리턴된 값은 *BIN4 유형입니다.

숫자 필드 인수는 날짜 기간의 경우 8자릿수 및 정밀도 0인 팩 십진수(*DEC)이고, 시간소인 기간의 경우 20자릿수 및 정밀도 6인 팩 십진수(*DEC)로 정의됩니다. 숫자 상수 인수에는 8자릿수 다음에 소수점이 있거나 14자릿수 다음에 소수점과 6자릿수가 있어야 합니다.

제한된 내장 함수:

다음은 몇 가지 제한된 내장 함수를 나열한 것입니다.

다음 내장 함수는 QRYSLT 또는 GRPSLT 매개변수에 지정된 '같음' 또는 '같지 않음' 관계 연산자의 두 번째 피연산자로만 지원됩니다.

%NULL

%NULL은 인수를 허용하지 않으며 레코드 내의 필드가 널값을 포함하는지 여부에 따라 레코드를 선택하거나 생략하는 데 사용됩니다.

예:

```
OPNQRYF
     FILE(library/file)
     QRYSLT('charfld = %NULL')
```

이 조회는 'charfld'에 널값을 포함하는 모든 레코드를 선택합니다.

다음 세 가지 내장 함수는 QRYSLT 또는 GRPSLT 매개변수에 지정된 '같음' 관계 연산자의 두 번째 피연산자로만 지원됩니다.

%RANGE(낮은 값 높은 값)

%RANGE는 필드나 식의 값에 대한 하한 및 상한 경계를 식별하는 데 사용됩니다. %RANGE는 연산자가 같음인 관계의 오른쪽으로 지정되어야 합니다. 낮은 값 및 높은 값 인수는 관계의 왼쪽으로 지정된 필드나 표현식의 유형과 일치하도록 필드명, 문자 스트링 또는 숫자 리터럴이어야 합니다. 예를 들어, 숫자 필드 NBRFLD의 값 범위가 10에서 20인 레코드만 선택할 경우 다음과 같이 지정합니다.

```
'nbrfld = %RANGE(10 20)'
```

낮은 값 인수가 높은 값 인수보다 크면 관계는 논리 값 '거짓'을 표시합니다.

%VALUES(허용된 값...)

%VALUES는 필드나 표현식에 허용된 값 리스트를 식별하는 데 사용됩니다. %VALUES는 연산자가 같음인 관계의 오른쪽으로 지정되어야 합니다. 허용된 값 인수는 관계의 왼쪽으로 지정된 필드나 표현식의 유형과 일치하도록 문자 스트링 또는 숫자 리터럴이어야 합니다. 예를 들어, CHARFLD 필드의 두 번째 문자 값이 'A', 'E', 'I', 'O' 또는 'U'인 레코드만 선택할 경우 다음과 같이 지정합니다.

```
'%SST(charfld 2 1) = %VALUES("A" "E" "I" "O" "U")'
```

%WLDCRD("패턴 스트링" "와일드 문자")

%WLDCRD는 관계의 왼쪽으로 지정되어야 하는 문자 또는 16진 필드 또는 스트링 표현식(단일 문자 스트링 리터럴로 구성된 표현식은 제외)의 와일드 카드 스캔을 수행하는 패턴을 지정하는 데 사용됩니다. %WLDCRD는 연산자가 같음인 관계의 오른쪽으로 지정되어야 합니다. 패턴 스트링 인수는 관계의 왼쪽과 일치하도록 문자 스트링, DBCS 또는 그래픽 리터럴이어야 합니다. 와일드 문자 인수는 패턴 스트링에 사용되는 '와일드 카드' 문자를 지정하는 선택적 매개변수입니다.

문자 자료 전용으로 지정된 경우(DBCS 자료 없음) 와일드 문자 인수는 정확히 두 문자의 문자 스트링 리터럴이어야 합니다. 첫 번째 문자는 검색 스트링에서 임의의 단일 문자와 일치하는 값입니다. 두 번째 문자는 문자 0개 이상의 서브스트링과 일치하는 값입니다. 두 문자는 서로 달라야 하지만 두 문자 중 하나를 패턴 스트링으로 표시할 필요는 없습니다. 와일드 문자 인수가 생략된 경우 디폴트는 단일 문자와 일치하기 위한 밑줄('_')과 문자 0개 이상의 서브스트링과 일치하기 위한 별표('*')에 해당됩니다.

와일드 문자 인수가 DBCS 자료 전용으로 지정된 경우(문자 자료 없음) 이 인수는 정확히 2바이트 문자 두 개로 된 2바이트 문자 스트링 리터럴이어야 합니다. 첫 번째 2바이트 문자는 검색 스트링의 2바이트 문자 하나와 일치하는 값입니다. 두번째 2바이트 문자는 문자 0개 이상의 서브스트링과 일치하는 값입니다. 2바이트 문자 두 개는 서로 달라야 하지만 두 문자 중 하나를 패턴 스트링으로 표시할 필요는 없습니다. 와일드 문자 인수가 생략된 경우 디폴트는 2바이트 문자 한 개와 일치하기 위한 DBCS 밑줄과 2바이트 문자 0개 이상의 서브스트링과 일치하기 위한 DBCS 별표에 해당됩니다.

와일드 문자 인수가 문자 및 DBCS 자료 모두에 지정된 경우 이전 규칙 이외에도 인수는 먼저 단일 바이트 문자 스트링 리터럴(단일 바이트 문자 두 개)을 포함한 다음 2바이트 문자 스트링(2바이트 문자 두 개)을 포함해야 합니다.

이 경우 첫 번째 문자는 문자 스트링의 단일 바이트 문자와 일치하고, 두 번째 문자는 임의 개수의 단일 바이트나 2바이트 문자의 서브스트링과 일치합니다. 첫 번째 2바이트 문자는 문자 스트링의 2바이트 문자와 일치합니다. 두 번째 2바이트 문자는 임의 개수의 단일 바이트나 2바이트 문자의 서브스트링과 일치합니다.

다음 예에서는 문자 필드 CHARFLD에 'T'가 있고 그 다음은 문자 두 개와 'E'가 필드의 아무 곳에나 표시되는 레코드만 선택합니다.

```
'charfld = %WLDCRD(''*T__E*'')'
```

주: 패턴 스트링의 시작과 끝에 있는 별표가 필드에서 첫 번째 및 마지막 위치를 제외한 다른 곳에 'T'와 'E'가 표시되도록 허용할 필요는 없습니다.

문자 필드 CHARFLD가 스트링 'ABC'로 시작하고, 그 다음에는 하나 이상의 다른 문자가 있고, 그 다음에는 스트링 'XYZ'가 있는(반드시 필드 끝일 필요는 없음) 레코드만 선택하려면 다음을 지정하십시오.

```
'charfld = %WLDCRD(''ABC_*XYZ*'')'
```

CHARFLD 필드의 두 번째 문자가 별표('*')이고, 마지막 문자가 밑줄('_')이며, 그 사이의 어딘가에 문자 'M'이 표시되는 레코드만 선택하려면 다음과 같이 지정하십시오.

```
'charfld = %WLDCRD(''#*.M._'' ''#.'')'
```

프로그램에서의 기본 데이터베이스 파일 조작

이 주제에서는 읽기, 갱신, 쓰기 및 삭제 조작을 포함한 기본 데이터베이스 파일 조작에 대해 설명합니다.

파일에서의 위치 설정

작업에 의하여 파일이 열린 후, 시스템은 그 작업을 위해 파일에서의 위치를 유지보수합니다. 파일 위치는 파일을 처리하는 데 사용됩니다.

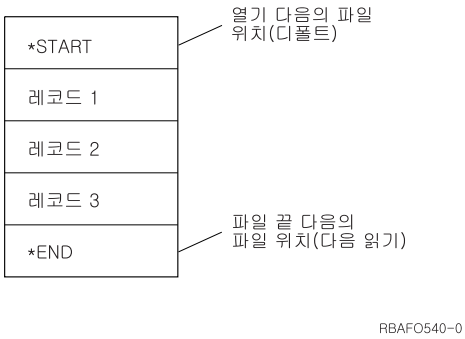
예를 들어 시스템은 파일 위치를 사용하여 프로그램으로 리턴할 레코드를 결정할 수 있습니다. 그런 후 시스템은 방금 읽은 레코드에 파일 위치를 설정하여 다음의 순차 레코드를 요구하는 또 다른 읽기 조작이 정확한 레코드를 리턴시키도록 합니다. 시스템은 각 작업에 대해 모든 파일 위치를 유지합니다. 그 외에도 각 작업은 동일한 파일 내에서 여러 개의 위치를 가질 수 있습니다.

파일 위치는 우선 RCLRSC(데이터베이스 파일 대체) 명령의 POSITION 매개변수에 지정된 위치에 설정됩니다. OVRDBF 명령을 사용하지 않거나 POSITION 매개변수에 디폴트 값을 사용할 경우 파일 위치는 멤버의 액세스 경로에서 첫 번째 레코드 바로 앞에 설정됩니다.

해당 고급 언어 프로그램의 파일 위치 지정 조작(예를 들어, RPG/400 언어의 SETLL이나 COBOL/400 언어의 START)을 사용하여 프로그램이 현재 파일 위치를 변경할 수 있습니다. 또한 프로그램은 POSDBF(데이터베이스 파일 위치 지정) 명령을 사용하여 파일 위치를 변경할 수 있습니다.

주: OVRDBF 명령에 의한 파일 위치 지정은 다음 번에 파일이 열릴 때까지는 발생하지 않습니다. 파일은 제어 언어(CL) 프로그램 내에서 한 번만 열릴 수 있으므로 이 명령은 RCVF(파일 수신) 명령을 통해 읽히는 내용에 영향을 미치는 단일 CL 프로그램 내에서는 사용할 수 없습니다.

파일 끝에서 마지막 읽기 이후, 프로그램이 파일을 앞에서 또는 뒤에서 읽을 것인지에 따라 파일 멤버는 *START 또는 *END 파일 위치로 지정됩니다. 다음 다이어그램은 *START 및 *END 파일 위치를 나타내고 있습니다.*START가 맨 위에, 그 다음에 세 개의 레코드 그리고 맨 아래에 *END가 나옵니다.



RBAFO540-0

읽기 조작, 자료 끝 강제(force-end-of-data) 조작, 고급 언어 위치 지정 조작 또는 파일 위치를 변경시키는 특정 CL 명령만이 파일 위치를 변경할 수 있습니다. 추가, 갱신 및 삭제 조작은 파일 위치를 변경하지 않습니다. 읽기 조작 이후, 파일은 새로운 레코드에 위치됩니다. 그러면 이 레코드가 프로그램에 리턴됩니다. 읽기 조작이 완료된 후, 파일은 프로그램으로 방금 리턴된 레코드에 위치됩니다. 멤버가 입력을 위해 열려는 경우 자료 끝 강제(force-end-of-data) 조작은 파일의 마지막 레코드(*END)뒤에 파일을 위치시키고 프로그램에 파일 끝(end-of-file) 메세지를 보냅니다.

순서 읽기 조작의 경우 액세스 경로에서 다음 레코드 또는 이전 레코드를 찾아내는 데 현재 파일 위치가 사용됩니다. 키에 의한 읽기(read by key) 또는 상대 레코드 번호에 의한 읽기 조작에는 파일 위치가 사용되지 않습니다. 열릴 당시 POSITION(*NONE)이 지정되면 파일의 시작 위치가 설정되지 않습니다. 이 경우 순차적으로 읽으려면 프로그램에 파일 위치를 설정해야 합니다.

OVRDBF 명령에서 파일에 대해 파일 끝 지연(end-of-file delay)이 지정된 경우 프로그램이 최종 레코드를 읽을 때 파일이 *START나 *END에 놓이지 않습니다. 파일은 읽힌 최종 레코드에 위치되어 있습니다. 파일 끝 지연 처리가 지정된 파일은 자료 끝 강제(force-end-of-data:EFOD)가 발생하거나 작업 종료 제어가 발생하는 경우에만 파일이 *START나 *END에 위치됩니다.

POSDBF 명령을 사용하여 OPNDBF(데이터베이스 파일 열기) 명령이나 OPNQRYF(조회 파일 열기) 명령으로 열린파일에 현재 위치를 설정하거나 변경할 수도 있습니다.

관련 개념

제어 언어(CL)

222 페이지의『파일 끝에 도달했을 때의 추가 레코드 대기』

파일 끝 지연(end-of-file delay)은 파일 끝 상태가 발생한 후에도 데이터베이스 파일(논리 파일 또는 실제 파일)에서 계속 순차적으로 읽어나가는 방법입니다.

관련 참조

데이터베이스 파일 대체(OVRDBF) 명령

데이터베이스 레코드 읽기

iSeries 시스템은 데이터베이스 레코드를 읽는 여러 가지 방법을 제공하고 있습니다.

주: 일부 고급 언어에서는 시스템에서 사용 가능한 모든 읽기 조작을 지원하지 않습니다. 데이터베이스 레코드 읽기에 대한 자세한 정보는 고급 언어 주제 콜렉션을 참조하십시오.

도달순 액세스 경로를 사용하여 데이터베이스 레코드 읽기:

이러한 읽기 조작은 파일이 도달순 액세스 경로로 정의되었거나 OPNDBF(데이터베이스 파일 열기) 명령 또는 OPNQRYF(조회 파일 열기) 명령에 키순 액세스 경로 무시 옵션을 지정하여 키순 액세스 경로로 파일이 정의된 경우에 허용됩니다.

주: 시스템은 고급 언어를 사용하여 지정하는 명령문을 기준으로 읽기 조작을 수행합니다. 사용 중인 고급 언어가 다음의 읽기 조작을 모두 허용하지 않을 수 있습니다. 허용되는 조작을 확인하려면 고급 언어 주제 콜렉션을 참조하십시오.

관련 개념

116 페이지의『키순 액세스 경로 무시』

키 필드를 파일에 정의하면 시스템은 자동으로 키순 액세스 경로를 사용합니다. 그러나 때로는 ACCPTH 매개변수를 사용하여 키순 액세스 경로를 무시하면 성능이 향상될 수 있습니다.

다음 조작 읽기:

이 조작은 파일을 도달순 액세스 경로에서 삭제되지 않은 다음 레코드에 배치하고 그 레코드를 가져옵니다.

파일의 현재 위치와 활동 중인 이전 레코드 사이의 삭제 레코드는 건너뛰게 됩니다. (RPG/400 언어의 READ 문과 COBOL/400 언어의 READ NEXT문이 이러한 조작의 예입니다.)

이전 조작 읽기:

이 조작은 파일을 도달순 액세스 경로에서 이전 활성 레코드에 배치하고 그 레코드를 가져옵니다.

파일의 현재 위치와 활동 중인 이전 레코드 사이의 삭제 레코드는 건너뛰게 됩니다. (RPG/400 언어의 READP 문과 COBOL/400 언어의 READ PRIOR문이 이러한 조작의 예입니다.)

첫 번째 조작 읽기:

이 조작은 파일을 도달순 액세스 경로에서 첫 번째 활성 레코드에 배치하고 그 레코드를 가져옵니다.

마지막 조작 읽기:

이 조작은 파일을 도달순 액세스 경로에서 마지막 활성 레코드에 배치하고 그 레코드를 가져옵니다.

동일한 조작 읽기:

이 조작은 파일의 현재 위치에 의해 식별되는 레코드를 가져옵니다. 파일 위치는 변경되지 않습니다.

상대 레코드 번호별 조작 읽기:

이 조작은 파일을 상대 레코드 번호에 의해 식별된 도달순 액세스 경로의 레코드에 배치하고 그 레코드를 가져옵니다.

상대 레코드 번호는 활동 레코드를 식별해야 하며 멤버에서 활동 중인 최대 상대 레코드 번호보다 작거나 같아야 합니다. 또한 이 조작은 현재 파일 위치에서 지정된 수의 레코드를 더하거나 빼서 식별된 도달순 액세스 경로의 레코드를 읽습니다. (RPG/400 언어의 CHAIN문과 COBOL/400 언어의 READ문이 이 조작의 예입니다.)파일이 상대 레코드 처리에 의해 처리되는 경우 삭제 레코드를 재사용하기 위해서는 파일을 작성 또는 변경하는 데 있어서 특별히 주의해야 합니다.

관련 개념

삭제 레코드 재사용

데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

키순 액세스 경로를 사용하여 데이터베이스 레코드 읽기:

이러한 읽기 조작은 키순 액세스 경로로 데이터베이스 레코드를 가져오는 데 사용할 수 있습니다.

키순 액세스 경로 사용 시, 읽기 조작은 삭제 레코드가 차지하는 기억장치에 위치할 수 없습니다.

주: 시스템은 고급 언어를 사용하여 지정하는 명령문을 기준으로 읽기 조작을 수행합니다. 사용 중인 고급 언어가 다음의 읽기 조작을 모두 허용하지 않을 수 있습니다. 해당 언어에서 허용하는 조작을 확인하려면 고급 언어 주제 콜렉션을 참조하십시오.

다음 조작 읽기:

이 조작은 키순 액세스 경로에서 다음 레코드를 가져옵니다.

레코드 형식명이 지정되면 이 조작은 키순 액세스 경로에서 레코드 형식과 대응되는 다음 레코드를 가져옵니다. 파일의 현재 위치가 이전 레코드를 찾는 데 사용됩니다. (RPG/400 언어의 READ문과 COBOL/400 언어의 READ NEXT문이 이러한 조작의 예입니다.)

이전 조작 읽기:

이 조작은 키순 액세스 경로에서 이전 레코드를 가져옵니다.

레코드 형식명이 지정되면 이 조작은 키순 액세스 경로에서 레코드 형식과 대응되는 이전 레코드를 가져옵니다. 파일의 현재 위치가 이전 레코드를 찾는 데 사용됩니다. (RPG/400 언어의 READP문과 COBOL/400 언어의 READ PRIOR문이 이러한 조작의 예입니다.)

첫 번째 조작 읽기:

이 조작은 키순 액세스 경로에서 첫 번째 레코드를 가져옵니다.

레코드 형식명이 지정되면 이 조작은 지정된 형식명을 가진 액세스 경로의 첫 번째 레코드를 가져옵니다.

마지막 조작 읽기:

이 조작은 키순 액세스 경로에서 마지막 레코드를 가져옵니다.

레코드 형식명이 지정되면 이 조작은 지정된 형식명을 가진 액세스 경로의 마지막 레코드를 가져옵니다.

동일한 조작 읽기:

이 조작은 현재 파일 위치에 의해 식별되는 레코드를 가져옵니다. 파일에서의 위치는 변경되지 않습니다.

키 조작에 의해 읽기:

이 조작은 키 값으로 식별되는 레코드를 가져옵니다.

같은 값, 같거나 작은 값, 같은 키 값 이전 읽기, 같은 키 값 다음 읽기, 다음 또는 이전의 키 조작이 지정될 수 있습니다. 형식명이 지정되면 시스템은 지정된 키 값과 레코드 형식명을 가진 레코드를 탐색합니다. 형식명이 지정되지 않으면 지정된 키 값으로 키순 액세스 경로 전체를 탐색합니다. 파일에 대한 키 정의에 복수 키 필드가 포함되어 있으면 부분 키가 지정될 수 있습니다. (사용될 키 필드의 수나 키 길이 중 하나를 지정하면 됩니다.) 그러면 총칭 키 탐색을 수행할 수 있습니다. 프로그램이 키 필드 수를 지정하지 않을 경우 시스템은 디폴트 키 필드 수를 가정합니다. 이 디폴트는 레코드 형식명이 프로그램에 의하여 전달되는지에 따라 달라집니다. 레코드 형식명이 전달되면 디폴트 키 필드 수는 해당 형식에 대해 정의된 키 필드의 총 수가 됩니다. 레코드 형식명이 전달되지 않으면 디폴트 키 필드 수는 액세스 경로의 모든 레코드 형식에서 공통되는 키 필드의 최대 수가 됩니다. 프로그램은 시스템에서 가정한 키 필드 수와 대응되도록 키 자료를 충분히 제공해야 합니다. (RPG/400 언어의 CHAIN문과 COBOL/400 언어의 READ문이 이 조작의 예입니다.)

상대 레코드 번호별 조작 읽기:

키순 액세스 경로의 경우 상대 레코드 번호를 사용할 수 있습니다. 열린 멤버가 키순 액세스 경로를 가지고 있는 경우라도 이것은 입력순의 상대 레코드 번호입니다.

멤버에 복수 레코드 형식이 있는 경우 레코드 형식명이 지정되어야 합니다. 이 경우 연관된 실제 파일 멤버에서 지정된 레코드 형식에 대응되는 레코드를 요구하게 됩니다. 열린 멤버에 선택/생략문이 들어 있고 상대 레코드 번호에 의하여 식별된 레코드가 키순 액세스 경로로부터 생략되었으면, 프로그램에 오류 메세지가 전달되면서 조작이 허용되지 않습니다. 조작이 완료된 후, 파일은 상대 레코드 번호에 의하여 식별된 실제 레코드에 있는 키순 액세스 경로의 키 값에 위치됩니다. 또한 이 조작은 현재 파일 위치에서 일정한 수의 레코드를 더하거나 빼서 식별된 키순 액세스 경로의 레코드를 가져옵니다. (RPG/400 언어의 CHAIN문과 COBOL/400 언어의 READ문이 이 조작의 예입니다.)

논리 파일이 추가 조작과 액세스 경로를 공유하는 경우의 읽기:

논리 파일의 자료 서술 스펙(DDS)에 선입선출(FIFO), 후입선출(LIFO) 또는 선변경선출(FCFO) 키워드가 지정되지 않은 경우 이 논리 파일은 작성 중인 논리 파일보다 더 많은 키가 있는 액세스 경로를 암시적으로 공유할 수 있습니다.

이렇게 기존 액세스 경로에서 키의 부분 세트를 공유하면 이러한 부분 공유 키순 액세스 경로를 사용하는 데이터베이스 읽기 조작에 대해 감지된 문제점이 야기될 수 있습니다. 다음은 예상되는 문제점입니다.

- 읽어야 할 레코드가 프로그램에 리턴되지 않음.
- 레코드가 프로그램에 여러 번 리턴됨.

실제로 발생되는 경우를 살펴보면, 사용자의 프로그램이나 현재 활동 중인 다른 프로그램이 부분 공유 키순 액세스 경로 내의 키인 실제 파일 필드를 갱신하고 있으나 이것이 사용자의 프로그램에 의해 사용 중인 논리 파일에 대한 실제 키(actual key)가 아닌 경우가 있습니다. (갱신 중인 필드는 사용자 프로그램이 사용 중인 논리 파일에 알려진 키의 수를 초과합니다.) 사용자 프로그램이나 다른 프로그램에 의해 논리 파일에 대한 실제 키 필드(actual key field)를 갱신하면 항상 위와 같은 결과가 발생됩니다. 부분 공유 키순 액세스 경로와의 차이점은 논리 파일에 알려진 키의 수를 초과하는 실제 파일 필드를 갱신하면 동일한 결과를 야기시킨다는 점입니다.

부분 공유 키순 액세스 경로에 의해 발생된 이 결과를 받아들일 수 없는 경우 FIFO, LIFO 또는 FCFO 키워드를 논리 파일에 대한 DDS에 추가한 후 논리 파일을 다시 작성할 수 있습니다.

파일 끝에 도달했을 때의 추가 레코드 대기:

파일 끝 지연(end-of-file delay)은 파일 끝 상태가 발생한 후에도 데이터베이스 파일(논리 파일 또는 실제 파일)에서 계속 순차적으로 읽어나가는 방법입니다.

순차적으로 읽고 있는 파일에 파일 끝 상태가 발생하고(예: 다음/이전 레코드), 사용자가 파일 끝 지연 시간을 지정한 경우(OVRDBF(데이터베이스 파일 대체) 명령의 EOFDLY 매개변수), 시스템은 지정된 시간 동안 대기합니다.

지연 시간이 종료되면 새로운 레코드가 파일에 추가되었는지 확인하기 위해 또 다른 읽기 조작이 수행됩니다. 레코드가 추가되었으면 파일 끝 상태가 다시 발생할 때까지 레코드가 정상적으로 처리됩니다. 레코드가 파일에 추가되지 않았으면 시스템은 지정된 시간 동안 대기합니다. 선택/생략 스펙이 있는 논리 파일에 대해 파일 끝 지연을 사용하여 파일이 열릴 때 키순 액세스 경로가 사용되지 못하도록 할 경우 특별히 고려해야 할 사항이 있습니다. 이 경우 파일 끝에 도달하면 시스템은 근거가 되는 실제 파일에 추가된 레코드 중 논리 파일의 선택/생략 스펙에 부합되는 레코드만 검색합니다.

키순 액세스 경로를 가진 파일에 대해 파일 끝 지연을 사용하여 파일이 열릴 때 키순 액세스 경로가 사용되도록 할 경우에도 특별히 고려해야 할 사항이 있습니다. 이 경우 파일 끝에 도달하면 시스템은 파일에 추가된 레코드 또는 파일에서 갱신된 레코드 중 키순 액세스 경로를 사용한 읽기 조작의 스펙에 부합되는 레코드만 검색합니다.

오름차순으로 된 숫자 키 필드가 있는 키순 파일에 파일 끝 지연이 사용된 경우를 예로 들어 봅시다. 어플리케이션 프로그램은 키순 액세스 경로를 사용하여 파일에 있는 레코드를 읽습니다. 어플리케이션 프로그램은 다음 읽기 조작을 수행하여 키 값이 99인 레코드를 가져옵니다. 어플리케이션 프로그램이 또 하나의 다음 읽기 조작을 수행하고 파일에는 더 이상 레코드가 없는 상태가 됩니다. 따라서 시스템은 지정된 파일 끝 지연 시간이

지난 후에 다시 파일을 읽으려 합니다. 레코드가 파일에 추가되거나 갱신되고 레코드가 99보다 작은 키 값을 가지면, 시스템은 이 레코드를 검색하지 않습니다. 레코드가 파일에 추가되거나 갱신되고 레코드가 99 이상의 키 값을 가지면 시스템이 레코드를 검색하게 됩니다.

파일 끝 지연 시간이 10초 이상인 경우 작업은 대기 시간 동안 주 기억장치로부터 제거될 수 있습니다. 적절한 작업이 주 기억장치에서 이동되기를 원치 않을 경우 작업이 사용 중인 CLASS에 대해 CRTCLS(클래스 작성) 명령에 PURGE(*NO)를 지정하십시오.

어떤 작업에 파일 끝 지연이 진행중인지 알려주기 위해 '활동 중인 작업에 대한 작업(WRKACTJOB)' 화면의 상태 필드에 레코드를 대기하고 있는 작업의 파일 끝 대기(end-of-file wait) 또는 파일 끝 활동 레벨(end-of-file activity level)이 나옵니다.

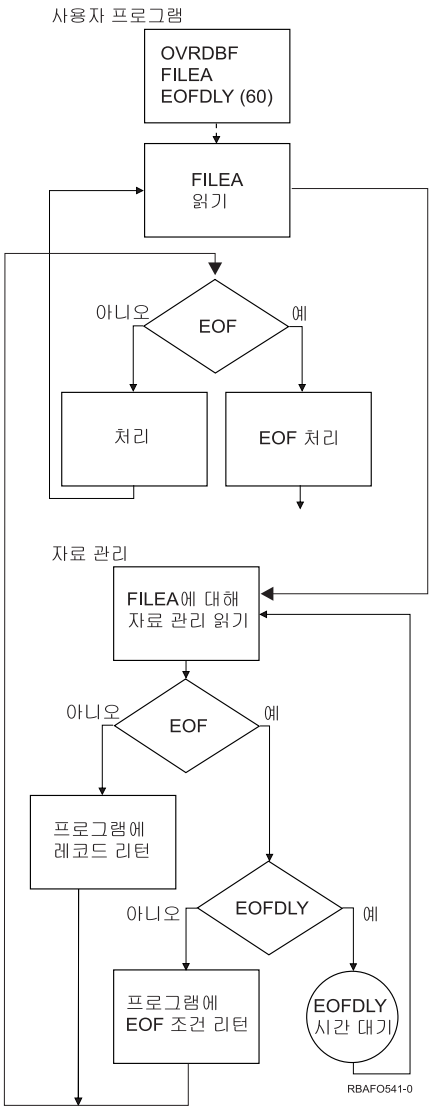
작업이 파일 끝 지연 및 확약 제어를 사용하는 경우 작업은 보다 긴 시간 동안 레코드를 잠가 둘 수 있습니다. 이로 인해 다른 작업이 이 레코드의 액세스할 때 잠겨질 수 있는 가능성이 높아집니다. 이러한 이유 때문에 동일한 작업에서 파일 끝 지연 및 확약 제어를 사용할 때에는 주의해야 합니다.

파일이 공유되는 경우 파일 끝 지연을 지정하는 OVRDBF 명령은 파일이 *맨 처음* 열리기 전에 수행되어야 합니다. 공유 파일이 열린 이후 지정되는 대체 속성은 무시되기 때문입니다.

OVRDBF 명령에 지정된 파일 끝 지연 때문에 더 많은 레코드를 대기하고 있는 작업을 종료하는 방법은 여러 가지가 있습니다.

- 어플리케이션 프로그램에서 마지막 레코드로 인식될 파일 끝 지연을 가진 레코드를 파일에 기록하십시오. 그러면 어플리케이션 프로그램이 자료 끝 강제(FEOD) 조작을 지정할 수 있습니다. FEOD 조작으로 프로그램은 정상적인 파일 끝 처리를 완료할 수 있습니다.
- ENDJOB(작업 종료) 명령에 OPTION(*CNTRLD)를 지정하고 DELAY 매개변수 값 시간을 EOFDLY 시간보다 길게 지정하여 작업을 제어 종료하십시오. 지정된 DEALY 매개변수 시간은 EOFDLY 시간이 종료되는 시간과 파일에 입력된 새로운 레코드가 처리되는 시간 및 어플리케이션에서 요구되는 파일 끝 처리에 소모되는 시간을 모두 수용할 수 있어야 합니다. 새로운 레코드가 처리되면 시스템은 파일 끝 신호를 보내고 정상적인 파일 끝 상태가 발생하게 됩니다.
- ENDJOB 명령에 OPTION(*IMMED)을 지정하십시오. 파일 끝 처리는 수행되지 않습니다.
- 작업이 대화식인 경우 최종 요구를 종료하려면 시스템 요구 키를 누르십시오.

다음은 파일 끝 지연 조작의 한 예를 표시합니다.



EOFDLY 매개변수의 실제 처리는 표시된 것보다 더 복잡합니다. ENDJOB 명령의 OPTION(*CNTRLD)이 긴 지연 시간과 함께 사용되면 실제 파일 끝(EOF)을 강제로 적용할 수 있기 때문입니다.

새로운 레코드가 파일에 추가될 때마다 작업은 활동 상태가 되지 않습니다. 지정된 파일 끝 지연 시간이 지나야 작업은 활동 상태가 됩니다. 작업이 활동 상태가 되면 시스템은 새로운 레코드가 있는지 검사합니다. 새로운 레코드가 추가되었으면 어플리케이션 프로그램이 제어를 갖게 되어 새로운 레코드를 모두 처리한 후 다시 기다립니다. 이로 인해 처리 시, 작업은 일괄처리 작업의 특성을 지니게 됩니다. 예를 들어 작업은 요구를 일괄적으로 처리합니다. 일괄처리가 완료되면 작업은 비활동 상태가 됩니다. 지연 시간이 짧으면 작업을 시작하고 새로운 레코드가 있는지 검사하는 데 필요한 내부 처리 때문에 시스템의 부수적인 작업 처리가 지나치게 많아질 수 있습니다. 일반적으로 파일 끝 지연 도중 대기중인 작업에 대해 사용되는 시스템의 부수적인 작업 처리는 많지 않습니다.

주: 작업이 비활동(대기 중) 상태이면 작업은 긴 대기 상태에 놓이는데, 이는 작업이 활동 레벨로부터 해제되었다는 것을 의미합니다. 긴 대기 상태가 충족된 이후, 시스템은 작업을 활동 레벨에서 다시 스케줄합니다.

관련 개념

작업 관리

잠긴 레코드 해제:

레코드를 갱신하거나 삭제하는 경우 또는 파일 내의 다른 레코드를 읽을 경우 시스템은 잠긴 레코드를 자동으로 해제합니다. 그러나 이러한 조작을 수행하지 않고도 잠긴 레코드를 해제해야 할 경우가 있습니다.

일부 고급 언어에서는 잠긴 레코드를 해제하는 조작을 지원합니다. 레코드 잠금 해제에 대한 자세한 정보는 고급 언어 주제 콜렉션을 참조하십시오.

주: 작업이 확약 제어하에서 수행되고 있는 경우에는 잠금 규칙이 다릅니다.

관련 개념

확약 제어

데이터베이스 레코드 갱신

갱신 조작으로 논리 파일이나 실제 파일의 기존 데이터베이스 레코드를 변경할 수 있습니다.

RPG/400 언어의 UPDAT문과 COBOL/400 언어의 REWRITE문은 이 조작의 예입니다. 데이터베이스를 갱신하기 전에 레코드를 먼저 읽고 잠가야 합니다. 잠금은 219 페이지의 『도달순 액세스 경로를 사용하여 데이터베이스 레코드 읽기』 또는 220 페이지의 『키순 액세스 경로를 사용하여 데이터베이스 레코드 읽기』에서 나열된 읽기 조작에 갱신 옵션을 지정하면 가능합니다.

갱신 옵션이 지정된 여러 가지의 읽기 조작을 발행하는 경우 각 읽기 조작은 새로운 레코드를 위치해 그 레코드를 잠그기 전에 이전 레코드의 잠금을 해제하게 됩니다. 갱신 조작을 수행할 때 시스템은 사용자가 현재 잠긴 레코드를 갱신 중인 것으로 가정합니다. 따라서 사용자는 갱신 조작에서 갱신될 레코드를 식별할 필요가 없습니다. 갱신 조작이 수행된 후, 시스템이 잠금을 해제합니다.

주: 작업이 확약 제어하에서 수행되고 있는 경우에는 잠금 규칙이 다릅니다.

갱신 조작이 즉시 유지보수가 지정된 액세스 경로의 키 필드를 변경하면 그 액세스 경로는 고급 언어가 허용할 경우 갱신됩니다. (일부 고급 언어에서는 갱신 조작에서의 키 필드 변경을 허용하지 않습니다.)

갱신을 위해 이미 잠겨진 레코드에 대해 읽기 조작을 요구하는 경우와 작업이 확약 제어 레벨 *ALL/ 또는 *CS(커서 안정성)하에서 수행 중인 경우 파일 작성 또는 대체 명령에 지정된 WAITRCD 매개변수에 의해 지정된 시간이 초과되거나 그 레코드가 해제될 때까지 대기해야 합니다. 잠금이 해제되지 않은 채 WAITRCD 시간이 초과한 경우에는 프로그램에 예외가 리턴되면서 파일, 멤버, 상대 레코드 번호 및 잠금 상태인 작업을 명시하는 메세지가 작업에 보내집니다. 레코드를 읽는 중인 작업이 확약 제어 레벨 *ALL 또는 *CS하에서 수행 중이 아니면 작업은 갱신을 위해 잠겨진 레코드를 읽을 수 있습니다.

갱신 중인 파일에 그 파일과 연관된 갱신 트리거가 있으면 레코드를 갱신하기 전후에 트리거 프로그램이 호출됩니다.

갱신 중인 파일이 참조 제한조건과 연관되어 있으면, 갱신 조작에 영향을 미칩니다.

관련 개념

확약 제어

306 페이지의 『데이터베이스의 자동 이벤트 트리거』

*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』

이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

데이터베이스 레코드 추가

쓰기 조작을 사용하여 새 레코드를 실제 데이터베이스 파일 멤버에 추가할 수 있습니다.

RPG/400 언어의 WRITE문과 COBOL/400 언어의 WRITE문은 이 조작의 예입니다. 실제 파일 멤버 또는 실제 파일 멤버를 기초로 한 논리 파일 멤버에 새로운 레코드를 추가할 수 있습니다. 복수 형식 논리 파일을 사용할 때는 어떤 실제 파일 멤버에 레코드를 추가할 것인지를 시스템에 알리기 위해 레코드 형식명을 제공해야 합니다.

새로운 레코드는 주로 실제 파일 멤버 끝에 추가됩니다. 그 다음 사용 가능한 상대 레코드 번호(삭제 레코드를 포함한)가 새로운 레코드에 할당됩니다. 일부 고급 언어에서는 삭제 레코드 위치에 새로운 레코드를 써넣을 수 있게 합니다(예를 들어, 파일 구성이 RELATIVE로 정의되었을 때 COBOL/400의 WRITE문).삭제된 레코드 위치에 레코드 쓰기에 대한 자세한 내용은 고급 언어 주제 콜렉션을 참조하십시오.

레코드가 추가될 실제 파일이 삭제 레코드를 재사용하면 시스템은 삭제 레코드가 있던 슬롯(slot)으로 레코드를 삽입하려고 시도합니다. 삭제 레코드를 재사용하기 위해서는 파일을 작성하거나 변경하기 전에 파일이 삭제 레코드 공간의 재사용 대상인지의 여부를 결정하기 위해 제한사항 및 도움 정보를 검토해야 합니다.

키순 액세스 경로를 가진 파일 멤버에 새로운 레코드를 추가하면 새로운 레코드는 키순 액세스 경로에서 레코드 키에 의해 정의된 위치에 즉시 나옵니다. 선택/생략 값을 포함하고 있는 논리 멤버에 레코드를 추가할 경우 생략 값은 멤버의 액세스 경로에 새로운 레코드가 나오지 않도록 할 수 있습니다.

레코드를 추가하려는 파일에 그 파일과 연관된 삽입 트리거가 있으면, 레코드를 삽입하기 전후에 트리거 프로그램이 호출됩니다.

추가중인 파일이 참조 제한조건과 연관되어 있으면, 레코드 삽입에 영향을 미칩니다.

CRTPF(실제 파일 작성) 및 CRTSRCPF(소스 실제 파일 작성) 명령의 SIZE 매개변수는 실제 파일 멤버에 추가될 수 있는 레코드 수를 결정합니다.

관련 개념

115 페이지의 『삭제 레코드 재사용』

데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

306 페이지의 『데이터베이스의 자동 이벤트 트리거』

*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』

이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

복수 형식 파일에 추가할 레코드 형식 식별:

어플리케이션이 데이터베이스에 추가될 레코드에 대해 레코드 형식명 대신 파일명을 사용하는 경우 또한 사용된 파일이 둘 이상의 레코드 형식을 가진 논리 파일인 경우 데이터베이스의 레코드 위치를 결정하기 위해 형식 선택 프로그램을 작성해야 합니다.

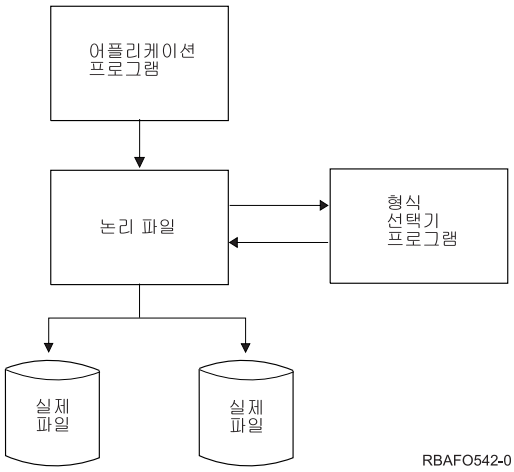
형식 선택 프로그램은 제어 언어(CL) 프로그램이나 고급 언어 프로그램이 될 수 있습니다. 다음의 경우가 모두 참인 경우에는 반드시 형식 선택 프로그램을 사용해야 합니다.

- 논리 파일이 결합 파일이 아니고, 뷰(view) 논리 파일도 아닙니다.
- 논리 파일이 여러 개의 실제 파일에 기초를 둡니다.
- 프로그램이 추가 조작에서 레코드 형식명 대신 파일명을 사용합니다.

이러한 상황에서 형식 선택 프로그램을 작성하지 않을 경우 데이터베이스에 레코드를 추가하려고 하면 프로그램은 오류를 발생시키면서 종료됩니다.

주: 형식 선택 프로그램은 파일이 여러 개의 멤버를 갖고 있는 경우에는 하나의 멤버를 선택하는 데 사용할 수 없습니다. 이 프로그램은 하나의 레코드 형식만을 선택할 수 있습니다.

어플리케이션 프로그램이 데이터베이스 파일에 레코드를 추가하려고 할 때 시스템은 형식 선택 프로그램을 호출합니다. 형식 선택 프로그램은 레코드를 검사하고 사용될 레코드 형식을 지정합니다. 그런 후 시스템은 지정된 레코드 형식명을 사용하여 데이터베이스 파일에 레코드를 추가합니다.



다음의 예에서는 RPG/400 언어로 작성된 형식 선택 프로그램문을 보여주고 있습니다.

```
CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments+++...
++++*
      C           *ENTRY    PLIST
C                     PARM           RECORD 80
C* The length of field RECORD must equal the length of
C* the longest record expected.
C                     PARM           FORMAT 10
C                     MOVELRECORD    BYTE    1
C           BYTE      IFEQ 'A'
C                     MOVEL'HDR'     FORMAT
      C                     ELSE
C                     MOVEL'DTL'     FORMAT
C                     END
```

형식 선택자는 첫 번째 매개변수의 레코드를 수신하므로 이 필드는 형식 선택자에서 예상하는 가장 긴 레코드 길이로 선언되어야 합니다. 형식 선택자는 레코드의 어느 부분이나 액세스하여 레코드 형식 이름을 알아 낼 수 있습니다. 이 예에서 형식 선택자는 문자 A에 대한 레코드에서 첫 번째 문자를 확인합니다. 문자가 A가 아니면, 형식 선택자는 레코드 형식명 DTL을 두 번째 매개변수(FORMAT)로 이동합니다.

형식 선택자는 10자 필드인 두 번째 매개변수를 사용하여 시스템으로 레코드 형식을 전달합니다. 시스템이 레코드 형식명을 알게 될 때 레코드를 데이터베이스에 추가하게 됩니다.

다음과 같은 경우에는 형식 선택자가 필요하지 않습니다.

- 갱신 조작만 수행하는 경우. 갱신 조작의 경우 프로그램이 이미 레코드를 검색했으며, 시스템에서 레코드를 가져온 실제 파일을 인식하는 경우.
- 사용자 어플리케이션이 추가 또는 삭제 조작에 파일명 대신 레코드 형식명을 지정하는 경우.
- 어플리케이션 프로그램에서 사용되는 모든 레코드가 한 개의 실제 파일에 들어 있는 경우.

형식 선택 프로그램을 작성하려면 프로그램을 작성한 언어에 대한 프로그램 작성 명령을 사용하십시오. 작성 명령에는 USRPRF(*OWNER)를 지정할 수 없습니다. 형식 선택자는 소유자의 사용자 프로파일이 아닌 사용자의 사용자 프로파일 하에서 실행되어야 합니다.

또한 보안 및 무결성을 위한 목적과 성능에 심각한 영향을 줄 수 있는 이유로 인해 형식 선택 프로그램내에 호출이나 입출력 조작이 있어서는 안 됩니다.

형식 선택자의 이름은 CRTLF(논리 파일 작성), CHGLF(논리 파일 변경) 또는 OVRDBF(데이터베이스 파일 대체) 명령의 FMTSLR 매개변수에 지정됩니다. 파일이 작성될 때 형식 선택 프로그램은 존재할 필요가 없으나 어플리케이션 프로그램이 수행될 때에는 존재해야 합니다.

관련 개념

49 페이지의 『복수 형식 파일에 레코드 추가 방법 제어』
복수 형식 논리 파일에 레코드를 추가하려면 레코드를 기록할 기준 실제 파일의 멤버를 확인해야 합니다.

관련 태스크

45 페이지의 『논리 파일 작성』
이 주제는 자료 서술 스펙(DDS)을 사용하여 논리 파일을 작성하는 방법을 표시합니다.

자료 끝 강제 조작 사용:

자료 끝 강제(FECO) 조작을 사용하면 프로그램에 의한 모든 파일 변경을 보조 기억장치에 강제로 기록할 수 있습니다. 또한 자료 끝 강제(FEOD) 조작은 파일이 입력 조작을 위해 열려 있는 경우 파일의 시작이나 파일의 끝에 위치하도록 합니다.

일반적으로, 변경 내용을 보조 기억장치에 강제로 기록시키는 시점은 시스템이 결정합니다. 그러나 FEOD 조작을 사용하여 모든 변경이 보조 기억장치에 강제로 기록되도록 할 수 있습니다.

*START는 현재 열려 있는 데이터베이스 파일 멤버 내의 시작 위치를 멤버의 첫 번째 레코드 바로 앞으로 설정합니다. (첫 번째 순차 읽기 조작은 현재 멤버의 첫 번째 레코드를 읽습니다.) MBR(*ALL) 처리가 OVRDBF(데이터베이스 파일 대체) 명령에 유효한 경우 이전 읽기 조작이 이전 멤버에서 마지막 레코드를 읽습니다. 이전 읽기 조작이 수행되고 이전 멤버가 존재하지 않으면 파일 끝 메세지(CPF5001)가 전달됩니다. *END는 현재 열려 있는 데이터베이스 파일 멤버 내의 위치를 멤버의 최종 레코드 바로 뒤로 설정합니다. (이전 읽기 조작이 현재 멤버의 최종 레코드를 읽습니다.) MBR(*ALL) 처리가 OVRDBF 명령에 유효한 경우 다음 읽기 조작이 다음 멤버에서 첫 번째 레코드를 읽습니다. 다음 읽기 조작이 수행되고 다음 멤버가 존재하지 않으면 파일 끝 메세지(CPF5001)가 발생합니다.

파일에 삭제 트리거가 있으면 자료 끝 강제(force-end-of-data) 조작이 허용되지 않습니다. 파일이 참조 상위 관계의 일부이면 FEOD 조작이 허용되지 않습니다.

FEOD 조작에 대한 자세한 내용은 해당 고급 언어 주제 콜렉션을 참조하십시오. (일부 고급 언어는 FEOD 조작을 지원하지 않습니다.)

관련 개념

306 페이지의 『데이터베이스의 자동 이벤트 트리거』

*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』

이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

데이터베이스 레코드 삭제

삭제 조작을 사용하면 기존의 데이터베이스 레코드를 삭제할 수 있습니다.

RPG/400 언어의 DELET문과 COBOL/400 언어의 DELETE문은 이 조작의 예입니다. 데이터베이스 레코드를 삭제하려면 레코드를 먼저 읽고 잠가야 합니다. 219 페이지의 『도달순 액세스 경로를 사용하여 데이터베이스 레코드 읽기』 또는 220 페이지의 『키순 액세스 경로를 사용하여 데이터베이스 레코드 읽기』에 나오는 읽기 조작상에 갱신 옵션을 지정하여 레코드를 잠글 수 있습니다. 삭제하기 위해 레코드를 잠그는 데 관한 규칙과 삭제할 레코드를 식별하는 데 관한 규칙은 갱신 조작에 관한 규칙과 동일합니다.

주: 일부 고급 언어에서는 먼저 레코드를 읽을 필요가 *없습니다*. 이러한 언어를 사용하면 삭제 명령문에서 삭제할 레코드를 지정할 수 있습니다. 예를 들어 RPG/400 언어에서는 레코드를 먼저 읽지 않고도 삭제할 수 있습니다.

데이터베이스 레코드가 삭제될 때에는 실제 레코드가 삭제된 것으로 표시됩니다. 이는 삭제 조작이 논리 파일을 통해 수행될 경우에도 마찬가지입니다. 삭제된 레코드는 읽을 수 *없습니다*. 레코드가 포함되어 있는 모든 키순 액세스 경로로부터 레코드가 제거됩니다. 삭제 레코드의 상대 레코드 번호는 그대로 남아 있습니다. 실제 파일 멤버 내의 다른 모든 상대 레코드 번호는 변경되지 않습니다.

삭제 레코드에 의해 사용된 공간은 파일 내에 그대로 있으나 다음과 같은 경우가 될 때까지는 다시 사용되지 않습니다.

- RGZPFM(실제 파일 멤버 재구성) 명령은 파일 멤버에 이러한 공간을 압축하고 지우기 위해 실행됩니다.
- 프로그램이 상대 레코드 번호로 레코드를 파일에 써넣고 사용된 상대 레코드 번호가 삭제 레코드의 상대 레코드 번호와 같을 경우.

주: 파일이 지정된 삭제 레코드 공간 재사용 속성을 가진다면 시스템은 자동적으로 삭제 레코드 공간을 재사용하려고 시도합니다.

시스템은 삭제 레코드의 자료를 검색하는 것을 허용하지 않습니다. 그러나 삭제 레코드와 연관된 위치(상대 레코드 번호)에 새로운 레코드를 써넣을 수는 있습니다. 이 쓰기 조작은 삭제 레코드를 새로운 레코드로 대체합니다. 파일의 특정 위치(상대 레코드 번호)에 레코드를 작성하는 방법에 대한 자세한 내용은 고급 언어 주제 콜렉션을 참조하십시오.

레코드를 삭제 레코드의 상대 레코드 번호에 써넣기 위해서는 그 상대 레코드 번호가 실제 파일 멤버 내에 존재해야 합니다. 고급 언어의 삭제 조작으로 파일의 레코드를 삭제할 수 있습니다. INZPFM(실제 파일 멤버 초기화) 명령으로도 파일 내의 레코드를 삭제할 수 있습니다. INZPFM 명령은 삭제 레코드에 실제 파일 멤버 전체를 초기화할 수 있습니다.

삭제중인 파일에 그 파일과 연관된 삭제 트리거가 있으면 레코드를 삭제하기 전후에 트리거 프로그램이 호출됩니다.

파일이 참조 제한조건 관계의 일부일 경우 레코드 삭제에 영향을 미칩니다.

관련 개념

239 페이지의『실제 파일 재구성』

이 주제에서는 i5/OS 오퍼레이팅 시스템에서 실제 파일을 재구성하는 방법과 재구성 조작을 일시중단하거나 취소하는 방법 및 선택할 수 있는 재구성 조작 유형에 대해 설명합니다. 실제 파일 재구성에 대한 고려사항도 포함됩니다.

115 페이지의『삭제 레코드 재사용』

데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

238 페이지의 『실제 파일 멤버에서 자료 초기화』
프로그램에서 상대 레코드 처리를 사용하려면 데이터베이스 파일에는 프로그램에서 사용되는 최상위 상대 레코드 번호와 같은 수의 레코드 위치가 있어야 합니다. 상대 레코드 번호 처리를 사용하는 프로그램은 때때로 레코드들이 초기화될 것을 요구합니다.

306 페이지의 『데이터베이스의 자동 이벤트 트리거』
| *트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의
| 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

291 페이지의 『참조 제한조건으로 자료 무결성 확인』
이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

데이터베이스 파일 닫기

프로그램이 데이터베이스 파일 멤버의 처리를 완료하고 나면, 파일을 닫아야 합니다. 데이터베이스 파일을 닫는다는 것은 프로그램과 파일 사이의 연결을 해제하는 것입니다.

닫기 조작은 모든 레코드 잠금과 모든 파일 멤버 잠금을 해제하며, 열린 자료 경로(ODP)를 통한 모든 변경 내용을 보조 기억장치에 강제로 기록한 후, ODP를 파기합니다(공유 파일이 닫히고 ODP는 계속 열려 있으면 기능이 다름).

프로그램에서 데이터베이스 파일을 닫으려면 다음 방법 중 하나를 사용하십시오.

- 고급 언어 닫기문

  대부분의 고급 언어에서는 데이터베이스 파일을 닫을 수 있습니다. 고급 언어 프로그램에서 데이터베이스 파일을 닫는 방법에 대한 자세한 내용은 해당 고급 언어 주제 콜렉션을 참조하십시오.

- CLOF(파일 닫기) 명령

  CLOF 명령을 사용하여 OPNDBF(데이터베이스 파일 열기) 또는 OPNQRYF(조회 파일 열기) 명령을 사용하여 열었던 데이터베이스 파일을 닫을 수 있습니다.

- RCLRSC(자원 재생) 명령

  RCLRSC 명령은 모든 잠금(단, 확약 제어하에서 변경은 되었으나 아직 확약되지 않은 레코드에 대한 잠금은 예외)을 해제하며, 모든 변경 내용을 보조 기억장치에 강제로 기록한 후, 해당 파일에 대한 ODP를 파기합니다. RCLRSC 명령을 사용하여 호출한 프로그램이 호출된 프로그램의 파일을 닫게 만들 수 있습니다. (예를 들어, 호출된 프로그램이 자체의 파일을 닫지 않은 채 호출한 프로그램으로 리턴하는 경우 호출한 프로그램이 호출된 프로그램의 파일을 닫을 수 있습니다.) 그러나 프로그램에서 파일을 닫는 일반적인 방법은 고급 언어 닫기 조작을 사용하거나 CLOF 명령을 사용하는 것입니다. 통합 언어 환경에서 자원 재생에 관한 자세한 내용은 ILE 개념 서적을 참조하십시오.

작업이 정상적으로 종료되고(예를 들어, 사용자가 종료함) 그 작업에 연관된 모든 파일이 닫혀지지 않았으면, 시스템은 자동적으로 그 작업과 연관되어 열려 있는 나머지 파일을 닫고, 모든 변경 내용을 보조 기억장치에

강제로 기록한 후, 해당 파일에 대한 모든 레코드 잠금을 해제합니다. 작업이 이상 종료하면 시스템이 그 작업과 연관되는 모든 파일들을 닫고, 보조 기억장치로 모든 변경을 강행합니다.

한 프로세스가 또다른 프로세스에서 보유한 파일을 잠그려고 시도할 때 닫기 데이터베이스 파일 나감 프로그램이 호출됩니다. 나감은 잠금을 보유 중인 프로세스에서 호출됩니다.

관련 개념

124 페이지의 『동일한 작업 또는 활성 그룹에서 데이터베이스 파일 공유』
기본적으로 데이터베이스 관리 시스템에서는 하나의 파일이 동시에 많은 사용자에 의해 읽히고 변경되도록 되어 있습니다. 그러나 SHARE 매개변수를 통해 동일한 작업이나 활성 그룹의 데이터베이스 파일을 공유할 수 있습니다.

제어 언어(CL)

ILE 개념 PDF

관련 참조

CLOF(파일 닫기) 명령

RCLRSC(자원 재생) 명령

프로그램에서 데이터베이스 파일 오류 모니터링

데이터베이스 어플리케이션이 데이터베이스 파일에서 조치를 수행할 경우 오류 방지를 위한 적절한 조치를 취할 수 있도록 프로그램이 감지한 파일 오류에 대한 메세지를 모니터링해야 합니다.

각 고급 언어(HLL)는 이러한 메세지 모니터링을 위한 고유한 프로시듀어를 제공하므로 오류 메세지 모니터링을 구현하려면 사용 중인 HLL에 대한 문서를 참조하십시오.

데이터베이스 파일 처리 중에 오류 상태가 감지될 경우 다음 이벤트 중 하나 이상이 발생합니다.

- 파일을 처리하는 프로그램의 프로그램 메세지 대기행렬에 메세지를 송신할 수 있습니다.
- 시스템 오퍼레이터 메세지 대기행렬에 조회 메세지를 송신할 수 있습니다.
- 그 외에도, 파일 오류 및 진단 정보는 파일 피드백 영역에 리턴 코드 및 상태 정보로서 프로그램에 표시될 수 있습니다.

예를 들어, COBOL 언어가 프로그램에 정의된 경우 파일 상태 필드에 리턴 코드가 설정됩니다.

오류 메세지의 시스템 처리

이 메세지들을 모니터링하지 않으면 시스템이 오류를 처리합니다.

시스템은 프로그램 내에 적절한 오류 리턴 코드도 설정합니다. 오류에 따라 시스템은 작업을 종료할 수도 있고, 오퍼레이터에게 추가의 조치를 요구하는 메세지를 보낼 수도 있습니다.

파일 위치지정에 대한 오류 메세지 효과

프로그램에서 데이터베이스 파일 멤버를 처리하는 동안 메세지가 프로그램으로 전송될 경우 파일에서의 위치는 상실되지 않습니다.

위치는 메세지가 전달되기 이전에 위치하였던 레코드에 그대로 있게 됩니다.

- 파일 끝 상태에 도달하여 메세지가 프로그램에 보내진 후에는 파일이 *START 또는 *END에 위치됨.
- 읽기 조작에서의 변환 맵핑 메세지 이후, 파일은 메세지를 발생시킨 자료가 들어 있는 레코드에 위치됨.

모니터링할 메세지 판별

프로그래밍 언어로 오류 메세지를 모니터링할 수 있는 경우 모니터링할 메세지를 선택할 수 있습니다. 다음은 모니터링할 수 있는 오류 메세지의 간단한 샘플입니다.

표 47. 모니터링 가능한 오류 메세지 샘플

메세지 ID	설명
CPF5001	파일 끝에 도달함
CPF5006	레코드가 없음
CPF5007	레코드가 삭제됨
CPF5018	최대 파일 크기에 도달함
CPF5025	*START 또는 *END를 벗어나서 읽음
CPF5026	중복 키
CPF5027	다른 작업에서 레코드를 사용 중
CPF5028	레코드 키가 변경됨
CPF5029	자료 맵핑 오류
CPF502B	트리거 프로그램에서의 오류
CPF502D	참조 제한조건 위반
CPF5030	멤버가 일부 손상됨
CPF5031	레코드 잠금의 최대수를 초과함
CPF5032	레코드가 작업에 이미 할당되었음
CPF5033	선택/생략 오류
CPF5034	다른 멤버의 액세스 경로에 중복된 키
CPF503A	참조 제한조건 위반
CPF5040	생략 레코드가 검색되지 않음
CPF5072	멤버 내의 결합 값이 변경됨
CPF5079	확약 제어 자원 한계를 초과함
CPF5084	확약되지 않은 키의 중복 키
CPF5085	다른 액세스 경로의 확약되지 않은 키의 중복 키
CPF5090	고유의 액세스 경로 문제점이 멤버에의 액세스를 방지함
CPF5097	키 맵핑 오류

DSPMSGD(메세지 설명 표시) 명령을 사용하여 이러한 메세지에 대한 자세한 설명을 표시할 수 있습니다.

관련 개념

CL 프로그램 또는 프로시듀어의 메세지 모니터

제어 언어(CL)

관련 참조

DSPMSGD(메세지 설명 표시) 명령

데이터베이스 파일 관리

이 주제에서는 파일, 해당 설명 및 속성을 관리하여 시스템에서 데이터베이스 파일 제어를 유지보수하는 방법에 대해 설명합니다. 자료에 대한 제어를 유지보수하여 유실되지 않게 보호하고, 제한사항으로 자료 무결성을 보증하고, 자료 변경 시 트리거 이벤트를 설정하는 방법에 관한 정보가 포함됩니다.

데이터베이스 파일 관리를 위한 기본 조작

이 주제에서는 데이터베이스 파일 관리를 위한 두 가지 기본 조작(파일 복사 및 파일 이동)에 대해 설명합니다.

파일 복사

iSeries Navigator의 표 복사 조작을 사용하거나 CPYF(파일 복사) 명령을 사용하여 파일을 복사할 수 있습니다.

iSeries Navigator를 사용하여 파일(표) 복사

다른 스키마로 표를 복사하면 동일한 표의 인스턴스가 두 개 작성됩니다. 다른 스키마로 표를 복사하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 스키마를 확장하십시오.
4. 표 컨테이너를 클릭하십시오.
5. 표를 마우스 오른쪽 버튼으로 클릭하고 **복사**를 클릭하십시오.
6. 표를 복사하려는 스키마를 마우스 오른쪽 버튼으로 클릭하고 **붙여넣기**를 클릭하십시오.

분산 자료 관리(DDM)는 표를 실제로 이동하거나 복사하기 위해 iSeries Navigator에서 사용됩니다. 소스 시스템의 오퍼레이팅 시스템이 버전 4 릴리스 4 이상을 실행 중이고 목표 시스템의 오퍼레이팅 시스템이 버전 4 릴리스 2 이상을 실행 중인 경우 TCP/IP에서 DDM을 사용하여 조작이 수행됩니다. 그렇지 않으면 SNA에서 DDM을 사용하여 조작이 수행됩니다. SNA에서 DDM을 사용하는 이동이나 복사 조작의 경우 iSeries Access가 시스템을 인식하는 이름이 DDM에서 사용하는 APPC 또는 APPN 장치 설명에 지정된 리모트 위치명과 동일해야 합니다. TCP/IP에서 DDM을 사용하는 이동이나 복사 조작의 경우 TCP 통신이 시스템 간에 작동되어야 합니다. TCP/IP의 경우 iSeries Access에 알려진 시스템 간에 TCP/IP가 작동 가능해야 함을 기억하십시오.

CPYF(파일 복사) 명령으로 파일 복사

CPYF(파일 복사) 명령은 데이터베이스나 외부 장치 파일 모두나 일부를 데이터베이스나 외부 장치 파일에 복사합니다.

관련 개념

제어 언어(CL)

관련 참조

파일 복사(CPYF) 명령

파일 이동

iSeries Navigator에서 표 이동 조작을 사용하여 또는 MOVOBJ(오브젝트 이동) 명령을 사용하여 한 라이브러리에서 다른 라이브러리로 파일을 이동할 수 있습니다.

iSeries Navigator를 사용하여 파일(표) 이동

파일(표)을 다른 라이브러리로 이동하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 스키마를 확장하십시오.
4. **표** 컨테이너를 클릭하십시오.
5. 표를 마우스 오른쪽 버튼으로 클릭하고 **잘라내기**를 클릭하십시오.
6. 표를 이동하려는 스키마를 마우스 오른쪽 버튼으로 클릭하고 **붙여넣기**를 클릭하거나, 표를 끌어서 동일한 시스템에나 다른 시스템에 있는 또다른 라이브러리에 놓을 수 있습니다.

주: 표를 새로운 위치에 이동시키는 것이 소스 시스템으로부터 표를 항상 제거하지는 않습니다. 예를 들어, 소스 표에 대한 읽기 권한은 있으나 삭제 권한은 없는 경우 표가 목표 시스템으로 이동합니다. 그러나 소스 시스템에서 표가 삭제되지 않으므로 표 인스턴스가 두 개가 됩니다.

분산 자료 관리(DDM)는 표를 실제로 이동하거나 복사하기 위해 iSeries Navigator에서 사용됩니다.소스 시스템의 오퍼레이팅 시스템이 버전 4 릴리스 4 이상을 실행 중이고 목표 시스템의 오퍼레이팅 시스템이 버전 4 릴리스 2 이상을 실행 중인 경우 TCP/IP에서 DDM을 사용하여 조작이 수행됩니다. 그렇지 않으면 SNA에서 DDM을 사용하여 조작이 수행됩니다. SNA에서 DDM을 사용하는 이동이나 복사 조작의 경우 iSeries Access가 시스템을 인식하는 이름이 DDM에서 사용하는 APPC 또는 APPN 장치 설명에 지정된 리모트 위치명과 동일해야 합니다. TCP/IP에서 DDM을 사용하는 이동이나 복사 조작의 경우 TCP 통신이 시스템 간에 작동되어야 합니다. TCP/IP의 경우 iSeries Access에 알려진 시스템 간에 TCP/IP가 작동 가능해야 함을 기억하십시오.

MOVOBJ(오브젝트 이동) 명령을 사용하여 파일 이동

MOVOBJ(오브젝트 이동) 명령은 현재 할당된 라이브러리에서 오브젝트를 제거하여 다른 라이브러리에 위치시킵니다. 이동된 오브젝트 유형은 OBJTYPE 매개변수에 지정됩니다.

관련 개념

분산 자료 관리

제어 언어(CL)

관련 참조

MOVOBJ(오브젝트 이동) 명령

데이터베이스 멤버 관리

이 주제는 데이터베이스 파일 멤버 관리를 위한 기본 조작에 대해 설명합니다. 실제 파일에 고유한 멤버 조작에 대해서도 설명합니다.

파일에 입력이나 출력 조작을 수행하기 전에 파일에 최소한 하나의 멤버가 있어야 합니다. 일반적으로 데이터베이스 파일에는 파일이 작성될 때 작성된 한 멤버만 포함됩니다. 사용자가 다른 이름을 지정하지 않는 한, 이 멤버명은 파일명과 동일합니다. 데이터베이스 파일에 대한 대부분의 조작에서는 사용되는 멤버가 파일의 첫 번째 멤버로 가정되며, 대부분의 파일에는 멤버가 하나뿐이므로 일반적으로 사용자가 멤버명을 지정하거나 신경 쓸 필요가 없습니다.

한 파일에 두 개 이상의 멤버가 있으면 각 멤버가 파일에 있는 자료의 서브세트로서 서비스를 제공합니다. 이것은 자료를 쉽게 정렬하는 방법을 제공합니다. 예를 들어, 미수금 파일을 정의합니다. 1년 동안의 자료를 이 파일 안에 보관하지만 자료를 한 달 단위로 처리한다고 하면 한 달별로 이름을 붙여 12개의 멤버를 갖는 실제 파일을 작성한 후 각 달의 자료를 별도로 처리할 수 있습니다(개별 멤버별로). 또한 멤버를 여러 개 또는 모두를 한 번에 처리할 수도 있습니다.

모든 데이터베이스 파일에 공통된 멤버 조작

시스템을 통해 파일 정의를 변경할 수 있습니다. 제어 언어(CL) 명령을 사용하여 이 조작의 대부분을 수행할 수 있습니다.

관련 개념

제어 언어(CL)

파일에 멤버 추가:

데이터베이스 파일에 멤버를 추가하는 방법은 다음과 같이 다양합니다.

- 자동적으로.CRTPF(실제 파일 작성) 또는 CRTLF(논리 파일 작성) 명령을 사용하여 파일 작성 시 디폴트는 자동으로 멤버(파일과 동일한 이름을 가짐)를 새로 작성된 파일에 추가는 것입니다. CRTSRCPF(소스 실제 파일 작성) 명령의 디폴트는 새로 작성된 파일에 멤버를 추가하지 않는 것입니다. 데이터베이스 파일 작성 명령에서 MBR 매개변수를 사용하여 멤버명을 달리 지정할 수도 있습니다. 파일이 작성될 때 멤버가 추가되지 않도록 하려면 MBR 매개변수에 *NONE을 지정하십시오.
- 명시적으로 파일이 작성된 후 ADDPFM(실제 파일 멤버 추가) 또는 ADDLFM(논리 파일 멤버 추가) 명령을 사용하여 멤버를 추가할 수 있습니다.
- CPYF(파일 복사) 명령. 복사중인 멤버가 복사될 파일에 존재하지 않을 경우 CPYF 명령을 사용하면 파일에 멤버가 추가됩니다.

관련 참조

실제 파일 작성(CRTPF) 명령

논리 파일 작성(CRTLF) 명령

실제 파일 멤버 추가(ADDPFM) 명령

파일 복사(CPYF) 명령

멤버 속성 변경:

CHGPFM(실제 파일 멤버 변경) 또는 CHGLFM(논리 파일 멤버 변경) 명령을 사용하여 실제 또는 논리 파일 멤버의 특정 속성을 변경할 수 있습니다.

실제 파일 멤버에 대해 매개변수 SRCTYPE(멤버의 소스 유형), EXPDATE(멤버의 만기일), SHARE(작업 내에 멤버 공유 여부) 및 TEXT(멤버의 텍스트 설명)를 변경할 수 있습니다. 논리 파일 멤버에 대해서는 SHARE 및 TEXT 매개변수를 변경할 수 있습니다.

주: CHGPF(실제 파일 변경) 및 CHGLF(논리 파일 변경) 명령을 사용하여 여러 가지 다른 파일 속성을 변경할 수 있습니다. 예를 들어, CHGPF 명령에 SIZE 매개변수를 사용하여 파일의 각 멤버에 허용되는 최대 크기를 변경할 수 있습니다.

관련 참조

실제 파일 멤버 변경(CHGPFM) 명령

논리 파일 멤버 변경(CHGLFM) 명령

CHGPF(실제 파일 변경) 명령

CHGLF(논리 파일 변경) 명령

멤버 재명명:

RNMM(멤버 재명명) 명령은 실제 또는 논리 파일의 기존 멤버명을 변경합니다. 그러나 파일명은 변경되지 않습니다.

관련 참조

멤버 재명명(RNMM) 명령

파일에서 멤버 제거:

RMVM(멤버 제거) 명령은 데이터베이스 파일에서 멤버와 해당 내용을 제거합니다.

멤버가 제거되면 시스템에서 더 이상 이 멤버를 사용할 수 없게 됩니다. 이것은 멤버로부터 자료만을 지우거나 삭제하는 것과는 다릅니다. 멤버가 존재하는 한, 프로그램은 멤버를 계속 사용할 수 있습니다(예, 멤버에 자료 추가).

관련 참조

멤버 제거(RMVM) 명령

실제 파일 멤버 조작

이 주제에서는 실제 파일 멤버에 대해 고유한 멤버 조작을 소개합니다. 조작 중인 실제 파일 멤버가 참조 제한사항과 연관되어 있을 경우 조작이 영향을 받을 수 있습니다.

관련 개념

291 페이지의 『참조 제한조건으로 자료 무결성 확인』
이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

실제 파일 멤버에서 자료 초기화:

프로그램에서 상대 레코드 처리를 사용하려면 데이터베이스 파일에는 프로그램에서 사용되는 최상위 상대 레코드 번호와 같은 수의 레코드 위치가 있어야 합니다. 상대 레코드 번호 처리를 사용하는 프로그램은 때때로 레코드들이 초기화될 것을 요구합니다.

INZPFM(실제 파일 멤버 초기화) 명령을 사용하여 두 가지 레코드 유형 중 하나로 멤버를 초기화할 수 있습니다.

- 디폴트 레코드
- 삭제 레코드

INZPFM 명령에 RECORDS 매개변수를 사용하여 원하는 레코드 유형을 지정할 수 있습니다.

디폴트 레코드를 사용하여 레코드를 초기화하면 새로운 각 레코드의 필드는 파일이 작성되었을 때 정의된 디폴트 필드 값으로 초기화됩니다. 디폴트 필드 값이 정의되지 않은 경우에는 숫자 필드는 0으로, 문자 필드는 공백으로 채워집니다.

가변 길이 문자 필드는 0 길이 디폴트 값을 가집니다. 널 가능 필드의 디폴트는 널값입니다. 디폴트 값이 정의되지 않은 경우에 날짜, 시간 및 시간소인에 대한 디폴트 값은 현재 날짜, 시간 또는 시간소인입니다. 프로그램 서술 파일의 디폴트 값은 모두 공백입니다.

주: 실제 파일 멤버 또는 연관된 논리 파일 멤버에 대한 DDS에 UNIQUE 키워드가 지정된 경우 하나의 디폴트 레코드를 초기화할 수 있습니다. 그렇지 않으면 일련의 중복 키 레코드가 작성됩니다.

레코드가 디폴트 레코드로 초기화되면 상대 레코드 번호를 사용하여 레코드를 읽고 자료를 변경할 수 있습니다.

레코드가 삭제 레코드로 초기화되면 삭제 레코드 중 하나의 상대 레코드 번호를 사용하여 레코드를 추가함으로써 자료를 변경할 수 있습니다. (삭제되지 않은 상대 레코드 번호로 레코드를 추가할 수는 없습니다.)

삭제 레코드는 읽을 수 없고, 단지 멤버 내에서 자리만 차지하고 있을 뿐입니다. 삭제 레코드 위에 새로운 레코드를 넣어 삭제 레코드를 변경할 수 있습니다.

관련 개념

229 페이지의 『데이터베이스 레코드 삭제』
삭제 조작을 사용하면 기존의 데이터베이스 레코드를 삭제할 수 있습니다.

관련 참조

실제 파일 멤버 초기화(INZPFM) 명령

실제 파일 멤버에서 자료 지우기:

CLRPFM(실제 파일 멤버 지우기) 명령은 실제 파일 멤버에서 자료를 제거합니다. 지우기 조작을 완료하면 멤버 설명만 남고 자료는 없어집니다.

관련 참조

실제 파일 멤버 지우기(CLRPFM) 명령

실제 파일 재구성:

이 주제에서는 i5/OS 오퍼레이팅 시스템에서 실제 파일을 재구성하는 방법과 재구성 조작을 일시중단하거나 취소하는 방법 및 선택할 수 있는 재구성 조작 유형에 대해 설명합니다. 실제 파일 재구성에 대한 고려사항도 포함됩니다.

iSeries Navigator를 사용하여 표 재구성:

이 주제에서는 iSeries Navigator를 사용하여 표를 재구성하는 방법과 재구성 창에 지정할 수 있는 옵션을 표시합니다.

표를 재구성하면 이상적인 실제 조직으로 표를 복원합니다. 데이터베이스 표의 이상적 조직은 자주 사용되는 색인에서 관련 키 값으로 순서화되어 페이지에 놓여 있는 행에 대한 것입니다. 삭제 레코드를 압축하거나 표 키에 의해 또는 선택된 색인에 의해 표를 재구성할 수 있습니다.

iSeries Navigator를 사용하여 표를 재구성하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스 폴더를 확장하십시오.
4. **스키마** 폴더를 확장하십시오.
5. 재구성하려는 표가 있는 스키마를 클릭하십시오.
6. 표를 클릭하십시오.
7. 세부사항 분할 창에서 재구성하려는 표를 마우스 오른쪽 버튼으로 클릭하고 **재구성**을 선택하십시오.

재구성 창에서 다음 옵션 중 하나를 선택하여 표에서 행을 재구성하는 방법을 지정하십시오.

- 도착 행 순서를 보존하지 않고 삭제된 행 압축: 삭제된 행이 남아 있지 않을 때까지 표의 끝에 있는 유효한 행이 삭제된 행으로 이동하도록 지정합니다.
- 삭제된 행을 압축하고 도착 행 순서 보존: 표에서 삭제된 첫 번째 행 다음의 유효한 모든 행이 삭제된 모든 행을 표 앞으로 이동되어 모든 삭제된 행을 압축하도록 지정합니다.
- 표 키로: 표에 있는 여러 행을 표의 액세스 경로 키값으로 재배열합니다. 표에 반드시 1차 키가 있거나 키 순 실제 파일이어야 합니다.
- 라이브러리에서 선택된 색인으로: 표의 행들이 지정 표를 사용하여 작성된 색인 또는 키가 있는 논리 파일의 키값에 따라 배열됩니다. 존재하는 색인만 선택할 수 있습니다. 사용자의 색인 리스트는 사용자가 선택한 라이브러리에 의해 결정됩니다.

다음과 같이 재구성 창에서 기타 옵션을 지정하여 재구성 조작의 성능 및 동시성을 제어할 수 있습니다.

- 재구성해야 하는 파티션된 파일의 파티션(또는 복수 멤버 실제 파일의 멤버) 지정
- 재구성 조작을 일시중단한 후 재시작할 수 있는지 여부 지정

  재구성을 일시중단할 수 있도록 지정하지 않으면 재구성 조작 기간 동안 표가 배타적으로 할당되므로 작업을 즉시 종료해야만 일시중단될 수 있습니다.

  재구성을 일시중단할 수 있도록 지정하는 경우:
  - 재구성 조작을 일시중단하는 경우 행이 유실되지 않았는지를 확인하도록 확약 제어에서 행이 이동되므로 파일을 저널해야 합니다.
  - 재구성 조작 기간 동안 다른 사용자가 표를 읽거나 변경할 수 있는지 여부도 지정할 수 있습니다. 재구성 동안 이동되는 열에서 짧은 기간 동안 잠금이 확보됩니다. 또한 동시 작업이 행에서 잠금을 확보할 경우 레코드 잠금 시간종료가 발생할 수 있습니다. 파일에 대한 레코드 잠금 대기 시간을 변경하거나 OVRDBF(데이터베이스 파일 대체) 명령을 사용하여 적절한 레코드 대기 시간을 지정할 수 있습니다.
- 다음과 같이 색인을 유지보수하는 방법을 지정하십시오.
  - 재구성 조작을 일시중단할 수 있도록 지정하는 경우 재구성 조작 동안 모든 색인을 유지보수하도록 지정할 수도 있습니다. 색인 리빌드가 필요하지 않습니다.
  - 그렇지 않으면 색인을 동기식 또는 비동기식으로 리빌드하도록 지정할 수 있습니다. EDTRBDAP(액세스 경로 재구성 편집) 명령을 사용하여 비동기식으로 빌드된 색인의 진행을 확인할 수 있습니다.

관련 개념

120 페이지의『레코드 잠금』

iSeries용 DB2 Universal Database에는 레코드에 대한 내장 무결성이 있습니다. 예를 들어, PGMA가 갱신 조작을 위해 레코드를 읽으면 그 레코드가 잠깁니다. PGMA가 그 레코드를 해제할 때까지는 다른 프로그램이 갱신 조작을 위해 동일한 레코드를 읽을 수 없지만 조회용으로는 읽을 수 있습니다.

RGZPFM(실제 파일 멤버 재구성) 명령을 사용하여 실제 파일 재구성:

이 주제에서는 RGZPFM(실제 파일 멤버 재구성) 명령 사용을 요약합니다.

RGZPFM 명령을 사용하여 다음 작업을 수행할 수 있습니다.

- 삭제 레코드가 차지하는 공간을 사용할 수 있도록 삭제 레코드를 제거합니다.
- 레코드를 검색하는 데 필요한 시간을 최소화하기 위해 순차적으로 레코드 액세스하는 순서로 파일의 레코드를 재구성합니다. 이는 KEYFILE 매개변수를 사용하여 수행됩니다. 이는 주로 도달순 이외의 순서로 액세스되는 파일의 경우에 유리합니다. 다음 키 필드 중 하나를 사용하여 멤버를 재구성할 수 있습니다.
  - 실제 파일의 키 필드
  - 실제 파일을 기초로 한 논리 파일의 키 필드
- 소스 파일 멤버를 재구성하고, 새로운 소스 순서 번호를 삽입하며, 소스 날짜 필드를 재설정합니다(RGZPFM 명령에서 SRCOPT 및 SRCSEQ 매개변수를 사용함).
- 재구성 조작을 취소할 수 없도록 지정하는 경우 실제 파일 형식의 가변 길이 필드로 이전에 사용되어 프래그먼트된 파일의 가변 부분의 공간을 재생하십시오.

관련 참조

실제 파일 멤버 재구성(RGZPFM) 명령

『사용상 주의사항: 파일 재구성』

다음은 파일 재구성 시 고려사항 리스트입니다.

예: 실제 파일 재구성:

이 주제의 예는 RGZPFM(실제 파일 멤버 재구성) 명령으로 실제 파일을 재구성하는 방법을 표시합니다.

예를 들어, 다음 RGZPFM 명령은 논리 파일의 액세스 경로를 사용하여 실제 파일의 첫 번째 멤버를 재구성합니다.

```
RGZPFM  FILE(DSTPRODLB/ORDHDRP)
                KEYFILE(DSTPRODLB/ORDFILL  ORDFILL)
```

실제 파일 ORDHDRP는 도달순 접근 경로를 갖고 있습니다. 그것은 논리 파일 ORDFILL의 액세스 경로를 사용하여 재구성되었습니다. *Order* 필드를 키 필드로 가정하십시오. 다음 예는 레코드가 배열된 방법을 표시합니다.

이 표는 원래 ORDHDRP 파일을 표시합니다. RGZPFM 명령이 실행되기 전에 레코드 3이 삭제되었음에 유의하십시오.

상대 레코드 번호	**Cust**	**Order**	**Ordate. . .**
1	41394	41882	072480. . .
2	28674	32133	060280. . .
3	삭제된	레코드	
4	56325	38694	062780. . .

이 표는 *Order* 필드를 오름차순 키 필드로 사용하여 재구성된 ORDHDRP 파일을 표시합니다.

상대 레코드 번호	**Cust**	**Order**	**Ordate. . .**
1	28674	32133	060280. . .
2	56325	38694	062780. . .
3	41394	41882	072480. . .

사용상 주의사항: 파일 재구성:

다음은 파일 재구성 시 고려사항 리스트입니다.

1. 도달순 액세스 경로를 가진 파일이 키순 액세스 경로를 사용하여 재구성되면 도달순 액세스 경로가 변경됩니다. 즉, 파일 내의 레코드들이 사용된 키순 액세스 경로의 순서대로 다시 배치됩니다. 사용 중인 키순 액세스 경로와 거의 일치하는 실제 순서로 자료를 재구성하면 자료의 순차 처리 성능을 향상시킬 수 있습니다.
2. 파일을 재구성하면 삭제 레코드가 압축되어 후속 상대 레코드 번호가 변경됩니다.
3. 선변경선출(FCFO), 선입선출(FIFO) 또는 후입선출(LIFO) DDS 키워드가 지정된 액세스 경로는 실제 파일 안에서는 실제 레코드 순서가 기준이 되므로 중복 키 필드가 있는 레코드 순서는 키순 액세스 경로를

사용하여 실제 파일을 재구성한 다음에 변경될 수 있습니다. 중복 키 필드가 있는 레코드의 순서는 KEYFILE 매개변수에 지정된 액세스 경로에 대해서만 유지보수됩니다. KEYFILE 매개변수에 지정된 액세스 경로에 LIFO DDS 키워드가 있으면 중복 키 필드는 재구성 조작을 취소(일시중단)할 수 있도록 지정할 경우에만 유지보수됩니다.

4. 재구성 조작을 취소할 수 없고 RGZPFM(실제 파일 멤버 재구성) 명령을 실행하는 작업을 종료하도록 지정하면 실제 파일 멤버에 대한 모든 액세스 경로를 재구성해야 할 수 있습니다. 재구성을 취소할 수 있는 상태에서 RGZPFM 명령을 취소하도록 지정하면 재구성 조작 동안 유지보수되지 않는 해당 액세스 경로만 리빌드해야 할 수 있습니다.
5. 한 행에 RGZPFM 명령을 두 번 사용하면 첫 번째 시도 후 파일의 총 *크기*와 두 번째 시도 후 총 *크기*에 차이가 있다는 것을 알 수 있습니다. 이것은 재구성 파일에 할당된 총 공간이 이후의 삽입 조작을 위해 여분의 허용하는 공간의 예상 값이기 때문입니다. 처음으로 레코드를 재구성한 후에 할당된 공간이 정확하게 계산됩니다.

재구성 조작 유형:

이 주제에서는 재구성 조작 유형과 기능을 나열합니다. 이러한 기능을 고려하여 사용할 조작을 선택해야 합니다.

- ALWCANCEL(*NO) - 재구성의 일반 유형입니다. 자료의 전체 사본을 작성할 수 있으므로 공간의 양이 최대 두 배까지 필요합니다. 이 옵션을 취소(일시중단)할 수 없고 병렬로 완전히 실행할 수도 없습니다. 파일을 배타적으로 사용해야 합니다.
- ALWCANCEL(*YES) - 자료의 전체 사본이 필요하지 않도록 자료 행을 파일에서 이동합니다. 그러나 파일을 저널해야 하므로 저널 항목에 기억장치가 필요합니다. 저널 리시버 임계값을 사용하여 특정 저널 리시버에서 사용되는 기억장치 양을 최소화할 수 있습니다.

  이 옵션은 취소(일시중단)하거나 재시작할 수 있습니다. DB2 UDB 대칭형 멀티프로세싱 옵션이 설치되어 있는 경우에는 병렬로 실행할 수 있습니다. 재구성 조작에서 사용된 자원의 양을 제어하기 위해 iSeries Navigator에서 CHGQRYA CL 명령 또는 조회 속성 변경을 사용하여 조회 속성을 변경할 수 있습니다.

  이 옵션에서는 시스템에 기억장치를 리턴하도록 재구성을 완료한 후에 몇 초 동안만 배타적 사용이 필요합니다. 배타적 잠금을 확보할 수 없는 경우 공간을 회복할 수 없음을 표시하는 경고 메세지가 작업 기록부에 송신됩니다. 공간을 회복하기 위해 동시 사용자가 파일에 액세스하지 않을 때 다시 재구성을 발행할 수 있습니다. 재구성 조작에서 재구성을 시작하기 전에 즉시 공간 회복을 시도합니다. 동시 자료 변경사항이 초기 재구성 조작 이후에 발생한 경우 공간의 일부만 회복할 수 있습니다.

  LOCK(*EXCLRD) 또는 LOCK(*SHRUPD)이 지정되면 동시 사용자가 파일의 행을 잠그거나 변경할 수 있으므로 재구성 조작의 결과가 정확한지 보증할 수 없습니다.

사용하려는 재구성 조작 유형은 몇 가지 요소에 따라 다릅니다. 예를 들어, 공간을 회복하는 것이 목적입니까 아니면 행 순서가 중요합니까? 재구성 조작의 취소(일시중단) 가능 여부가 중요합니까? 파일에 대한 동시 액세스 허용이 중요합니까? 이러한 요소에 기초하여 가장 적절한 옵션을 판별하려면 다음 표를 사용하십시오. 숨겨진 항목(별표로도 식별됨)은 특히 선택사항에 합당한 키 파일 옵션의 특성입니다.

	ALWCANCEL(*NO)		ALWCANCEL(*YES)		
	KEYFILE(*NONE)	**KEYFILE(*FILE 또는 키 파일)**	**KEYFILE (*RPLDLTRCD)**	**KEYFILE(*NONE)**	**KEYFILE(*FILE 또는 키 파일)**
취소 및 재시작	아니오	아니오	예*	예*	예*
동시 액세스	아니오	아니오	예*	예*	예*
병렬 처리	색인 리빌드에서만	색인 리빌드에서만	자료 이동 및 자료 리빌드*	자료 이동 및 자료 리빌드*	자료 이동 및 자료 리빌드*
비 병렬 성능	매우 빠름	빠름	매우 빠름*	느림*	가장 느림*
임시 기억장치	2배 자료 기억장치	2배 자료 기억장치	저널 리시버 기억장치*	저널 리시버 기억장치*	저널 리시버 기억장치*
LIFO KEYFILE 색인 처리	N/A	반전된 복제	N/A*	N/A*	보존된 복제 순서*
색인 처리(비 KEYFILE)	동기 또는 비동기 리빌드*	동기 또는 비동기 리빌드*	색인이나 동기 또는 비동기 리빌드 유지보수*	색인이나 동기 또는 비동기 리빌드 유지보수*	색인이나 동기 또는 비동기 리빌드 유지보수*
정확한 마지막 행 위치	예*	예*	LOCK(*EXCL)이고 재시작되지 않는 경우에만	LOCK(*EXCL)이고 재시작되지 않는 경우에만	LOCK(*EXCL)이고 재시작되지 않는 경우에만
사용된 CPU 및 I/O의 양	가장 적음*	다음으로 가장 적음*	가장 적음	많음	가장 많음
가변 길이 세그먼트 재구성	좋음*	좋음*	나쁨	나쁨	나쁨
참조 무결성 상위 및 FILE LINK CONTROL DataLink 허용	예*	예*	아니오	아니오	아니오
QTEMP 및 데이터베이스 상호 참조 파일 허용	예*	예*	아니오	아니오	아니오
복제 비용	최소(하나의 저널 항목)*	최소(하나의 저널 항목)*	많음(이동된 모든 행의 저널 항목)	가장 많음(이동된 모든 행의 저널 항목)	가장 많음(이동된 모든 행의 저널 항목)

재구성 조작 일시중단 또는 취소:

재구성 조작을 일시중단하거나 취소해야 할 경우가 있습니다.

재구성 조작을 취소할 수 없고 RGZPFM(실제 파일 멤버 재구성) 명령을 실행하는 동안 다음 상황 중 하나가 발생하도록 지정하는 경우 레코드를 전혀 재구성할 수 없게 될 수 있습니다.

- 시스템이 이상 종료하는 경우
- RGZPFM 명령이 들어 있는 작업이 *IMMED 옵션으로 종료되는 경우
- RGZPFM 명령이 수행 중이던 서브시스템이 *IMMED 옵션으로 종료되는 경우
- 시스템이 *IMMED 옵션으로 중단되는 경우

또한 RGZPFM 명령을 실행 중일 경우 큰 오브젝트와 연관된 레코드(LOB)는 재구성되지 않을 수 있으며 삭제된 레코드가 파일에 남아 있을 수 있습니다.

재구성을 취소할 수 있는 상태에서 앞의 상황 중 하나가 발생하거나 재구성 조작을 취소하도록 지정한 경우 이 조작은 부분적으로만 완료될 수 있습니다. 나중에 동일한 재구성 조작을 발생할 경우 인터럽트된 위치에서 재구성을 계속할 수 있습니다. 그러나 재구성 조작을 취소한 다음에 중요한 변경사항이 발생하면 재구성 조작이 계속되지 않고 처음부터 다시 시작됩니다.

재구성이 종료되기 전에 시스템이 처리한 양과 사용자가 SRCOPT 매개변수에서 지정한 내용에 따라서도 재구성되는 멤버의 상태가 결정됩니다.

SRCOPT 매개변수가 지정되면 멤버에 다음 중 하나가 발생할 수 있습니다.

- 멤버가 완전히 재구성됩니다. 재구성 조작이 정상적으로 종료되었음을 알리는 완료 메세지가 사용자의 작업 기록부로 전달됩니다.
- 전혀 재구성되지 않거나 부분적으로만 재구성됩니다. 재구성 조작이 정상적으로 이루어지지 못했음을 알리는 메세지가 사용자의 작업 기록부로 전달됩니다. 이 경우 RGZPFM(실제 파일 멤버 재구성) 명령을 다시 실행하십시오.
- 멤버가 재구성되기는 했지만 일부 순서 번호만 변경됩니다. 멤버가 재구성되었으나 순서 번호가 모두 변경되지 않았음을 알리는 완료 메세지가 사용자의 작업 기록부로 전달됩니다. 이 경우 KEYFILE(*NONE)을 지정하여 RGZPFM 명령을 다시 수행하십시오.

SRCOPT 매개변수가 지정되지 않았을 경우 멤버가 완전히 재구성되거나 전혀 재구성되지 않습니다. 재구성된 파일의 개수를 판별하고 RGZPFM 명령을 다시 실행할 수 있도록(필요한 경우) iSeries Navigator에서 빠른 보기 또는 DSPPFM(실제 파일 멤버 표시) 명령을 사용하여 파일 내용을 표시할 수 있습니다.

실제 파일 멤버에 있는 삭제 레코드의 수를 줄이기 위해 삭제된 레코드 공간을 재사용하여 파일을 작성하거나 변경할 수 있습니다.

관련 개념

115 페이지의『삭제 레코드 재사용』
데이터베이스 파일에 삭제 레코드를 재사용하려는 경우가 있습니다. 이 경우 REUSEDLT 매개변수를 사용할 수 있습니다.

실제 파일 멤버에서 레코드 표시:

DSPPFM(실제 파일 멤버 표시) 명령은 실제 파일 멤버의 레코드를 표시합니다.

DSPPFM 명령은 도달순으로 실제 파일의 자료를 표시하는 데 사용할 수 있습니다. DSPPFM 명령은 다음 작업에 사용할 수 있습니다.

- 문제점 분석
- 디버깅
- 레코드 조회

파일이 키순으로 되어 있거나 입력순으로 되어 있는지에 관계없이 소스 파일이나 자료 파일을 표시할 수 있습니다. 파일이 키순 파일일 경우에도 레코드는 입력순으로 표시됩니다. 파일을 넘겨서 레코드 번호로 특정 레코드를 찾거나 레코드의 다른 부분을 보기 위해 화면을 좌우로 이동시킬 수 있습니다. 또한 기능 키를 눌러 화면에서 문자 자료나 16진 자료를 볼 수도 있습니다.

조회 프로그램이 설치되어 있는 경우 STRQRY(조회 시작) 명령을 사용하여 레코드를 선택하고 표시할 수도 있습니다.

SQL 언어가 설치되어 있을 경우 STRSQL(SQL 시작) 명령을 사용하여 레코드를 대화식으로 선택하고 표시할 수 있습니다.

관련 참조

실제 파일 멤버 표시(DSPPFM) 명령

데이터베이스 속성 및 상호 참조 정보 사용

iSeries의 통합 데이터베이스는 파일 속성 및 상호 참조 정보를 제공합니다.

상호 참조 정보에는 다음과 같은 내용이 포함됩니다.

- 프로그램에서 사용되는 파일
- 자료나 액세스 경로를 위해 다른 파일에 의존하는 파일
- 파일 속성
- 파일에 정의된 필드
- 파일과 연관된 제한조건
- 파일에 대한 키 필드

이 주제에 설명되는 각 명령은 화면이나 인쇄 출력에 정보를 표시하거나 상호 참조 정보를 데이터베이스 파일에 기록하여 프로그램이나 유틸리티(예: 조회)에서 분석하는 데 사용될 수 있습니다.

RTVMBRD(멤버 설명 검색) 명령을 사용하여 어플리케이션에서 사용할 데이터베이스 파일의 멤버에 대한 정보를 검색할 수 있습니다.

관련 개념

제어 언어(CL)

관련 참조

멤버 설명 검색(RTVMBRD) 명령

데이터베이스 파일 정보 표시

iSeries Navigator에서 표 설명 표시 조작으로 데이터베이스 파일 및 장치 파일에 대한 파일 속성을 표시할 수 있습니다.

iSeries Navigator에서 표 설명 표시를 사용하여 파일 속성 표시:

이 주제에서는 iSeries Navigator에서 표 설명 표시를 사용하여 데이터베이스 파일 속성을 표시하는 방법을 보여줍니다.

데이터베이스 파일 속성을 표시하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 라이브러리를 확장하십시오.
4. 정보를 표시하려는 표 또는 뷰가 들어 있는 라이브러리를 클릭하십시오.

5. 표, 뷰 또는 색인을 마우스 오른쪽 버튼으로 클릭하고 **설명**을 클릭하십시오.

설명 창에서 일반, 할당, 사용, 활동 및 세부사항을 선택할 수 있습니다.

DSPFD(파일 설명 표시) 명령을 사용하여 파일 속성 표시:

DSPFD(파일 설명 표시) 명령을 사용하여 데이터베이스 파일의 속성을 표시할 수 있습니다. 정보를 표시하거나 인쇄할 수 있으며, 데이터베이스 출력 파일(OUTFILE)에 기록할 수도 있습니다.

DSPFD 명령으로 제공되는 정보는 다음과 같습니다(괄호 안은 매개변수 값).

- 기본 속성(*BASATR)
- 파일 속성(*ATR)
- 액세스 경로 스펙(*ACCPTH, 논리 및 실제 파일 전용)
- 선택/생략 스펙(*SELECT, 논리 파일 전용)
- 결합 논리 파일 스펙(*JOIN, 결합 논리 파일 전용)
- 대체 배열 순서 스펙(*SEQ, 실제 및 논리 파일 전용)
- 레코드 형식 스펙(*RCDFMT)
- 멤버 속성(*MBR, 실제 및 논리 파일 전용)
- 스풀링 속성(*SPOOL, 프린터 및 디스켓 파일 전용)
- 멤버 리스트(*MBRLIST, 실제 및 논리 파일 전용)
- 파일 제한조건(*CST)
- 트리거(*TRG)

관련 참조

파일 설명 표시(DSPFD) 명령

파일에 필드 설명 표시:

DSPFFD(파일 필드 설명 표시) 명령을 사용하여 데이터베이스 파일 및 장치 파일에 대한 필드 정보를 표시할 수 있습니다. 정보를 표시하거나 인쇄할 수 있으며, 데이터베이스 출력 파일(OUTFILE)에 기록할 수도 있습니다.

관련 참조

파일 필드 설명 표시(DSPFFD) 명령

시스템에서 파일 간 관계 표시:

DSPDBR(데이터베이스 관계 표시) 명령을 사용하여 시스템의 파일 간 관계를 표시할 수 있습니다. 이러한 정보는 표시되거나 인쇄되거나 데이터베이스 출력 파일(OUTFILE)에 기록될 수 있습니다.

DSPDBR 명령을 사용하여 사용자의 데이터베이스 조직에 대해 다음 정보를 표시할 수 있습니다.

- 특정 레코드 형식을 사용하는 데이터베이스 파일(실제 및 논리) 리스트

- 자료를 공유하기 위해 지정된 파일에 종속되는 데이터베이스 파일(실제 및 논리) 리스트
- 자료를 공유하거나 액세스 경로를 공유하기 위해 지정된 멤버에 종속되는 멤버(실제 및 논리) 리스트
- 이 파일과 참조 제한조건 관계에 있는 종속 파일인 실제 파일 리스트

예를 들어 ORDHDRP 레코드 형식을 가진 실제 파일 ORDHDRP와 연관된 모든 데이터베이스 파일 리스트를 표시하려면 다음의 DSPDBR 명령을 입력하십시오.

```
DSPDBR FILE(DSTPRODLB/ORDHDRP) RCDFMT(ORDHDR)
```

주: 이 화면에 대한 세부사항은 제어 언어(CL) 주제에서 DSPDBR 명령 설명을 참조하십시오.

이 화면에는 RCDFMT 매개변수에 레코드 형식명이 지정될 때의 헤더 정보와 지정된 레코드 형식을 사용하는 파일에 관한 정보 등이 표시됩니다.

DSPDBR 명령의 MBR 매개변수에 멤버명이 지정되면 종속되는 멤버가 표시됩니다.

DSPDBR 명령이 디폴트 MBR(*NONE) 매개변수 값으로 지정된 경우 종속 자료 파일이 표시됩니다. 공유 액세스 경로를 표시하려면 멤버명을 지정해야 합니다.

DSPDBR 명령 출력은 관련된 공유 유형을 식별합니다. 명령의 결과가 표시되는 경우 공유 유형의 이름이 표시됩니다. 명령의 결과가 데이터베이스 파일에 기록되는 경우 공유 유형 코드(아래와 같음)가 출력 파일 레코드의 *WHTYPE* 필드에 들어가게 됩니다.

유형	코드	설명
제한조건	C	실제 파일은 제한조건을 통해 연관되는 다른 실제 파일의 자료에 종속됨.
자료	D	파일이나 멤버가 다른 파일 내의 멤버의 자료에 종속됨.
액세스 경로 공유	I	파일 멤버가 액세스 경로를 공유함
액세스 경로 소유자	O	액세스 경로가 공유되면 파일의 멤버 중 하나가 소유자로 간주됨. 액세스 경로의 소유자는 액세스 경로에 사용되는 기억장치에 대해 책임을 지게 됨. 표시된 멤버가 소유자로서 지정되었으면, 하나 이상의 파일 멤버가 액세스 경로 공유를 나타내는 I로 지정됨.
SQL 뷰	V	SQL 뷰 또는 멤버가 다른 SQL 뷰에 종속됨.

관련 개념

제어 언어(CL)

관련 참조

데이터베이스 관계 표시(DSPDBR) 명령

프로그램에서 사용하는 파일 표시:

DSPPGMREF(프로그램 참조 표시) 명령을 사용하여 프로그램에서 사용하는 파일, 자료 영역 및 기타 프로그램을 판별할 수 있습니다. 이 정보는 컴파일된 프로그램에 대해서만 사용 가능하며, 데이터베이스 출력 파일(OUTFILE)에 기록하거나 인쇄하거나 표시할 수 있습니다.

프로그램이 작성되면 프로그램에서 사용되는 특정 오브젝트에 관한 정보가 저장되어 DSPPGMREF 명령으로 이 정보를 사용할 수 있습니다.

다음 표는 고급 언어 및 유틸리티가 저보를 저장하는 오브젝트를 나타냅니다.

언어 또는 유틸리티	파일	프로그램	자료 영역	주 참조
BASIC	예	예	아니오	1
C/400 언어	아니오	아니오	N/A	
CL	예	예	예	2
COBOL/400 언어	예	예	아니오	3
CSP	예	예	아니오	4
DFU	예	N/A	N/A	
FORTRAN/400 언어	아니오	아니오	N/A	
Pascal	아니오	아니오	N/A	
PL/I	예	예	N/A	3
RPG/400 언어	예	예	예	5
SQL 언어	예	N/A	N/A	

주:

1. 외부 서술 파일 참조, 프로그램 및 자료 영역이 저장됩니다.
2. 파일, 프로그램 또는 자료 영역을 참조하는 모든 시스템 명령은 명령 정의에 제어 언어(CL) 프로그램에서 명령이 컴파일될 경우 정보를 저장하도록 지정합니다. 변수가 사용될 경우 변수명이 오브젝트명(예를 들면, &FILE)으로 사용됩니다. 표현식이 사용될 경우 오브젝트명은 *EXPR로 저장됩니다. 사용자 정의 명령도 명령에서 지정된 파일, 프로그램 또는 자료 영역의 정보를 저장할 수 있습니다. 제어 언어(CL) 주제에서 PARM 또는 ELEM 명령문의 FILE, PGM 및 DTAARA 매개변수에 대한 설명을 참조하십시오.
3. 프로그램명으로 COBOL/400 ID가 사용되는 경우에는(동적 호출, 예: `CALL PGM1`) 프로그램명이 저장되지 않고 리터럴이 사용되는 경우(정적 호출, 예: `CALL 'PGM1'`)에만 프로그램명이 저장됩니다.
4. CSP 프로그램도 오브젝트 유형 `*MSGF`, `*CSPMAP` 및 `*CSPTBL`에 대한 정보를 저장합니다.
5. 로컬 자료영역의 사용은 저장되지 않습니다.

저장된 파일 정보에는 사용 유형별 항목(번호)이 포함됩니다. DSPPGMREF 명령의 데이터베이스 파일 출력에서(OUTFILE 매개변수 사용 시 작성) 다음과 같이 지정됩니다.

코드 의미

1 입력

2 출력

3 입출력

4 갱신

8 지정 안됨

코드를 조합하여 사용하는 경우도 있습니다. 예를 들어, 코드 7로 지정된 파일은 입력, 출력 및 갱신에 사용됩니다.

관련 개념

제어 언어(CL)

관련 참조

DSPPGMREF(프로그램 참조 표시) 명령

시스템 상호 참조 파일 표시:

시스템에서 관리하는 데이터베이스 파일을 사용하여 기본 속성 및 데이터베이스 파일 요구사항을 결정할 수 있습니다. 이러한 파일에 포함된 필드를 표시하려면 DSPFFD(파일 필드 설명 표시) 명령을 사용하십시오.

시스템은 다음 정보를 포함하는 8개의 데이터베이스 파일을 관리합니다.

- 기본 데이터베이스 파일 속성 정보(QSYS/QADBXREF)
- 시스템의 모든 데이터베이스 파일(QTEMP 라이브러리에 있는 데이터베이스 파일은 제외하고)에 대한 상호 참조 정보
- 데이터베이스 파일 필드 정보(QSYS/QADBIFLD)
- 데이터베이스 파일 키 필드 정보(QSYS/QADBKFLD)
- 참조 제약조건 파일 정보(QSYS/QADBFCST)
- 참조 제약조건 필드 정보(QSYS/QADBCCST)
- SQL 패키지 정보(QSYS/QADBPKG)
- 리모트 데이터베이스 디렉토리 정보(QSYS/QADBXRDBD)

주: 이러한 파일들을 사용하는 권한은 보안 담당자로 제한됩니다. 그러나 모든 사용자는 각 파일에 대해 빌드된 논리 파일 중 하나(또는 유일한 논리 파일)를 사용하여 자료를 볼 수 있는 권한이 있습니다. 이러한 파일들에 대한 권한은 이 파일들이 항상 열려있기 때문에 변경될 수 없습니다.

관련 참조

파일 필드 설명 표시(DSPFFD) 명령

명령의 출력을 데이터베이스 파일에 직접 기록

여러 제어 언어(CL) 명령에 OUTFILE 매개변수를 지정하여 출력 실제 파일에 이 명령의 출력을 저장할 수 있습니다. 이제 프로그램이나 유틸리티(예: 조회)에서 출력 파일을 사용하여 자료를 분석할 수 있습니다.

예를 들어, DSPPGMREF(프로그램 참조 표시) 명령의 출력을 실제 파일로 보낸 후, 이를 조회하여 어떤 프로그램이 특정한 파일을 사용하는지를 알아낼 수 있습니다.

명령에 OUTFILE 매개변수가 지정되면 실제 파일이 작성됩니다. 초기에는 소유자(명령을 수행한 사용자)만이 파일을 사용할 수 있도록 개인 권한으로 파일이 작성됩니다. 그러나 다른 데이터베이스 파일에 대해서와 마찬가지로 소유자가 다른 사용자들에게 이들 파일에 대한 권한부여할 수 있습니다.

시스템은 OUTFILE 매개변수를 지정할 수 있는 각 명령에 대한 레코드 형식을 식별하는 모델 파일을 제공합니다. 이미 존재하지 않는 파일의 OUTFILE 매개변수에 파일명을 지정하면 시스템은 모델 파일과 동일한 레

코드 형식을 사용하여 파일을 작성합니다. 기존의 출력 파일에 대해 파일명이 지정되면 레코드 형식이 모델 파일의 레코드 형식과 동일한지를 시스템이 검사합니다. 레코드 형식이 일치하지 않으면 시스템이 작업으로 메세지를 보내며 명령은 완료되지 않습니다.

주: OUTFILE 매개변수에 시스템 제공 모델 파일을 지정하지 말고 사용자가 작성한 파일을 출력 파일로 사용해야 합니다.

이 명령에 대한 출력 파일과 모델 파일들의 이름을 제공하는 명령 리스트에 대해서는 CL 참조서 주제를 참조하십시오.

주: 모든 시스템 제공 모델 파일은 QSYS 라이브러리에 있습니다.

DSPFFD(파일 필드 설명 표시) 명령을 사용하여 시스템 제공 모델 파일의 레코드 형식에 포함된 필드를 표시할 수 있습니다.

관련 개념

제어 언어(CL)

관련 참조

파일 필드 설명 표시(DSPFFD) 명령

예: 명령 출력 파일:

이 예는 제어 언어(CL) 명령의 출력을 데이터베이스 파일로 직접 기록하는 방법을 표시합니다.

DSPPGMREF(프로그램 참조 표시) 명령을 사용하여 모든 라이브러리에서 컴파일된 모든 프로그램에 대한 정보를 수집하고 DBROUT라는 데이터베이스 파일에 출력할 수 있습니다.

```
DSPPGMREF  PGM(*ALL/*ALL)  OUTPUT(*OUTFILE)  OUTFILE(DSTPRODLB/DBROUT)
```

조회를 사용하여 출력 파일을 처리하거나 또 다른 방법으로 파일로부터 정보를 선택하는 논리 파일을 작성합니다. 다음 예는 이러한 논리 파일에 대한 DDS를 표시합니다. 파일명에 따라 레코드가 선택됩니다.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
          A* Logical file DBROUTL for query
     A
          A          R DBROUTL                   PFILE(DBROUT)
          A          S WHFNAM                    VALUES('ORDHDRL' 'ORDFILL')
     A
```

DSPFD(파일 설명 표시) 명령에 대한 출력 파일:

DSPFD(파일 설명 표시) 명령은 지정된 매개변수에 따라 고유한 출력 파일을 제공합니다.

LIBA 라이브러리에 있는 모든 파일에 대한 액세스 경로 정보를 수집하려면 다음을 지정하십시오.

```
DSPFD   FILE(LIBA/*ALL)  TYPE(*ACCPTH)  OUTPUT(*OUTFILE) +
                OUTFILE(LIBB/ABC)
```

파일 ABC가 라이브러리 LIBB 안에 작성되며 시스템 제공 파일 QSYS/QAFDACCP에서와 같은 필드 설명으로 외부 서술됩니다. 이때 파일 ABC는 LIBA에 액세스 경로를 가지고 있는 파일 내의 각 키 필드에 대한 레코드를 포함합니다.

DSPFD 명령이 다음과 같이 코딩된 경우:

```
DSPFD   FILE(LIBX/*ALL)  TYPE(*ATR)  OUTPUT(*OUTFILE) +
                FILEATR(*PF)  OUTFILE(LIBB/DEF)
```

파일 DEF가 라이브러리 LIBB 안에 작성되며, QSYS/QAFDPHY에 있는 것과 같은 필드 설명으로 외부 서술됩니다. 이때 파일 DEF에는 라이브러리 LIBX에 있는 각 실제 파일에 대한 레코드를 포함합니다.

DSPFFD 명령을 사용하여 IBM에서 제공되는 각 모델 파일의 필드명을 표시할 수 있습니다. 예를 들어, 액세스 경로 모델 파일(TYPE 매개변수에 *ACCPTH가 지정됨)의 필드 설명을 표시하려면 다음과 같이 지정하십시오.

```
DSPFFD   QSYS/QAFDACCP
```

관련 개념

제어 언어(CL)

관련 참조

파일 설명 표시(DSPFD) 명령

DSPJRN(저널 표시) 명령에 대한 출력 파일:

DSPJRN(저널 표시) 명령은 고유한 출력 파일을 제공합니다.

관련 개념

제어 언어(CL)

관련 참조

저널 표시(DSPJRN) 명령

DSPPRB(문제점 표시) 명령에 대한 출력 파일:

DSPPRB(문제점 표시) 명령은 레코드 유형에 따라 고유한 출력 파일을 제공합니다.

출력 파일은 다음 레코드 유형에 따라 달라집니다.

- 기본 문제점 자료 레코드(*BASIC). 문제점 유형, 상태, 기계 유형/모델/일련 번호, 제품 ID, 문의처 정보 및 자료 추적에 대한 내용 등이 들어 있습니다.
- 장애 지점, 분리 또는 응답 FRU 레코드(*CAUSE). 응답 FLU가 있으면 이것이 사용되고, 그렇지 않은 경우 분리 FRU가 있으면 이것이 사용됩니다. 응답 FRU와 분리 FRU가 없으면 장애 지점 FRU가 사용됩니다.
- PTF 고정 레코드(*FIX)
- 사용자가 입력한 텍스트(주석 레코드)(*USRTXT)
- 지원 자료 ID 레코드(*SPTDTA)

원인, 수정 및 사용자 텍스트 정보와 지원 자료가 기본 문제점 자료와 상관될 수 있도록 모든 5가지 출력 파일의 레코드에는 문제점 ID가 있습니다. 특정 출력 파일에는 오직 하나의 자료 유형이 기록될 수 있습니다. 원인, 수정, 사용자 텍스트 및 지원 자료 출력 파일은 특정 문제에 대해 복수의 레코드가 있을 수 있습니다.

관련 개념

제어 언어(CL)

관련 참조

문제점 표시(DSPPRB) 명령

데이터베이스 파일 설명 및 속성 변경

이 주제에서는 데이터베이스 파일 설명 및 속성 변경 방법에 대해 설명합니다. 파일 설명에서 필드 변경에 대한 고려사항도 포함됩니다.

파일 설명에서 필드 변경의 영향

이때 레벨 ID를 결정하기 위해 레코드 형식 설명의 정보가 사용됩니다. 파일 설명에서 필드를 변경하면 레벨 ID가 변경됩니다. 키 필드나 선택/생략 필드의 내용을 변경하면 새 액세스 경로를 사용하는 프로그램에 예상치 못한 결과를 가져올 수 있습니다.

외부 서술 자료를 사용하는 프로그램이 컴파일될 때 컴파일러는 파일의 파일 설명을 컴파일하는 프로그램에 복사합니다. 프로그램을 수행할 때 시스템은 파일에 현재 정의되어 있는 레코드 형식이 프로그램이 컴파일되었을 때의 형식과 같은지 확인합니다. 디폴트 값은 레벨 검사를 수행합니다. 시스템은 해당 파일이 작성될 때 각 레코드 형식에 고유한 레벨 ID를 할당합니다. 이때 레벨 ID를 결정하기 위해 레코드 형식 설명의 정보가 사용됩니다. 이 정보에는 레코드 형식의 총 길이, 레코드 형식명, 정의된 필드의 개수 및 순서, 자료 유형, 필드의 크기, 필드명, 필드의 소수 자릿수 및 필드가 널값을 허용하는지 여부가 포함됩니다. 레코드 형식에서 이 정보를 변경하면 레벨 ID가 변경됩니다.

다음의 DDS 정보는 레벨 ID에 아무런 영향을 미치지 않으므로 파일을 사용하는 프로그램을 다시 컴파일하지 않고서도 이들을 변경할 수 있습니다.

- TEXT 키워드
- COLHDG 키워드
- CHECK 키워드
- EDTCDE 키워드
- EDTWRD 키워드
- REF 키워드
- REFFLD 키워드
- CMP, RANGE 및 VALUES 키워드
- TRNTBL 키워드
- REFSHIFT 키워드
- DFT 키워드

- CCSID 키워드
- 결합 스펙 및 결합 키워드
- 키 필드
- 액세스 경로 키워드
- 선택/생략 필드

키 필드나 선택/생략 필드를 변경해도 레벨 검사가 유발되지는 않지만 이러한 변경이 새로운 액세스 경로를 사용하는 프로그램에 예기치 못한 결과를 가져올 수 있다는 점에 유의하십시오. 예를 들어, 키 필드를 고객 번호에서 고객명으로 변경할 경우 레코드가 검색되는 순서가 바뀌어 파일을 처리하는 프로그램에 예기치 못한 문제가 발생할 수도 있습니다.

레벨 검사가 지정되면(또는 디폴트 값으로서), 파일이 열릴 때 사용될 파일의 레벨 ID가 프로그램 내의 파일 레벨 ID와 비교됩니다. ID가 서로 다를 경우 변경된 조건을 알려주는 메세지가 프로그램에 전달되며, 이 변경은 프로그램에 영향을 미칠 수 있습니다. 변경사항을 반영하도록 프로그램을 다시 컴파일할 수 있습니다.

또 하나의 방법으로는 변경된 사항이 프로그램에 영향을 미쳤는지를 알기 위해 파일 설명을 표시하는 것이 있습니다. DSPFFD(파일 필드 설명 표시) 명령을 사용하여 설명을 표시하거나, 소스 입력 유틸리티(SEU)가 있을 경우 파일에 대한 DDS를 포함하는 소스 파일을 표시할 수 있습니다.

파일에 정의된 형식 레벨 ID는 DSPFD(파일 설명 표시) 명령을 사용하여 표시할 수 있습니다. 레벨 ID를 표시하는 경우 주의해야 할 점은 비교되는 것은 파일 ID가 아닌 레코드 형식 ID라는 점입니다.

파일 변경이 프로그램에 매번 영향을 미치는 것은 아닙니다. 예를 들어, 파일 끝에 필드를 추가하고 프로그램이 이 새로운 필드를 사용하지 않는 경우 프로그램을 다시 컴파일할 필요가 없습니다. 변경이 프로그램에 영향을 미치지 않는 경우 CHGPF(실제 파일 변경) 또는 CHGLF(논리 파일 변경) 명령에서 LVLCHK(*NO)를 지정하여 파일에 대한 레벨 검사를 하지 않게 하거나 OVRDBF(데이터베이스 파일로 대체) 명령에서 LVLCHK(*NO)를 지정하여 레벨 검사 없이 프로그램을 수행하도록 만들 수 있습니다.

레벨 검사를 수행하는 것이 바람직한 방법이라는 것을 명심해야 합니다. LVLCHK(*YES)를 사용하는 것은 데이터베이스의 무결성을 보장하는 좋은 방법이며, LVLCHK(*NO)를 사용하면 때때로 예측할 수 없는 결과를 가져올 수도 있습니다.

관련 참조

파일 필드 설명 표시(DSPFFD) 명령

파일 설명 표시(DSPFD) 명령

CHGPF(실제 파일 변경) 명령

CHGLF(논리 파일 변경) 명령

데이터베이스 파일 대체(OVRDBF) 명령

실제 파일 설명 및 속성 변경

실제 파일 설명을 변경하고 파일을 다시 작성하면 레벨 ID가 변경될 수 있습니다. 레벨 ID가 변경되면 논리 파일을 사용하여 재컴파일을 피하거나 프로그램을 다시 컴파일할 수 있습니다.

예를 들어, 파일 설명에 필드를 추가하거나 기존 필드의 길이를 변경하면 레벨 ID가 변경됩니다. 레벨 ID가 변경되면 실제 파일을 사용하는 프로그램을 다시 컴파일하면 됩니다. 프로그램이 재컴파일되면 새 레벨 검사 ID를 사용합니다.

실제 파일의 원래 레코드 형식으로 프로그램에 자료를 제시하는 논리 파일을 작성하여 프로그램을 다시 컴파일하지 않을 수도 있습니다. 이 경우 논리 파일은 변경 전의 실제 파일과 동일한 레벨 검사 ID를 갖게 됩니다.

예를 들어, 실제 파일 레코드 형식에 필드를 추가하려고 합니다. 프로그램을 다시 컴파일하지 않으려면 다음 단계를 수행하십시오.

1. DDS를 변경하여 새로운 필드를 포함하는 새로운 실제 파일(LIBA의 FILEB)을 작성합니다.

   ```
   CRTPF FILE(LIBA/FILEB) MBR(*NONE)...
   ```

   FILEB에는 멤버가 없습니다. (이전 파일 FILEA는 라이브러리 LIBA 내에 있으며, 하나의 멤버 MBRA가 들어 있습니다.)
2. 이전 실제 파일의 멤버를 새로운 실제 파일에 복사합니다.

   ```
   CPYF    FROMFILE(LIBA/FILEA) TOFILE(LIBA/FILEB)
                    FROMMBR(*ALL) TOMBR(*FROMMBR)
                  MBROPT(*ADD) FMTOPT(*MAP)
   ```

   FROMMBR(*ALL) 및 TOMBR(*FROMMBR)가 지정되었으므로 새로운 실제 파일의 멤버는 자동적으로 이전 실제 파일의 멤버명과 동일하게 명명됩니다. FMTOPT 매개변수는 필드명으로 필드의 자료를 복사(*MAP)하도록 지정합니다.
3. 원래의 실제 파일처럼 보이는 새로운 논리 파일(FILEC)을 서술합니다. (논리 파일 레코드 형식에 새로운 실제 파일 필드는 포함되지 *않습니다.)* PFILE 키워드에 FILEB를 지정하십시오. (FILEA와 FILEC는 동일한 형식을 가지므로 레벨 검사가 수행될 때 논리 파일의 레벨 ID와 프로그램의 레벨 ID가 일치합니다.)
4. 새로운 논리 파일을 작성합니다.

   ```
   CRTLF FILE(LIBA/FILEC)...
   ```
5. 다음 조작 중 하나를 수행할 수 있습니다.
   a. 해당 작업에서 OVRDBF(데이터베이스 파일 대체) 명령을 사용하여 프로그램에서 참조되는 이전 실제 파일을 논리 파일로 대체합니다. (OVRDBF 명령 매개변수에 관해서는 113 페이지의 『데이터베이스 파일 처리: 런타임 고려사항』에 자세히 설명되어 있습니다.)

      ```
      OVRDBF FILE(FILEA) TOFILE(LIBA/FILEC)
      ```

   b. 프로그램에서의 파일명이 대체될 필요가 없도록 이전 실제 파일을 삭제한 후, 논리 파일을 이전 실제 파일의 이름으로 재명명하십시오.

      ```
      DLTF FILE(LIBA/FILEA)
      RNMOBJ   OBJ(LIBA/FILEC) OBJTYPE(*FILE)
                        NEWOBJ(FILEA)
      ```

다음 표는 세 파일에 사용된 레코드 형식의 관계를 표시합니다.

표 48. FILEA(기존 실제 파일)

FLDA	FLDB	FLDC	FLDD

FILEB에 FLDB1이 레코드 형식으로 추가되었습니다.

표 49. FILEB(새로운 실제 파일)

		FLDB1		

FILEC는 FILEA의 레코드 형식을 공유합니다. FLDB1이 논리 파일의 레코드 형식에는 사용되지 않습니다.

표 50. FILEC(논리 파일)

FLDA	FLDB	FLDC	FLDD

실제 파일의 변경으로 인해 파일을 다시 작성해야 할 경우, 이를 삭제하고 새 실제 파일을 작성하기 전에 이를 참조하는 모든 논리 파일을 먼저 삭제해야 합니다. 실제 파일이 다시 작성된 후 이를 참조하는 논리 파일을 다시 작성하거나 복원할 수 있습니다.

예 1: 실제 파일 설명 및 속성 변경:

다음 예는 동일한 이름의 새로운 실제 파일을 다른 라이브러리에 작성하는 방법을 표시합니다.

1. 새로운 실제 파일을 다른 레코드 형식으로 이전 실제 파일이 있는 라이브러리가 아닌 다른 라이브러리에 작성합니다. 새 파일 이름은 이전 파일 이름과 같아야 합니다. (이전 실제 파일 FILEPFC는 라이브러리 LIBB에 들어 있으며, 두 개의 멤버 MBRC1과 MBRC2를 가집니다.)

   ```
   CRTPF FILE(NEWLIB/FILEPFC) MAXMBRS(2)...
   ```

2. 기존의 실제 파일의 멤버를 새로운 실제 파일에 복사합니다. TOMBR(*FROMMBR) 및 FROMMBR(*ALL)이 지정되었으므로 새로운 실제 파일의 멤버는 자동적으로 이전 실제 파일의 멤버와 같게 명명됩니다.

   ```
   CPYF      FROMFILE(LIBB/FILEPFC) TOFILE(NEWLIB/FILEPFC)
                      FROMMBR(*ALL) TOMBR(*FROMMBR)
                       FMTOPT(*MAP *DROP) MBROPT(*ADD)
   ```

3. 이전 논리 파일이 들어 있는 라이브러리가 아닌 다른 라이브러리에 새로운 논리 파일을 서술하고 작성합니다. 새 논리 파일 이름은 이전 논리 파일 이름과 같아야 합니다. 레코드 형식을 변경할 필요가 없는 경우에는 FORMAT 키워드를 사용하여 현재 논리 파일과 동일한 레코드 형식을 사용할 수 있습니다. CRTDUPOBJ(오브젝트 복사 작성) 명령을 사용하여 라이브러리 LIBB의 이전 논리 파일 FILELFC로부터 또 다른 논리 파일을 작성할 수도 있습니다.

   ```
   CRTLF FILE(NEWLIB/FILELFC)
   ```

4. 기존의 논리 파일 및 실제 파일을 삭제합니다.

   ```
   DLTF FILE(LIBB/FILELFC)
   DLTF FILE(LIBB/FILEPFC)
   ```

5. 다음 명령을 사용하여 새로 작성된 파일을 원래의 라이브러리로 옮깁니다.

```
MOVOBJ OBJ(NEWLIB/FILELFC) OBJTYPE(*FILE) TOLIB(LIBB)
MOVOBJ OBJ(NEWLIB/FILEPFC) OBJTYPE(*FILE) TOLIB(LIBB)
```

예 2: 실제 파일 설명 및 속성 변경:

다음 예는 동일한 라이브러리에 새 버전의 파일을 작성하는 방법을 표시합니다.

동일한 라이브러리에 새 버전의 파일을 작성하려면 다음 단계를 수행하십시오.

1. 새로운 실제 파일을 다른 레코드 형식으로 이전 실제 파일이 들어 있는 라이브러리에 작성합니다. 기존의 파일과 새로운 파일의 파일명은 서로 달라야 합니다. (기존의 실제 파일 FILEPFA는 라이브러리 LIBA에 들어 있으며, 두 개의 멤버 MBRA1과 MBRA2를 가집니다.)

   ```
   CRTPF FILE(LIBA/FILEPFB) MAXMBRS(2)...
   ```

2. 기존의 실제 파일의 멤버를 새로운 실제 파일에 복사합니다.

   ```
   CPYF     FROMFILE(LIBA/FILEPFA) TOFILE(LIBA/FILEPFB)
                    FROMMBR(*ALL) TOMBR(*FROMMBR)
                    FMTOPT(*MAP *DROP) MBROPT(*REPLACE)
   ```

3. 새로운 논리 파일을 기존의 논리 파일이 들어 있는 라이브러리에 작성합니다. 기존의 파일과 새로운 파일의 파일명은 서로 달라야 합니다. (레코드 형식을 변경할 필요가 없는 경우에는 FORMAT 키워드를 사용하여 현재 논리 파일과 동일한 레코드 형식을 사용할 수 있습니다.) PFILE 키워드는 1단계에서 작성한 새 실제 파일을 참조해야 합니다. 이전 논리 파일 FILELFA는 LIBA 라이브러리에 있습니다.

   ```
   CRTLF FILE(LIBA/FILELFB)
   ```

4. 기존의 논리 파일 및 실제 파일을 삭제합니다.

   ```
   DLTF FILE(LIBA/FILELFA)
   DLTF FILE(LIBA/FILEPFA)
   ```

5. 새로운 논리 파일을 기존의 논리 파일의 이름으로 재명명합니다. (실제 파일도 재명명하려는 경우에는 논리 파일에 대한 DDS를 변경하여 PFILE 키워드가 새로운 실제 파일명을 참조하도록 해야 합니다.)

   ```
   RNMOBJ(LIBA/FILELFB) OBJTYPE(*FILE) NEWOBJ(FILELFA)
   ```

6. 논리 파일 멤버의 이름을 변경해야 할 경우 CRTLF(논리 파일 작성) 명령에 디폴트가 사용되었다고 가정하면 다음 명령을 발행하십시오.

   ```
   RNMM FILE(LIBA/FILELFA) MBR(FILELFB) NEWMBR(FILELFA)
   ```

CHGPF(실제 파일 변경) 명령을 사용하여 실제 파일 및 그 멤버의 일부 속성을 변경할 수 있습니다.

관련 개념

제어 언어(CL)

논리 파일 설명 및 속성 변경

일반적으로, 레벨 ID에 변경을 가져올 논리 파일을 변경할 경우(예를 들어, 새 필드 추가, 필드 삭제 또는 필드 길이 변경 등) 해당 논리 파일을 사용하는 프로그램을 다시 컴파일해야 합니다.

그러나 레벨 ID에 변경을 가져올 논리 파일에 대한 변경인 경우에도(예를 들어, 파일의 끝에 프로그램에서 사용되지 않는 필드를 추가) 프로그램을 재컴파일하지 않아도 되는 경우가 있습니다. 그러나 이러한 경우에는 변

경된 파일을 사용하는 프로그램을 실행하기 위해 레벨 검사를 하지 않도록 해야 합니다. 이것은 바람직하지 않은 방식으로서, 추후 자료에 오류가 발생할 가능성이 높아집니다.

프로그램을 다시 컴파일하지 않으려면 현재 논리 파일(변경하지 않은)을 보유하고 추가된 필드를 가진 새로운 논리 파일을 작성하면 됩니다. 그러면 프로그램은 아직 존재하는 이전 파일을 참조하게 됩니다.

CHGLF(논리 파일 변경) 명령을 사용하면 CRTLF(논리 파일 작성) 명령에 지정된 논리 파일과 해당 멤버의 속성 대부분을 변경할 수 있습니다.

관련 참조

CHGLF(논리 파일 변경) 명령

논리 파일 작성(CRTLF) 명령

데이터베이스 회복 및 복원

이 주제에서는 시스템에서 자료가 손상된 후 데이터베이스 회복 또는 복원하는 iSeries 기능에 대해 설명합니다.

관련 개념

118 페이지의『보조 기억장치에 자료 및 액세스 경로 기록』

일반적으로 iSeries용 DB2 Universal Database는 변경된 자료가 주 기억장치에서 보조 기억장치로 기록되는 시점을 결정합니다. 그러나 데이터베이스 변경 내용이 보조 기억장치로 기록되는 시점을 제어할 수 있습니다.

백업 및 회복 PDF

백업 및 회복

데이터베이스 파일의 자료 회복

iSeries 시스템은 데이터베이스 파일에서 자료를 회복하는 데 저널링 및 확약 제어를 사용합니다.

저널 관리:

저널 관리를 사용하면 하나 이상의 데이터베이스 파일에서 발생하는 자료 변경 내용을 모두 기록할 수 있습니다. 그런 다음 저널을 사용해 회복할 수 있습니다.

데이터베이스 파일이 손상되어 사용할 수 없는 상태에서 저널이 있으면 파일의 거의 모든 활동을 재구성할 수 있습니다. 저널을 사용하여 파일에 대한 변경 내용을 제거할 수도 있습니다.

관련 개념

저널 관리

저널:

저널 사용 중 파일이 변경되면 변경 내용이 파일에 기록되기 전에 시스템이 저널 리시버에 변경 내용을 기록하고 리시버를 보조 기억장치에 기록합니다. 그러므로 저널 리시버는 항상 최신의 데이터베이스 정보를 가지고 있습니다.

저널은 특정 레코드 또는 전 파일에 대해 레코드 활동을 입력합니다. 각 항목에는 활동 소스(사용자, 작업, 프로그램, 시간 및 날짜 등)를 나타내는 제어 정보가 들어 있는 바이트들이 추가로 포함됩니다. 하나의 레코드에 영향을 미치는 변경일 경우 제어 정보 다음에 레코드 이미지가 포함됩니다. 변경 전의 레코드 이미지도 원한다면 포함시킬 수 있습니다. STRJRNPF(실제 파일 저널 시작) 명령에 IMAGES 매개변수를 지정하여, 변경 전후의 레코드 이미지를 모두 저널링할 것인지 아니면 변경 후의 레코드 이미지만을 저널링할 것인지를 제어할 수 있습니다.

저널되는 모든 데이터베이스 파일은 시스템이 시작(IPL 시)될 때 자동적으로 저널과 동기화됩니다. 시스템 세션이 이상 종료될 경우 일부 데이터베이스 변경 내용은 저널에 포함되지만 이 변경 내용의 일부는 데이터베이스 파일에 반영되지 않을 수 있습니다. 이 경우 시스템은 자동적으로 저널로부터 데이터베이스를 갱신합니다.

저널은 데이터베이스 파일을 신속하고 용이하게 저장하도록 합니다. 예를 들어, 전체 파일을 매일 저장하는 대신 이 파일에 대한 변경 내용을 포함하는 저널 리시버를 저장하십시오. 그러나 여전히 파일 전체를 매주 저장할 수 있습니다. 이러한 방법을 사용하면 매일 저장 조작을 수행하는 데 걸리는 시간을 감소시킬 수 있습니다.

관련 개념

저널 관리

저널에 대한 작업:

이 주제에서는 저널 작업에 사용할 수 있는 제어 언어(CL) 명령 및 iSeries Navigator 기능에 대해 설명합니다.

다음 CL 명령을 사용하여 저널에 대해 작업할 수 있습니다.

- 저널된 변경사항이 들어 있는 손상된 또는 사용할 수 없는 데이터베이스 파일 멤버를 회복하려면 APYJRNCHG(저널된 변경사항 적용) 및 RMVJRNCHG(저널된 변경사항 제거) 명령을 사용하십시오.
- 저널 리시버에 기록된 변경 내용을 지정 실제 파일 멤버에 적용하려면 APYJRNCHG 명령을 사용하십시오. 실제 파일에 어떤 손상이 발생했는지와 파일이 마지막으로 저장된 이후의 활동량에 따라 RMVJRNCHG 명령을 사용하여 파일에서 변경 내용을 제거하는 것이 더 편할 수도 있습니다.
- 저널 항목을 데이터베이스 파일로 전환하려면 DSPJRN(저널 표시) 명령을 사용하십시오. 이러한 파일은 활동 보고서, 감사 추적, 보안 및 프로그램 디버깅 등에 사용됩니다.

관련 개념

저널 관리

제어 언어(CL)

관련 참조

APYJRNCHG(저널 변경사항 적용) 명령

RMVJRNCHG(저널 변경사항 제거) 명령

저널 표시(DSPJRN) 명령

iSeries Navigator를 사용하여 저널 작성:

저널 작성은 사용자 시스템에서 저널의 새로운 인스턴스를 야기합니다. 저널링 정보를 시작하기 전에 저널에 대한 표 저널링을 시작해야 합니다.

표를 저널에 저널링하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 새 저널을 작성할 데이터베이스와 라이브러리를 확장하십시오.
4. 라이브러리 오브젝트를 마우스 오른쪽 버튼으로 클릭하고 **새 저널**을 클릭하십시오.
5. 새 저널 창에서 **이름** 필드에 이름을 지정하십시오.
6. **설명** 필드에 설명을 지정하십시오.
7. 저널 리시버를 저장할 라이브러리를 선택하십시오.

이제 디폴트 값을 사용하여 저널을 작성할 수 있습니다. **확장**을 클릭하여 저널 디폴트 값을 편집할 수 있습니다. 디폴트 값을 사용하여 저널을 작성하려면 **확인**을 클릭하십시오.

저널 디폴트 값을 편집하려면 우선 저널을 작성하고 다음 단계를 수행하십시오.

1. 새 저널 창에서 **확장**을 클릭하십시오.
2. 저널 메세지 대기행렬 필드에서 **저널 메세지 대기행렬**을 선택하십시오. 디폴트는 시스템 오퍼레이터 메세지 대기행렬입니다. 또다른 메세지 대기행렬을 지정할 수 있습니다. 그러나 메세지가 송신될 때 사용자가 지정한 메세지 대기행렬을 사용할 수 없는 경우 메세지는 시스템 오퍼레이터 메세지 대기행렬로 송신됩니다.
3. 저널 메세지 대기행렬을 저장할 라이브러리를 지정하십시오.
4. **설명** 필드에서 설명을 편집하거나 지정하십시오.
5. **관리할 리시버** 옵션을 선택하십시오. **시스템**이나 **사용자**를 선택할 수 있습니다.
6. 작업, 프로그램, 사용자 프로파일 정보를 기록하지 않으려면 **항목의 고정 부분 최소화**를 선택하십시오. 이렇게 하면 각 저널 항목의 크기를 최소화하지만 다른 저널 명령에서 사용할 수 있는 선택 기준을 제한시킵니다.
7. 시스템 재시작 회복에 필요한 내부 저널 항목만을 자동으로 제거하려는 경우 **내부 항목 제거**를 선택하십시오. 이 항목 제거는 저널 리시버의 크기를 축소시킵니다.
8. 새 저널 리시버는 저널과 동시에 작성됩니다. **새 리시버**를 클릭하여 리시버의 디폴트 값을 편집할 수 있습니다.
9. **확장** 옵션을 완료한 다음 **확인**을 클릭하여 새 저널 창으로 리턴하십시오.
10. **확인**을 클릭하여 저널을 작성하십시오.

iSeries Navigator를 사용하여 저널 리시버 작성:

저널 리시버는 저널이 기록 중인 정보를 포함하는 파일입니다. 저널을 작성하면 리시버가 자동으로 작성됩니다. 그러나 수동으로 리시버를 스왑하려는 경우 우선 새 저널 리시버를 작성해야 합니다.

iSeries Navigator를 사용하여 저널 리시버를 작성하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 리시버를 추가하려는 저널이 포함된 데이터베이스와 라이브러리를 확장하십시오.
4. 리시버를 추가할 저널을 마우스 오른쪽 버튼으로 클릭하고 **등록 정보**를 클릭하십시오.
5. **리시버**를 클릭하십시오.
6. 저널용 리시버... 창에서 **신규**를 클릭하십시오.
7. 이름(10자로 제한), 리시버를 포함할 라이브러리, 설명 및 기억장치 공간 임계값을 지정하십시오.
8. **확인**을 클릭하여 새 저널 리시버 창을 닫으십시오.
9. **확인**을 클릭하여 저널용 리시버... 창을 닫으십시오.
10. **확인**을 클릭하여 저널 등록 정보 창을 닫으십시오.

새 저널을 작성하는 경우 다음 단계를 수행하여 새 리시버도 작성할 수 있습니다.

1. 확장 저널 속성 창에서 **새 리시버**를 클릭하십시오.
2. 새 저널 리시버 창에서 이름(10자로 제한), 리시버를 포함할 라이브러리, 설명 및 기억장치 공간 임계값을 지정하십시오.
3. **확인**을 클릭하여 새 저널 리시버 창을 닫으십시오.
4. **확인**을 클릭하여 확장 저널 속성 창을 닫으십시오. 저널 리시버에 대한 값을 지정하지 않은 경우 디폴트 값으로 작성됩니다.

새 저널 및 새 저널 리시버 작성 시 사용된 값:

저널 및 저널 리시버는 확장 저널 속성 또는 저널 등록정보 창에 지정한 값을 사용하여 작성되거나 변경됩니다. 임의 값을 지정하지 않은 경우 저널 및 저널 리시버는 디폴트 값을 사용하여 작성됩니다.

저널의 경우:

- 저널은 초점을 가진 라이브러리에서 작성됩니다.
- 저널에 대한 기억장치 공간은 저널 라이브러리의 ASP의 기억장치 공간과 동일한 보조 기억장치 풀(ASP)에서 할당됩니다. 이 값은 변경될 수 없습니다.
- 저널과 연관된 메세지 대기행렬은 시스템 오퍼레이터 메세지 대기행렬입니다.
- 저널 리시버의 스와핑이 설정되어 시스템이 자동으로 스와핑을 수행합니다.
- 저널 리시버는 스왑 처리 중 시스템에 의해 자동으로 삭제되지 않습니다.
- 저널 항목의 고정 부분은 최소화되지 않는 반면에 내부 저널 항목은 제거됩니다.
- 저널에 대한 공용 권한은 라이브러리에 대한 디폴트 공용 권한에서 얻습니다.
- 저널에 대한 디폴트 텍스트 설명이 작성되지 않습니다.

저널 리시버의 경우:

- 저널 리시버에 대한 기억장치 공간은 저널 리시버의 라이브러리의 ASP의 기억장치 공간과 동일한 보조 기억장치 풀(ASP)에서 할당됩니다. 이 값은 변경될 수 없습니다.
- 저널 리시버에 대한 기억장치 공간 임계값은 500MB로 설정되어 있습니다.
- 저널 리시버에 대한 공용 권한은 라이브러리에 대한 디폴트 공용 권한에서 얻습니다.
- 디폴트 텍스트 설명은 저널 리시버용으로 작성됩니다.

iSeries Navigator를 사용하여 리모트 저널 추가:

리모트 저널은 저널 정보를 각 시스템으로 복제하도록 허용합니다. 리모트 저널은 기존 저널과 연관됩니다. 소스 시스템에서의 저널은 로컬 저널 또는 또다른 리모트 저널 중 하나일 수 있습니다.

iSeries Navigator를 사용하여 리모트 저널을 추가하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 **라이브러리**를 확장하십시오.
4. 리모트 저널을 추가하려는 저널이 들어 있는 라이브러리를 클릭하십시오.
5. 리모트 저널을 추가하려는 저널을 마우스 오른쪽 버튼으로 클릭하고 **등록 정보**를 클릭하십시오.
6. 저널 등록 정보 창에서 **리모트 저널**을 클릭하십시오.
7. 리모트 저널을 이 저널에 추가(연관)하려면 **추가**를 클릭하십시오.
8. 관계형 데이터베이스(RDB) 디렉토리 항목의 리스트를 표시하려면 리모트 저널 추가 창의 관계형 데이터베이스명 상자에서 아래 화살표를 클릭하십시오.
9. 저널 유형을 선택하십시오(유형 1 또는 유형 2).
10. 소스 시스템에서 사용되는 다른 라이브러리로 목표 시스템의 리모트 저널 리시버를 연관시키려면 **리시버 경로 재지정**을 클릭하십시오.
11. 목표 리시버 라이브러리 필드에서 리모트 저널 리시버가 위치되는 목표 시스템에 라이브러리를 지정하십시오.
12. **확인**을 클릭하십시오.

리모트 저널 유형은 경로 재지정 기능, 저널 리시버 복원 작업 및 리모트 저널 연관 특성에 영향을 미칩니다.

제한된 경로 재지정(유형 1): 추가 중인 리모트 저널이 유형 1이면 저널 라이브러리명을 로컬 저널의 라이브러리에서 다른 단일 라이브러리로 재지정할 수 있습니다. 주어진 로컬 저널과 연관된 모든 유형 1 리모트 저널은 같은 라이브러리에 있어야 합니다. 유형 1 리모트 저널을 유형 2 리모트 저널에 추가할 수 없습니다.

유연한 경로 재지정(유형 2): 추가 중인 리모트 저널이 유형 2이면 유형 2 리모트 저널의 이전 추가에 지정되어 있는 재지정 라이브러리와 같은 재지정 라이브러리나 다른 경로 재지정 라이브러리를 지정할 수 있습니다. 나중에 유형 2 리모트 저널을 추가하면 이전에 추가한 리모트 저널에 지정된 것과 다른 라이브러리 경로 재지정을 지정할 수 있습니다.

리모트 저널을 추가한 다음 이를 활성화해야 합니다.

주: 이 타스크는 RDB 디렉토리가 있으며 목표 시스템에 사용자 프로파일이 있는 것으로 가정한 것입니다.

iSeries Navigator를 사용하여 리모트 저널 제거:

리모트 저널을 제거하면 소스 시스템에 있는 저널에서 해당 리모트 저널이 연관해제되고, 저널이나 저널 리시버가 삭제되지는 않습니다. 제거하기 전에 리모트 저널을 비활성화해야 합니다.

iSeries Navigator를 사용하여 리모트 저널 활성화:

리모트 저널을 추가한 다음 이를 활성화해야 합니다.

리모트 저널을 활성화하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 **라이브러리**를 확장하십시오.
4. 활성화하려는 연관된 리모트 저널이 들어 있는 라이브러리를 클릭하십시오.
5. 저널을 마우스 오른쪽 버튼으로 클릭하고, **등록 정보**를 선택하십시오
6. 저널 등록 정보 창에서 **리모트 저널**을 클릭하십시오.
7. 창의 리모트 저널에 있는 리모트 저널 리스트에서 해당 리모트 저널을 선택한 다음 **활성화**를 클릭하여 선택한 리모트 저널을 활성화하십시오.
8. 활성화 창에서 전달 모드, 시작 리시버 및 활성화 요청에 대한 처리 모드를 선택하십시오.
9. **확인**을 클릭하십시오.

주:

1. 처음 리모트 저널을 활성화하면 목표 시스템에 하나 이상의 저널 리시버를 작성합니다.
2. 리모트 저널을 활성화하면 소스 및 리모트 저널 간에 연결을 설정하여 저널 항목 복제를 시작할 수 있습니다.
3. 리모트 저널은 활성화되기 전에 비활동 상태이어야 합니다. 또한 활성화하고 있는 리모트 저널은 저널 항목을 다른 리모트 저널로 이미 복제 중이지 않아야 합니다.

iSeries Navigator를 사용하여 리모트 저널 비활성화:

리모트 저널을 비활성화할 경우 자료 콜렉션이 중단됩니다.

동기적으로 유지보수된 저널을 비활성하려는 경우 즉시 또는 제어 종료가 요청되는지 여부에 관계없이 리모트 저널 기능이 종료됩니다. 리모트 저널이 회복 처리 단계에 있는 경우 즉시 또는 제어 종료가 요청되는지 여부에 관계없이 리모트 저널 기능은 즉시 종료됩니다. (회복 처리는 리모트 저널이 활성화되기 전에 소스 저널의 저널 리시버에 존재하는 저널 항목을 복제하는 프로세스를 나타냅니다.) 리모트 저널이 비동기 지연이나 동기 지연의 전달 모드를 갖는 경우 리모트 저널은 회복 상태입니다.

제어

제어된 방식으로 리모트 저널 기능을 비활성화합니다. 이것은 리모트 저널을 비활성화하기 전에 소스 시스

템에서 목표 시스템으로 송신되도록 이미 대기행렬로 보낸 모든 저널 항목을 복제해야 한다는 의미입니다. 대기행렬로 보낸 모든 항목이 복제될 때까지 리모트 저널은 비활동 지연 상태에 있게 됩니다. 비활동 지연 상태에 있는 동안 리모트 저널 창은 비활동 지연 정보를 제공합니다. 대기행렬로 보낸 모든 항목이 목표 시스템으로 송신되면 시스템은 저널 메세지 대기행렬로 메세지를 송신하며, 리모트 저널 기능이 종료되었음을 나타냅니다.

즉시

리모트 저널 기능을 즉시 비활성화합니다. 이것은 리모트 저널을 비활성화하기 전에는 이미 대기행렬로 보낸 저널 항목을 시스템이 계속 복제해서는 안된다는 것을 의미합니다.

iSeries Navigator를 사용하여 표의 저널 정보 표시:

이 주제는 iSeries Navigator를 사용하여 표에 저널 정보를 표시하는 프로시듀어를 포함합니다.

표에 저널 정보를 표시하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 **라이브러리**를 확장하십시오.
4. 저널 정보를 표시하려는 표가 들어 있는 라이브러리를 클릭하십시오.
5. 표를 마우스 오른쪽 버튼을 클릭하고 **저널링**을 선택하십시오.
6. 표가 저널된 적이 없는 경우 저널 및 라이브러리명을 해당 상자에 입력하거나 **찾아보기** 버튼을 클릭하고 표에서 사용하려는 저널의 위치를 검색하여 사용하려는 저널을 선택할 수 있습니다.
7. 사전 이미지를 저널하려면 **변경 전 저널 이미지** 옵션을 선택하십시오.
8. 저널 중인 열기 및 닫기 항목을 생략하려면 **열기 및 닫기 항목 제외** 옵션을 선택하십시오.

iSeries Navigator를 사용하여 리시버 스와핑:

스와핑 저널 리시버는 시스템에 의해 자동으로 작성되고 저널에 접속된 새 저널 리시버로 현재 저널 리시버를 대체합니다. 저널링할 리시버를 스왑하기 위해 사용할 수 있는 두 가지 방법이 있습니다. 첫 번째 방법으로는 새 리시버의 속성을 변경할 수 없지만 두 번째 방법으로는 가능합니다.

첫 번째 방법을 선택할 경우 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 **라이브러리**를 확장하십시오.
4. 사용하려는 저널을 마우스 오른쪽 버튼으로 클릭하고 **리시버 스왑**을 클릭하십시오.

   이 옵션의 경우 시스템에서 리시버 작성 시 새 이름이 생성되며, 사용자는 새 리시버의 속성을 변경할 수 *없습니다.*

두 번째 방법을 선택할 경우 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업할 데이터베이스와 **라이브러리**를 확장하십시오.
4. 사용하려는 저널을 두 번 클릭하십시오.
5. **리시버**를 클릭하십시오. 이 창은 저널과 연관된 모든 리시버를 표시합니다.
6. 새 리시버를 추가하려면 **신규**를 클릭하십시오.
7. **확인**을 클릭하여 저널 리시버 창을 닫으십시오. 다시 **확인**을 클릭하면 신규 저널 리시버가 그 상태를 접속됨으로 변경합니다.

   이 옵션의 경우 사용자는 새 리시버의 속성을 변경할 수 없습니다.

iSeries Navigator를 사용하여 표에 대한 저널링 시작/중지:

저널을 작성한 다음에는 표에서 저널링을 시작해야 합니다. 저널을 삭제하려면 중단시키십시오.

iSeries Navigator를 사용하여 표에 대한 저널링을 시작/중지하려면 다음 단계를 수행하십시오.

1. iSeries Navigator에서 사용할 시스템을 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 편집하려는 저널이 포함된 데이터베이스와 라이브러리를 확장하십시오.
4. 편집하려는 저널이 있는 라이브러리를 클릭하십시오.
5. 저널을 마우스 오른쪽 버튼으로 클릭하고 **표 저널링 시작 및 종료**를 클릭하십시오.
6. 표(파일)의 저널링을 시작하려면 **표** 리스트에서 저널하려는 표를 선택한 다음 **추가**를 클릭하십시오또는 **표** 리스트에서 표를 끌기 조작하여 **저널할 표** 리스트에 놓을 수 있습니다.
7. 표의 저널링을 끝내려면 **이미 저널 중인 표** 리스트에서 더 이상 저널하지 않으려는 표를 선택한 후 **제거**를 클릭하십시오.
8. 모든 표의 저널링을 즉시 끝내려면 **이미 저널링 중인 표** 리스트에 나열된 모든 표를 선택하기 위해 **모두 선택**을 클릭한 후 **제거**를 클릭하십시오.
9. **확인**을 클릭하여 저널링 시작/종료 창을 닫으십시오.

다음 단계를 수행하여 표나 파일에 대한 저널링을 시작/중지할 수 있습니다.

1. 라이브러리 트리 리스트에서 저널링을 시작하거나 중단하려는 오브젝트를 마우스 오른쪽 버튼으로 클릭하고 **저널링**을 클릭하십시오.
2. 선택된 오브젝트에 대한 저널링을 중단하려면 **중단**을 클릭하십시오.
3. 오브젝트에서 저널링을 시작하려면 다음과 같이 하십시오.
   a. 오브젝트와 연관시킬 저널을 선택하십시오. **찾아보기**를 클릭하여 저널을 찾아보기할 수 있습니다.
   b. 저널이 위치한 라이브러리를 선택하십시오. **찾아보기**에서 저널을 선택하면 이 필드는 자동으로 채워집니다.
   c. 사전 이미지를 저널하려면 **변경 전 저널 이미지** 옵션을 선택하십시오.

d. 저널 중인 열기 및 닫기 항목을 생략하려면 **열기 및 닫기 항목 제외** 옵션을 선택하십시오.

e. 선택된 오브젝트에 대한 저널링을 시작하려면 **시작**을 클릭하십시오.

4. **확인**을 클릭하여 확장 저널 속성 창을 닫으십시오. 저널 리시버에 대한 값을 지정하지 않은 경우 디폴트 값으로 작성됩니다.

확약 제어로 자료 무결성 확인:

확약 제어는 데이터베이스 파일에 대한 일련의 변경 내용을 하나의 단위(트랜잭션)로 정의하여 처리할 수 있습니다. 확약 제어는 작업이나 시스템이 종료해도 복잡한 어플리케이션 트랜잭션이 논리적으로 동기화되는지 확인합니다. 2단계 확약 제어는 데이터베이스 파일과 같이 다중 시스템에서 확약될 수 있는 자원이 동기화 상태로 남아 있도록 보장해줍니다.

확약 및 롤백 조작을 실행하여 데이터베이스에서 확약 제어를 구현합니다. SQL을 사용하여 COMMIT 및 ROLLBACK 명령문을 사용할 수 있습니다.

관련 개념

확약 제어

관련 참조

COMMIT

ROLLBACK

트랜잭션:

트랜잭션은 단일 변경 내용으로 표시되는 일련의 변경 내용(예: 정기 예금에서 당좌 예금으로 예금 전환)입니다.

트랜잭션은 다음과 같이 분류됩니다.

- 조회: 파일이 변경되지 않습니다.
- 단순 처리: Enter 키를 누를 때마다 파일이 하나씩 변경됩니다.
- 복합 처리: Enter 키를 누를 때마다 두 개 이상의 파일이 변경됩니다.
- 복합 처리: Enter 키를 누를 때마다 한 개 이상의 파일이 변경됩니다. 이러한 변경은 트랜잭션의 논리적 그룹의 일부만을 나타냅니다.

트랜잭션 처리중 파일에 대한 변경 내용은 확약 제어 사용 시 저널됩니다.

시스템이나 작업이 이상 종료되면 저널링 자체만으로는 최종 레코드 변경 내용만이 손실되었음을 확인할 수 있습니다. 그러나 복합 트랜잭션 동안에 시스템이나 작업이 이상 종료되면 파일은 불완전한 논리 트랜잭션을 반영합니다. 예를 들면, 어떤 작업이 파일 A에 있는 레코드는 갱신하였으나 파일 B에서 대응되는 레코드를 변경하기 전에 작업이 이상 종료될 경우입니다. 이런 경우 논리 트랜잭션은 두 개의 갱신으로 구성되지만 작업이 이상 종료되기 전 갱신 한 개만 완료합니다.

확약 제어 사용의 이점:

복합 어플리케이션을 회복하기 위해서는 어플리케이션에 대한 보다 자세한 지식이 필요합니다. 프로그램은 재시작될 수 없습니다. 이 경우 확약 제어가 이러한 문제점을 해결하는 데 도움이 됩니다.

최종 복합 트랜잭션이 시작되기 직전의 상태로 파일을 되돌려 놓기 위해 어플리케이션 프로그램이나 자료 파일 유틸리티로 레코드 변경을 수행해야 할 경우가 있습니다. 동시에 여러 사용자가 파일에 액세스하는 경우 이러한 작업은 더 복잡해 집니다. 이런 경우 확약 제어가 도움이 됩니다. 확약 제어는 복합 트랜잭션 동안 사용자로부터 레코드를 잠급니다. 이렇게 함으로써 트랜잭션이 완료될 때까지 레코드를 다른 사용자들이 사용할 수 없도록 합니다. 트랜잭션의 끝에서 프로그램이 확약 조작을 발행하여 잠겼던 레코드를 해제합니다. 그러나 확약 조작이 수행되기 전에 시스템이 이상 종료되면 마지막으로 확약 조작을 수행한 다음 해당 작업에 대한 모든 레코드 변경 내용은 롤백됩니다. 또한 계속 잠겨 있도록 영향을 받은 모든 레코드가 해제됩니다. 다시 말해, 데이터베이스에 대한 변경 내용이 명확한 처리 경계로 롤백하는 것입니다.

사용상 주의사항: 확약 제어:

확약 및 롤백 조작은 여러 가지 iSeries 프로그래밍 언어(RPG/400, COBOL/400, PL/I, SQL 및 i5/OS 제어 언어(CL) 포함)로 사용할 수 있습니다. 기본이 되는 실제 파일이 다른 저널로 저널링될 때 확약 제어 하에 출력을 위해 논리 파일을 열 수 있습니다. 확약 제어는 일괄처리 환경에서도 사용될 수 있습니다.

그러나 레코드 변경이 같은 저널로 저널링되는 기본 실제 파일에 영향을 미칠 경우 위반에 대한 체크가 지연됩니다. 레코드 변경이 같은 저널로 저널링되지 않은 기본 실제 파일에 영향을 미칠 경우와 이로 인해 중복 키 또는 참조 제한조건 위반이 발생할 경우 입/출력 조작 중에 오류가 발생합니다. 예를 들어, 고유 키가 있는 실제 파일 A가 저널 X로 저널링되고 고유 키가 있는 실제 파일 B가 저널 Y로 저널링되는 것으로 가정하십시오. 논리 파일 C가 실제 파일 A와 B에서 작성되어 확약 제어 하에 열립니다. 논리 파일 C를 사용하여 수행된 삭제 조작은 키 K가 있는 실제 파일 A에서 레코드를 제거합니다. 트랜잭션이 확약되기 전에 키 K를 사용하여 실제 파일 A로 레코드를 다시 추가할 수 있습니다. 그러나 트랜잭션이 확약되기 전에 키 K가 있는 실제 파일 B에 레코드를 추가하려고 하면 실패합니다. 실제 파일 A와 B는 서로 다른 저널에 저널링되기 때문입니다.

대화식 트랜잭션 회복에서 도움이 되는 것과 마찬가지로 일괄처리 작업 회복에서도 확약 제어가 도움이 됩니다.

관련 개념

확약 제어

액세스 경로 회복 시 시간 단축

사용자가 액세스 경로를 사용할 수 있으려면 시스템이 먼저 그 무결성을 확인해야 합니다. 시스템이 액세스 경로를 사용할 수 없다고 판단하면 시스템은 액세스 경로를 회복하려 합니다. 액세스 경로가 회복되는 시기를 제어할 수 있습니다.

특히 재구성해야 할 액세스 경로가 많거나 액세스 경로가 큰 경우에는 액세스 경로를 회복하는 데 많은 시간이 필요할 수도 있으며, 여러 가지 방법으로 이 회복 시간을 줄일 수 있습니다.

저널링 액세스 경로는 재구축 액세스 경로보다 가끔 빠릅니다. SMAPP(시스템 관리 액세스 경로 보호) 지원을 사용할 경우 액세스 경로 저널링을 활용하기 위해 STRJRNAP(액세스 경로 저널링 시작) 명령과 같은 저널링 명령을 사용할 필요가 없습니다. SMAPP 지원은 IPL 중에 액세스 경로를 재구성하기보다는 시스템 이상 종료 후에 액세스 경로를 회복합니다.

액세스 경로 저장:

액세스 경로를 저장하여 액세스 경로 회복 시간을 줄일 수 있습니다. SAVCHGOBJ(변경된 오브젝트 저장), SAVLIB(라이브러리 저장) 및 SAVOBJ(오브젝트 저장) 명령에서 ACCPTH(액세스 경로) 매개변수를 사용하면 액세스 경로를 저장할 수 있습니다.

대개, 시스템은 논리 파일에 대한 설명만 저장하며 액세스 경로를 다음 조건하에서만 저장합니다.

- ACCPTH(*YES)가 지정됨.
- 논리 파일 하의 모든 실제 파일이 저장되어 동일한 라이브러리에 있음.
- 논리 파일이 MAINT(*IMMED) 또는 MAINT(*DLY)임.

주:

ACCPTH(*YES) 매개변수를 지정하면 논리 파일 자체는 저장되지 않습니다. 논리 파일은 명시적으로 저장해야 합니다.

관련 개념

백업 및 회복 PDF

액세스 경로 복원:

시스템에는 액세스 경로 복원 기능이 있으며 일반적으로 액세스 경로를 재구성하는 것보다 복원하는 것이 더 빠릅니다.

시스템은 다음 경우에 액세스 경로를 복원할 수 있습니다.

- 액세스 경로가 이미 저장된 경우
- 기초를 두고 있는 모든 실제 파일이 동시에 복원되는 경우 액세스 경로를 복원할 수 있습니다.

예를 들어, 논리 파일이 레코드 500 000개를 포함하는 실제 파일에 빌드되었다고 가정합니다. 논리 파일의 크기가 약 15MB인 DSPOBJD(오브젝트 설명 표시) 명령을 통해 판별하였습니다.

이 때 논리 파일에 대한 액세스 경로를 재구성하는 데에는 약 50분 정도가 소요됩니다. 테이프에서 같은 액세스 경로를 복원하는 데 약 1분 정도가 걸립니다. (이것은 시스템이 분당 약 10 000개의 색인 항목을 빌드하는 것으로 가정합니다.)

액세스 경로를 복원한 다음에는 최신 저널 변경 내용을 적용하여 파일을 갱신해야 할 경우도 있습니다. 예를 들어, 시스템은 시간당 약 80 000개에서 100 000개의 저널 항목을 파일에 적용합니다. 이것은 항목이 적용되는 실제 파일 각각에 하나의 액세스 경로만이 작성되어 있다고 가정합니다. 이 비율은 실제 파일에 존재하는

*IMMED 유지보수의 각 액세스 경로에 비례하여 낮아집니다. 이렇게 회복 시간이 추가로 걸린다 해도 액세스 경로를 재구성하는 것보다는 액세스 경로를 복원하는 것이 훨씬 신속함을 알 수 있습니다.

관련 개념

저널 관리

액세스 경로 저널:

액세스 경로를 저널링하면 시스템이 이상 종료된 후 재구성해야 하는 액세스 경로의 수를 줄임으로써 회복 시간을 대폭 줄일 수 있습니다. 액세스 경로의 *크기*가 클수록 재구성하는 데 시간이 더 걸리므로 액세스 경로를 저널링하는 것이 좋습니다.

데이터베이스 파일을 저널하면 사용자는 파일의 레코드에 대한 변경 이미지를 저널에 기록합니다. 시스템은 이 레코드 이미지를 사용하여 이상 시스템 종료후에 파일을 회복할 수 있습니다.

비정상 종료 후 시스템에서 액세스 경로가 파일의 자료와 동기화되지 않은 것을 발견할 수도 있습니다. 액세스 경로가 자료로 동기화되어 있지 않으면 시스템은 이들 두 개가 동기화되어 사용 가능하도록 시스템이 액세스 경로를 재구성합니다.

액세스 경로가 저널되면 시스템이 액세스 경로와 그 자료 사이의 동기점을 제공하기 위해 액세스 경로의 이미지를 저널에 기록합니다. 이런 정보가 저널에 있으면, 시스템은 데이터베이스 파일과 액세스 경로를 모두 회복할 수 있습니다. 시스템은 둘을 동기화합니다. 이 경우 사용자는 액세스 경로 재구성에 오랜 시간이 소요되지 않도록 합니다.

추가적으로, 다른 시스템 회복 기능은 액세스 경로 저널링으로 작업합니다. 예를 들어 시스템에는 디스크 장치의 장애와 교체로부터 회복하는 데 필요한 시간을 줄이기 위한 여러 가지 옵션이 있습니다. 사용자 보조 기억 장치 풀 및 체크섬 보호 등이 이 옵션에 해당됩니다. 이 옵션은 전 시스템이 디스크 실패로 인해 재로드해야 하는 기회를 줄여줍니다. 그러나 시스템이 다음 실패 디스크로 대체를 시작한 경우 사용자는 여전히 액세스 경로를 재구축할 필요가 있습니다. 액세스 경로 저널링 및 일부 회복 옵션을 사용하면 시스템 전체를 다시 로드하거나 액세스 경로를 재구성해야 할 필요성을 덜 수 있습니다.

액세스 경로 저널링에 앞서, 해당 액세스 경로와 연관된 실제 파일을 저널해야 합니다. 또한 액세스 경로와 연관된 실제 파일에 대해서 동일한 저널을 사용해야 합니다. 액세스 경로 저널링 시작은 쉽습니다.

- 시스템 관리 액세스 경로 보호(SMAPP) 기능을 사용할 수 있습니다.
- STRJRNAP(액세스 경로 저널링 시작) 명령으로 자체 저널링 환경을 관리할 수 있습니다.
  - 지정된 파일에 대한 액세스 경로 저널링을 시작하기 위해서는 STRJRNAP 명령을 사용하십시오. 즉시(*IMMED) 또는 지연(*DLY) 유지보수 속성을 가진 액세스 경로를 저널할 수 있습니다.
  - 일단 저널링이 시작하면 액세스 경로가 삭제될 때까지 또는 해당 액세스 경로에 대한 ENDJRNAP(액세스 경로 저널링 종료) 명령이 실행될 때까지는 시스템은 액세스 경로를 보호합니다.

액세스 경로 저널링은 추가 출력 조작을 최소화합니다. 예를 들어, 시스템은 동일한 출력 조작에서 변경된 레코드와 변경된 액세스 경로의 저널 자료를 기록하게 됩니다. 그러나 사용자의 액세스 경로 저널링이 시작될 때

저널 리시버를 사용자 보조 기억장치 풀에 분리해 두는 것을 고려해야 할 것입니다. 저널 리시버를 각자의 사용자 보조 기억장치 풀에 넣어두면, 디스크 고장으로부터 이들을 보호하며 최고의 저널링 성능을 가져올 수 있습니다.

관련 개념

저널 관리

관련 참조

STRJRNAP(저널 액세스 경로 시작) 명령

시스템 관리 액세스 경로 보호:

시스템 관리 액세스 경로 보호(SMAPP)는 액세스 경로에 대한 자동 보호 기능을 제공합니다. SMAPP 지원을 사용할 경우 액세스 경로 저널링을 활용하기 위해 STRJRNAP(액세스 경로 저널링 시작) 명령과 같은 저널링 명령을 사용할 필요가 없습니다.

SMAPP 지원은 IPL 중에 액세스 경로를 재구성하기보다는 시스템 이상 종료 후에 액세스 경로를 회복합니다. 보내진 시스템은 자동적으로 SMAPP 지원을 켭니다. 시스템은 SMAPP 지원 값을 70분으로 설정합니다.

시스템은 기본 보호를 위해 액세스 경로를 판별합니다.

- 사용자에 위해 제공된 목표 액세스 경로 회복 시간 또는
- 시스템에서 제공되는 디폴트 시간

목표 액세스 경로 회복 시간을 시스템 전체에 적용되는 값 또는 보조 기억장치 풀(ASP) 기준으로 지정할 수 있습니다. 사용자 정의 저널의 액세스 경로는 이미 보호되어 있으므로 SMAPP 보호에 적합하지 않습니다.

관련 개념

저널 관리

액세스 경로 리빌드:

데이터베이스 액세스 경로를 리빌드하는 데는 10 000 레코드당 약 1분이 소요됩니다. 그러나 일부 요소가 액세스 경로 리빌드 예상 시간에 영향을 줄 수 있습니다.

다음 요소는 액세스 경로 리빌드 예산 시간에 영향을 줍니다.

- 기억장치 풀 크기. 대형 기억장치 풀에서 작업을 수행함으로써 리빌드 시간을 향상시킬 수 있습니다.
- 시스템 모델. 처리 장치의 속도는 주요 요인입니다.
- 키 길이. 키 길이가 길어지면 액세스 경로가 주요 정보를 더 많이 구성하고 저장해야 하므로 액세스 경로 리빌드 시간이 길어집니다.
- 선택/생략값. 시스템이 각 레코드가 선택/생략값에 맞는지를 비교해야 하므로 선택/생략 처리로 액세스 경로의 재구성 시간이 늦어집니다.
- 레코드 길이. 레코드 길이가 길면 시스템이 더 많은 자료를 검토하므로 액세스 경로의 재구성 시간이 늦어집니다.

- 자료를 포함하는 기억장치 장치. 실제 자료가 들어 있는 기억장치와 액세스 경로가 저장되는 장치의 상대적인 속도는 액세스 경로를 재구성하는 데 소요되는 시간에 영향을 미칩니다.
- 파일 내의 레코드 순서. 시스템은 액세스 경로를 사용하여 보다 신속하게 정보를 찾을 수 있도록 액세스 경로를 재구성하려고 합니다. 파일에 있는 레코드의 순서는 액세스 경로를 효율적으로 유지보수하기 위해 시도하는 동안 시스템이 액세스 경로를 얼마나 신속하게 구축할 수 있는지에 영향을 미칩니다.

액세스 경로 재구성 시점 제어:

시스템이 비정상적으로 종료되면 다음 초기 프로그램 로드(IPL) 시 시스템에서 액세스 경로 회복이 필요한 파일을 자동으로 나열합니다. 액세스 경로 재구성 시점을 제어할 수 있습니다.

액세스 경로 재구성 여부를 결정할 수 있습니다.

- IPL 중
- IPL 후
- 파일을 처음 사용하는 경우

사용자는 또한 다음 사항을 결정할 수도 있습니다.

- 액세스 경로가 재구성되는 스케줄링 순서 변경
- 무한정으로 액세스 경로의 재구성 보류
- *IPL 임계값(threshold)*과 같거나 작은 순서 값을 가진 액세스 경로가 재구성되는 동안 IPL 처리를 계속
- EDTRBDAP(액세스 경로 재구성 편집) 명령을 사용하여 시스템에서 IPL 처리가 완료된 후 액세스 경로 재구성 제어

IPL 임계값은 어느 액세스 경로가 IPL 중에 재구성되는지를 판별합니다. IPL 임계값과 같거나 작은 순서값을 가진 모든 액세스 경로는 IPL 중에 재구성됩니다. IPL 임계값을 99로 변경하는 것은 1-99의 순서 값을 가진 모든 액세스 경로가 IPL 중에 재구성됨을 뜻합니다. IPL 임계값을 0으로 변경하는 것은 시스템이 그 자체의 IPL을 완료한 후에도 저널되고 있는 액세스 경로와 시스템 파일에 대한 액세스 경로를 제외하고는 재구성되는 액세스 경로가 없음을 의미합니다.

파일에 대한 액세스 경로 회복 값은 파일 작성 및 파일 변경 명령의 RECOVER 매개변수에서 지정된 값에 의해 결정됩니다. *IPL(IPL 중 재구성)에 대한 디폴트 회복 값은 25이고, AFTIPL(IPL 후 재구성)에 대한 디폴트 값은 75입니다. 따라서, RECOVER(*IPL)은 25로 표시됩니다. 초기의 IPL 임계값은 50이며, 이는 액세스 경로 재구성 시 매개변수가 영향을 미치도록 합니다. 액세스 경로의 재구성 편집 화면에서 이 값을 대체할 수 있습니다.

IPL 직후에 재구성될 필요가 없는 파일에 대해서는 후에 파일이 재구성되도록 지정하십시오. 시스템이 IPL을 보다 신속히 완료할 수 있도록 이는 IPL시 재구성되어야 하는 파일의 수를 감소시켜야 합니다.

예를 들어, 액세스 경로가 재구성되어야 하는 모든 파일에 대해 파일이 처음 사용될 때 액세스 경로가 재구성되도록 지정할 수 있습니다. 이 경우 IPL시에는 어떤 액세스 경로도 재구성되지 않습니다. 먼저 재구성하려는 파일을 사용하는 프로그램들만을 실행함으로써 액세스 경로의 재구성 순서를 제어할 수 있습니다. 이 방법은 IPL 시간을 줄이고 사용 가능한 몇 가지 어플리케이션 중 첫 번째를 더 빠르게 만들어 줍니다. 그러나 액세스

경로 재구성 전체 시간은 더 길 수 있습니다. (액세스 경로가 재구성되는 동안 다른 작업이 실행될 수 있으며, 액세스 경로를 재구성하는 데 사용할 수 있는 주 기억장치가 줄어들 수 있습니다.)

관련 참조

EDTRBDAP(액세스 경로 리빌드 편집) 명령

액세스 경로 재구성 시간을 줄이기 위한 파일 설계:

파일 설계에 따라 액세스 경로 회복 시간이 줄어들 수 있습니다.

예를 들어 대형의 마스터 파일을 이력 파일과 트랜잭션 파일 두 개로 분리할 수 있습니다. 시스템은 새로운 자료 추가를 위해 트랜잭션 파일을 사용합니다. 시스템은 조회만을 위해 이력 파일을 사용합니다. 매일 트랜잭션 자료를 이력 파일과 통합한 후, 다음날의 자료를 위해 트랜잭션 파일을 비웁니다. 이러한 설계로 액세스 경로를 재구성하는 데 소요되는 시간을 줄일 수 있습니다.

그러나 시스템이 이상 종료되는 경우 작은 트랜잭션 파일에 대한 액세스 경로를 재구성해야할 필요가 있습니다. 그러나 대부분 읽기 전용으로만 사용되는 큰 이력 파일의 액세스 경로는 그 자료와 재구성되어야 할 필요성이 훨씬 적을 것입니다. 그러므로 이 액세스 경로를 감소할 필요를 줄일 수 있습니다.

액세스 경로 재구성 시간을 줄이기 위한 파일 설계를 사용할 것인지, 액세스 경로 저널링과 같은 시스템 제공 기능을 사용할 것인지를 결정할 때는 양쪽의 장단점을 충분히 고려해야 합니다. 위 파일 설계는 더 복잡한 설계가 필요합니다. 상황에 따라서 더 복잡한 어플리케이션을 설계하는 것보다는 액세스 경로 저널링과 같은 시스템 제공 기능을 사용하는 것이 더 나은 방법일 수도 있습니다.

액세스 경로를 재구성하지 않기 위한 기타 방법:

액세스 경로를 저널링하지 않거나 시스템 관리 액세스 경로 보호(SMAPP)를 활용하지 않을 경우 액세스 경로를 재구성할 가능성을 줄이는 시스템 기능을 사용하는 것이 좋습니다.

시스템은 액세스 경로가 재구성되어야 하는지를 판별하기 위해 파일 동기화 인디케이터를 사용합니다. 일반적으로 동기화 인디케이터는 액세스 경로와 연관 자료 동기화를 나타내기 위해 켜져 있습니다. 작업이 액세스 경로에 영향을 미치는 파일을 변경하면 시스템이 파일의 동기화 인디케이터를 끕니다. 시스템이 이상 종료되면 동기화 인디케이터가 꺼져 있는 파일에 대해서 액세스 경로를 재구성해야 합니다.

재구성해야 할 액세스 경로의 수를 줄이기 위해서는 정기적으로 자료와 액세스 경로를 동기화할 필요가 있습니다. 파일과 액세스 경로를 동기화시키는 방법으로는 다음과 같은 몇 가지가 있습니다.

- 전체 파일 닫기. 파일에 대해 시스템 전반에 걸쳐 수행된 최종 전체(즉 공유되지 않은) 닫음은 액세스 경로와 자료를 동기화시킵니다.
- 액세스 경로 강제. 강제 액세스 경로(FRCAOCPTH) 매개변수를 데이터베이스 파일 작성, 변경 또는 대체 명령에서 지정하십시오.
- 2이상의 강제 기록 수. 강제 기록율(FRCRATIO) 매개변수를 데이터베이스 파일 작성, 변경 또는 대체 명령에서 지정하십시오.

- 자료 끝 강제. 프로그램에서 자료 끝 강제 조작을 실행하게 되면 파일의 자료와 해당 액세스 경로를 동기화할 수 있습니다(일부 고급 언어에는 자료 끝 강제 조작이 없습니다. 자세한 내용은 고급 언어 주제 콜렉션을 참조하십시오.

이미 언급한 메소드 중 하나를 수행하는 것은 액세스 경로 및 자료를 동기화합니다. 그러나 파일의 자료를 다음 변경은 동기화 인디케이터를 다시 끌 수 있습니다.

각 방법은 성능상 손실이 클 수도 있으므로 주의해서 사용해야 합니다. 액세스 경로를 보호하기 위한 1차 수단으로서 액세스 경로를 저장하거나 SMAPP를 사용하는 것과 함께 액세스 경로를 저널링하는 것을 고려하십시오.

시스템 종료 후 데이터베이스 회복

시스템이 이상 종료된 후, 시스템은 몇 가지 자동 회복 단계를 따릅니다. 시스템은 디렉토리를 재구축하고 파일이 저널되도록 저널을 동기화합니다. 시스템은 IPL 중이나 IPL 후에 회복 조작을 수행합니다.

IPL 중 데이터베이스 파일 회복:

초기 프로그램 로드(IPL) 중 시스템에서는 회복 기능만이 활성 상태입니다.

IPL 도중 데이터베이스 파일 회복은 다음 작업으로 구성됩니다.

- 시스템 종료 시 진행되고 있던 다음 기능들이 완료됩니다.
  - 파일 삭제
  - 멤버 제거
  - 멤버 재명명
  - 오브젝트 이동
  - 오브젝트 재명명
  - 오브젝트 소유자 변경
  - 멤버 변경
  - 권한부여
  - 권한 취소
  - 실제 파일 저널링 시작
  - 액세스 저널링 시작
  - 실제 파일 저널링 종료
  - 액세스 경로 저널링 종료
  - 저널 변경
  - 저널 삭제
  - SQL 뷰 회복
  - 실제 파일 제한조건 제거
- 시스템 종료 시 진행하고 있던 다음 기능들은 원래 상태로 되돌려집니다(다시 수행해야 함).

  - 파일 작성
  - 멤버 추가
  - 파일 변경
  - 저널 작성
  - 저널 복원
  - 실제 파일 제한조건 추가
- 오퍼레이터가 IPL(유인 IPL)을 수행할 경우 오퍼레이터의 화면에 액세스 경로 편집 재작성이 표시됩니다. 이 화면에서 오퍼레이터는 즉시 또는 지연 유지보수를 가진 사용 중이던 파일의 RECOVER 옵션을 편집할 수 있습니다. 모든 액세스 경로가 유효하거나 IPL이 무인(unattended) 모드에서 이루어지는 경우에는 화면이 표시되지 않습니다.
- 다음의 액세스 경로;
  - 즉시 또는 지연된 유지보수를 갖고 있는 액세스 경로
  - IPL 중 회복되도록 지정된(RECOVER 옵션에서 또는 '액세스 경로의 재구성 편집' 화면에서의 변경에 의해) 액세스 경로
  - 액세스 경로 저널링 시작할 때 재구축되는 액세스 경로 및 보내지는 메세지. 저널 리시버를 각자의 사용자 보조 기억장치 풀에 넣어두면, 디스크 고장으로부터 이들을 보호하며 최고의 저널링 성능을 가져올 수 있습니다.

관련 개념

저널 관리

IPL 후 데이터베이스 파일 회복:

초기 프로그램 로드(IPL) 완료 후 이 데이터베이스 파일 회복이 실행됩니다. 대화식 및 일괄처리 작업은 회복과 함께 계속됩니다.

IPL 후 데이터베이스 파일 회복은 다은 작업으로 구성됩니다.

- IPL 후 재구성을 지정하는 즉시 또는 지연 유지보수 파일에 대한 액세스 경로가 재구성됩니다.
- 시스템 이력 기록부는 재구성 조작의 성공이나 실패 여부를 나타내는 메세지를 보냅니다.
- IPL 완료 후 액세스 경로를 재구성하려면 EDTRBDAP(액세스 경로 재구성 편집) 명령을 사용하십시오.
- IPL 완료 후, EDTCPCST(검사 지연 제한조건 편집) 명령은 검사 지연 상태에 있는 실제 파일 제한조건의 리스트를 표시합니다.이 명령은 검사 지연 제한조건의 확인 순서를 지정합니다.

주: 파일에 저널링을 사용하지 않을 경우 IPL 회복 후 레코드가 있을 수도 있고 없을 수도 있습니다.

- 추가된 레코드의 경우 IPL 회복후에 추가된 N번째 레코드가 존재하면 N 이전에 추가된 모든 레코드도 존재합니다.
- 갱신 및 삭제 레코드의 경우 IPL 회복 후에 N번째 레코드에 대한 갱신이나 삭제 조작이 표시되면 N번째 레코드 이전에 갱신되거나 삭제된 레코드가 반드시 데이터베이스에 표시되는 것은 아닙니다.

- REUSEFLT(*YES)의 경우 추가된 레코드는 갱신으로 처리되므로 이러한 레코드는 IPL 회복 후에 존재하지 않을 수 있습니다.

관련 참조

EDTRBDAP(액세스 경로 리빌드 편집) 명령

EDTCPCST(확인 지연 제한사항 편집) 명령

데이터베이스 회복 시 기억장치 풀 페이징 옵션 효과:

공유 풀 페이징 옵션은 최적의 성능을 위해 시스템이 각 기억장치의 페이징 특성을 동적으로 조절할지의 여부를 결정합니다.

- 시스템은 페이징 옵션이 *FIXED인 경우 페이징 특성을 동적으로 조절하지 않습니다.
- 시스템은 페이징 옵션이 *CALC인 경우 페이징 특성을 동적으로 조절합니다.
- 또한 어플리케이션 프로그래밍 인터페이스를 통해 페이징 특성을 제어할 수 있습니다. 자세한 내용은 어플리케이션 프로그래밍 인터페이스(API) 주제에서 QWCCHGTN(풀(pool) 조정 정보 API 변경)을 참조하십시오.

*FIXED이외의 공유 풀 페이징 옵션은 시스템 장애 시 저널되지 않은 실제 파일(PF)에 대한 자료손실을 초래할 수 있습니다. 실제 파일을 저널하지 않았을 때 *CALC나 USRDFN 페이징 옵션의 경우 시스템 장애시, 저장되지 않은 메모리의 자료 손실은 더욱 증가될 수 있습니다. 이러한 옵션의 경우 파일 변경 내용을 빈번하게 보조 기억장치에 기록하지 않을 수 있습니다. *FIXED 옵션을 사용한 저널되지 않은 파일에는 자료 손실의 위험이 있으며, 이 위험성은 *CALC나 사용자 정의(USRDFN) 페이징 옵션의 경우 더 높아집니다. 페이징 옵션에 대한 자세한 내용은 성능 주제의 성능 자동 조정을 참조하십시오.

관련 개념

자동 성능 조정

어플리케이션 프로그래밍 인터페이스(API)

성능

관련 참조

QWCCHGTN(풀(pool) 조정 정보 API 변경) 명령

데이터베이스 파일 회복 옵션 표:

이 주제는 데이터베이스 파일의 회복 옵션에 대해 간단히 소개합니다.

	지정된 RECOVER 매개변수		
액세스 경로/유지보수	***NO**	***AFTIPL**	***IPL**
키순 액세스 경로/즉시 또는 지연 유지보수	• IPL 시 데이터베이스가 회복되지 않음 • 파일이 즉시 사용 가능함 • 파일이 처음 열릴 때 액세스 경로가 재구성됨	• IPL 후에 액세스 경로가 재구성됨	• IPL 중에 액세스 경로가 재구성됨

지정된 RECOVER 매개변수			
액세스 경로/유지보수	***NO**	***AFTIPL**	***IPL**
키순 액세스 경로 리빌드 유지보수	• IPL 시 데이터베이스가 회복되지 않음 • 파일이 즉시 사용 가능함 • 파일이 처음 열릴 때 액세스 경로가 재구성됨	• 적용 안됨. 리빌드 유지보수에 대해 어떤 회복도 이루어지지 않음	• 적용 안됨. 리빌드 유지보수에 대해 어떤 회복도 이루어지지 않음
도달순 액세스 경로	• IPL 시 데이터베이스가 회복되지 않음 • 파일이 즉시 사용 가능함	• 적용 안됨. 도달순 액세스 경로에 대해 어떤 회복도 이루어지지 않음	• 적용 안됨. 도달순 액세스 경로에 대해 어떤 회복도 이루어지지 않음

데이터베이스 저장 및 복원

지원되는 장치 및 매체 또는 저장 파일을 사용하여 데이터베이스 파일 저장 및 복원될 수 있습니다.

저장 파일(또는 미디어)는 저장 정보의 특별 형식으로 쓰여진 복사를 수신합니다. 사용자의 시스템이나 다른 iSeries 시스템에서 나중에 사용할 미디어를 저장 또는 제거할 수 있습니다. 복원된 정보는 매체나 저장 파일에서 시스템 사용자가 정보에 액세스하는 기억장치로 읽힙니다.

저장 파일이란 저장 조작의 목표나 복원 조작의 소스가 될 수 있는 디스크 상주 파일입니다. 저장 파일은 무인 저장(unattened save) 조작을 허용합니다. 오퍼레이터가 테이프나 디스켓을 로드할 필요가 없습니다. 그러나 정기적으로 SAVSAVFDTA(저장 파일 자료 저장) 명령을 사용하여 저장 파일을 테이프나 디스켓에 저장하십시오. 정기적으로 사이트로부터 테이프 및 디스켓을 제거 및 저장하십시오. 이 매체는 사이트 재해의 경우에 회복을 도울 수 있습니다.

관련 개념

백업 및 회복 PDF

백업 및 회복

관련 참조

SAVSAVFDTA(저장 파일 자료 저장) 명령

저장 및 복원 시 데이터베이스 고려사항

이 주제에서는 저장 및 복원 기능으로 작업 시 추가 정보를 제공합니다.

- 오브젝트를 저장 파일에 저장할 때에 저장 명령에 UPDHST(*NO)를 지정하여 시스템이 저장 조작의 날짜와 시간을 갱신하지 못하게 할 수 있습니다.
- 오브젝트를 복원할 때 시스템은 항상 복원 조작의 날짜와 시간으로 오브젝트 설명을 갱신합니다. DETAIL(*FULL)과 함께 DSPOBJD(오브젝트 설명 표시) 명령을 사용하여 오브젝트 설명과 기타 저장/복원 관련 정보를 표시하십시오.
- 저장 파일에 오브젝트를 표시하려면 DSPSAVF(저장 파일 표시) 명령을 사용하십시오.
- 매체에 오브젝트를 표시하려면 DSPDKT(디스켓 표시) 또는 DSPTAP(테이프 표시) 명령에 DATA(SAVRST)를 지정하십시오.

- 데이터베이스 파일의 최근 저장/복원 날짜를 표시하려면 DSPFD FILE(파일명) TYPE(*MBR)을 입력하십시오.

또한 보조 기억장치로 레코드 자동 쓰기를 고려하십시오.

보조 기억장치에 자료 강제 쓰기

파일 작성 명령과 데이터베이스 파일 대체 명령의 FRCRATIO(강제 기록 수) 매개변수는 레코드가 보조 기억장치에 기록되는 빈도를 표시합니다.

강제 기록 수가 1이면 파일에 대한 모든 추가, 갱신 및 삭제 요구가 보조 기억장치에 기록됩니다. 그러나 이 옵션을 사용하면 시스템 성능이 저하될 수 있으므로 파일을 저장하고 저널링을 이용하는 것이 데이터베이스 파일을 보호하는 1차적인 방법으로 고려하십시오.

관련 개념

백업 및 회복 PDF

백업 및 회복

관련 참조

오브젝트 설명 표시(DSPOBJD) 명령

DSPSAVF(저장 파일 표시) 명령

DSPTAP(테이프 표시) 명령

소스 파일 사용

이 주제에서는 소스 파일에 자료를 입력하고 유지보수하는 방법과 소스 파일을 사용하여 시스템에서 다른 오브젝트(예: 파일 또는 프로그램)를 작성하는 방법을 설명합니다.

관련 개념

16 페이지의 『소스 파일 설정』
이 주제에서는 소스 파일 설정 방법과 소스 파일을 사용하는 이유를 설명합니다.

소스 파일에 대한 작업

이 주제에서는 다양한 방법을 사용하여 자료를 입력하고 유지보수하는 방법을 설명합니다.

관련 태스크

45 페이지의 『논리 파일 작성』
이 주제는 자료 서술 스펙(DDS)을 사용하여 논리 파일을 작성하는 방법을 표시합니다.

소스 입력 유틸리티(SEU) 사용:

소스 입력 유틸리티(SEU)를 사용하여 소스 파일에 소스를 입력하고 변경할 수 있습니다. SEU는 iSeries용 IBM WebSphere Development Studio의 일부입니다.

SEU를 사용하여 데이터베이스 파일에 소스를 입력하는 경우 SEU가 각 소스 레코드에 순서 번호와 날짜 필드를 추가합니다.

SEU를 사용하여 소스 파일을 갱신하는 경우 기존 레코드 사이에 레코드를 추가할 수 있습니다. 예를 들어, 레코드 0003.00과 0004.00 사이에 레코드를 하나 추가하려면 추가되는 레코드의 순서 번호는 0003.01이 될 수 있습니다. SEU는 이러한 방법으로 자동적으로 새로 추가된 소스문을 배열합니다.

레코드가 소스 파일에 처음 들어가면, 날짜 필드는 모두 존 십진 0으로 됩니다(DFT 키워드와 함께 지정된 DDS가 사용되지 않는 한). SEU를 사용하면 레코드를 변경할 때 레코드의 날짜 필드도 변경됩니다. 데이터베이스 소스 파일 갱신 방법에 대한 정보는 V5R1 보충 매뉴얼 웹 사이트에서 *AS/400용 ADTS: 소스 입력 유틸리티* 매뉴얼을 참조하십시오.

장치 소스 파일 사용:

테이프와 디스켓 파일은 소스 파일로서 작성할 수 있습니다. 장치 파일이 소스 파일로 사용되는 경우 레코드 길이에는 순서 번호와 날짜 필드가 포함되어야 합니다.

최대 레코드 길이 제한은 항상 이 12자를 포함하여 고려되어야 합니다. 예를 들어, 테이프 레코드의 최대 레코드 길이는 32 766입니다. 자료가 소스 입력으로 처리될 경우 실제 테이프 자료 레코드의 최대 길이는 32 754(32 766 - 12)가 됩니다.

입력을 위해 소스 장치 파일을 여는 경우 시스템이 순서 번호와 날짜 필드를 추가하며 이 때 날짜 필드는 0으로 채워집니다.

출력을 위해 장치 파일을 열고 파일이 소스 파일로 정의된 경우 장치에 자료를 기록하기 전에 시스템이 순서 번호와 날짜를 삭제합니다.

소스 파일 자료 복사:

CPYSRCF(소스 파일 복사) 및 CPYF(파일 복사) 명령을 사용하여 자료를 소스 파일 멤버에 기록하거나 멤버로부터 기록할 수 있습니다. 데이터베이스 소스 파일을 이와 연관된 삽입 트리거가 있는 다른 데이터베이스 소스 파일로 복사할 때 트리거 프로그램이 복사된 각 레코드에 대해 호출됩니다.

CPYSRCF(소스 파일 복사) 명령을 사용하여 소스 파일에 및 소스 파일에서 복사:

CPYSRCF(소스 파일 복사) 명령은 데이터베이스 소스 파일과 함께 작동하도록 설계되었습니다. 기능면에서는 CPYF(파일 복사) 명령과 유사하나 CPYSRCF 명령에서는 소스 파일을 복사할 때 일반적으로 사용되는 디폴트가 제공됩니다.

예를 들어, CPYSRCF 명령에는 TOMBR 매개변수가 FROMMBR 매개변수와 동일한 것으로 가정하는 디폴트가 있으며 TOMBR 레코드가 언제나 대체되도록 하는 디폴트가 있습니다. 또한 CPYSRCF 명령은 TOFILE(*PRINT)가 지정될 때 고유 인쇄 형식을 지원합니다. 그러므로 데이터베이스 소스 파일을 복사할 경우 CPYSRCF 명령을 사용하는 것이 편리합니다.

CPYSRCF 명령은 From 파일 CCSID에서 to-파일 CCSID로 자료를 자동 변환합니다.

관련 참조

소스 파일 복사(CPYSRCF) 명령

파일에 및 파일로부터 복사에 CPYF(파일 복사) 명령 사용:

CPYF(파일 복사) 명령은 소스 파일에 및 소스 파일로부터 복사의 CPYSRCF(소스 파일 복사) 명령에 추가 기능을 제공합니다.

CPYF 명령은 다음 작업을 허용합니다.

- 데이터베이스 소스 파일로부터 장치 파일에 복사
- 장치 파일로부터 데이터베이스 소스 파일에 복사
- 소스 파일이 아닌 데이터베이스 파일과 소스 데이터베이스 파일간의 복사
- 소스 멤버를 16진 형식으로 인쇄
- 선택값을 사용하여 소스 복사

관련 참조

파일 복사(CPYF) 명령

복사에 사용되는 순번:

데이터베이스 소스 파일에 복사할 경우 SRCOPT 매개변수를 사용하여 순번을 갱신하고 날짜를 0으로 초기화 할 수 있습니다.

디폴트로서, 시스템은 첫 번째 레코드에 순서 번호 1.00을 할당하며 나머지 레코드에 대해 순서 번호를 1.00씩 증가시킵니다. SRCSEQ 매개변수를 사용하여 소수의 증분값을 설정하고, 번호 재설정이 시작될 순서 번호를 지정할 수 있습니다. 예를 들어, SRCSEQ 매개변수에 증분값을 .10으로 정하고 순서 번호 100.00에서 번호가 시작되도록 지정하면 복사된 레코드의 순서 번호는 100.00, 100.10, 100.20 등으로 매겨집니다.

시작값 .01과 증분값 .01이 지정되면 최대 999,999개까지의 레코드가 고유한 순서 번호를 갖게 됩니다. 순서 번호가 최대(9999.99)에 이르면 나머지 레코드는 모두 순서 번호 9999.99를 갖게 됩니다.

다음은 동일한 파일에 있는 한 멤버에서 다른 멤버에게 소스를 복사하는 데 관한 예입니다. MBRB가 존재하지 않을 경우에는 추가되고, 존재하는 경우에는 레코드가 모두 대체됩니다.

```
CPYSRCF FROMFILE(QCLSRC) TOFILE(QCLSRC) FROMMBR(MBRA) +
                TOMBR(MBRB)
```

다음은 총칭 멤버명을 한 파일에서 다른 파일로 복사하는 데 대한 예입니다. PAY로 시작되는 모든 멤버가 복사됩니다. 해당 멤버가 존재하지 않을 경우에는 추가되고, 존재하는 경우에는 모든 레코드가 대체됩니다.

```
CPYSRCF FROMFILE(LIB1/QCLSRC) TOFILE(LIB2/QCLSRC) +
                FROMMBR(PAY*)
```

다음은 멤버 PAY1을 프린터 파일 QSYSPRT(*PRINT의 디폴트)로 복사하는 데 대한 예입니다. 서비스 입력 유틸리티(SEU)에 사용되는 것과 유사한 형식이 소스문을 인쇄하는 데 사용됩니다.

```
CPYSRCF FROMFILE(QCLSRC) TOFILE(*PRINT) FROMMBR(PAY1)
```

장치 소스 파일로부터 데이터베이스 소스 파일에 복사하면 순서 번호들이 추가되며 날짜들이 0으로 초기화됩니다. 순서 번호는 1.00부터 시작하여 1.00씩 증가합니다. 복사되는 파일의 레코드 수가 9999개를 넘는 경우 SRCOPT 및 SRCSEQ 매개변수가 지정되지 않은 한, 순서 번호는 다시 1.00부터 시작하여 계속하여 증가됩니다.

데이터베이스 소스 파일로부터 장치 소스 파일에 복사하면 날짜 및 순서 번호 필드들은 제거됩니다.

비iSeries 시스템에서 자료 로드 및 언로드:

CPYFRMIMPF(가져온 파일에서 복사) 및 CPYTOIMPF(가져온 파일로 복사) 명령을 사용하여 비iSeries 시스템에서 자료를 가져오기(로드)하거나 이 시스템으로 자료를 내보내기(언로드)할 수 있습니다.

CPYFRMIMPF 및 CPYTOIMPF 명령을 사용하여 iSeries 이외의 데이터베이스에 있는 자료를 외부 설명된 iSeries용 DB2 Universal Database 데이터베이스 파일로 가져오려면 다음 단계를 수행하십시오.

1. 복사할 자료에 대해 가져오기 파일을 작성합니다. 가져오기 파일은 하나의 필드를 갖는 데이터베이스 소스 파일이거나 외부 설명된 데이터베이스 파일이 될 수 있습니다. 필드의 자료 유형은 반드시 CHARACTER, IGC OPEN, IGC EITHER, IGC ONLY 또는 UCS-2여야 합니다.
2. 자료를 가져오기 파일로(또는 파일에서) 전송합니다. 이 프로세스 동안 시스템은 필요하다면 ASCII에서 EBCDIC로 변환을 수행합니다. 다음과 같은 여러 방법으로 자료를 전송할 수 있습니다.
   - TCP/IP 파일 전송
   - iSeries Access 지원(파일 전송, ODBC)
   - CPYFRMTAP(테이프 파일에서 복사) 명령
3. 자료를 복사해 넣을 외부 설명된 iSeries용 DB2 UDB 데이터베이스 파일 또는 분산 자료 관리(DDM) 파일을 작성합니다.
4. CPYFRMIMPF 명령을 사용하여 가져오기 파일에서 iSeries 데이터베이스 파일로 자료를 복사합니다. DB2 UDB 대칭형 멀티프로세싱 제품이 시스템에 설치되어 있으면, 시스템이 그 파일을 병행해서 복사할 것입니다.

iSeries 데이터베이스 자료를 다른 시스템으로 내보내려면 CPYTOIMPF 명령을 사용하여 데이터베이스 파일에 있는 자료를 가져오기 파일로 복사합니다. 그리고 나서 그 자료를 내보내기하려는 시스템으로 자료를 전송합니다.

관련 참조

가져오기 파일에서 복사(CPYFRMIMPF) 명령

가져오기 파일로 복사(CPYTOIMPF) 명령

프로그램에서 소스 파일 사용:

프로그램에서 소스 파일을 처리할 수 있습니다. 다른 데이터베이스 파일과 마찬가지로 소스 파일의 외부 정의를 사용하고, 해당 파일에 대해 입/출력 조작을 수행할 수 있습니다.

소스 파일은 외부 서술 데이터베이스 파일입니다. 따라서 프로그램에 소스 파일을 명명하여 컴파일하는 경우 소스 파일 설명이 프로그램 인쇄 출력에 자동으로 포함됩니다. 예를 들어, 소스 파일 QDDSSRC에서 멤버 FILEA의 레코드를 읽고 갱신하는 것으로 가정하십시오. 이 파일을 처리할 프로그램을 작성할 때 시스템은 소스 파일로부터 *SRCSEQ*, *SRCDAT* 및 *SRCDTA* 필드 등을 포함시킵니다.

주: DSPFFD(파일 필드 설명 표시) 명령을 사용하여 파일에 정의된 필드를 표시할 수 있습니다.

QDDSSRC 파일의 FILEA 멤버를 처리하는 프로그램은 다음을 수행할 수 있습니다.

- 파일 멤버를 엽니다(다른 데이터베이스 파일 멤버의 경우와 같음).
- 소스 파일 레코드를 읽고 갱신합니다(실제의 소스 자료가 저장되어 있는 *SRCDTA* 필드를 변경할 수도 있음).
- 소스 파일 멤버를 닫습니다(다른 데이터베이스 파일 멤버의 경우와 같음).

관련 개념

246 페이지의『파일에 필드 설명 표시』

DSPFFD(파일 필드 설명 표시) 명령을 사용하여 데이터베이스 파일 및 장치 파일에 대한 필드 정보를 표시할 수 있습니다. 정보를 표시하거나 인쇄할 수 있으며, 데이터베이스 출력 파일(OUTFILE)에 기록할 수도 있습니다.

소스 파일을 사용하여 오브젝트 작성

작성 명령으로 소스 파일을 사용하여 오브젝트를 작성할 수 있습니다. 소스 파일을 사용하여 오브젝트를 작성하는 경우 작성 명령에 소스 파일명을 지정할 수 있습니다.

예를 들어, 제어 언어 프로그램을 작성하려면 CRTCLPGM(제어 언어 프로그램 작성) 명령을 사용합니다. 작성 명령은 SRCFILE 매개변수를 통해 소스가 저장되는 곳을 지정합니다.

작성 명령은 사용자가 이 프로시듀어를 수행할 경우 소스 파일명과 멤버명을 지정할 필요가 없도록 설계되었습니다.

1. 작성 중인 오브젝트 유형에 대해 디폴트 소스 파일명을 사용합니다. 사용 중인 명령에 대한 디폴트 소스 파일을 찾으려면 IBM에서 제공하는 소스 파일을 참조하십시오.
2. 소스 멤버명을 작성될 오브젝트명과 동일하게 지정합니다.

예를 들어, 명령 디폴트를 사용하여 제어 언어(CL) 프로그램 PGMA를 작성하려면 다음을 입력합니다.

```
CRTCLPGM PGM(PGMA)
```

시스템은 PGMA에 대한 소스가 QCLSRC 소스 파일의 PGMA 멤버 안에 있는 것으로 가정합니다. QCLSRC 파일이 들어 있는 라이브러리는 라이브러리 리스트에 의해 결정됩니다.

또 하나의 예로, 다음의 CRTPF(실제 파일 작성) 명령은 데이터베이스 소스 파일 FRSOURCE를 사용하여 파일 DSTREF를 작성합니다. 소스 멤버는 DSTREF로 명명됩니다. SRCMBR 매개변수가 지정되지 않았으므로 시스템은 소스 멤버명 DSTREF가 작성될 오브젝트명과 같다고 가정합니다.

```
CRTPF FILE (QGPL/DSTREF) SRCFILE(QGPL/FRSOURCE)
```

관련 개념

19 페이지의『IBM 제공 소스 파일』

사용자 편의를 위해 i5/OS 라이센스 프로그램 및 기타 라이센스 프로그램은 각 소스 유형에 데이터베이스 소스 파일을 제공합니다.

일괄처리 작업의 소스문에서 오브젝트 작성:

작성 명령이 일괄처리 작업에 포함된 경우 인라인 자료 파일을 명령에 대한 소스 파일로 사용할 수 있습니다.

그러나 소스 파일로 사용되는 인라인 자료 파일의 레코드 수는 10,000개를 초과할 수 없습니다. 인라인 자료 파일에는 이름이 있거나 또는 없을 수 있습니다. 이름이 지정된 인라인 자료 파일은 //DATA 명령에 고유 파일명이 지정됩니다. 인라인 자료 파일에 대한 자세한 내용은 데이터베이스 파일 관리 주제를 참조하십시오.

이름이 지정되지 않은 인라인 자료 파일은 고유한 이름이 없는 파일이며, 모두 QINLINE으로 명명됩니다. 다음은 소스 파일로 사용되는 인라인 자료 파일의 예입니다.

```
//BCHJOB
CRTPF FILE(DSTPRODLB/ORD199) SRCFILE(QINLINE)
//DATA FILETYPE(*SRC)
  .
  .   (소스문)
  .
//
//ENDBCHJOB
```

위의 예에서 //DATA 명령에는 파일명이 지정되지 않았습니다. 스풀링 읽기 기능에 의해 작업이 처리될 때 이름이 없는 스풀 파일이 작성되었습니다. 이름이 없는 파일에 액세스하려면 CRTPF 명령에서 소스 파일명으로 QINLINE을 지정해야 합니다. 또한 //DATA 명령은 인라인 파일이 소스 파일임을 지정합니다(FILETYPE 매개변수로 *SRC가 지정됨).

//DATA 명령에 파일명을 지정하면 CRTPF 명령의 SRCFILE 매개변수에 동일한 이름을 지정해야 합니다. 예를 들면,

```
//BCHJOB
CRTPF FILE(DSTPRODLB/ORD199) SRCFILE(ORD199)
//DATA FILE(ORD199) FILETYPE(*SRC)
  .
  .   (소스문)
  .
//
//ENDBCHJOB
```

프로그램이 인라인 파일을 사용하는 경우 시스템은 지정된 이름을 가진 첫 인라인 파일을 탐색합니다. 파일이 발견되지 않으면 프로그램은 이름이 지정되지 않은 첫 번째 파일(QINLINE)을 사용합니다.

작성 명령에 소스 파일명을 지정하지 않으면 IBM 제공 소스 파일에 필요한 소스 자료가 들어 있는 것으로 가정합니다. 예를 들어, 제어 언어(CL) 프로그램을 작성하면서 소스 파일명을 지정하지 않으면 IBM 제공 소스 파일 QCLSRC가 사용됩니다. 소스 자료는 QCLSRC 안에 있어야 합니다.

소스 파일이 데이터베이스 파일인 경우 필요한 소스 자료가 들어 있는 소스 멤버를 지정할 수 있습니다. 소스 멤버를 지정하지 않을 경우 소스 자료는 작성중인 오브젝트명과 같은 이름의 멤버 안에 있어야 합니다.

관련 개념

데이터베이스 파일 관리

오브젝트 작성 시 사용된 소스 파일 멤버 판별:

소스로부터 오브젝트가 작성되면 소스 파일, 라이브러리 및 멤버에 대한 정보가 오브젝트에 보유됩니다. 오브젝트 작성 이전에 소스 멤버가 최종 변경된 날짜 및 시간도 오브젝트에 저장됩니다.

오브젝트 안의 정보는 DETAIL(*SERVICE)을 지정한 DSPOBJD(오브젝트 설명 표시) 명령을 통해 표시될 수 있습니다.

이 정보를 통해 어떤 소스 멤버가 사용되었는지와 오브젝트가 작성된 이후 기존 소스 멤버가 변경되었는지를 알 수 있습니다.

또한 다음 명령을 사용하여 오브젝트를 작성하는 데 사용된 소스가 현재 소스 멤버 안에 있는 소스와 동일한 것인지 확인할 수 있습니다.

- TYPE(*MBR)을 사용하는 DSPFD(파일 설명 표시) 명령. 이 화면에는 소스 멤버에 대한 날짜 및 시간이 모두 표시됩니다. DSPOBJD 명령에서 표시된 소스 파일 날짜/시간 값이 최종 소스 갱신 날짜/시간 값과 비교되는 데 사용되어야 합니다.
- DETAIL(*SERVICE)을 사용하여 DSPOBJD(오브젝트 설명 표시) 명령. 이 화면에는 오브젝트를 작성하는 데 사용된 소스 멤버의 날짜와 시간이 표시됩니다.

주: 출력 파일에 기록된 자료를 사용하여 소스와 오브젝트 날짜가 같은지 알아 보려면 DSPOBJD DETAIL(*SERVICE) 명령의 출력 파일에 있는 필드 *ODSRCD*(소스 날짜) 및 *ODSRCT*(소스 시간)를 DSPFD TYPE(*MBR) 명령의 출력 파일에 있는 필드 *MBUPDD*(멤버 갱신 날짜) 및 *MBUPDT*(멤버 갱신 시간)와 비교하면 됩니다.

관련 참조

오브젝트 설명 표시(DSPOBJD) 명령

파일 설명 표시(DSPFD) 명령

소스 파일 관리

이 주제에는 소스 파일 관리 시 고려사항이 포함됩니다.

소스 파일 속성 변경:

이 주제에서는 소스 파일 속성을 변경하는 여러 가지 접근방식을 소개합니다.

소스 입력 유틸리티(SEU)를 사용하여 데이터베이스 소스 파일을 관리하는 경우 데이터베이스 소스 파일 변경 방법에 대한 정보는 V5R1 보충 매뉴얼 웹 사이트에서 *AS/400®용 ADTS: 소스 입력 유틸리티* 매뉴얼을 참조하십시오. SEU를 사용하지 않고 데이터베이스 소스 파일을 유지보수하는 경우에는 기존 멤버 전체를 대체해야 합니다.

소스 파일이 디스켓에 있으면 이를 데이터베이스 파일에 복사하여 SEU를 사용해 변경한 후, 다시 디스켓에 복사할 수 있습니다. SEU를 사용하지 않는 경우에는 기존 소스 파일을 삭제한 후에 새로운 소스 파일을 작성해야 합니다.

소스 파일을 변경하면 소스 파일로부터 이전에 작성된 오브젝트가 현재 소스와 일치되지 않습니다. 기존의 오브젝트는 반드시 삭제한 다음 변경된 소스 파일을 사용하여 다시 작성해야 합니다. 예를 들어, 280 페이지의 『소스 파일을 사용하여 오브젝트 작성』에서 작성된 소스 파일 FRSOURCE를 변경하면 소스 원본 파일에서 작성된 파일 DSTREF를 삭제한 후 새 소스 파일을 사용하여 다시 작성하여 DSTREF가 변경된 FRSOURCE 소스 파일과 일치하도록 해야 합니다.

소스 파일 멤버 자료 재구성:

입력순 소스 파일을 사용하는 경우에는 일반적으로 소스 파일 멤버를 재구성할 필요가 없습니다. 고유 순서 번호를 할당하려면 몇 가지 매개변수를 지정해야 합니다.

모든 레코드에 고유한 순서 번호를 할당하려면 RGZPFM(실제 파일 멤버 재구성) 명령에 다음 매개변수를 지정하십시오.

- KEYFILE(*NONE)(레코드가 재구성되지 않음)
- SRCOPT(*SEQNBR)(순서 번호가 변경됨)
- .10 또는 .01 등의 소수값으로 지정된 SRCSEQ(모든 순서 번호가 고유함)

주: 삭제 레코드가 있는 경우 압축되어 제거됩니다.

키순 액세스 경로가 지정된 논리 파일이 실제 파일을 기초로 하여 작성되었으면, 도달순 액세스 경로를 가진 소스 파일이 순서 번호순으로 재구성될 수 있습니다.

관련 참조

실제 파일 멤버 재구성(RGZPFM) 명령

소스문 변경 시점 판별:

명령문이 변경되는 경우 각 소스 레코드에는 소스 입력 유틸리티(SEU)에 의해 자동으로 갱신되는 날짜 필드가 포함되어 있습니다. 이를 통해 명령문이 최종 변경된 시점을 판별할 수 있습니다.

대부분의 고급 언어 컴파일러는 이 날짜를 컴파일러 리스트에 인쇄합니다. CPYF(파일 복사) 및 CPYSRCF(소스 파일 복사) 명령도 이 날짜를 인쇄합니다.

각 소스 멤버 설명에는 두 개의 날짜 및 시간 필드가 포함됩니다. 첫 번째 날짜/시간 필드는 갱신된 후 닫힌 때의 멤버 변경사항을 반영합니다.

두 번째 날짜/시간 필드는 멤버에 대한 모든 변경사항을 반영합니다. 이 필드에는 SEU, CPYF, CPYSRCF와 같은 명령, 권한 변경 그리고 파일 상태 변경 등으로 생긴 모든 변경사항이 포함됩니다. 예를 들어, CHGPF(실제 파일 변경) 명령의 FRCRATIO 매개변수는 멤버 상태를 변경합니다. 이 날짜/시간 필드는 멤버의 저장 여부를 결정하는 SAVCHGOBJ(변경된 오브젝트 저장) 명령에 의해 사용됩니다. 날짜/시간 필드는 TYPE(*MBR)을 지정하는 DSPFD(파일 설명 표시) 명령으로 화면에 표시될 수 있습니다. 소스 멤버를 표시하는 두 개의 변경된 날짜/시간 필드가 있습니다.

- 최종 소스 갱신 날짜/시간. 이 값은 멤버 내의 소스 자료 레코드에 대한 모든 변경사항을 반영합니다. 소스가 갱신되는 경우 날짜/시간값에 1,2초 가량 시간차가 있더라도 최종 변경 날짜/시간 값이 함께 갱신됩니다.
- 최종 변경 날짜/시간. 이 값은 멤버에 대한 모든 변경사항을 반영합니다. 여기에는 SEU, CPYF와 CPYSRCF 명령, 권한 변경 및 파일 상태 변경 등으로 생기는 모든 변경사항이 포함됩니다. 예를 들어, CHGPF 명령의 FRCRATIO 매개변수는 멤버 상태를 변경하며, 따라서 최종 변경 날짜/시간 값에 반영됩니다.

문서화를 위해 소스 파일 사용:

온라인 문서를 작성 및 갱신하기 위하여 IBM 제공 소스 파일인 QTXTSRC를 사용할 수 있습니다.

소스 입력 유틸리티(SEU)와 사용 가능한 다른 어플리케이션(예: QRPGSRC 또는 QCLSRC)과 마찬가지로 QTXTSRC 멤버를 작성하고 갱신할 수 있습니다. QTXTSRC 파일은 특히 해설식 문서를 작성하는 데 편리하며, 이는 온라인으로 검색되거나 인쇄될 수 있습니다. SEU의 추가, 변경, 이동, 복사 및 포함 조작을 사용하여 소스 멤버에 넣는 텍스트를 쉽게 갱신할 수 있습니다. 나감 프롬트의 현재 소스 파일 인쇄 옵션에 예(Yes)를 지정하여 멤버 전체를 인쇄할 수 있습니다. 또한 소스 멤버의 일부 또는 전체를 인쇄하는 프로그램을 작성할 수도 있습니다.

제한조건으로 데이터베이스 무결성 제어

*제한조건*은 파일에 지정된 제한 또는 한계를 의미하며 레코드 추가, 변경 및 제거 시 데이터베이스의 자료에 일관성이 있는지 확인할 수 있습니다. 이 주제에서는 자료 일관성 확인을 위해 제한조건을 사용하는 방법에 대해 설명합니다.

- 고유한 제한조건 및 1차 키 제한조건을 사용하여 파일 액세스 경로 밖의 파일에 대해 강제된 고유 키를 작성할 수 있습니다.
- 제한조건 검사는 표현식의 자료를 테스트하여 자료의 유효성에 대해 또 하나의 검사를 제공합니다.

참조 제한조건을 추가할 때 1차 키와 고유 제한사항을 상위 키로 사용할 수 있습니다.

데이터베이스에 제한조건 설정

실제 파일 제한조건을 시용하여 데이터베이스에 유지보수되는 자료 무결성을 제어할 수 있습니다.iSeries Navigator를 사용하여 제한조건을 추가할 수도 있습니다.

실제 파일 제한조건을 추가하려면 ADDPFCST(실제 파일 제한조건 추가) 명령을 사용하십시오.

- 고유 제한조건을 추가하려면 Type 매개변수에 *UNQCST 값을 지정하십시오. 키 매개변수용으로 하나 또는 그 이상의 필드명을 지정해야 합니다.
- 1차 키 제한조건을 추가하려면 Type 매개변수에 *PRIKEY 값을 지정하십시오. 명령에 지정하는 키는 1차 액세스 경로가 됩니다. 파일에 공유될 수 있는 키순 액세스 경로가 없을 경우에는 시스템이 파일을 작성합니다. 키 매개변수용으로 하나 또는 그 이상의 필드명을 지정해야 합니다.
- 검사 제한조건을 추가하려면 Type 매개변수에 *CHKCST 값을 지정하십시오. 또한 CHKCST 검사 제한조건 매개변수 표현식을 지정하십시오. 검사 제한조건은 표현식이 SQL에 정의된 검사 조건에 사용된 것과 동일한 구문을 갖습니다. SQL로 제한조건 설정에 대한 정보는 iSeries용 DB2 Universal Database SQL 참조서를 참조하십시오.

iSeries Navigator를 사용하여 제한조건을 추가할 수도 있습니다.SQL 프로그래밍 주제에서 다음 주제를 참조하십시오.

- iSeries Navigator를 사용하여 키 제한조건 추가
- iSeries Navigator를 사용하여 검사 제한조건 추가

SQL CREATE TABLE 및 ALTER TABLE 명령문을 사용하여 제한사항을 추가할 수도 있습니다.

제한조건 설정에 대한 규칙

다음은 모든 실제 파일 제한조건에 적용되는 규칙 리스트입니다.

- 파일이 실제 파일이어야 합니다.
- 파일은 한 멤버의 최대인 MAXMBR(1)을 가질 수 있습니다.
- 파일에 멤버가 없을 경우 제한조건이 정의될 수 있습니다. 파일에 단지 하나의 멤버가 있어도 제한조건이 설정될 수 없습니다.
- 한 파일에서 1차 키 제한조건은 최대 한 개를 포함할 수 있으나 고유 제한조건은 여러 개를 포함할 수 있습니다.
- 파일당 최대 300개의 제한조건 관계가 있습니다. 이 최대값은 다음 제한조건의 합계입니다.
  - 고유 제한조건
  - 1차 키 제한조건
  - 검사 제한조건
  - 참조 제한조건이 상위(parent)로 관여할 것인지 아니면 종속으로 관여할 것인지와 제한조건이 정의되거나 설정될 것인지의 여부
- 제한조건명은 하나의 라이브러리 내에서 고유해야 합니다.
- 제한조건은 QTEMP 라이브러리의 파일에는 추가될 수 없습니다.
- 참조 제한조건은 같은 보조 기억장치 풀(ASP)의 상위 종속 파일을 가져야 합니다.

관련 개념

iSeries SQL용 DB2 UDB 참조서

SQL 프로그래밍

iSeries Navigator 시작

관련 참조

ADDPFCST(실제 파일 제한사항 추가) 명령

CREATE TABLE

ALTER TABLE

고유, 1차 키 또는 검사 제한조건 제거

RMVPFCST(실제 파일 제한조건 제거) 명령을 사용하여 실제 파일 제한조건을 제거할 수 있습니다. 제거하려는 제한조건 유형 및 사용 방법에 따라 명령을 효율적으로 사용할 수 있습니다.

실제 파일 제한조건을 제거하려면 RMVPFCST 명령을 사용하십시오.

- 고유 제한조건을 제거하려면 유형 매개변수에 *UNQCST 값을 지정하십시오.
- 1차 키 제한조건을 제거하려면 유형 매개변수에 *PRIKEY 값을 지정하십시오.
- 검사 제한조건을 제거하려면 유형 매개변수에 *CHKCST 값을 지정하십시오.

각 제한조건 유형의 CST(제한조건) 매개변수에 다음 값 중 하나를 지정할 수 있습니다.

- 유형 매개변수에 지정한 모든 제한조건을 제거하려면 CST(*ALL)을 지정하십시오.
- 특정 제한조건을 제거하려면 CST(제한조건명)을 지정하십시오.
- 검사 지연 상태인 제한조건만 삭제하려면 CST(*CHKPND)를 지정하십시오.
- 파일에서 모든 제한조건을 제거하려면 TYPE(*ALL)와 함께 CST(*ALL)을 지정하십시오.

또한 SQL(Structured Query Language) 또는 iSeries Navigator를 사용하여 제한조건을 제거할 수 있습니다.

제한조건 제거 시 고려사항

1차 키 또는 고유 제한조건을 제거하고 연관된 액세스 경로가 논리 파일에 의해 공유될 경우 공유된 경로의 소유권은 논리 파일로 이전합니다. 액세스 경로가 공유되지 않을 경우에는 제거됩니다.

RMVPFCST 명령으로 1차 키 제한조건을 제거하면 시스템은 키 스펙이 파일에서 제거되어야 하는지를 결정하기 위해 조회 메세지를 보냅니다. 응답 'K'는 키 스펙을 파일에 보유합니다. 파일은 키가 지정된 채 남아 있습니다. 'G' 응답은 명령이 완료되면 파일이 도달순 액세스 경로로 정렬된다는 것을 나타냅니다.

주: SQL ALTER TABLE 명령문으로 1차 키 제한조건을 제거하면 조회 메세지가 전달되지 않습니다. ALTER TABLE 완료 시 키 스펙이 항상 제거되고 파일에는 도달순 액세스 경로가 지정됩니다.

관련 개념

iSeries SQL용 DB2 UDB 참조서

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

RMVPFCST(실제 파일 제한사항 제거) 명령

제한조건 그룹으로 작업

WRKPFCST(실제 파일 제한조건으로 작업) 명령을 사용하여 제한조건 리스트를 표시하고 작업할 수 있습니다.

특정 파일에 존재하는 제한조건 리스트를 표시하려면 WRKPFCST 명령을 사용하십시오. 이 표시로부터, 제한조건을 변경 또는 제거할 수 있고 파일 제한조건을 검사 지연 상태에 놓는 레코드 리스트를 표시할 수 있습니다.

관련 참조

WRKPFCST(실제 파일 제한사항으로 작업) 명령

세부사항: 제한조건 그룹으로 작업:

제한조건 그룹으로 작업에 대한 추가 정보를 읽으십시오.

```
                          실제 파일 제한조건에 대한 작업

옵션을 입력한 후 Enter 키를 누르십시오.
   2=변경   4=제거   6=검사 지연 상태의 레코드 표시


Opt  제한조건     파일        라이브러리 유형     상태     지연     검사
  _   DEPTCST      EMPDIV7     EPPROD     *REFCST  EST/ENB  No
  _   ACCTCST      EMPDIV7     EPPROD     *REFCST  EST/ENB  Yes
  _   STAT84       EMPDIV7     EPPROD     *REFCST  DEF/ENB  No
  _   FENSTER      REVSCHED    EPPROD     *REFCST  EST/DSB  Yes
  _   IRSSTAT3     REVSCHED    EPPROD     *UNQCST
_   IFRNUMBER0 > REVSCHED    EPPROD     *UNQCST
  _   EVALDATE     QUOTOSCHEM  EPPROD     *REFCST  EST/ENB  No
  _   STKOPT       CANSCRONN9  EPPROD     *PRIKEY
  _   CHKDEPT      EMPDIV2     EPPROD     *CHKCST  EST/ENB  No

옵션 2, 4, 6 또는 명령에 대한 매개변수
===>_____________________________________________________
F3=나감  F4=프롬트  F5=화면정리  F12=취소  F15=정렬 방법
F16=위치 반복         F17=위치  F22=제한조건명 표시
```

실제 파일 제한조건으로 작업 표시는 WRKPFCST 명령에서 지정된 파일에 대해 정의된 모든 제한조건을 보여줍니다. 또한 이 화면은 제한조건명, 파일명 및 라이브러리명을 나열합니다. 추가적으로 다음 정보가 표시됩니다.

- 유형 열은 제한조건을 참조, 검사, 고유, 1차 키로 나타냅니다.
- 상태 열은 제한조건이 정의되는지 또는 설정되는지와 사용 가능한지 아니면 사용 불가능한지를 나타냅니다. 상태 열은 참조 및 제한조건 검색에만 적용됩니다.
- 검사 지연 열에는 제한조건의 검사 지연 상태가 들어 있습니다. 항상 설정되어 사용 가능하므로 고유 및 1차 키 제한조건에는 상태가 없습니다.

각 나열된 제한조건에 대해서는 다음의 조치를 취할 수 있습니다.

- 제한조건을 허용 상태로 제한조건을 변경 또는 검사하려면 변경(옵션 2)을 선택하십시오. 예를 들어, 현재 사용 불가능 상태인 제한조건을 사용 가능하게 할 수 있습니다. 이 옵션은 CHGPFCST(실제 파일 제한조건 변경) 명령과 같은 기능을 수행합니다.

- 제한조건을 제거하려면 제거(옵션 4)를 선택하십시오. 이 옵션은 RMVPFCST(실제 파일 제한조건 제거) 명령과 같은 기능을 수행합니다.
- 검사 지연 상태의 레코드 표시하려면 표시(옵션 6)를 선택하십시오. 이 옵션은 DSPCPCST(검사 지연 제한조건 표시) 명령과 같은 기능을 수행합니다. DSPCPCST 명령은 참조 및 제한조건 검색에만 적용됩니다.

검사 지연 상태의 제한조건으로 작업:

참조 제한조건 또는 검사 제한조건을 추가할 경우 제한조건 정의를 충족하는 지 확인하기 위해 시스템은 자동적으로 데이터베이스 파일의 모든 레코드를 확인합니다. 제한조건이 유효하지 않거나 확인되지 않는 경우 시스템은 이 제한조건을 검사 지연 상태로 지정합니다.

이러한 검사는 시스템이 복원되는 경우에도 수행됩니다.

검사 지연 상태의 제한조건으로 작업하려면 다음 단계를 수행하십시오.

1. CHGPFCST(실제 파일 제한조건 변경) 명령을 실행하고 제한조건 상태 매개변수에 *DISABLED를 지정하여 제한조건을 비활성 상태로 만듭니다.
2. DSPCPCST(검사 지연 제한조건 표시) 명령을 실행하여 제한조건에 검사 지연으로 표시하는 레코드 리스트를 표시합니다.

   주: 명령 실행 시간 길이는 파일이 포함하는 레코드 수에 따라 다릅니다.
3. EDTCPCST(검사 지연 제한조건 편집) 명령을 실행하여 검사 지연 상태의 제한조건 확인 스케줄을 작성합니다.
4. CHGPFCST 명령을 다시 실행하고 제한조건 상태 매개변수에 *ENABLED를 지정하여 제한조건을 활성 상태로 만듭니다.

   관련 참조

   CHGPFCST(실제 파일 제한사항 변경) 명령

   DSPCPCST(확인 지연 제한사항 표시) 명령

   EDTCPCST(확인 지연 제한사항 편집) 명령

제한조건을 검사 지연 상태로 지정하는 레코드 표시:

DSPCPCST(검사 지연 제한조건) 명령을 사용하여 제한조건을 검사 지연 상태로 지정하는 레코드를 표시할 수 있습니다.

제한조건 규칙을 따르지 않는 레코드를 검사하는 것은 종종 유효합니다. 필요한 경우 레코드나 제한조건을 변경할 수 있습니다.

주: 다음 단계를 수행하기 전에 CHGPFCST(실제 파일 제한조건 변경) 명령을 실행하여 제한조건을 작동 불가능하게 해야 합니다.

제한조건이 검사 지연 상태가 되게 하는 레코드 리스트를 표시하거나 인쇄하려면 DSPCPCST 명령을 실행하십시오.

관련 참조

DSPCPCST(확인 지연 제한사항 표시) 명령

CHGPFCST(실제 파일 제한사항 변경) 명령

검사 지연 상태의 제한조건 처리:

큰 데이터베이스 파일에 대해 작성된 제한조건은 시스템에서 확인하는 데 시간이 많이 걸릴 수 있습니다. 지연 상태에 있는 제한조건을 나열하고 필요한 확인을 위해 스케줄할 수 있습니다.

검사 지연 상태의 제한조건 리스트를 표시하고 편집하려면 다음 단계를 수행하십시오.

1. EDTCPCST(검사 지연 제한조건 편집) 명령 실행.
2. 처리하려는 제한조건 상태 검사
3. 제한조건이 실행 또는 준비 완료 상태가 아니라면, 일렬 필드의 *HLD 값을 1과 99 사이 값으로 변경하십시오.
4. Enter를 누르십시오.

```
                              검사 지연 제한조건 편집

순서를 입력한 후 Enter 키를 누르십시오.
순서:   1-99, *HLD

                                ------------제한조건-----------     확인      경과
  Seq  상태      Cst         파일        라이브러리  시간      시간
  1    RUN       EMP1        DEP         EPPROD      00:01:00  00:00:50
1    READY     CONST   >   DEP         EPPROD      00:02:00  00:00:00
*HLD CHKPND    FORTH   >   STYBAK      EPPROD      00:03:00  00:00:00
  *HLD CHKPND    CST88       STYBAK      EPPROD      00:10:00  00:00:00
  *HLD CHKPND    CS317       STYBAK      EPPROD      00:20:00  00:00:00
  *HLD CHKPND    KSTAN       STYBAK      EPPROD      02:30:00  00:00:00




                                                                          맨 아래
F3=나감  F5=화면정리  F12=취소  F13=전체 반복   F15=정렬 방법
F16=위치 반복         F17=위치  F22=제한조건명 표시
```

세부사항: 검사 지연 상태의 제한조건 처리

검사 지연 상태의 제한조건 처리에 대한 추가 정보에는 검사 지연 제한조건 편집 화면의 상태 필드, 제한조건 및 확인 시간 열이 포함됩니다.

검사 지연 제한조건 편집 화면의 상태 필드에는 다음 값 중 하나가 포함됩니다.

- **RUN**은 제한조건이 확인 중임을 나타냅니다.
- **READY**는 제한조건이 확인될 준비가 되어 있음을 나타냅니다.
- **NOTVLD**는 제한조건과 연관된 액세스 경로가 유효하지 않음을 나타냅니다. 일단 액세스 경로가 재구성되면 시스템은 제한조건을 자동으로 확인합니다. 이 값은 참조 제한조건에만 적용됩니다.
- **HELD**는 제한조건이 확인 중이 아님을 나타냅니다. 이 상태를 변경하려면 순서를 1-99의 값으로 변경해야 합니다.

- **CHKPND**는 시스템이 제한조건을 확인하려는 시도가 있었으나, 제한조건이 여전히 검사 지연 상태임을 나타냅니다. 이 상태를 변경하려면 순서를 1-99의 값으로 변경해야 합니다.

제한조건 열에는 제한조건명의 처음 5자가 들어 있습니다. 5자를 넘을 경우 > 기호가 이름 뒤에 추가됩니다. 해당 행에 커서를 두고 F22 키를 누르면 긴 이름 전체를 표시할 수 있습니다.

확인 시간 열은 시스템에 다른 작업이 없을 경우 제한조건을 확인하는 데 걸리는 시간을 보여줍니다. 경과 시간 열은 제한조건을 확인하는 데 이미 소요된 시간을 나타냅니다.

관련 참조

EDTCPCST(확인 지연 제한사항 편집) 명령

고유 제한사항

고유 제한조건은 열이 고유한 지 확인하기 위해 데이터베이스의 제어로서 활동합니다.

예를 들면 데이터베이스의 고유 제한조건으로 고객 식별을 지정할 수 있습니다. 같은 고객 번호로 새 고객을 작성할 경우 오류 메세지가 데이터베이스 관리자에게 보내집니다.

고유 제한조건은 파일 내의 레코드에 고유한 값을 가진 데이터베이스 파일의 필드 또는 필드 세트를 지정해야 합니다. 파일은 오름차순이어야 하며 널을 허용할 수 있습니다.

하나의 파일이 복수의 고유 제한조건을 가질 수는 있으나, 고유 제한조건을 중복할 수는 없습니다. 순서에 관계없이 동일한 키 필드는 중복 제한조건을 구성합니다.

참조 제한조건을 추가할 경우 고유 제한조건을 상위 키로 사용할 수 있습니다.

1차 키 제한조건

1차 키 제한조건은 파일에 1차 액세스 경로가 되도록 하는 특수 속성을 가진 고유 키입니다.

1차 키 제한조건은 파일 내의 레코드에 고유한 값을 가진 데이터베이스 파일의 필드 또는 필드 세트를 지정합니다. 파일은 오름차순이어야 하며 널을 허용할 수 있습니다. 1차 키가 널(null) 허용일 경우 검사 제한조건이 내부적으로 추가되어 필드에 널값을 추가할 수 없게 됩니다. 하나의 파일에 대해 하나의 1차 키 제한조건만을 정의할 수 있습니다.

참조 제한조건을 추가할 때 1차 키를 상위 키로 사용할 수 있습니다.

검사 제한조건

검사 제한조건을 사용하여 필드 값이 데이터베이스 요구사항에 맞도록 필드 값의 한계를 유지보수할 수 있습니다.

검사 제한조건은 사용자가 정의하는 검사 제한조건 표현식에 대해 자료를 검사하여 삽입이나 갱신 중에 자료 유효성을 확인합니다.

예를 들어, 필드에 검사 제한조건을 작성하여 해당 필드에 삽입되는 값이 1에서 100 사이가 되도록 정의할 수 있습니다. 값이 지정된 범위에 해당되지 않을 경우 데이터베이스에 대한 삽입이나 갱신 조작이 처리되지 않습니다.

검사 제한조건은 상태 관점에서 보면 참조 제한조건과 매우 비슷합니다.

- 정의됨/사용 가능. 파일에 제한조건 정의가 추가되었으며, 제한조건이 설정된 후에 제한조건이 강제 적용됩니다.
- 정의됨/사용 불가능. 파일에 제한조건 정의가 추가되었으나 제한조건이 강제 적용되지 않습니다.
- 설정됨/사용 가능. 파일에 제한조건이 추가되었으며 모든 파일이 강제 적용을 위해 존재합니다.
- 설정됨/사용 불가능. 파일에 제한조건이 추가되었으며, 모든 파일이 강제 적용을 위해 존재하지만 제한조건이 강제 적용되지 않습니다.

참조 제한조건같은 검사 제한조건은 검사 지연 상태를 가질 수 있습니다. 필드의 자료가 검사 제한조건 표현식을 위반하면 제한조건은 검사 지연 상태가 됩니다. 레코드를 삽입 또는 갱신할 경우 자료가 검사 제한조건 표현식을 위반하면 삽입 및 갱신 조작이 허용되지 않습니다.

하나 이상의 LOB(큰 오브젝트) 필드를 포함하는 검사 제한조건은 LOB 필드가 없는 검사 제한조건보다 조작 범위를 더욱 적게 제한합니다. 검사 제한조건에는 하나 이상의 LOB 필드가 포함됩니다. LOB 필드들은 다음에 대해 직접적인 비교에만 관련될 수 있습니다.

- 유형 및 최대 길이가 같은 기타 LOB 필드
- 리터럴 값
- 널값

서브스트링이나 연결 조작과 같이 파생된 조작으로 알려진 조작들은 검사 제한조건에서 LOB 필드에 대해 허용되지 않습니다. LOB 필드에 대해 파생된 조작을 시도하는 검사 제한조건을 추가하려고 하면 진단 메세지 CPD32E6이 전송됩니다.

참조 제한조건으로 자료 무결성 확인

이 주제에서는 데이터베이스에 유효한 자료만 포함되도록 데이터베이스에서 참조 제한조건을 사용하는 방법에 대해 설명합니다.

iSeries 데이터베이스에서 참조 제한조건을 사용하여 시스템의 참조 무결성을 수행할 수 있습니다. 참조 무결성은 데이터베이스에 유효한 자료만 있는지 확인할 때 사용되는 모든 기술 및 기법을 포함합니다.

참조 제한조건 추가

하나 멤버로 실제 파일에 참조 제한조건을 추가할 수 있습니다. 참조 제한조건은 파일 레벨 속성입니다. 그러므로 멤버가 존재하기 전체 제한조건을 작성해야 합니다.

참조 제한조건을 추가하기 전에:

참조 제한조건을 추가하기 전에 다음 조건에 해당되는지 확인해야 합니다.

- 상위 키가 될 수 있는 키를 가진 상위 파일이 있어야 합니다. 필드 속성이 종속 파일의 외부 키 필드 속성과 맞는다면 시스템이 파일에 1차 키 제한조건을 상위 파일에 추가하려고 시도합니다. 상위 파일에 1차 키 또는 고유 제한조건이 없는 경우에는 잠재 상위 키의 필드 속성이 종속 파일의 외부 키 필드 속성과 맞는다면 시스템이 1차 키 제한조건을 상위 파일에 추가하려고 시도합니다.
- 상위 파일의 속성과 일치하는 특정 속성을 가진 종속 파일이 있어야 합니다.
  - 정렬 순서(SRTSEQ)는 자료 유형 CHAR, OPEN, EITHER 및 HEX에 일치해야 합니다.
  - CCSID는 하나 또는 모두 65535가 아니면, 각 SRTSEQ 표에 일치해야 합니다.
  - 각 정렬 순서표는 정확하게 일치해야 합니다.
- 종속 파일에는 다음의 상위 키 속성과 일치하는 외부 키가 있어야 합니다.
  - 자료 유형
  - 길이
  - 정밀도(팩, 존 또는 2진)
  - CCSID(65535코드화 문자 세트 ID(CCSID)를 가지고 있지 않은 경우)
  - REFSHIFT(자료 유형이 OPEN, EITHER 또는 ONLY일 경우)

참조 제한조건의 상위 파일 정의:

상위 파일은 최대 한 개의 멤버를 가진 실제 파일이어야 합니다. 상위 파일을 정의할 경우 새 파일을 작성하거나 기존 파일을 사용할 수 있습니다.

상위 키의 개념은 참조 제한조건의 용어에만 적용합니다. 참조 제한조건이 종속 파일에 추가된 경우 상위 파일에 대해 상위 키가 필요합니다. 이를 위해 먼저 1차 키 제한조건이나 고유 제한조건을 이 키에 대한 해당 필드 세트가 있는 상위 파일에 추가해야 합니다. 참조 제한조건이 추가된 경우 탐색은 대응하기 위해 고유 제한조건(및 1차 키)을 수행합니다. 대응이 되면 그다음 제한조건의 액세스 경로는 참조 제한조건 관계의 상위 키로 사용됩니다.

새로운 실제 파일을 상위 파일로 작성하려면 다음 단계를 수행하십시오.

1. CRTPF(실제 파일 작성) 명령을 사용하여 파일을 작성하십시오.
2. ADDPFCST(실제 파일 제한조건 추가) 명령을 사용하여 1차 키 제한조건이나 고유 제한조건에 추가하십시오. 1차 키는 널(null)을 허용하지만 시스템은 필드의 널값 삽입을 막기 위해 암시 검사 제한조건을 작성합니다.

주: SQL CREATE TABLE문을 사용하면 이전의 여러 단계를 한 단계로 수행할 수 있습니다.

기존 파일을 상위 파일로 사용하려면 다음 옵션에서 선택하십시오.

- ADDPFCST 명령을 사용하여 파일에 1차 키 제한조건을 추가할 수 있습니다. TYPE 매개변수에 대해 *PRIKEY를 지정하십시오. 또한 KEY 매개변수에 키 필드나 필드들을 지정해야 합니다.

1차 키가 파일 제한조건이 이미 존재할 경우 하나의 파일은 하나의 1차 키만을 가질 수 있으므로 TYPE(*PRIKEY)이 지정된 ADDPFCST 명령은 실패하게 됩니다. 다른 1차 키 제한조건을 원하면 먼저 RMVPFCST(실제 파일 제한조건 제거) 명령으로 기존 1차 키 제한조건을 제거해야 합니다. 그 후에야 새로운 1차 키 제한조건을 추가할 수 있습니다.

- ADDPFCST 명령을 사용하여 파일에 고유 제한조건을 추가할 수 있습니다. TYPE 매개변수에 대해 *UNQCST를 지정하십시오. 또한 KEY 매개변수에 키 필드나 필드들을 지정해야 합니다. SQL ALTER TABLE 명령문을 사용하여 고유 제한조건을 추가할 수도 있습니다.

  상위 키로 사용할 1차 키 또는 고유 제한조건을 가지고 있지 않은 경우 시스템은 자동적으로 참조 제한조건을 추가할 때 1차 키 제한조건을 추가하려 할 것입니다.

  상위 파일이 고유 키순 액세스 경로를 가진다면(액세스 경로 필드는 키 수와 속성에 대해 외부 키 필드와 일치), 1차 키 제한조건은 상위 파일에 암시적으로 추가가 될 것입니다. 이는 참조 제한조건에 대해 상위 키가 될 것입니다.

  상위 파일이 도달순 액세스 경로 가진다면(상위 키에 대한 지정 필드가 속성에 외부 키 필드와 일치), 1차 키 제한조건은 상위 파일에 암시적으로 추가가 될 것입니다. 이는 참조 제한조건에 대해 상위 키가 될 것입니다.

1차 키를 정의할 수 없을 때의 처리 방법

1차 키 또는 고유 제한조건이 있는 기존 파일의 경우 어느 것도 상위 키로 충분하지 않으면 다음 옵션 중에서 선택할 수 있습니다.

상위 키를 정의할 수 없을 경우 다음 조치 중 하나를 선택할 수 있습니다.

- 파일을 삭제한 후 적절한 키로 다시 작성함.
- 작성된 파일에 고유 키 또는 1차 키 제한조건을 추가함.

  관련 참조

  실제 파일 작성(CRTPF) 명령

  ADDPFCST(실제 파일 제한사항 추가) 명령

  RMVPFCST(실제 파일 제한사항 제거) 명령

참조 제한조건의 종속 파일 정의:

종속 파일은 최대 하나의 멤버를 가진 실제 파일이어야 합니다. 종속 파일을 작성하려면 실제 파일대로 작성하거나 기존의 파일을 사용할 수 있습니다.

실제 제한조건을 작성할 때 종속 파일이 키순 액세스 경로를 가질 필요는 없습니다. 기존의 액세스 경로가 외부 키 기준에 맞지 않을 경우에는 시스템은 하나의 액세스 경로를 파일에 추가합니다.

참조 제한조건 규칙 지정:

참조 제한조건은 레코드를 삭제하거나 갱신할 때 시스템이 강제로 하려는 규칙을 지정합니다.

참조 제한조건으로 강제로 하려는 규칙을 지정하려면 다음 단계를 수행하십시오.

1. ADDPFCST(실제 파일 제한조건 추가) 명령을 실행하십시오.
2. DLTRULE 매개변수에 대해 값을 선택하여 레코드를 삭제할 때(*규칙 삭제*), 강제로 하려는 규칙을 지정하십시오.
3. UPDRULE 매개변수에 대해 값을 선택하여 레코드를 갱신할 때(*규칙 갱신*), 강제로 하려는 규칙을 지정하십시오.

iSeries Navigator를 사용하여 참조 제한조건을 추가할 수도 있습니다.

관련 개념

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

ADDPFCST(실제 파일 제한사항 추가) 명령

세부사항: 참조 제한조건 삭제 규칙 지정:

이 주제에서는 DLTRULE 매개변수에 사용할 수 있는 값 다섯 개를 나열합니다. 삭제 규칙은 상위 키 값을 삭제할 경우 시스템이 활동할 조치를 지정합니다. 삭제 규칙은 널 키 값에 영향을 주지 않습니다.

- *NOACTION(디폴트 값)
  - 상위 키 값에 대응하는 외부 키 값이 있을 경우에는 상위 파일에서의 레코드 삭제가 발생하지 않습니다.
- *CASCADE
  - 상위 파일에서의 레코드 삭제로 인해 상위 키 값이 외부 키 값과 일치하면 종속 파일의 레코드가 삭제됩니다.
- *SETNULL
  - 상위 파일에서의 레코드 삭제로 인해 널이 아닌 상위 키 값이 외부 키 값과 일치할 때 종속 파일의 레코드가 갱신됩니다. 선행 기준을 충족시키는 종속 레코드에 대해 외부 키 내의 모든 널 허용 필드가 널로 설정됩니다. 널이 아닌 속성을 가진 외부 키 필드는 갱신되지 않습니다.
- *SETDFT
  - 상위 파일에서의 레코드 삭제로 인해 널이 아닌 상위 키 값이 외부 키 값과 일치할 때 종속 파일의 레코드가 갱신됩니다. 선행 기준을 충족시키는 종속 레코드에 대해 외부 키 필드 또는 필드들이 해당 디폴트 값으로 설정됩니다.
- *RESTRICT
  - 상위 키 값에 대응하는 외부 키 값이 있을 경우에는 상위 파일에서의 레코드 삭제가 발생하지 않습니다.

  주: 삭제가 시도될 때 시스템은 삭제 *RESTRICT 규칙을 즉시 강제합니다. 조작이 논리적으로 종료될 때 시스템은 다른 제한조건을 강제합니다. 다른 제한조건의 경우 조작에는 삭제 조작 전후에 실행되는 트

리거 프로그램이 포함됩니다. 트리거 프로그램은 발생할 수 있는 참조 무결성 위반을 정정할 수 있습니다. 예를 들어, 상위 레코드가 존재하지 않으면 트리거 프로그램이 이를 추가할 수 있습니다. *RESTRICT 규칙은 위반을 예방하지 않습니다.

세부사항: 참조 제한조건 갱신 규칙 지정:

이 주제에서는 UPDRULE 매개변수에 사용할 수 있는 값을 나열합니다. UPDRULE 매개변수는 상위 파일과 종속 파일 간의 제한조건 관계에 대한 갱신 규칙을 식별합니다. 갱신 규칙은 시스템이 상위 파일을 갱신할 때 취해질 조치를 지정합니다.

- *NOACTION(디폴트 값)
  - 종속 파일에 대응하는 외부 키 값이 있을 경우에는 상위 파일에서의 레코드 갱신이 발생하지 않습니다.
- *RESTRICT
  - 널이 아닌 상위 키의 값이 외부 키 값과 일치하지 않을 경우에는 상위 파일에서의 레코드 갱신이 발생하지 않습니다.

  주: 갱신 조작을 시도할 때 시스템은 갱신 *RESTRICT 규칙을 즉시 강제합니다. 조작이 논리적으로 종료될 때 시스템은 다른 제한조건을 강제합니다. 예를 들어, 상위 레코드가 존재하지 않으면 트리거 프로그램이 이를 추가할 수 있습니다. *RESTRICT 규칙은 위반을 예방하지 않습니다.

세부사항: 참조 제한조건 규칙 저널링 요구사항 지정:

참조 제한조건과 연관되어 있는 파일에서 삽입, 갱신 또는 삭제를 수행 중이나 삭제 규칙, 갱신 규칙 또는 모두 *RESTRICT가 아닐 경우에는 저널링을 사용해야 합니다.

상위 파일과 종속 파일 모두 같은 저널 리시버에 저널되어야 합니다. 또한 사용자는 상위 파일과 종속 파일에 대한 저널링을 시작해야 할 책임이 있습니다.

사용자는 STRJRNPF(실제 파일 저널 시작) 명령으로 상위 파일과 종속 파일에 대한 저널링을 시작해야 할 수 있습니다.

삭제 규칙, 갱신 규칙 또는 모두 *RESTRICT가 아닌 참조 제한조건과 연관되어 있는 파일에서 레코드의 삽입, 갱신 또는 삭제를 수행할 경우에는 확약 제어가 필요합니다. 확약 제어를 시작하지 않은 경우에는 시스템은 사용자를 위해 자동으로 확약 주기를 시작하고 종료합니다.

관련 참조

실제 파일 저널 시작(STRJRNPF) 명령

세부사항: 참조 제한조건 추가:

다음은 참조 제한조건에 적용되는 제한 리스트입니다.

- 상위 파일은 실제 파일이어야 합니다.
- 상위 파일은 최대 한 개의 멤버(MAXMBR(1))를 가질 수 있습니다.
- 종속 파일은 실제 파일이어야 합니다.

- 종속 파일은 최대 한 개의 멤버(MAXMBR(1))를 가질 수 있습니다.
- 종속 파일과 상위 파일 중 하나 또는 모두 멤버가 없을 경우에도 제한조건을 정의할 수 있습니다. 두 파일 모두가 멤버를 갖는 경우가 아니면 제한조건이 정의될 수 없습니다.
- 한 파일에서 1차 키는 최대 한 개를 포함할 수 있으나 고유 제한조건은 여러 개를 포함할 수 있습니다.
- 파일당 최대 300개의 제한조건 관계가 있습니다. 이 최대값은 다음의 합계입니다.
  - 참조 제한조건이 상위(parent)로 관여할 것인지 아니면 종속으로 관여할 것인지와 제한조건이 정의되거나 설정될 것인지의 여부
  - 1차 키 제한조건을 포함하는 고유 제한조건
  - 검사 제한조건
- 외부 서술 파일만이 참조 제한조건에서 허용됩니다. 소스 파일은 허용되지 않습니다. 프로그램 서술 파일은 허용되지 않습니다.
- 삽입, 갱신 또는 삭제 기능을 가진 파일은 *RESTRICT 관계에서 허용되지 않습니다.
- 제한조건명은 하나의 라이브러리 내에서 고유해야 합니다.
- 제한조건을 QTEMP 라이브러리의 파일에는 추가할 수 없습니다.
- 상위 파일이 하나의 ASP에 있고 종속 파일이 다른 ASP 참조 제한조건을 추가할 수 없습니다.

세부사항: 제한조건 주기 회피:

*제한조건 주기*는 상위 파일의 자(descendent)가 원래 상위 파일에 대해 상위(parent)가 되는 제한조건 관계의 순서입니다.iSeries용 DB2 Universal Database 데이터베이스에서 제한조건 주기를 사용할 수는 있으나 사용하지 않는 것이 좋습니다.

참조 제한조건 확인

ADDPFCST(실제 파일 제한조건 추가) 명령으로 제한조건을 추가할 경우 시스템은 자동으로 참조 제한조건의 유효성을 확인합니다. 시스템은 외부 키의 널값이 아닌 모든값이 상위 파일의 대응값과 일치하는지 확인합니다.

확인이 성공적일 경우 제한조건 규칙은 사용자나 어플리케이션 프로그램에 의해 후속 액세스에 강제됩니다. 확인이 실패하면 제한조건이 검사 지연 상태로 마크됩니다. 제한조건이 ADDPFCST 명령으로 추가된 경우 제한조건은 검사 지연 상태이나, 작동 불가능 상태입니다.

주: 많은 양의 자료가 들어 있는 기존 파일에 참조 제한조건을 추가하는 것은 흔히 있는 일입니다. 많은 수의 레코드가 관련된 경우 ADDPFCST 명령은 완료하는 데 여러 시간이 걸릴 수 있습니다. 추가 처리는 파일에서 배타적으로 잠급니다. 참조 제한조건을 추가하기 전에 시간 요인 및 파일 가용성을 고려해야 합니다.

참조 제한조건을 작동 가능 또는 작동 불가능하게 설정

CHGPFCST(실제 파일 제한조건 변경) 명령이나 iSeries Navigator를 사용하여 참조 제한조건을 작동 가능 또는 작동 불가능하게 설정할 수 있습니다.

참조 제한조건 관계를 작동 가능 또는 작동 불가능하게 설정하려면 CHGPFCST 명령을 사용하십시오. 제한조건을 변경할 때 종속 파일을 지정하여야 합니다. 상위 파일을 지정해서는 제한조건을 사용 불가능하게 하거나 사용 가능하게 할 수 없습니다.

iSeries Navigator을 사용하여 참조 제한조건을 작동 가능 또는 작동 불가능으로 설정할 수도 있습니다.

제한조건을 사용 가능하게 하거나 사용 불가능하게 하려면 최소한 종속 파일에 대한 오브젝트 관리 권한(또는 ALTER 특권)을 갖고 있어야 합니다.

세부사항: 참조 제한조건을 작동 가능 또는 작동 불가능하게 설정

시스템에서 제한조건을 작동 가능 또는 작동 불가능으로 설정하면 상위 파일 및 종속 파일, 두 멤버, 두 액세스 경로가 잠깁니다. 작동 가능 또는 작동 불가능 조작이 완료되면 잠금이 제거됩니다.

사용 가능한 제한조건을 사용 가능하게 하거나 사용 불가능한 제한조건을 아무것도 수행할 수 없도록 사용 불가능하게 하는 조치는 정보용 메세지를 발행할 뿐입니다.

설정됨/사용 불가능 또는 검사 지연 제한조건 관계는 사용 가능해 질 수 있습니다. 작동 가능은 시스템이 제한조건을 다시 확인하도록 합니다. 확인 결과 상위 키와 외부 키 사이에 불일치가 발견되면 제한조건이 검사지연으로 표시됩니다.

사용자가 올바른 권한을 갖고 있을 경우 제한조건 관계를 사용 불가능하게 설정하면 상위 및 종속 파일에 대한 모든 파일 입/출력(I/O) 조작이 허용됩니다. 제한조건의 전체 하부 구조는 남아 있습니다. 상위 키 및 외부 키 액세스 경로는 그대로 유지보수됩니다. 그러나 사용 불가능한 관계에 있는 두 개의 파일에 대해서는 어떤 참조 강제도 수행되지 않습니다. 남아 있는 모든 사용 가능한 제한조건은 그대로 강제됩니다.

제한조건을 사용 불가능하게 하면 성능 임계 상황의 파일 I/O 조작이 보다 빠르게 이루어지도록 할 수 있습니다. 이런 상황에서는 트레이드 오프(trade-off)를 고려해야 합니다. 파일 자료는 참조적으로 유효하지 않을 수 있습니다. 제한조건이 작동할 수 있으면, 파링 크기에 따라 다르지만 시스템이 참조 제한조건 관계를 재확인하는 데 시간이 걸립니다.

주: 사용자와 어플리케이션은 설정 및 사용 불가능함 상태의 제한조건 관계를 가진 파일을 수정할 때 주의해야 합니다. 제한조건이 다시 사용 가능하게 될 때까지는 관계가 위반될 수 있으나 재검색할 수는 없습니다.

제한조건 관계가 작동 불가능한 동안 ALCOBJ(오브젝트 허용) 명령은 파일을 할당(잠글)할 수 있습니다. 참조 제한조건 관계가 사용 불가능한 동안 ALCOBJ로 파일 변경을 막을 수 있습니다. 이 명령에 배타적 읽기 잠금 상태(*EXCLRD)를 설정하면 다른 사용자가 파일을 읽을 수 있습니다. 제한조건이 다시 작동 가능하게 되면 DLCOBJ(오브젝트 할당해제) 명령이 파일 잠금을 해제합니다.

복수의 제한조건을 사용 가능하게 하거나 사용 불가능하게 할 때 이 조건들은 순차적으로 처리됩니다. 제한조건이 수정될 수 없으면, 진단 메세지가 수신되고 기능은 리스트상의 다음 제한조건으로 진행됩니다. 모든 제한조건이 처리되면 수정된 제한조건의 수를 나열하는 완료 메세지가 수신됩니다.

관련 개념

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

CHGPFCST(실제 파일 제한사항 변경) 명령

ALCOBJ(오브젝트 할당) 명령

DLCOBJ(오브젝트 할당해제) 명령

참조 제한조건 제거

여러 방법으로 참조 제한조건을 제거할 수 있습니다. 제거가 전체에 미치는 영향은 제거할 제한조건과 제한조건 주위의 특정 조건에 따라 다릅니다.

참조 제한조건을 제거하려면 다음 단계를 수행하십시오.

1. RMVPFCST(실제 파일 제한조건 제거) 명령을 실행하십시오.
2. CST 및 TYPE 매개변수를 사용하여 제거할 제한조건을 지정하십시오.
   - 모든 제한조건 또는 특정 제한조건을 지정하려면 CST 매개변수를 사용하십시오.
   - 제한조건의 특별 유형을 지정하려면 TYPE 매개변수를 사용하십시오.

iSeries Navigator를 사용하여 참조 제한조건을 제거할 수도 있습니다.

참조 제한조건을 제거할 때 시스템은 연관되는 외부 키와 액세스 경로가 파일로부터 제거합니다. 시스템의 논리 파일 또는 다른 제한조건이 외부 키 액세스 경로를 사용한다면 시스템은 외부 키 액세스 경로를 제거하지 않습니다.

참조 제한조건, 1차 키 제한조건 또는 고유 제한조건을 제거하고, 연관되는 액세스 경로가 논리 파일에 의해 공유될 경우 공유된 경로의 소유권은 논리 파일로 이전됩니다.

관련 개념

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

RMVPFCST(실제 파일 제한사항 제거) 명령

세부사항: CST 매개변수를 사용하여 제한조건 제거:

이 주제에서는 CST 매개변수로 제거할 수 있는 제한조건에 대해 설명합니다.

CST 매개변수를 사용하여 다음을 제거하도록 지정할 수 있습니다.

- TYPE(*ALL)이 지정된 파일과 연관된 모든 제한조건 CST(*ALL)
- 특정 참조 제한조건 CST(*제한조건명*)
- CST(*CHKPND) 검사 지연 상태의 참조 또는 검사 제한조건
- 제한조건 특정 TYPE과 연관된 모든 제한조건 CST(*ALL)

세부사항: TYPE 매개변수를 사용하여 제한조건 제거:

이 주제에서는 TYPE 매개변수로 제거할 수 있는 제한조건에 대해 설명합니다.

TYPE 매개변수를 사용하여 다음을 제거하도록 제한조건의 유형을 지정할 수 있습니다.

- 모든 유형: TYPE(*ALL)
  - CST(*ALL)의 경우 모든 제한조건
  - CST(*CHKPND)의 경우 검사 지연 상태의 모든 제한조건
  - CST(제한조건명)의 경우 명명된 제한조건
- 참조 제한조건: TYPE(*REFCST)
  - CST(*ALL)의 경우 모든 참조 제한조건
  - CST(*CHKPND)의 경우 검사 지연 상태의 모든 참조 제한조건
  - CST(제한조건명)의 경우 명명된 참조 제한조건
- 고유 제한조건: TYPE(*UNQCST)
  - CST(*ALL)의 경우 1차 키를 제외한 모든 고유 제한조건
  - CST(*CHKPND)에 적용할 수 없음—고유 제한조건은 검사 지연 상태에 있을 수 없습니다.
  - CST(제한조건명)가 지정될 경우 명명된 고유 제한조건
- 1차 키 제한조건: TYPE(*PRIKEY)
  - CST(*ALL)의 경우 1차 제한조건
  - CST(*CHKPND)에 적용할 수 없음--1차 제한조건은 검사 지연 상태에 있을 수 없습니다.
  - CST(제한조건명)가 지정될 경우 명명된 1차 제한조건
- 검사 제한조건: TYPE(*CHKCST)
  - CST(*ALL)의 경우 모든 검사 제한조건
  - CST(*CHKPND)의 경우 검사 지연 상태의 모든 검사 제한조건
  - CST(제한조건명)가 지정될 경우 명명된 검사 제한조건

세부사항: 참조 제한조건으로 자료 무결성 확인

여러 가지 이유로 데이터베이스 관리 시스템의 참조 무결성을 사용하기를 원할 수 있습니다.

참조 제한조건으로 자료 무결성을 확인해야 할 경우가 있습니다.

- 파일간의 자료값이 업무 규칙에 맞는지 확인하기 위해 예를 들어, 업무상 하나의 파일에는 고객의 리스트를, 다른 파일에는 고객 계정의 리스트를 가지고 있는 업무를 고려해 보십시오. 연관되는 고객이 존재하지 않는다면 추가 계정을 허용하지 않습니다. 마찬가지로 모든 계정을 삭제할 때까지는 고객을 삭제하는 것이 타당하지 않습니다.
- 자료 값들 간의 관계를 정의할 수 있게 하기 위해.
- 어떤 어플리케이션이 변경되든지간에 시스템이 자료 관계를 강제하도록 하기 위해.

- 검사를 데이터베이스로 이동시켜 고급 언어(HLL) 또는 SQL 레벨에서 수행되는 무결성 검사의 성능을 향상시키기 위해.

예: 참조 제한조건으로 자료 무결성 확인

다음은 참조 제한조건으로 자료 무결성을 확인하는 방법의 예입니다.

데이터베이스에는 사원 파일과 부서 파일이 포함되어 있습니다. 두 파일에는 모두 DEPTNO라고 명명된 부서 번호 필드가 들어 있습니다. 이 데이터베이스 파일의 관련 레코드는 department.DEPTNO와 동일한 employee.DEPTNO에 대한 레코드입니다. 이 예는 사원 파일의 각 사원이 부서 파일에 속하는 대응하는 부서를 가지고 있는 지 확인합니다. 참조 제한조건으로 이를 완수할 수 있습니다.

1. ADDPFCST(실제 파일 제한조건 추가) 명령을 사용하여 1차 키 제한조건 또는 고유 제한조건을 DEPTNO 필드에 대한 부서 파일에 추가하십시오. 이 1차 키 또는 고유 제한조건이 나중에 상위 키가 됩니다. 참조 제한조건은 아직 추가되지 않았으므로 상위 키는 아직 아닙니다.
2. ADDPFCST 명령을 사용하여 참조 제한조건을 사원 파일에 추가하십시오. 사원 파일은 종속 파일이 될 것입니다. 외부 키는 employee.DEPTNO가 됩니다. 부서 파일은 상위 키 department.DEPTNO를 가진 상위 파일이 될 것입니다. 1차 키 제한조건 또는 키로서 DEPTNO 필드와 고유 제한조건이 있으므로 제한조건은 참조 제한조건과 연관된 상위 키로서 작동됩니다.

참조 제한조건에는 상위 파일이나 종속 파일에서 레코드 삽입, 갱신 및 삭제 조작 시 따라야 할 갱신 및 삭제 규칙이 있습니다.

참조 무결성 용어

참조 무결성에 대해 논의하려면 몇 가지 용어에 대해 이해하고 있어야 합니다. 이 용어들은 용어간의 관계에 대한 이해를 도울 수 있는 순서로 되어 있습니다.

1차 키 제한조건

고유한, 오름차순으로써, 널값을 포함할 수 없는 데이터베이스 파일 내의 필드 또는 필드 세트. 1차 키는 1차 파일 액세스 경로입니다. 참조 제한조건을 추가할 때 1차 키를 상위 키로 사용할 수 있습니다. 1차 키 제한조건은 특수 속성을 가진 고유 키 제한조건입니다.

고유 제한조건

고유한, 오름차순으로써, 널값을 포함할 수 있는 데이터베이스 파일 내의 필드 또는 필드 세트.

상위 키

고유한 오름차순이어야 하며, 널값을 포함할 수도 있고 포함하지 않을 수도 있는 데이터베이스 파일 내의 필드 또는 필드 세트. 상위 파일의 상위 키는 참조 제한조건을 종속 파일에 추가하는 데 사용될 수 있습니다. 상위 키는 1차 키 이거나 고유 제한조건이어야 합니다.

외부 키

관련된 상위 파일의 값과 일치해야 하는 널이 아닌 각 값의 필드 또는 필드 세트.

그러나 속성(자료 유형, 길이 등)은 상위 키의 1차 키와 같아야 합니다.

상위 파일

상위 키를 포함하는 참조 제한조건 관계의 파일.

종속 파일

외부 키를 포함하는 참조 제한조건 관계의 파일. 종속 파일은 상위 파일에 종속됩니다. 즉, 종속 파일의 외부 키의 널값이 아닌 모든 값에 대해서는 상위 파일의 상위 키에 대응되는 널값이 아닌 값이 있어야 합니다.

검사 지연

데이터베이스가 참조 제한조건에 대해 다음이 참인지 확실히 알 수 없을 때 일어나는 상태. 종속 파일의 외부 키의 널값이 아닌 모든 값에 대해서는 상위 파일의 상위 키에 대응되는 널값이 아닌 값이 있어야 합니다.

삭제 규칙

상위 레코드를 삭제하려는 시도가 있을 때 데이터베이스가 취해야 하는 조치에 대한 정의

갱신 규칙

상위 레코드를 갱신하려는 시도가 있을 때 데이터베이스가 취해야 하는 조치에 대한 정의

참조 무결성 수행

설정되고 사용 가능한 제한조건과 연관된 파일에 대한 I/O 액세스는 여러 가지가 있습니다. 이는 파일에 제한조건 관계의 상위 키가 들어 있는지 아니면 외부 키가 들어 있는지에 따라 달라집니다. 시스템은 참조 무결성 강제를 모든 상위 및 종속 파일의 I/O 요구에 대해 수행합니다.

데이터베이스는 어플리케이션 프로그램 또는 시스템 명령(예: INZPFM(실제 파일 멤버 초기화) 명령) 또는 SQL 문 또는 파일 I/O 유틸리티(예: STRSEU)에 관한 제한조건 규칙을 수행합니다.

외부 키 수행:

제한조건 작성 시 지정된 삭제 및 갱신 규칙이 상위 키의 변경에 적용됩니다. 참조 무결성을 유지보수하기 위해 데이터베이스는 외부 키에 대한 갱신 및 삽입 조작에 아무 조치 취하지 않음(no-action) 규칙을 수행합니다. 시스템은 널이 아닌 모든 외부 키 값이 상위 키 값과 일치하도록 외부 키 갱신 및 삽입 시 이 규칙을 수행합니다.

시스템은 대응 상위 키가 새로운 외부 키 값에 대해 존재하지 않을 경우에는 참조 제한조건 위반을 리턴하며 종속 레코드를 삽입 또는 갱신하지 않습니다.

상위 키 수행:

참조 제한조건에 지정하는 규칙에 따라 데이터베이스가 상위 키에 삭제 및 갱신 조작을 처리하는 방법이 결정됩니다. 시스템은 상위 키의 고유 속성을 모든 상위 파일 I/O에 수행합니다.

규칙 삭제 수행:

레코드를 상위 파일에서 삭제할 경우 시스템은 종속 파일에서 종속 레코드(널이 아닌 외래 키 값과 일치하는)를 검사합니다. 시스템이 종속 레코드를 발견하면 삭제 규칙에 따라 시행할 조치가 결정됩니다.

- **조치 없음**. 시스템이 종속 레코드를 발견할 경우 제한조건 위반을 리턴하고 레코드를 삭제하지 않습니다.
- **연쇄**. 시스템은 종속 파일에서 발견한 종속 레코드를 삭제합니다.

- **널 설정**. 시스템은 발견한 모든 종속 레코드에서 외부 키의 널 허용 필드를 널로 설정합니다.
- **디폴트 설정**. 시스템은 일치하는 상위 키를 삭제할 경우 외부 키의 모든 필드를 디폴트 값으로 설정합니다.
- **제한**. 실행이 즉각적이라는 점을 제외하면 조치 없음과 동일합니다.

삭제 규칙의 부분 강제가 실패할 경우 전체 삭제 조작도 실패하며 연관된 모든 변경이 롤백됩니다. 예를 들어, 삭제 연쇄 규칙은 데이터베이스가 10개의 종속 레코드를 삭제하게 하지만 마지막 레코드를 삭제할 때 시스템 장애가 발생합니다. 데이터베이스는 상위 키 레코드의 삭제를 허용하지 않으며 삭제된 종속 레코드가 다시 삽입됩니다.

참조 제한조건 강제가 레코드를 변경하는 경우 연관된 저널 항목은 레코드 변경이 참조 제한조건에 의해 이루어졌음을 나타내는 인디케이터를 연관된 저널 값을 갖습니다. 예를 들어, 삭제 연쇄 규칙에 의해 삭제된 종속 레코드는 참조 제한조건 강제가 이루어지는 동안 레코드가 변경되었음을 나타내는 저널항목 인디케이터를 갖게 됩니다.

규칙 갱신 수행:

시스템이 상위 키가 상위 파일을 갱신할 때 종속 파일 내에 종속 레코드(대응하는 널이 아닌 외부 키 값)가 있는지 검사합니다. 종속 레코드가 발견되는 경우 제한조건 관계에 대한 갱신 규칙은 시행할 조치를 판별합니다.

- **조치 없음**. 시스템이 종속 레코드를 발견할 경우 제한조건 위반을 리턴하고 레코드를 갱신하지 않습니다.
- **제한**. 시스템이 위와 동일한 작업을 수행하지만 즉시 실행됩니다.

제한조건 상태

파일은 세 가지 제한조건 상태 중 하나에 있게 됩니다. 이 중 두 가지 상태에서 제한조건이 사용 가능하게 되거나 사용 불가능하게 됩니다.

- **비제한조건 관계 상태**. 이 상태에서는 파일에 대한 어떤 참조 제한조건도 존재하지 않습니다. 파일에 대한 제한조건 관계가 일단 존재하면 이에 관한 모든 정보가 제거됩니다.
- **정의된 상태**. 제한조건 관계가 종속 파일과 상위 파일 사이에서 정의됩니다. 제한조건 관계를 정의하는 데 있어서 둘 중 어느 파일에도 멤버를 작성할 필요는 없습니다. 정의된 상태에서는 다음의 제한조건이 있을 수 있습니다.
  - 정의됨/사용 가능. 정의됨/사용 가능 제한조건 관계는 정의만을 목적으로 합니다. 제한조건에 대한 규칙은 강제되지 않습니다. 이 상태에서의 제한조건은 설정된 상태가 되면 사용 가능하게 됩니다.
  - 정의됨/사용 불가능. 사용 불가능하다고 정의된 제한조건 관계는 정의만을 목적으로 합니다. 제한조건에 대한 규칙은 강제되지 않습니다. 이 상태에서 설정된 상태가 되면 제한조건은 사용 불가능하게 됩니다.
- **설정된 상태**. 종속 파일은 상위 파일과의 제한조건 관계를 가집니다. 외부 키와 상위 키간의 속성이 일치할 경우에만 제한조건이 설정됩니다. 두 파일 모두에 대해 멤버가 존재해야 합니다. 설정된 상태에서의 제한조건은 다음과 같습니다.
  - 설정됨/사용 가능. 사용 가능하게 설정된 제한조건 관계는 데이터베이스가 참조 무결성을 강제하게 합니다.

– 설정됨/사용 불가능. 사용 불가능하게 설정된 제한조건 관계는 데이터베이스가 참조 무결성을 강제하지 않도록 합니다.

참조 제한조건의 검사 지연 상태

*검사 지연*은 상위 키와 외부 키간에 잠재적인 불일치가 존재할 때 제한조건 관계의 상태입니다. 시스템에서 참조 무결성이 위반되었다고 판단하면 제한조건 관계는 검사 지연으로 표시됩니다.

예를 들면,

- 종속 파일의 자료만이 복원되고, 이 자료가 더 이상 시스템의 상위 파일과 동기화되지 않는(외부 키가 상위(parent)를 갖지 않음) 경우의 복원 조작
- 시스템 장애로 인해 대응하는 외부 키가 존재할 때 상위 키 값이 삭제될 수 있습니다. 이는 종속 및 상위 파일이 저널될 때만 발생할 수 있습니다.
- 외부 키 값에 상응하는 상위 키 값이 없습니다. 이는 이전에 한 번도 제한조건 관계의 일부가 된 적이 없었던 기존의 파일에 참조 제한조건을 추가할 때만 발생할 수 있습니다.

검사 지연 상태는 *NO 아니면 *YES 중 하나입니다.

검사 지연은 설정된 상태의 제한조건에만 적용됩니다. 설정 및 사용 가능 참조 제한조건은 *YES 또는 *NO의 검사 지연 상태를 가질 수 있습니다.

제한조건 관계를 검사 지연 상태에서 벗어나게 하려면 관계를 사용 불가능하게 하고, 키(외부, 상위 또는 둘 다) 자료를 정정한 후 제한조건을 다시 사용 가능하게 해야 합니다. 그러면 데이터베이스는 제한조건 관계를 재확인합니다.

관계가 검사 지연 상태에 있을 때 모 및 종속 파일은 사용을 제한하는 상태에 있게 됩니다. 상위 파일 I/O 제한조건은 종속 파일 제한조건과는 다릅니다. 검사 지연 제한조건은 설정 상태 및 사용 불가능 상태(항상 검사 지연 상태에 있음)에 있는 제한조건에는 적용되지 않습니다.

검사 지연 상태의 종속 파일 제한사항:

이 주제에서는 검사 지연 상태의 설정 및 사용 가능 참조 제한조건에 적용하는 제한조건을 간단하게 소개합니다.

검사 지연으로 표시된 제한조건 관계의 종속 파일에 대해서는 어떤 파일 입/출력(I/O) 조작도 수행할 수 없습니다. 종속 파일 및 상위 파일간의 파일 비일치를 올바르게 해야 합니다. 또한 시스템이 I/O 조작하기 이전에 그 관계를 검사 지연 상태가 되지 않게 해야 합니다. 사용자나 어플리케이션이 검사 지연 상태와 제한조건 위반에 대해 모를 수 있으므로 시스템은 그런 파일에서 레코드 읽기를 허용하지 않습니다.

검사 지연 상태의 사용 가능한 참조 제한조건이 있는 종속 파일에서 I/O 조작을 수행하려면 먼저 이 제한조건을 작동 불가능하게 만든 다음 원하는 I/O 조작을 수행해야 합니다.

검사 지연 상태의 상위 파일 제한사항:

이 주제에서는 검사 지연 상태의 설정 및 사용 가능 참조 제한조건에 적용하는 제한조건을 간단하게 소개합니다.

시스템이 검사지연으로 표시한 제한조건 관계의 상위 파일을 열 수는 있으나 수행 가능한 I/O 조작 유형에 제한이 있습니다. 레코드를 읽고 삽입할 수 있으나 삭제 또는 갱신할 수 없습니다.

검사 지연 상태의 사용 가능한 참조 제한조건이 있는 상위 파일에서 갱신 및 삭제 조작을 수행하려면 먼저 제한조건을 작동 불가능하게 만든 다음 원하는 I/O 조작을 수행해야 합니다.

참조 무결성 및 CL 명령

참조 무결성은 일부 CL 명령의 특성에 영향을 줍니다.

- **ADDPFM(실제 파일 멤버 추가)**:

  각 멤버를 갖지 않는 종속 파일과 상위 파일 사이에서 제한조건 관계가 정의되면 다음과 같은 상황이 발생합니다.

  – 멤버가 먼저 상위 파일에 추가되면 제한조건 관계는 정의된 상태로 유지됩니다.
  – 그런 다음 멤버가 종속 파일에 추가되면 외부 키 액세스 경로가 설정되고 제한조건 관계가 상위(parent)에 의해 설정됩니다.

- **CHGPF(실제 파일 변경)**:

  제한조건 관계가 파일에 대해 존재할 때는 CHGPF 명령에 사용 가능한 특정의 매개변수를 변경할 수 없습니다. 다음 매개변수는 제한됩니다.

  MAXMBRS
  제한조건 관계를 가지는 파일에 대한 최대 멤버 수는 하나 즉, MAXMBRS(1)입니다.

  CCSID
  제한조건과 연관되지 않은 파일의 CCSID는 변경될 수 없습니다. 파일이 제한조건과 연관되어 있을 경우 CCSID는 65535로만 변경될 수 있습니다.

- **CLRPFM(실제 파일 멤버 지우기)**:

  CLRPFM 명령은 레코드를 포함하고 있으며 사용 가능한 참조 제한조건과 연관되어 있는 상위 파일에 대해 발행될 때는 실패합니다.

- **FEOD(FORTRAN 자료의 끝 강제)**:

  FEOD 조작은 사용 가능한 참조 제한조건과 연관된 상위 파일에 대해 발행될 때 실패합니다.

- **CRTDUPOBJ(중복 오브젝트 작성)**:

  CST(*NO)가 지정된 경우 제한조건이 새 파일에서 중복되지 않습니다. CST(*YES)가 지정된 경우 제한조건이 중복됩니다. 다음 규칙은 제한조건이 중복되는 방법을 설명합니다.

  – 상위 파일이 같은 라이브러리 또는 다른 라이브러리에 대해 중복될 경우 정의된 참조 제한조건의 종속 파일을 위치시키기 위해 시스템 상호 참조 파일이 사용됩니다. 또한 시스템은 제한조건 관계를 설정하려고 시도를 합니다.

– 종속 파일이 중복될 경우에는 TOLIB가 제한조건 관계를 판별하는 데 사용됩니다.
  - 상위 파일과 종속 파일이 같은 라이브러리에 있을 경우에는 TOLIB의 상위 파일과 함께 참조 제한조건 관계가 설정됩니다.
  - 상위 파일과 종속 파일이 다른 라이브러리에 있을 경우에는 중복된 종속 파일의 참조 제한조건 관계가 원래 상위 파일과 함께 설정됩니다.

- **CPYF(파일 복사)**:

  CPYF 명령은 새로운 파일을 작성하며 원래 파일에 제한조건이 있을 때 제한조건은 새로운 파일로 복사되지 않습니다.

- **MOVOBJ(오브젝트 이동)**:

  MOVOBJ 명령은 하나의 라이브러리로부터 다른 라이브러리로 파일을 이동시킵니다. 시스템은 새로운 라이브러리의 파일에 대해 존재할 수도 있는 정의된 참조 제한조건을 설정하려고 시도합니다.

- **RNMOBJ(오브젝트 재명명)**:

  RNMOBJ 명령은 같은 라이브러리 내의 파일이나 라이브러리를 재명명합니다.

  재명명된 파일 또는 라이브러리에 대해 존재할 수 있는 정의된 참조 제한조건을 설정하려는 시도가 행해집니다.

- **DLTF(파일 삭제)**:

  DLTF 명령에는 참조 제한조건 관계의 처리 방법을 지정하는 선택적 키워드가 있습니다. RMVCST 키워드는 제한조건 관계에 있는 종속 파일에 적용됩니다. 키워드는 상위 파일이 삭제될 때 종속 파일의 얼마나 많은 제한조건 관계가 제거될 것인지를 지정합니다.

  ***RESTRICT**
  제한조건 관계가 상위 파일과 종속 파일 사이에서 정의되거나 설정되면 상위 파일은 삭제되지 않고 제한조건 관계도 제거되지 않습니다. 이것이 디폴트 값입니다.

  ***REMOVE**
  상위 파일이 삭제되고, 제한조건 관계와 정의가 제거됩니다. 상위 파일과 종속 파일 사이의 제한조건 관계가 제거됩니다. 종속 파일에 상응하는 외부 키 액세스 경로가 제한조건 정의와 같이 제거됩니다.

  ***KEEP**
  상위 파일은 삭제되고 참조 제한조건 관계 정의는 정의된 상태로 남습니다. 종속 파일에 상응하는 외부 키 액세스 경로와 제한조건 정의는 제거되지 않습니다.

- **RMVM(실제 파일 멤버 제거)**:

  제한조건 관계에 있는 상위 파일의 멤버가 제거될 때 제한조건 관계는 정의된 상태에 놓이게 됩니다. 외부 키 액세스 경로와 참조 제한조건 정의는 제거되지 않습니다. 모 멤버가 제거되었으므로 상위 키 액세스 경로도 제거되나, 모 제한조건 정의는 파일 레벨에 남습니다.

제한조건 관계에 있는 종속 파일의 멤버가 제거될 때 제한조건 관계는 정의된 상태에 놓이게 됩니다. 상위 키 액세스 경로와 제한조건 정의는 제거되지 않습니다. 종속 멤버가 제거되었으므로 외부 키 액세스 경로도 제거되나, 참조 제한조건 정의는 제거되지 않습니다.

- **저장 및 복원 명령**:

  상위 파일이 라이브러리에 복원될 경우 시스템은 정의된 참조 제한조건의 종속 파일을 위치시키기 위해 시스템 상호 참조 파일이 사용합니다. 제한조건 관계를 설정하려는 시도가 행해집니다.

  종속 파일이 복원될 경우 TOLIB가 제한조건 관계를 판별하는 데 사용됩니다.
  - 상위 파일과 종속 파일이 같은 라이브러리에 있을 경우 참조 제한조건 관계가 TOLIB의 상위 파일에 대해 설정됩니다.
  - 상위 파일과 종속 파일이 다른 라이브러리에 있을 경우 중복된 종속 파일의 참조 제한조건 관계가 원래 상위 파일에 대해 설정됩니다.

  제한조건 관계 내의 종속 및 상위 파일의 복원 순서와는 관계없습니다. (즉, 종속 파일보다 먼저 상위 파일이 복원되거나 상위 파일보다 먼저 종속 파일이 복원될 수 있습니다.) 제한조건 관계는 결과적으로 성립합니다.

데이터베이스의 자동 이벤트 트리거

*트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

변경 조작은 어플리케이션 프로그램에 있어서 고급 언어 명령문의 삽입, 갱신 또는 삭제입니다. 읽기 조작은 어플리케이션 프로그램에서 고급 언어 명령문 페치, 가져오기 또는 읽기입니다.

SQL 트리거를 사용할 수도 있습니다.

관련 개념

8 페이지의 『데이터베이스 자료 보호 및 모니터링』
시스템은 자료의 무결성 및 일관성을 향상시키는 피처를 제공합니다. 자료를 보호하는 두 가지 방법은 비즈니스 규칙과 자료 유형 규칙을 강제 적용하는 것입니다.

SQL 트리거

SQL 프로그래밍

관련 정보

iSeries용 DB2 Universal Database의 저장 프로시듀어, 트리거 및 사용자 정의 기능

트리거 사용

데이터베이스에서 트리거를 몇 가지 용도로 사용할 수 있습니다.

트리거를 사용하여 다음 타스크를 수행할 수 있습니다.

- 업무 규정 강제

- 입력 자료 확인
- 다른 파일에 새로 삽입된 열에 대한 고유값 생성(대리 기능)
- 감사 추적을 위해 다른 파일에 기록
- 상호 참조를 위한 다른 파일로부터의 조회
- 시스템 액세스 기능(예를 들면, 규칙이 위반될 때의 예외 메세지 인쇄)
- 자료 일관성을 위해 다른 파일에 자료 복제

트리거 사용 시 이점

트리거를 사용하면 업무에 여러 가지로 도움이 됩니다.

트리거의 장점은 다음과 같습니다.

- 빠른 어플리케이션 개발. 데이터베이스가 트리거를 저장하기 때문에 각 데이터베이스 어플리케이션으로 트리거를 코드할 필요는 없습니다.
- 업무 규칙의 글로벌 강제성. 일단 트리거가 정의되면 사용하는 어떤 어플리케이션에도 재사용됩니다.
- 쉬운 유지보수. 업무 규정이 변경되어도 각 어플리케이션 프로그램이 아닌 해당 트리거 프로그램만을 변경하면 됩니다.
- 클라이언트 서버 환경에서 성능 향상. 결과가 리턴되기 전에 모든 규칙은 서버에서 실행됩니다.

트리거 프로그램 작성

이 주제에서는 트리거 프로그램 작성 및 작업 방법에 대해 설명합니다.

| 트리거를 데이터베이스 파일에 추가하려면 다음 단계를 수행하십시오.

1. 트리거 프로그램을 제공하십시오. 고급 언어(HLL), SQL(Structured Query Language) 또는 제어 언어(CL)로 트리거 프로그램을 작성할 수 있습니다.
2. 다음 방법 중 하나를 사용하여 트리거를 추가하십시오.
   - ADDPFTRG(실제 파일 트리거 추가) 명령. 이 명령의 PGM(프로그램 트리거) 매개변수에 트리거 프로그램을 지정해야 합니다.
   - iSeries Navigator를 사용하여 트리거를 추가하십시오.
   - CREATE TRIGGER SQL 명령문.

   관련 참조

   ADDPFTRG(실제 파일 트리거 추가) 명령

   CREATE TRIGGER

iSeries Navigator를 사용하여 트리거 추가:

| iSeries Navigator를 사용하여 시스템 트리거 및 SQL 트리거를 정의할 수 있습니다. 또한 실제 데이터베이스
| 파일에 트리거를 작동 가능 또는 불가능으로 설정할 수 있습니다.

| 트리거는 지정된 변경 조작이 지정된 데이터베이스 파일에서 수행될 때 자동으로 실행되는 일련의 조치입니다.
| 설명 중에 사용되는 표는 실제 파일을 말합니다. 변경 또는 읽기 조작은 어플리케이션 프로그램의 고급 언어
| 명령문 또는 SQL INSERT, UPDATE 또는 DELETE 명령문 삽입, 갱신 또는 삭제입니다.

트리거를 추가하려면 다음 단계를 수행하십시오.

| 1. iSeries Navigator에서 서버 **Database** → **Schemas**를 확장하십시오.
2. 트리거를 추가할 표가 포함된 라이브러리를 클릭하십시오.
| 3. 트리거를 추가할 표를 마우스 오른쪽 버튼으로 클릭하고 **새** → **트리거**를 클릭하십시오.
| 4. 시스템 트리거를 추가하려면 **외부**를 클릭하십시오.
| 5. SQL 트리거를 추가하려면 **SQL**을 클릭하십시오.

관련 개념

306 페이지의 『데이터베이스의 자동 이벤트 트리거』
| *트리거*는 지정된 데이터베이스 파일에서 지정된 변경 또는 읽기 조작이 수행되면 자동으로 실행되는 일련의
| 조치입니다. iSeries에서 지원되는 고급 언어로 일련의 트리거 조치를 정의할 수 있습니다.

SQL 트리거

트리거 프로그램 작업 방식:

사용자나 어플리케이션이 연관된 트리거가 있는 데이터베이스 파일에서 변경 또는 읽기 조작을 발행할 때 조작은 적절한 트리거 프로그램이나 프로그램을 호출합니다.

변경 또는 읽기 조작은 두 개의 매개변수를 다음 표에 설명된 대로 트리거 프로그램에 전달합니다.

매개변수	설명	입출력	유형
1	이 트리거 프로그램을 호출 중인 현재 변경 조작에 대한 정보가 포함된 트리거 버퍼.	입력	CHAR(*)
2	트리거 버퍼 길이	입력	BINARY(4)

이 입력으로부터 트리거 프로그램이 원래 또는 새 레코드의 사본을 참조할 수 있습니다. 이 매개변수를 허용하는 트리거 프로그램을 코딩해야 합니다.

관련 개념

329 페이지의 『트리거 버퍼 섹션』
트리거 버퍼는 두 논리 섹션을 가집니다(정적 섹션 및 가변 섹션).

트리거 작업의 기타 정보:

이 주제에서는 트리거 프로그램에 대한 권장사항, 주의사항 및 오류 메세지에 대해 설명합니다. 모니터링 및 확약 제어에 대한 정보도 포함됩니다.

트리거 프로그램 권장사항:

다음은 트리거 프로그램에 대한 권장사항입니다.

- 작성한 사용자 프로파일에서 실행하도록 트리거 프로그램을 작성하십시오. 이러한 방법으로 프로그램의 같은 레벨의 권한을 가지고 있지 않은 사용자는 오류를 만나지 않습니다.
- USRPRF(*OWNER)과 *EXCLUDE 권한으로 프로그램을 작성한 후, 트리거 프로그램에 대한 권한을 USER(*PUBLIC)에게 부여하지 마십시오. 트리거 프로그램이 다른 사용자에 의해 변경되거나 대체되는 것을 방지하십시오. 트리거 프로그램을 실행시킬 사용자에게 트리거 프로그램에 대한 권한이 있는 경우에도 데이터베이스가 트리거 프로그램을 호출합니다.
- 프로그램이 통합 언어 환경(ILE)에서 실행되는 경우 프로그램을 ACTGRP(*CALLER)로 작성하십시오. 이렇게 하면 트리거 프로그램이 어플리케이션과 같은 확약 정의하에 실행됩니다.
- 어플리케이션의 확약 잠금 레벨과 같은 확약 잠금 레벨로 파일을 여십시오. 이렇게 하면 트리거 프로그램은 어플리케이션과 같은 확약 잠금 레벨하에서 실행됩니다.
- 실제 파일의 라이브러리에서 프로그램을 작성하십시오.
- 트리거 프로그램이 어플리케이션과 다른 활성 그룹하에서 실행될 경우 트리거 프로그램에서 확약 또는 롤백을 사용하십시오.
- 트리거 프로그램에서 오류가 발생하거나 검색될 경우 예외를 표시하십시오. 트리거 프로그램에서 오류 메세지가 표시되지 않으면 데이터베이스는 트리거가 성공적으로 실행되었다고 가정합니다. 이렇게 하면 사용자 자료가 불일치 상태로 종료됩니다.

트리거 프로그램을 코딩 시 사전 준비:

트리거 프로그램은 매우 강력할 수 있습니다. 트리거 프로그램 코딩 시에는 주의를 기울여야합니다.

테이프 드라이브와 같이 시스템 자원에 액세스하는 트리거 프로그램 설계 시에는 주의가 필요합니다. 예를 들어, 테이프 매체에 레코드 변경사항을 복사하는 트리거 프로그램은 유용하나, 테이프 드라이브가 준비되었는지 또는 올바른 테이프가 들어 있는지를 프로그램 자체로는 감지할 수 없습니다. 트리거 프로그램을 설계할 때는 이런 종류의 자원 문제를 고려해야 합니다.

또한 주의를 기울여 읽기 트리거를 사용하십시오. 읽기 트리거를 사용하면 읽히는 모든 레코드에 대해 트리거가 호출됩니다. 조회를 처리하는 중이면 이로 인해 조회가 레코드를 처리하는 횟수만큼 트리거가 호출될 수 있습니다. 이렇게 되면 시스템 성능에 영향을 미칩니다.

트리거 프로그램에서 주의하여 사용해야 할 기능:

일부 제어 언어(CL) 명령 및 기능은 트리거 프로그램에서 사용하지 않는 것이 좋으며 사용할 경우는 신중하게 고려해야 합니다.

이러한 CL 명령 및 기능으로는 다음과 같은 것이 있습니다.

- STRCMTCTL(확약 제어 시작)
- RCLSPLSTG(스풀 기억장치 재생)
- RCLRSC(자원 재생)
- CHGSYSLIBL(시스템 라이브러리 리스트 변경)
- DLTLICPGM, RSTLICPGM 및 SAVLICPGM(라이센스 프로그램 삭제, 복원 및 저장)

- (*NO)가 아닌 SAVACT가 있는 SAVLIB(라이브러리 저장)
- DKT 또는 TAP에 대한 모든 명령
- 모든 마이그레이션 명령
- 디버그 프로그램(보안 노출)
- 리모트 작업 항목(RJE)과 관련된 모든 명령
- 또다른 CL 또는 대화식 항목 호출-잠금 자원 한계에 도달할 수 있음.

트리거 프로그램에서 사용할 수 없는 명령, 명령문 및 조작:

트리거 프로그램에는 일부 명령, 명령문 및 조작을 포함할 수 없습니다.

시스템은 다음을 사용할 경우 예외사항을 리턴합니다.

- 트리거를 호출한 삽입, 갱신 또는 삭제 조작과 연관되어 있는 확약 정의는 COMMIT 조작을 허용하지 않습니다. COMMIT 조작은 작업에서의 다른 확약 정의에 대해서는 *허용됩니다*.
- 트리거를 호출한 삽입, 갱신 또는 삭제 조작과 연관되어 있는 확약 정의는 ROLLBACK 조작을 허용하지 않습니다. ROLLBACK 조작은 작업에서의 다른 확약 정의에 대해서는 *허용됩니다*.
- SQL CONNECT, DISCONNECT, SET CONNECTION 및 RELEASE문은 허용되지 *않습니다*.
- 트리거를 호출한 삽입, 갱신, 삭제 또는 읽기 조작과 연관되어 있는 확약 정의는 ENDCMTCTL CL 명령을 허용하지 않습니다. ENDCMTCTL CL 명령은 작업에서의 다른 확약 정의에 대해서는 *허용됩니다*.
- 트리거를 호출한 삽입, 갱신, 삭제 또는 읽기 조작과 연관되어 있는 동일한 확약 정의에 로컬 API 확약 자원(QTNADDCR)을 추가하려고 시도합니다.
- *SHARE로 열려 있으며 트리거 프로그램을 호출한 파일에 트리거 프로그램이 입/출력 조작을 수행하려고 시도합니다.
- 호출된 트리거 프로그램은 트리거를 호출하였으며 이미 기존의 리모트 자원을 가지고 있는 삽입, 갱신, 삭제 또는 읽기 조작과 동일한 확약 정의를 사용합니다. 그러나 시스템이 전체 트랜잭션을 롤백 요구 상태로 만듭니다.
  - 트리거 프로그램이 실패하여 이탈 메세지 AND를 신호할 경우
  - iSeries 이외의 위치이거나 이전 버전 3 릴리스 2 레벨에 대한 1차 이외의 확약 주기 동안 리모트 자원이 갱신된 경우
  - 트리거 프로그램은 트리거를 호출한 삽입, 갱신, 삭제 또는 읽기 조작과 연관되어 있는 확약 정의에 리모트 자원을 추가할 수 있습니다. 이는 LU62 리모트 자원(보호된
  - 대화) 및 DFM 리모트 자원(DDM 파일 열기)에 대해서는 허용되지만 DRDA® 리모트 자원에 대해서는 허용되지 않습니다.
  - 트리거 프로그램으로부터 리모트 자원을 변경할 때 실패하면 이탈 메세지를 표시하여 트리거 프로그램을 종료해야 합니다. 이렇게 함으로써 시스템은 모든 리모트 위치에 대해 전체 트랜잭션이 올바르게 롤백하는 지 확인할 수 있습니다. 트리거 프로그램이 이탈 메세지로 종료되지 않으면 여러 리모트 위치의 데이터베이스가 일치하지 않을 수 있습니다.

– 어플리케이션 프로그램의 확약 잠금 레벨이 트리거 프로그램에 전달됩니다. 어플리케이션 프로그램과 같은 잠금 레벨하에서 트리거 프로그램을 실행하십시오.
– 트리거 프로그램과 어플리케이션 프로그램은 같은 활성 그룹 내 또는 다른 활성 그룹 내에서 실행될 수 있습니다. 트리거 프로그램과 어플리케이션 프로그램 사이의 일관성을 위해서 트리거 프로그램을 ACTGRP(*CALLER)로 컴파일하십시오.
– 트리거 프로그램은 다른 프로그램을 호출하거나 다른 프로그램에 내포될 수 있습니다. (즉, 트리거 프로그램 내의 한 명령문이 다른 트리거 프로그램을 호출합니다.) 그 외에도, 트리거 프로그램 자체가 트리거 프로그램을 호출할 수도 있습니다. 삽입, 갱신, 삭제 또는 읽기를 위한 트리거의 최대 내포 레벨은 200입니다. 트리거 프로그램이 확약 제어하에서 실행될 때 다음 상황이 오류 결과를 가져옵니다.
  - 변경 조작이나 트리거 프로그램에서의 조작에 의해 이전에 변경되었던 같은 파일의 같은 레코드를 갱신합니다.
  - 하나의 변경 조작 내에서 같은 레코드에 대해 상반되는 조작. 예를 들어, 변경 조작은 레코드를 삽입한 후 트리거 프로그램으로 레코드를 삭제합니다.

  주:

  1. 변경 조작이 확약 제어하에서 수행되지 않을 경우 시스템은 변경 조작을 보호합니다. 그러나 시스템은 트리거 프로그램 안에서 같은 레코드를 갱신하는 것을 모니터링하지 않습니다.
  2. ADDPFTRG(실제 파일 트리거 추가) 명령의 ALWREPCHG(*NO|*YES) 매개변수는 확양 제어하의 반복 변경을 제어합니다. 디폴트 값에서 ALWREPCHG(*YES)로 변경하면 트리거 프로그램과 연관이 있는 같은 레코드나 갱신된 레코드가 반복적으로 변경할 수 있습니다.
– 또한 ADDPFTRG(실제 파일 트리거 추가) 명령의 반복 변경 허용(ALWREPCHG(*YES)) 매개변수도 데이터베이스 삽입 및 갱신 조작 전에 호출되도록 정의된 트리거 프로그램에 영향을 줄 수 있습니다. 트리거 프로그램이 트리거 버퍼의 새로운 레코드를 갱신하며 ALWREPCHG(*YES)가 지정된 경우에는 연관 실제 파일의 실제적인 삽입 또는 갱신 조작이 수정된 새로운 레코드 이미지를 사용합니다. 이 옵션은 자료 검증 및 자료 정정을 위해 설계된 트리거 프로그램에 큰 도움이 됩니다. 트리거 프로그램이 실제 파일 레코드 이미지를 수신하기 때문에(논리 파일의 경우에도), 트리거 프로그램은 그 레코드 이미지의 어느 필드라도 변경할 수 있습니다.
– 실제 파일에서 변경되거나 읽은 각 행에 대해 트리거 프로그램이 호출됩니다.
– 실제 파일이나 종속 논리 파일이 SEQONLY(*YES) 처리로 열리고 실제 파일에 이와 연관된 트리거 프로그램이 있을 경우 시스템은 SEQONLY(*NO)로 열기를 변경하여 삽입된 각 행에 대해 트리거 프로그램을 호출하도록 합니다.

트리거 프로그램 사용 모니터링:

iSeries용 DB2 Universal Database는 트리거 프로그램을 데이터베이스 파일과 연관시키는 기능을 제공합니다. 트리거 프로그램 기능은 업계에서 고급 기능을 갖춘 데이터베이스 관리자에 대해 공통되는 것입니다.

트리거 프로그램을 데이터베이스 파일과 연관시킬 경우 트리거 프로그램이 실행되는 시기를 지정합니다. 예를 들어, 새로운 레코드가 파일에 추가될 때마다 트리거 프로그램을 실행할 고객 순서 파일을 설정할 수 있습니다. 고객의 미해결 잔고가 대변 한도를 초과할 경우 트리거 프로그램은 고객에 대한 경고장을 인쇄하여 메세지를 대변 관리자에게 보냅니다.

트리거 프로그램은 어플리케이션 기능을 제공하고 정보를 관리하기 위한 생산적인 방법입니다. 트리거 프로그램은 또한 시스템에 『트로이 목마』를 만들려는 정도를 벗어나는 의도를 가진 사용자에게 기능을 제공하기도 합니다. 파괴적인 프로그램이 시스템의 데이터베이스 파일에서 특정 이벤트가 발생할 때 실행하려고 대기하고 있을 수 있습니다.

주: 역사적으로 트로이 목마는 그리스 군사들로 가득 찬 속이 비어있는 커다란 목마입니다. 목마가 트로이 성벽 내로 들어온 후 안에 있던 군사들이 나와 트로이의 군사와 싸웠습니다. 컴퓨터 세계에서는 파괴적 기능을 감추고 있는 프로그램을 종종 트로이 목마라고 합니다.

시스템이 제공될 때 트리거 프로그램을 데이터베이스 파일에 추가할 수 있는 기능을 제한합니다. 오브젝트 권한을 세밀하게 관리하려는 경우 일반 사용자에게 트리거 프로그램을 데이터베이스 파일에 추가하기에 충분한 권한이 없습니다. (*iSeries 보안 참조서* 서적의 부록 D에서 ADDPFTRG(실제 파일 트리거 추가) 명령을 포함한 모든 명령에 필요한 권한을 참조할 수 있습니다.

PRTTRGPGM(트리거 프로그램 인쇄) 명령을 사용하여 특정 라이브러리나 모든 라이브러리에 있는 모든 트리거 프로그램의 리스트를 인쇄할 수 있습니다. 다음은 보고서의 예를 나타낸 것입니다.

트리거 프로그램(전체 보고서)

```
지정된 라이브러리  . . . . . :   CUSTLIB
                              트리거          트리거          트리거          트리거          트리거
라이브러리      파일          라이브러리      프로그램        시간            이벤트          조건
CUSTLIB         MB106         ARPGMLIB        INITADDR        이전            갱신            항상
CUSTLIB         MB107         ARPGMLIB        INITNAME        이전            갱신            항상
```

시스템에 이미 존재하는 트리거 프로그램을 평가하기 위한 기본으로 초기 보고서를 사용할 수 있습니다. 그 다음, 변경된 보고서를 정기적으로 인쇄하여 새로운 트리거 프로그램이 시스템에 추가되었는지 볼 수 있습니다.

트리거 프로그램을 평가할 경우 다음을 고려하십시오.

- 트리거 프로그램 작성자는? DSPOBJD(오브젝트 설명 표시) 명령을 사용하여 이를 판별할 수 있습니다.
- 프로그램이 수행하는 것은? 소스 프로그램을 보거나 프로그램 작성자에게 알려 이를 판별합니다. 예를 들어, 트리거 프로그램이 사용자가 누구인지 체크합니까? 아마도, 트리거 프로그램은 시스템 자원에 대한 액세스 권한을 얻기 위해 특정 사용자(QSECOFR)를 기다리고 있을 것입니다.

정보의 기본을 설정하고 나면, 변경된 보고서를 정기적으로 인쇄하여 시스템에 추가된 새로운 트리거 프로그램을 모니터링할 수 있습니다. 다음은 변경된 보고서의 예를 나타낸 것입니다.

```
트리거 프로그램(변경된 보고서)
지정된 라이브러리  . . . . . :   LIBX
최종 변경된 보고서  . . . . . :   96/01/21  14:33:37
                              트리거          트리거          트리거          트리거          트리거
```

라이브러리	파일	라이브러리	프로그램	시간	이벤트	조건
INVLIB	MB108	INVPGM	NEWPRICE	이후	삭제	항상
INVLIB	MB110	INVPGM	NEWDSCNT	이후	삭제	항상

확약 제어 하의 트리거 및 어플리케이션 프로그램:

이 주제에서는 트리거 및 어플리케이션 프로그램이 확약 제어의 영향을 받을 경우 조건에 대해 설명합니다.

트리거 프로그램과 어플리케이션 프로그램이 같은 확약 정의 하에서 실행될 경우 트리거 프로그램이 실패하면 트리거 프로그램과 연관되어 있는 모든 명령문이 롤백됩니다. 여기에는 내포된 트리거 프로그램의 모든 명령문도 포함됩니다. 작업 변경 기원 또한 뒤로 화면이동합니다. 이것은 오류를 만났을 때 트리거 프로그램이 예외를 신호하도록 요구합니다.

트리거 프로그램과 어플리케이션 프로그램이 다른 확약 정의하에서 수행될 때 어플리케이션 프로그램의 COMMIT문은 자체 확약 정의에만 영향을 미칩니다. 프로그래머는 COMMIT문을 사용하여 트리거 프로그램에 있는 변경사항을 반드시 확약해야 합니다.

삽입 또는 갱신 레코드 조작이 확약 제어하에서 수행되면 특정 중복 키 오류를 발견하는 처리가 조작의 논리 끝까지 지연되므로 그 시간 동안 오류가 해결될 수 있는 가능성을 제공합니다. 관련 호출 프로그램과 동일한 확약 정의에서 실행하는 트리거 프로그램의 경우에 조작의 논리 끝은 단일 또는 블록화 삽입, 갱신 또는 삭제 레코드 조작이 호출 프로그램에 의해 수행된 이후에 발생하며, 트리거 프로그램 이전 또는 이후에 호출된 곳에서 제어가 리턴합니다. 결과적으로, 트리거 프로그램을 호출한 삽입, 갱신 또는 삭제 레코드와 동일한 확약 정의를 사용하는 트리거 프로그램에서 중복 키 오류를 검출할 수 없습니다.

확약 제어 하에 있지 않은 트리거 및 어플리케이션 프로그램:

이 주제에서는 트리거 및 어플리케이션 프로그램이 확약 제어의 영향을 받지 않을 경우 조건에 대해 설명합니다.

두 가지 프로그램이 모두 확약 제어하에서 실행되지 않을 경우 트리거 프로그램에서의 오류는 오류가 발생할 때의 파일 상태로 둡니다. 롤백이 발생하지 않습니다.

트리거 프로그램이 확약 제어하에서 실행되지 않고 어플리케이션 프로그램이 확약 제어하에서 수행될 경우 어플리케이션 프로그램의 COMMIT문은 어플리케이션 프로그램에 의해 이루어진 변경사항만을 확약합니다.

- 트리거 프로그램에서 확약 조작이 수행됩니다.
- 활성 그룹의 종료 시. 대개의 경우 활성 그룹이 종료될 때 내재적인 확약이 수행됩니다. 그러나 비정상적인 시스템 장애가 발생할 경우에는 롤백이 수행됩니다.

트리거 프로그램 오류 메세지:

트리거 프로그램 실행중 실패하면 빠져나가기 전에 적절한 이탈 메세지를 신호해야 합니다. 그렇지 않으면 어플리케이션은 트리거 프로그램이 성공적으로 수행되었다고 가정합니다.

메세지는 시스템으로부터 표시된 원래 메세지일 수도 있고, 트리거 프로그램 작성자에 의해 작성된 메세지일 수도 있습니다.

예: 트리거 프로그램:

이 주제에서는 C, COBOL 및 RPG로 작성된 트리거 프로그램 예를 보여 줍니다.

다음 예의 트리거 프로그램은 ATMTRANS 파일에 대한 쓰기 갱신 및 삭제 조작으로 트리거됩니다. 이 트리거 프로그램은 ILE C, ILE COBOL 및 RPG/400으로 작성됩니다. ILE RPG 예에 대해서는 iSeries용 DB2 Universal Database  레드북에서 저장 프로시듀어, 트리거 및 사용자 정의 기능을 참조하십시오.

이 어플리케이션에는 다음 유형의 트랜잭션이 포함됩니다.

1. 어플리케이션은 삽입 트리거를 실행하는 ATMTRANS 파일에 세 개의 레코드를 삽입합니다. 삽입 트리거는 변경사항을 반영하기 위해 ATMS 파일과 ACCTS 파일에 정확한 금액을 추가합니다.
2. 그 다음, 어플리케이션은 갱신 트리거를 실행하는 두 개의 취소를 실행합니다.
   a. 어플리케이션이 갱신 트리거를 실행하는 계좌 번호 20001과 ATM 번호 10001로부터 $25.00를 인출합니다. 갱신 트리거는 ACCTS와 ATMS 파일로부터 $25.00를 빼냅니다.
   b. 어플리케이션이 갱신 트리거를 실행하는 계좌 번호 20002와 ATM 번호 10002로부터 $900.00를 인출합니다. 갱신 트리거는 어플리케이션에 예외를 신호하여 트랜잭션이 실패했음을 나타냅니다.
3. 결과적으로 어플리케이션이 삭제 트리거를 실행하는 ATMTRANS 파일로부터 ATM 번호를 삭제합니다. 삭제 트리거는 ACCTS 파일에서 상응하는 ACCTID를 삭제하고 ATMS 파일에서 ATMID를 삭제합니다.

예: RPG로 작성된 삽입 트리거:

이 예에서 RPG 트리거 프로그램은 레코드를 ATMTRANS 파일에 삽입합니다.

주: 코드 예제를 사용하여 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의하십시오.

```
* Program Name : INSTRG
              * This is an insert trigger for the application
              * file.  The application inserts the following three
       * records into the ATMTRANS file.
       *
              * ATMID   ACCTID   TCODE    AMOUNT
       * -------------------------------
              * 10001   20001      D      100.00
              * 10002   20002      D      250.00
              * 10003   20003      D      500.00
       *
              * When a record is inserted into ATMTRANS, the system calls
              * this program, which updates the ATMS and
              * ACCTS files with the correct deposit or withdrawal amount.
              * The input parameters to this trigger program are:
              *  - TRGBUF : contains trigger information and newly inserted
              *             record image of ATMTRANS.
              *  - TRGBUF Length : length of TRGBUF.
       *
            H        1
       *
              * Open the ATMS file and the ACCTS file.
       *
            FATMS    UF  E                    DISK         KCOMIT
```

```
     FACCTS   UF  E                    DISK         KCOMIT
*
       * DECLARE THE STRUCTURES THAT ARE TO BE PASSED INTO THIS PROGRAM.
*
     IPARM1       DS
       * Physical file name
     I                                       1  10 FNAME
       * Physical file library
     I                                      11  20 LNAME
       * Member name
     I                                      21  30 MNAME
       * Trigger event
     I                                      31  31 TEVEN
       * Trigger time
     I                                      32  32 TTIME
       * Commit lock level
     I                                      33  33 CMTLCK
       * Reserved
     I                                      34  36 FILL1
       * CCSID
     I                                   B  37  400CCSID
       * Reserved
     I                                      41  48 FILL2
       * Offset to the original record
     I                                   B  49  520OLDOFF
       * length of the original record
     I                                   B  53  560OLDLEN
       * Offset to the original record null byte map
     I                                   B  57  600ONOFF
       * length of the null byte map
     I                                   B  61  640ONLEN
       * Offset to the new record
     I                                   B  65  680NOFF
       * length of the new record
     I                                   B  69  720NEWLEN
       * Offset to the new record null byte map
     I                                   B  73  760NNOFF
       * length of the null byte map
     I                                   B  77  800NNLEN
       * Reserved
     I                                      81  96 RESV3
       * Old record ** not applicable
     I                                      97 112 OREC
       * Null byte map of old record
     I                                     113 116 OOMAP
       * Newly inserted record of ATMTRANS
     I                                     117 132 RECORD
       * Null byte map of new record
     I                                     133 136 NNMAP
     IPARM2       DS
     I                                   B   1   40LENG
******************************************************************
       * SET UP THE ENTRY PARAMETER LIST.
******************************************************************
     C           *ENTRY    PLIST
     C                     PARM           PARM1
     C                     PARM           PARM2
******************************************************************
```

```
            * Use NOFF, which is the offset to the new record, to
            * get the location of the new record from the first
            * parameter that was passed into this trigger program.
            *   - Add 1 to the offset NOFF since the offset that was
            *     passed to this program started from zero.
            *   - Substring out the fields to a CHARACTER field and
            *     then move the field to a NUMERIC field if it is
            *     necessary.
      *****************************************************************
           C                    Z-ADDNOFF      O       50
           C                    ADD  1         O
      *****************************************************************
            *         - PULL OUT THE ATM NUMBER.
      *****************************************************************
           C           5        SUBSTPARM1:O   CATM    5
      *****************************************************************
            *         - INCREMENT "O", WHICH IS THE OFFSET IN THE PARAMETER
            *           STRING.  PULL OUT THE ACCOUNT NUMBER.
***************************************************************
           C                    ADD  5         O
           C           5        SUBSTPARM1:O   CACC    5
      *****************************************************************
            *         - INCREMENT "O", WHICH IS THE OFFSET IN THE PARAMETER
            *           STRING.  PULL OUT THE TRANSACTION CODE.
***************************************************************
           C                    ADD  5         O
           C           1        SUBSTPARM1:O   TCODE   1
      *****************************************************************
            *         - INCREMENT "O", WHICH IS THE OFFSET IN THE PARAMETER
            *           STRING.  PULL OUT THE TRANSACTION AMOUNT.
***************************************************************
           C                    ADD  1         O
           C           5        SUBSTPARM1:O   CAMT    5
           C                    MOVELCAMT      TAMT    52
      *************************************************************
            *  PROCESS THE ATM FILE.                    ****************
*************************************************************
            * READ THE FILE TO FIND THE CORRECT RECORD.
           C           ATMN     DOUEQCATM
           C                    READ ATMS                     61EOF
           C                    END
           C   61               GOTO EOF
            * CHANGE THE VALUE OF THE ATM BALANCE APPROPRIATELY.
           C           TCODE    IFEQ 'D'
           C                    ADD  TAMT      ATMAMT
           C                    ELSE
           C           TCODE    IFEQ 'W'
           C                    SUB  TAMT      ATMAMT
           C                    ELSE
           C                    ENDIF
           C                    ENDIF
            * UPDATE THE ATM FILE.
           C           EOF      TAG
           C                    UPDATATMFILE
           C                    CLOSEATMS
      *************************************************************
            *  PROCESS THE ACCOUNT FILE.                ****************
*************************************************************
```

```
      * READ THE FILE TO FIND THE CORRECT RECORD.
     C           ACCTN     DOUEQCACC
     C                     READ ACCTS                    62  EOF2
     C                     END
     C   62                GOTO EOF2
      * CHANGE THE VALUE OF THE ACCOUNTS BALANCE APPROPRIATELY.
     C           TCODE     IFEQ 'D'
     C                     ADD  TAMT      BAL
     C                     ELSE
     C           TCODE     IFEQ 'W'
     C                     SUB  TAMT      BAL
     C                     ELSE
     C                     ENDIF
     C                     ENDIF
      * UPDATE THE ACCT FILE.
     C           EOF2      TAG
     C                     UPDATACCFILE
     C                     CLOSEACCTS
*
     C                     SETON                     LR
```

어플리케이션에 의한 삽입 후, ATMTRANS 파일에는 다음과 같은 자료가 포함됩니다.

ATMID	ACCTID	TCODE	AMOUNT
10001	20001	D	100.00
10002	20002	D	250.00
10003	20003	D	500.00

삽입 트리거 프로그램에 의해 ATMTRANS 파일로부터 갱신된 후, ATMS 파일과 ACCTS 파일에는 다음과 같은 자료가 포함됩니다.

ATMN	LOCAT	ATMAMT
10001	MN	300.00
10002	MN	750.00
10003	CA	750.00

ACCTN	BAL	ACTACC
20001	200.00	A
20002	350.00	A
20003	500.00	C

예: COBOL로 작성된 갱신 트리거:

이 예에서는 ATMTRANS 파일에서 레코드가 갱신될 때 ILE COBOL 트리거 프로그램 실행을 표시합니다.

주: 코드 예제를 사용하여 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의하십시오.

```
   100        IDENTIFICATION DIVISION.
   200        PROGRAM-ID. UPDTRG.
 300 ************************************************************************
   400 **** Program Name : UPDTRG                                           *
   500 *****                                                               *
```

```
  600 ***** This trigger program is called when a record is updated     *
  700 ***** in the ATMTRANS file.                                        *
 800 ***** This program will check the balance of ACCTS and            *
  900 ***** the total amount in ATMS.If either one of the amounts       *
 1000 ***** is not enough to meet the withdrawal, an exception          *
 1100 ***** message is signalled to the application.                    *
1200 ***** If both ACCTS and ATMS files have enough money, this        *
 1300 ***** program will update both files to reflect the changes.      *
1400 *****                                                             *
 1500 ***** ATMIDs of 10001 and 10002 will be updated in the ATMTRANS   *
 1600 ***** file with the following data:                               *
 1700 *****                                                           *
 1800 ***** ATMID   ACCTID   TCODE    AMOUNT                          *
 1900 ***** -------------------------------                           *
 2000 ***** 10001   20001      W       25.00                          *
 2100 ***** 10002   20002      W      900.00                          *
 2200 ***** 10003   20003      D      500.00                          *
 2300 *****                                                           *
 2400 *****************************************************************
 2500       *************************************************************
 2600        ENVIRONMENT DIVISION.
 2700        CONFIGURATION SECTION.
 2800        SOURCE-COMPUTER. IBM-AS400.
2900        OBJECT-COMPUTER. IBM-AS400.
3000        SPECIAL-NAMES. I-O-FEEDBACK IS FEEDBACK-JUNK.
3100        INPUT-OUTPUT SECTION.
 3200        FILE-CONTROL.
 3300            SELECT ACC-FILE ASSIGN TO DATABASE-ACCTS
 3400                   ORGANIZATION IS INDEXED
 3500                   ACCESS IS RANDOM
 3600                   RECORD KEY IS ACCTN
 3700                   FILE STATUS IS STATUS-ERR1.
 3800
 3900            SELECT ATM-FILE ASSIGN TO DATABASE-ATMS
 4000                   ORGANIZATION IS INDEXED
 4100                   ACCESS IS RANDOM
 4200                   RECORD KEY IS ATMN
 4300                   FILE STATUS IS STATUS-ERR2.
 4400
 4500       *************************************************************
 4600       *                   COMMITMENT CONTROL AREA.                *
4700       *************************************************************
 4800        I-O-CONTROL.
 4900            COMMITMENT CONTROL FOR ATM-FILE, ACC-FILE.
 5000
 5100       *************************************************************
 5200       *                  DATA DIVISION                            *
 5300       *****************************************************************
 5400
 5500        DATA DIVISION.
 5600        FILE SECTION.
 5700        FD  ATM-FILE
 5800            LABEL RECORDS ARE STANDARD.
 5900        01  ATM-REC.
 6000        COPY DDS-ATMFILE OF ATMS.
 6100
 6200        FD  ACC-FILE
 6300            LABEL RECORDS ARE STANDARD.
```

```
 6400        01  ACC-REC.
 6500        COPY DDS-ACCFILE OF ACCTS.
 6600
 7000
 7100       ************************************************************
 7200       *             WORKING-STORAGE SECTION                      *
 7300       ************************************************************
 7400        WORKING-STORAGE SECTION.
 7500        01  STATUS-ERR1         PIC XX.
 7600        01  STATUS-ERR2         PIC XX.
 7700        01  TEMP-PTR USAGE IS POINTER.
 7800
 7900        01 NUMBERS-1.
 8000          03 NUM1          PIC 9(10).
 8100          03 NUM2          PIC 9(10).
 8200          03 NUM3          PIC 9(10).
 8300
 8400        01  FEEDBACK-STUFF      PIC X(500) VALUE SPACES.
 8500
 8600       ************************************************************
 8700       * MESSAGE FOR SIGNALLING ANY TRIGGER ERROR                  *
 8800       * - Define any  message ID and message file in the following*
 8900       *   message data.                                           *
 9000      ************************************************************
 9100        01 SNDPGMMSG-PARMS.
 9200          03 SND-MSG-ID        PIC X(7)    VALUE "TRG9999".
 9300          03 SND-MSG-FILE      PIC X(20)   VALUE "MSGF      LIB1    ".
 9400          03 SND-MSG-DATA      PIC X(25)   VALUE "Trigger Error".
 9500          03 SND-MSG-LEN       PIC 9(8) BINARY  VALUE 25.
 9600          03 SND-MSG-TYPE      PIC X(10)   VALUE "*ESCAPE   ".
 9700          03 SND-PGM-QUEUE     PIC X(10)   VALUE "*     ".
 9800          03 SND-PGM-STACK-CNT PIC 9(8) BINARY  VALUE 1.
 9900          03 SND-MSG-KEY       PIC X(4)    VALUE "    ".
10000         03 SND-ERROR-CODE.
10100           05 PROVIDED          PIC 9(8) BINARY VALUE 66.
10200           05 AVAILABLE         PIC 9(8) BINARY VALUE 0.
10300           05 RTN-MSG-ID        PIC X(7) VALUE " ".
10400           05 FILLER            PIC X(1) VALUE " ".
10500           05 RTN-DATA          PIC X(50) VALUE " ".
10600
10700      ************************************************************
10800      *             LINKAGE         SECTION                      *
10900      *  PARM 1 is the trigger buffer                            *
11000      *  PARM 2 is the length of the trigger buffer              *
11100      ************************************************************
11200       LINKAGE SECTION.
11300       01 PARM-1-AREA.
11400         03 FILE-NAME        PIC X(10).
11500         03 LIB-NAME         PIC X(10).
11600         03 MEM-NAME         PIC X(10).
11700         03 TRG-EVENT        PIC X.
11800         03 TRG-TIME         PIC X.
11900         03 CMT-LCK-LVL      PIC X.
12000         03 FILLER           PIC X(3).
12100         03 DATA-AREA-CCSID  PIC 9(8)  BINARY.
12200         03 FILLER           PIC X(8).
12300         03 DATA-OFFSET.
12400           05 OLD-REC-OFF      PIC 9(8)  BINARY.
```

```
12500            05 OLD-REC-LEN      PIC 9(8)  BINARY.
12600            05 OLD-REC-NULL-MAP PIC 9(8)  BINARY.
12700            05 OLD-REC-NULL-LEN PIC 9(8)  BINARY.
12800            05 NEW-REC-OFF      PIC 9(8)  BINARY.
12900            05 NEW-REC-LEN      PIC 9(8)  BINARY.
13000            05 NEW-REC-NULL-MAP PIC 9(8)  BINARY.
13100            05 NEW-REC-NULL-LEN PIC 9(8)  BINARY.
13200            05 FILLER           PIC X(16).
13300          03 RECORD-JUNK.
13400            05 OLD-RECORD       PIC X(16).
13500            05 OLD-NULL-MAP     PIC X(4).
13600            05 NEW-RECORD       PIC X(16).
13700            05 NEW-NULL-MAP     PIC X(4).
13800
13900          01 PARM-2-AREA.
14000          03  TRGBUFL             PIC X(2).
14100
14200          01 INPUT-RECORD2.
14300          COPY DDS-TRANS OF ATMTRANS.
14400
14500          05 OFFSET-NEW-REC2   PIC 9(8)  BINARY.
14600
14700 ****************************************************************
14800 ******                   PROCEDURE DIVISION                    *
14900 ****************************************************************
15000        PROCEDURE DIVISION USING PARM-1-AREA, PARM-2-AREA.
15100        MAIN-PROGRAM SECTION.
15200        000-MAIN-PROGRAM.
15300            OPEN I-O ATM-FILE.
15400            OPEN I-O ACC-FILE.
15500
15600            MOVE 0 TO BAL.
15700
15800       ************************************************************
15900       * SET UP THE OFFSET POINTER AND COPY THE NEW RECORD.        *
16000       ************************************************************
16100            SET TEMP-PTR TO ADDRESS OF PARM-1-AREA.
16200            SET TEMP-PTR UP BY NEW-REC-OFFSET.
16300            SET ADDRESS OF INPUT-RECORD2 TO TEMP-PTR.
16400            MOVE INPUT-RECORD2 TO INPUT-RECORD.
16500
16600       ************************************************************
16700       * READ THE RECORD FROM THE ACCTS FILE                      *
16800       ************************************************************
16900            MOVE ACCTID TO ACCTN.
17000            READ ACC-FILE
17100                 INVALID KEY PERFORM 900-OOPS
17200                 NOT INVALID KEY PERFORM 500-ADJUST-ACCOUNT.
17300
17400       ************************************************************
17500       * READ THE RECORD FROM THE ATMS FILE.                      *
17600       ************************************************************
17700            MOVE ATMID TO ATMN.
17800            READ ATM-FILE
17900                 INVALID KEY PERFORM 950-OOPS
18000                 NOT INVALID KEY PERFORM 550-ADJUST-ATM-BAL.
18100            CLOSE ATM-FILE.
18200            CLOSE ACC-FILE.
```

```
18300            GOBACK.
18400
18500 ****************************************************************
18600 ****************************************************************
18700 ****************************************************************
18800 ****************************************************************
18900 ****** THIS PROCEDURE IS USED IF THERE IS NOT ENOUGH MONEY IN THE  ****
19000 ****** ACCTS FOR THE WITHDRAWAL.                                    ****
19100 ****************************************************************
19200        200-NOT-ENOUGH-IN-ACC.
19300            DISPLAY "NOT ENOUGH MONEY IN ACCOUNT.".
19400            CLOSE ATM-FILE.
19500            CLOSE ACC-FILE.
19600            PERFORM 999-SIGNAL-ESCAPE.
19700            GOBACK.
19800
19900 ****************************************************************
20000 ****** THIS PROCEDURE IS USED IF THERE IS NOT ENOUGH MONEY IN THE
20100 ****** ATMS FOR THE WITHDRAWAL.
20200 ****************************************************************
20300        250-NOT-ENOUGH-IN-ATM.
20400            DISPLAY "NOT ENOUGH MONEY IN ATM.".
20500            CLOSE ATM-FILE.
20600            CLOSE ACC-FILE.
20700            PERFORM 999-SIGNAL-ESCAPE.
20800            GOBACK.
20900
21000 ****************************************************************
21100 ****** THIS PROCEDURE IS USED TO ADJUST THE BALANCE FOR THE ACCOUNT OF
21200 ****** THE PERSON WHO PERFORMED THE TRANSACTION.
21300 ****************************************************************
21400        500-ADJUST-ACCOUNT.
21500            IF TCODE = "W" THEN
21600               IF (BAL < AMOUNT) THEN
21700                  PERFORM 200-NOT-ENOUGH-IN-ACC
21800               ELSE
21900                  SUBTRACT AMOUNT FROM BAL
22000                  REWRITE  ACC-REC
22100            ELSE IF TCODE = "D" THEN
22200               ADD AMOUNT TO BAL
22300               REWRITE  ACC-REC
22400            ELSE DISPLAY "TRANSACTION CODE ERROR, CODE IS: ", TCODE.
22500
22600 ****************************************************************
22700 ****** THIS PROCEDURE IS USED TO ADJUST THE BALANCE OF THE ATM FILE ***
22800 ****** FOR THE AMOUNT OF MONEY IN ATM AFTER A TRANSACTION.          ***
22900 ****************************************************************
23000        550-ADJUST-ATM-BAL.
23100            IF TCODE = "W" THEN
23200               IF (ATMAMT < AMOUNT) THEN
23300                  PERFORM 250-NOT-ENOUGH-IN-ATM
23400               ELSE
23500                  SUBTRACT AMOUNT FROM ATMAMT
23600                  REWRITE  ATM-REC
23700            ELSE IF TCODE = "D" THEN
23800               ADD AMOUNT TO ATMAMT
23900               REWRITE  ATM-REC
24000            ELSE DISPLAY "TRANSACTION CODE ERROR, CODE IS: ", TCODE.
```

```
24100
24200 ************************************************************  *******
24300 ****** THIS PROCEDURE IS USED IF THERE THE KEY VALUE THAT IS USED IS **
24400 ****** NOT FOUND IN THE ACCTS FILE.                                  **
24500 ********************************************************************
24600       900-OOPS.
24700           DISPLAY "INVALID KEY: ", ACCTN, "  ACCOUNT FILE STATUS: ",
24800                   STATUS-ERR1.
24900           CLOSE ATM-FILE.
25000           CLOSE ACC-FILE.
25100           PERFORM 999-SIGNAL-ESCAPE.
25200           GOBACK.
25300
25400 ********************************************************************
25500 ****** THIS PROCEDURE IS USED IF THERE THE KEY VALUE THAT IS USED IS **
25600 ****** NOT FOUND IN THE ATM FILE.                                    **
25700 ********************************************************************
25800       950-OOPS.
25900           DISPLAY "INVALID KEY: ", ATMN, "  ATM FILE STATUS: ",
26000                   STATUS-ERR2.
26100           CLOSE ATM-FILE.
26200           CLOSE ACC-FILE.
26300           PERFORM 999-SIGNAL-ESCAPE.
26400           GOBACK.
26500
26600 ********************************************************************
26700 ****** SIGNAL ESCAPE TO THE APPLICATION                      ********
26800 ********************************************************************
26900       999-SIGNAL-ESCAPE.
27000
27100           CALL "QMHSNDPM" USING SND-MSG-ID,
27200                                 SND-MSG-FILE,
27300                                 SND-MSG-DATA,
27400                                 SND-MSG-LEN,
27500                                 SND-MSG-TYPE,
27600                                 SND-PGM-QUEUE,
27700                                 SND-PGM-STACK-CNT,
27800                                 SND-MSG-KEY,
27900                                 SND-ERROR-CODE.
28000      *DISPLAY RTN-MSG-ID.
28100      *DISPLAY RTN-DATA.
28200
```

갱신 트리거 프로그램에 의해 ATMTRANS 파일이 갱신된 후, ATMS와 ACCTS 파일에는 다음과 같은 자료가 포함됩니다. 계좌에 충분한 금액이 없으므로 ATMID 10002에 대한 갱신이 실패합니다.

ATMN	LOCAT	ATMAMT
10001	MN	275.00
10002	MN	750.00
10003	CA	750.00

ACCTN	BAL	ACTACC
20001	175.00	A
20002	350.00	A
20003	500.00	C

예: ILE C로 작성된 삭제 트리거:

이 예에서는 ATMTRANS 파일에서 레코드가 삭제될 때 ILE C 트리거 프로그램 실행을 표시합니다.

주: 코드 예제를 사용하여 345 페이지의 『코드 라이센스 및 면책사항 정보』의 조건에 동의하십시오.

```
/*************************************************************/
/* Program Name - DELTRG                                     */
/* This program is called when a delete operation occurs in  */
/* the ATMTRANS file.                                        */
/*                                                           */
/* This program will delete the records from ATMS and ACCTS  */
/* based on the ATM ID and ACCT ID that are passed in from   */
/* the trigger buffer.                                       */
/*                                                           */
/* The application will delete ATMID 10003 from the ATMTRANS */
/* file.                                                     */
/*                                                           */
/*************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <recio.h>
#include "applib/csrc/msghandler" /* message handler include       */
#include "qsysinc/h/trgbuf"       /* trigger buffer include without*/
                                  /* old and new records           */
Qdb_Trigger_Buffer_t *hstruct;      /* pointer to the trigger buffer */
char *datapt;

#define KEYLEN 5

/*************************************************************/
/* Need to define file structures here since there are non-  */
/* character fields in each file. For each non-character     */
/* field, C requires boundary alignment. Therefore, a _PACKED */
/* struct should be used in order to access the data that    */
/* is passed to the trigger program.                         */
/*                                                           */
/*************************************************************/

/** record area for ATMTRANS                **/
_Packed struct rec {
              char atmid[5];
              char acctid[5];
              char tcode[1];
              char amount[5];
                   } oldbuf, newbuf;

/** record area for ATMS                    **/
_Packed struct rec1{
              char atmn[5];
              char locat[2];
              char atmamt[9];
                   } atmfile;

/** record area for ACCTS                   **/
_Packed struct rec2{
```

```
             char acctn[5];
             char bal[9];
             char actacc[1];
                  } accfile;


/*****************************************************************/
/*****************************************************************/
/* Start of the Main Line Code.  *********************************/
/*****************************************************************/
/*****************************************************************/
main(int argc, char **argv)
{
_RFILE  *out1;                /* file pointer for ATMS        */
_RFILE  *out2;                /* file pointer for ACCTS       */
_RIOFB_T *fb;                 /* file feedback pointer        */
char record[16];             /* record buffer            */
_FEEDBACK fc;                 /* feedback for message handler */
_HDLR_ENTRY hdlr = main_handler;
                              /*******************************/
                              /* active exception handler    */
                              /*******************************/
CEEHDLR(&hdlr, NULL, &fc);;
                              /*******************************/
                              /* ensure exception handler OK  */
                              /*******************************/
if (fc.MsgNo != CEE0000)
 {
   printf("Failed to register exception handler.₩n");
   exit(99);
 }

/* set pointer to the input parameter     */
hstruct = (Qdb_Trigger_Buffer_t *)argv[1];
datapt  = (char *) hstruct;

/* Copy old and new record from the input parameter  */

 if ((strncmp(hstruct ->trigger_event,"2",1)== 0)|| /* delete event */
     (strncmp(hstruct -> trigger_event,"3",1)== 0)) /* update event */
   { obufoff = hstruct ->old_record_offset;
     memcpy(&oldbuf,datapt+obufoff,; hstruct->old_record_len);
   }
 if ((strncmp(hstruct -> trigger_event,"1",1)== 0) ||  /* insert event */
     (strncmp(hstruct -> trigger_event,"3",1)== 0))    /* update event */
   { nbufoff = hstruct ->new_record_offset;
     memcpy(&newbuf,datapt+nbufoff,; hstruct->new_record_len);
   }

/*************************************************/
/* Open ATM and ACCTS    files                   */
/*                                               */
/* Check the application's commit lock level. If it  */
/* runs under commitment control, then open both     */
/* files with commitment control. Otherwise, open    */
/* both files without commitment control.            */
/*************************************************/
if(strcmp(hstruct->commit_lock_level,"0") == 0)    /* no commit    */
```

```
 {
   if ((out1=_Ropen("APPLIB/ATMS","rr+")) == NULL)
     {
       printf("Error opening ATM file");
       exit(1);
     }
   if ((out2=_Ropen("APPLIB/ACCTS","rr+")) == NULL)
     {
       printf("Error opening ACCTS file");
       exit(1);
     }
  }
 else                  /* with commitment control   */
 {
   if ((out1=_Ropen("APPLIB/ATMS","rr+,commit=Y")) == NULL)
     {
       printf("Error opening ATMS file");
       exit(1);
     }
   if ((out2=_Ropen("APPLIB/ACCTS","rr+,commit=Y")) == NULL)
     {
       printf("Error opening ACCTS file");
       exit(1);
     }
  }

/* Delete the record based on the input parameter  */
fb =_Rlocate(out1,&oldbuf.atmid,KEYLEN,__DFT);
if (fb->num_bytes != 1)
 {
   printf("record not found in ATMS₩n");
   _Rclose(out1);
   exit(1);
 }
 _Rdelete(out1);         /* delete record from ATMS */
 _Rclose(out1);

fb =_Rlocate(out2,&oldbuf.acctid,KEYLEN,__DFT);
if (fb->num_bytes != 1)
 {
   printf("record not found in ACCOUNTS₩n");
   _Rclose(out2);
   exit(1);
 }
 _Rdelete(out2);         /* delete record from ACCOUNTS  */
 _Rclose(out2);

 } /* end of main */
```

어플리케이션에 의한 삭제 후, ATMTRANS 파일에는 다음과 같은 자료가 포함됩니다.

ATMID	ACCTID	TCODE	AMOUNT
10001	20001	W	25.00
10002	20002	W	900.00

삭제 트리거 프로그램에 의해 ATMTRANS 파일로부터 삭제된 후, ATMS 파일과 ACCTS 파일에는 다음과 같은 자료가 포함됩니다.

ATMN	LOCAT	ATMAMT
10001	MN	275.00
10002	MN	750.00

ACCTN	BAL	ACTACC
20001	175.00	A
20002	350.00	A

```
/*****************************************************************/
/*   INCLUDE NAME : MSGHANDLER                                   */
/*                                                               */
/*   DESCRIPTION  : Message handler to signal an exception message*/
/*                  to the caller of this trigger program.       */
/*                                                               */
/*   Note: This message handler is a user defined routine.       */
/*                                                               */
/*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <recio.h>
#include <leawi.h>

#pragma linkage (QMHSNDPM, OS)
void QMHSNDPM(char *,             /* Message identifier            */
              void *,             /* Qualified message file name   */
              void *,             /* Message data or text          */
              int,                /* Length of message data or text */
              char *,             /* Message type                  */
              char *,             /* Call message queue            */
              int,                /* Call stack counter            */
              void *,             /* Message key                   */
              void *,             /* Error code                    */
              ...);               /* Optionals:
                                       length of call message queue
                                        name
                                       Call stack entry qualification
                                       display external messages
                                        screen wait time           */
/*****************************************************************/
/********  This is the start of the exception handler function.    */
/*****************************************************************/
  void main_handler(_FEEDBACK *cond, _POINTER *token, _INT4 *rc,
                    _FEEDBACK *new)
   {
                           /***************************************/
                           /* Initialize variables for call to    */
                           /* QMHSNDPM.                           */
                           /* User defines any message ID and     */
                           /* message file for the following data  */
                           /***************************************/
  char      message_id[7] = "TRG9999";
  char      message_file[20] = "MSGF      LIB1      ";
```

```
   char      message_data[50] = "Trigger error              ";
   int       message_len = 30;
   char      message_type[10] = "*ESCAPE   ";
   char      message_q[10] = "_C_pep    ";
   int       pgm_stack_cnt = 1;
   char      message_key[4];
                            /****************************************/
                            /* Declare error code structure for     */
                            /* QMHSNDPM.                            */
                            /****************************************/
   struct error_code {
     int bytes_provided;
     int bytes_available;
     char message_id[7];
   } error_code;

   error_code.bytes_provided = 15;
                            /****************************************/
                            /* Set the error handler to resume and  */
                            /* mark the last escape message as      */
                            /* handled.                             */
                            /****************************************/
   *rc = CEE_HDLR_RESUME;
                            /****************************************/
                            /* Send my own *ESCAPE message.         */
                            /****************************************/
   QMHSNDPM(message_id,
            &message_file,
            &message_data,
            message_len,
            message_type,
            message_q,
            pgm_stack_cnt,
            &message_key,
            &error_code );
                            /****************************************/
                            /* Check that the call to QMHSNDPM      */
                            /* finished correctly.                  */
                            /****************************************/
  if (error_code.bytes_available != 0)
      {
        printf("Error in QMHOVPM : %s₩n", error_code.message_id);
      }
   }

/*************************************************************/
/*   INCLUDE NAME : TRGBUF                                   */
/*                                                           */
/*   DESCRIPTION  : The input trigger buffer structure for the  */
/*                  user's trigger program.                  */
/*                                                           */
/*   LANGUAGE     : ILE C                                    */
/*                                                           */
/*************************************************************/
/*************************************************************/
/*   Note: The following type definition only defines the fixed */
/*         portion of the format. The data area of the original */
/*         record, null byte map of the original record, the    */
/*         new record, and the null byte map of the new record  */
```

```
/*         is varying length and immediately follows what is    */
/*         defined here.                                        */
/************************************************************/
 typedef _Packed struct Qdb_Trigger_Buffer {
           char  file_name[10];
           char  library_name[10];
           char  member_name[10];
           char  trigger_event[1];
           char  trigger_time[1];
           char  commit_lock_level[1];
           char  reserved_1[3];
           int   data_area_ccsid;
           char  reserved_2]8];
           int   old_record_offset;
           int   old_record_len;
           int   old_record_null_byte_map;
           int   old_record_null_byte_map_len;
           int   new_record_offset;
           int   new_record_len;
           int   new_record_null_byte_map;
           int   new_record_null_byte_map_len;
                 } Qdb_Trigger_Buffer_t;
```

트리거 프로그램: 예에서 사용된 데이터베이스의 자료 구조:

이 주제에서는 이 어플리케이션에 사용되는 자료 구조에 대해 설명합니다.

- ATMTRANS : /* Transaction record */

```
ATMID      CHAR(5) (KEY)  /* ATM** machine ID number     */
ACCTID     CHAR(5)        /* Account number            */
TCODE      CHAR(1)        /* Transaction code          */
AMOUNT     ZONED          /* Amount to be deposited or */
                                                  /* withdrawn                 */
```

- ATMS : /* ATM machine record */

```
ATMN       CHAR(5)  (KEY) /* ATM machine ID number     */
LOCAT      CHAR(2)        /* Location of ATM           */
ATMAMT     ZONED          /* Total amount in this ATM  */
                                                  /* machine                   */
```

ATMN	LOCAT	ATMAMT
10001	MN	200.00
10002	MN	500.00
10003	CA	250.00

- ACCTS: /* Accounting record */

```
ACCTN      CHAR(5)  (KEY) /* Account number            */
BAL        ZONED          /* Balance of account        */
ACTACC     CHAR(1)        /* Status of Account         */
```

ACCTN	BAL	ACTACC
20001	100.00	A
20002	100.00	A
20003	0.00	C

트리거 버퍼 섹션:

트리거 버퍼는 두 논리 섹션을 가집니다(정적 섹션 및 가변 섹션).

정적 섹션은 0-95의 오프셋(십진수)을 차지합니다. 정적 섹션에는 다음 항목이 포함됩니다.

- 실제 파일명, 멤버명, 트리거 이벤트, 트리거 시간, 확약 잠금 레벨, 현재 레코드의 CCSID 및 상대 레코드 번호가 들어 있는 트리거 템플리트
- 레코드 영역과 널 바이트 맵의 오프셋 및 길이

가변 섹션에는 기존 레코드, 기존 널 바이트 맵, 새 레코드 및 새 널 바이트 맵에 대한 영역이 포함됩니다.

다음 표는 트리거 버퍼의 필드에 대한 개요를 나타냅니다.

오프셋		유형	필드
소수 자릿수	16진수		
0	0	CHAR(10)	실제 파일명
10	A	CHAR(10)	실제 파일 라이브러리명
20	14	CHAR(10)	실제 파일 멤버명
30	1E	CHAR(1)	트리거 이벤트
31	1F	CHAR(1)	트리거 시간
32	20	CHAR(1)	확약 잠금 레벨
33	21	CHAR(3)	예약됨
36	24	BINARY(4)	자료의 CCSID
40	28	BIN(4)	상대 레코드 번호(RRN)
44	2C	CHAR(4)	예약됨
48	30	BINARY(4)	원래 레코드 오프셋
52	34	BINARY(4)	원래 레코드 길이
56	38	BINARY(4)	원래 레코드의 널 바이트 맵 오프셋
60	3C	BINARY(4)	원래 레코드의 널 바이트 맵 길이
64	40	BINARY(4)	새로운 레코드 오프셋
68	44	BINARY(4)	새로운 레코드 길이
72	48	BINARY(4)	새로운 레코드의 널 바이트 맵 오프셋
76	4C	BINARY(4)	새로운 레코드의 널 바이트 맵 길이
80	50	CHAR(*)	예약됨
*	*	CHAR(*)	원래 레코드
*	*	CHAR(*)	원래 레코드의 널 바이트 맵
*	*	CHAR(*)	새로운 레코드
*	*	CHAR(*)	새로운 레코드의 널 바이트 맵

트리거 버퍼 필드 설명:

이 주제에서는 트리거 버퍼에 포함된 필드를 알파벳순으로 설명합니다.

자료의 CCSID

새로운 레코드 또는 원래 레코드에 있는 자료의 CCSID. 자료가 데이터베이스에 의해 작업 CCSID로 변환됩니다. SBCS 자료는 단일 바이트 연관 CCSID로 변환됩니다. DBCS 자료는 2바이트 연관 CCSID로 변환됩니다.

확약 잠금 레벨

현재 어플리케이션 프로그램의 확약 잠금 레벨. 사용 가능한 값은 다음과 같습니다.

'0' *NONE

'1' *CHG

'2' *CS

'3' *ALL

파일 라이브러리명

데이터베이스 파일이 상주하는 라이브러리명입니다.

파일 멤버명

데이터베이스 파일 멤버명입니다.

파일명 변경 중 또는 읽기 중인 실제 파일명입니다.

새로운 레코드

변경 조작의 결과로 실제 파일에 삽입되거나 갱신되는 레코드의 사본. 새로운 레코드는 삽입 또는 갱신 조작에만 적용됩니다.

새로운 레코드 길이

최대 길이는 32 766바이트입니다.

새로운 레코드의 널 바이트 맵

이 구조에는 새로운 레코드의 각 필드에 대한 NULL 값 정보가 들어 있습니다. 각각의 바이트가 하나의 필드를 나타냅니다. 각 바이트에 대해 사용 가능한 값은 다음과 같습니다.

'0' NULL이 아님

'1' NULL

새로운 레코드의 널 바이트 맵 길이

이 길이는 실제 파일의 필드에 있는 필드의 수와 같습니다.

새로운 레코드의 널 바이트 맵 오프셋

새로운 레코드의 널 바이트 맵의 위치. 오프셋 값은 트리거 버퍼의 시작부터입니다. 레코드의 새로운 값이 변경 조작(예: 삭제 조작)에 적용되지 않을 경우 이 필드는 적용되지 않습니다.

새로운 레코드 오프셋

새로운 레코드의 위치. 오프셋 값은 트리거 버퍼의 시작부터입니다. 레코드의 새로운 값이 변경 조작(예: 삭제 조작)에 적용되지 않을 경우 이 필드는 적용되지 않습니다.

원래 레코드

갱신, 삭제 또는 읽기 전의 원래 실제 레코드의 사본. 원래 레코드는 갱신, 삭제 및 읽기 조작에만 적용됩니다.

원래 레코드 길이

최대 길이는 32 766바이트입니다.

원래 레코드의 널 바이트 맵

이 구조에는 원래 레코드의 각 필드에 대한 NULL 값 정보가 들어 있습니다. 각각의 바이트가 하나의 필드를 나타냅니다. 각 바이트에 대해 사용 가능한 값은 다음과 같습니다.

'0' NULL이 아님

'1' NULL

원래 레코드의 널 바이트 맵 길이

이 길이는 실제 파일의 필드에 있는 필드의 수와 같습니다.

원래 레코드의 널 바이트 맵 오프셋

원래 레코드의 널 바이트 맵의 위치. 오프셋 값은 트리거 버퍼의 시작부터입니다. 레코드의 원래 값이 변경 조작(예: 삽입 조작)에 적용되지 않을 경우 이 필드는 적용되지 않습니다.

원래 레코드 오프셋

원래 레코드의 위치. 오프셋 값은 트리거 버퍼의 시작부터입니다. 레코드의 원래 값이 조작(예: 삽입 조작)에 적용되지 않을 경우 이 필드는 적용되지 않습니다.

상대 레코드 번호(RRN)

갱신하거나 삭제할 레코드의 상대 레코드 번호(*BEFORE 트리거) 또는 삽입, 갱신, 삭제, 또는 읽은 레코드의 상대 레코드 번호(*AFTER 트리거).

트리거 이벤트

트리거 프로그램을 호출한 이벤트. 사용 가능한 값은 다음과 같습니다.

'1' 삽입 조작

'2' 삭제 조작

'3' 갱신 조작

'4' 읽기 조작

트리거 시간

| 트리거 프로그램 호출 시 데이터베이스 파일의 조작에 대해 상대적인 시간입니다. 사용 가능한 값은
| 다음과 같습니다.

'1' 변경 또는 읽기 조작 이후

'2' 변경 조작 전

| **'3'** 변경 조작 대신

트리거 추가

이 주제에서는 트리거를 데이터베이스 파일에 추가하는 방법을 보여 줍니다.

트리거를 추가하려면 다음 단계를 수행하십시오.

1. 적절한 권한과 파일이 적절한 자료 기능을 가지고 있는지 확인하십시오.
2. 트리거 프로그램을 특정 데이터베이스 파일과 연관시키려면 다음 방법 중 하나를 사용하십시오.
   - iSeries Navigator를 사용하여 새 파일을 작성하거나 기존 파일의 등록 정보를 편집하십시오.
   - ADDPFTRG(실제 파일 트리거 추가) 명령을 사용하십시오.
   - CREATE TRIGGER SQL 명령문을 사용합니다.

주: 트리거 프로그램이 QTEMP 라이브러리에 상주하면 트리거 프로그램을 실제 파일과 연관시킬 수 없습니다.

트리거 프로그램과 파일 사이에 관계를 작성한 다음 실제 파일을 통해 작성된 데이터베이스 파일, 파일 멤버 및 논리 파일에 대해 변경 또는 읽기 조작이 시작되면 시스템은 트리거 프로그램을 호출합니다.

데이터베이스 파일 하나에 트리거를 300개까지 연관시킬 수 있습니다. 각 삽입, 삭제 또는 갱신 조작은 조작이 발생하기 전후에 복수 트리거를 호출할 수 있습니다. 각 읽기 조작은 조작 발생 이후에 복수 트리거를 호출할 수 있습니다. 이 주제에서는 트리거를 파일에 추가하는 방법을 표시합니다.

논리 파일 하나에 트리거를 세 개까지 연관시킬 수 있습니다. 즉, 삽입, 갱신 또는 삭제 조작마다 INSTEAD OF 트리거 한 개를 작성할 수 있습니다. 조작 수행 대신 트리거가 호출됩니다.

조회가 발행한 읽기 조작 이후에 호출된 트리거 수가 실제로 리턴된 레코드 수와 같지 않을 수 있습니다. 이것은 올바른 레코드 수를 리턴하기 전에 각 읽기 조작에 대해 트리거가 호출되도록 야기하며 조회에서 다른 수의 레코드를 읽을 수 있기 때문입니다.

SQL 갱신 조작에는 쓰기 조작 앞에 오는 동시 읽기 조작이 포함됩니다. SQL 갱신 조작에 대한 읽기 트리거는 실행되지 않습니다. 쓰기 조작 앞에 오는 이 읽기 조작을 수행할 수 있도록 갱신 트리거를 지정하십시오.

트리거 권한 및 자료 기능

필수 권한은 다음과 같습니다.

- 파일에 대한 오브젝트 관리 또는 변경 권한
- 파일에 대한 오브젝트 조작 권한
- 파일에 대한 자료 읽기 권한
- CRTPFTRG ALWREPCHG(*YES)가 지정된 경우 자료 권한 및 파일에 대한 오브젝트 조작 권한 갱신
- 파일 라이브러리에 대한 실행 권한
- 트리거 프로그램에 대한 실행 권한
- 트리거 프로그램 라이브러리에 대한 실행 권한

트리거를 추가하기 전에 파일에는 적합한 자료 기능이 있어야 합니다.

- CRTPF ALWUPD(*NO)는 *UPDATE 트리거와 상반됩니다.
- CRTPF ALWDLT(*NO)는 *DELETE 트리거와 상반됩니다.

관련 참조

ADDPFTRG(실제 파일 트리거 추가) 명령

트리거 표시

DSPFD(파일 설명 표시) 명령은 파일과 연관된 트리거 리스트를 제공합니다. 이 리스트를 보려면 TYPE(*TRG) 또는 TYPE(*ALL)을 지정하십시오.

DSPFD 명령은 다음 정보를 제공합니다.

- 트리거 프로그램의 수
- 트리거명 및 라이브러리
- 트리거 상태
- 트리거 프로그램명 및 라이브러리
- 트리거 이벤트
- 트리거 시간
- 트리거 갱신 조건
- 트리거 유형
- 트리거 모드
- 트리거 용지 방향
- 트리거 작성 날짜/시간
- 트리거 갱신 열의 수
- 트리거 갱신 열의 리스트

관련 참조

파일 설명 표시(DSPFD) 명령

트리거 제거

RMVPFTRG(실제 파일 트리거 제거) 명령, SQL DROP TRIGGER 명령문 및 iSeries Navigator를 사용하여 트리거를 제거할 수 있습니다.

RMVPFTRG 명령을 사용하여 파일과 트리거 프로그램의 연관 관계를 제거하십시오. 연관 관계가 제거된 다음 시스템은 실제 파일에 변경 또는 읽기 조작이 수행되어도 아무런 조치를 취하지 않습니다. 그러나 트리거 프로그램은 시스템에 남아 있습니다.

관련 개념

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

RMVPFTRG(실제 파일 트리거 제거) 명령

실제 파일 트리거를 작동 가능 또는 작동 불가능하게 설정

CHGPFTRG(실제 파일 트리거 변경) 명령 및 iSeries Navigator를 사용하여 트리거를 작동 가능 또는 작동 불가능하게 설정할 수 있습니다.

CHGPFTRG 명령을 사용하여 명명된 트리거를 작동 가능 또는 작동 불가능하게 하거나 파일에 대한 모든 트리거를 작동 가능 또는 작동 불가능하게 설정하십시오. 트리거를 작동 불가능하게 설정하면 변경 조작이 실제 파일에 발생할 경우 트리거 프로그램이 호출되지 않습니다. 트리거를 작동 가능하게 설정하면 변경 조작이 실제 파일에 발생할 때 트리거 프로그램이 다시 호출됩니다. iSeries Navigator를 사용하여 트리거를 작동 가능 또는 작동불가능으로 설정할 수 있습니다.

| 논리 파일의 트리거는 작동 가능이나 작동 불가능으로 설정할 수 없습니다.

관련 개념

iSeries Navigator 시작

SQL 프로그래밍

관련 참조

CHGPFTRG(실제 파일 트리거 변경) 명령

트리거 및 CL 명령과의 관계

트리거는 여러 가지 방법으로 CL 명령과 대화합니다.

SAVOBJ/RSTOBJ(기본 파일 저장/복원)

- 저장/복원 기능은 저장/복원시 트리거 프로그램을 탐색하지 않습니다. 프로그램을 관리하는 것은 사용자의 책임입니다. 런타임 시 시스템이 트리거 프로그램을 복원하지 않았으면 시스템은 트리거 프로그램명, 실제 파일명 및 트리거 이벤트와 함께 심각한 오류를 리턴합니다.
| • 전체 라이브러리(*ALL)가 저장되고, 파일 및 모든 트리거 프로그램이 같은 라이브러리에 있으며, 다른 라이
| 브러리로 복원될 경우 모든 트리거 프로그램명은 파일에서 새 라이브러리를 반영하기 위해 변경됩니다.

SAVOBJ/RSTOBJ(트리거 프로그램 저장/복원)

- 트리거 프로그램이 다른 라이브러리에 복원될 경우에는 트리거 프로그램이 원래 라이브러리에서 발견되지 않으므로 변경 조작이 실패합니다. 오류는 트리거 프로그램명, 실제 파일명 및 트리거 이벤트 정보를 리턴합니다.

  이러한 상황에서는 두 가지의 회복 방법이 사용됩니다.
  - 트리거 프로그램을 같은 라이브러리로 복원
  - 새로운 라이브러리에서 같은 이름의 새로운 트리거 작성

DLTF(파일 삭제)

| • 파일이 삭제되면 이 파일과 해당 트리거 프로그램 사이의 연관 관계가 제거됩니다. 시스템 트리거의 경우
| 트리거 프로그램이 시스템에 남아 있습니다.

CPYF(파일 복사)

- To-파일이 삽입 트리거와 연관되어 있을 경우 삽입된 각각의 레코드는 트리거 프로그램이 호출되도록 합니다.
- To-파일이 삭제 트리거 프로그램과 연관되어 있는 상태에서 MBROPT(*REPLACE)가 CPYF 명령에서 지정되면 복사 조작이 실패합니다.
- CREATE(*YES)가 지정된 복사는 트리거 정보를 전달하지 않습니다.

CRTDUPOBJ(중복 오브젝트 작성)

| TRG(*NO)가 지정된 경우 트리거가 새 파일에서 중복되지 않습니다. TRG(*YES)가 지정된 경우 트리거가 중
| 복됩니다. 다음 규칙은 트리거가 중복되는 방법을 설명합니다.

- 실제 파일과 트리거 프로그램이 원래 같은 라이브러리에 있는 경우 트리거 프로그램이 새로운 라이브러리에 없더라도 트리거 프로그램 라이브러리는 항상 새로운 라이브러리로 변경됩니다. 또한 다음 규칙이 적용됩니다.
  - CRTDUPOBJ 명령이 실제 파일과 실제 파일의 트리거 프로그램에 중복된 경우 새로운 트리거 프로그램이 새로운 실제 파일과 연관됩니다.
  - CRTDUPOBJ 명령이 실제 파일에만 중복된 경우 TO 라이브러리의 같은 프로그램명의 트리거 프로그램은 새로운 실제 파일과 연관됩니다. 해당 이름에 의한 트리거 프로그램이 TO 라이브러리에 없을지라도 그렇습니다. 트리거 프로그램의 라이브러리는 변경됩니다.
  - CRTDUPOBJ 명령이 트리거 프로그램에만 중복된 경우 새로운 트리거 프로그램은 어떤 실제 파일과도 연관되지 않습니다.
- 실제 파일과 트리거 프로그램이 원래 같은 라이브러리에 있는 경우
  - 이전 트리거 프로그램은 새로운 실제 파일과 연관됩니다. 새로운 실제 파일이 트리거 프로그램으로 같은 라이브러리에 중복되더라도, 이전 트리거 프로그램은 여전히 새로운 실제 파일과 연관됩니다.
- 프로그램이 QTEMP 라이브러리에 있으면 트리거 프로그램을 추가할 수 없습니다. 데이터베이스 파일의 경우 CRTDUPOBJ 명령이 TO 라이브러리에서 트리거 프로그램을 찾으려고 시도합니다. 새 라이브러리로 지정된 QTEMP와 함께 CRTDUPOBJ 명령을 사용하면 CRTDUPOBJ가 가능한 한 많은 오브젝트를 작성하려고 시도합니다. 이때 파일이 작성되기는 하지만 트리거가 추가되지 않고 파일이 멤버없이 QTEMP에 남습니다.

CLRPFM(실제 파일 멤버 지우기)

- 실제 파일이 삭제 트리거와 연관되어 있을 경우 CLRPFM 조작이 실패합니다.

INZPFM(실제 파일 멤버 초기화)

- 실제 파일이 삽입 트리거와 연관되어 있을 경우 INZPFM 조작이 실패합니다.

FEOD(FORTRAN 자료의 끝 강제)

- 실제 파일이 삭제 트리거와 연관되어 있을 경우 FEOD 조작이 실패합니다.

APYJRNCHG/RMVJRNCHG(저널 변경 적용/저널 변경 삭제)

- 실제 파일이 트리거 유형 중 하나와 연관되어 있을 경우 APYJRNCHG 및 RMVJRNCHG 조작은 트리거 프로그램을 호출하지 않습니다. 따라서 트리거 프로그램 내의 모든 파일이 저널되어야 합니다. 그런 다음 APYJRNCHG 또는 RMVJRNCHG 명령을 사용할 때 이러한 파일이 모두 지정되었는지 확인하십시오. 이렇게 하면 어플리케이션 프로그램과 트리거 프로그램에 대한 모든 실제 파일 변경사항이 일치하게 됩니다.

  주: 어떤 트리거 프로그램 기능이라도 데이터베이스 파일과 연관되지 않고, 명시적으로 저널될 수 없는 경우에는 레코드 관련 정보로 저널 항목을 전송하십시오. SNDJRNE(저널 항목 전송) 명령 또는 QJOSJRNE(저널 항목 전송) API를 사용하십시오. 일관성을 확인하기 위해 데이터베이스 파일이 회복하는 동안 이 정보를 사용하십시오.

트리거와 참조 무결성과의 관계

실제 파일은 트리거 및 그와 연관된 참조 제한조건을 모두 가질 수 있습니다. 트리거 조치와 참조 제한조건 사이의 실행 순서는 파일과 연관된 제한조건 및 트리거에 따라 다릅니다.

어떤 경우 시스템은 시스템이 이후 트리거 프로그램을 호출하기 전 참조 제한조건을 지정합니다. 이는 제한조건이 RESTRICT 규칙을 지정한 경우입니다.

어떤 경우 제한조건이 적용되기 전에 트리거 프로그램(내포 트리거 프로그램 포함)의 모든 명령문이 실행됩니다. 이것은 NO ACTION, CASCADE, SET NULL 및 SET DEFAULT 참조 제한조건 규칙에 대해서는 참(true)입니다. 이 규칙이 지정될 때 시스템은 트리거 프로그램의 내포된 결과에 기초하여 파일의 제한조건을 평가합니다. 예를 들어, 어플리케이션이 다음과 같은 제한조건과 트리거가 있는 EMP 파일에 사원 레코드를 삽입합니다.

- 참조 제한조건은 EMP 파일에 삽입된 사원 레코드의 부서 번호가 DEPT 파일에 반드시 존재하도록 지정합니다.
- EMP 파일에 삽입할 때마다, 트리거 프로그램이 부서 번호가 DEPT 파일에 존재하는 지 검사합니다. 존재하지 않으면 트리거 프로그램은 번호를 삽입합니다.

EMP 파일에 대한 삽입이 발생할 때 시스템은 먼저 트리거 프로그램을 호출합니다. DEPT 파일에 부서 번호가 존재하지 않으면 트리거 프로그램이 새로운 부서 번호를 DEPT 파일에 삽입합니다. 그런 다음, 시스템은 참조 제한조건을 평가합니다. 이 경우에는 부서 번호가 DEPT 파일에 있으므로 삽입이 성공적으로 수행된 것입니다.

트리거와 참조 제한조건이 모두 같은 실제 파일에 정의되어 있을 때는 몇 가지 제한사항이 있습니다.

- 삭제 트리거가 실제 파일과 연관되어 있을 경우 그 파일은 삭제 규칙이 CASCADE인 참조 제한조건의 종속 파일이어서는 안됩니다.
- 갱신 트리거가 실제 파일과 연관되어 있을 경우 이 실제 파일에서의 어떤 필드도 삭제 규칙이 SET NULL 또는 SET DEFAULT인 참조 제한조건의 외부 키가 될 수 없습니다.

트리거 프로그램 또는 참조 제한조건 확인 도중에 실패가 발생할 경우 모든 파일이 같은 확약 정의하에서 실행되고 있다면 변경 조작과 연관된 모든 트리거 프로그램이 롤백됩니다. 트리거 프로그램 내의 모든 파일과 참조 무결성 네트워크가 같은 확약 정의하에서 실행될 때에는 참조 제한조건이 보증됩니다. 확약 제어 없이 또는 혼합 시나리오에서 파일을 열 경우 원치 않는 결과가 일어나기도 합니다.

트리거를 사용하여 참조 제한조건과 업무 규정을 강제할 수 있습니다. 예를 들면, 트리거를 사용하여 실제 파일에 갱신 연쇄 제한조건을 시뮬레이트할 수 있습니다. 그러나 시스템 참조 무결성 기능으로 정의된 제한조건이 제공한 것과 같은 기능을 갖지는 못합니다. 트리거로 정의할 경우 다음의 참조 무결성 장점을 잃을 수도 있습니다.

- 제한조건을 검사 지연 상태로 만들었던, 하나 이상의 참조 제한조건을 위반하는 열이 종속 파일에 들어 있어도 파일 조작은 계속 허용됩니다.
- 제한조건이 검사 지연 상태에 놓일 경우 사용자에게 알려 주는 기능
- 어플리케이션이 COMMIT(*NONE)하에서 실행되고 연쇄 삭제 도중에 오류가 발생할 때 모든 변경사항은 데이터베이스에 의해 롤백됩니다.
- 제한조건과 연관된 파일을 저장하는 동안, 데이터베이스 네트워크 내의 같은 라이브러리에 저장되어 있는 모든 종속 파일이 저장됩니다.

데이터베이스 분배

별도로 비용을 지불해야 하는 DB2 Multisystem에서는 불완전 결합 환경에 놓인 다중 시스템상의 데이터베이스 파일 분배에 있어서 간단하고도 직접적인 방법을 제공합니다.

DB2 Multisystem에서는 분산 iSeries 시스템상의 사용자들이 분산 데이터베이스가 마치 자신의 특정 시스템에 있는 것처럼 분산 데이터베이스에 실시간 조회 및 갱신 액세스를 할 수 있습니다. DB2 Multisystem은 사용자 정의 키 필드 또는 필드에 따라 적합한 시스템에 새로운 레코드를 놓습니다. DB2 Multisystem은 시스템 제공 또는 사용자 정의 해싱(hashing)알고리즘을 기초로 시스템을 선택합니다.

조회 성능은 그 환경의 여러 노드를 액세스하는 요소에 의해 향상됩니다. 예를 들어, 4개의 시스템상에 분산된 데이터베이스에 대한 조회에는 대략 15분이 걸립니다. 그러나 성능은 조회에 결합이나 그룹핑이 포함된 경우 상당히 달라집니다. 또한 성능은 여러 노드간 자료의 균형에 영향을 받습니다. 다중 시스템은 각 시스템에서 동시에 조회를 수행합니다. DB2 Multisystem에서는 대용량 데이터베이스에서의 조회 시간을 크게 줄일 수 있습니다.

관련 개념

DB2 Multisystem

2바이트 문자 세트 고려사항

이 주제에서는 iSeries 시스템에 데이터베이스를 적용할 경우 2바이트 문자 세트(DBCS) 고려사항을 설명합니다.

DBCS는 각 문자를 2바이트로 표시하는 문자 세트입니다. DBCS는 여러 고유 문자나 기호가 들어 있는 자국어를 지원합니다(1바이트로 나타낼 수 있는 문자의 최대수는 256자임). 그러한 언어의 예로는 한글, 일본어 및 한자 등이 있습니다.

DBCS 필드 자료 유형

2바이트 문자 세트(DBCS) 자료에는 괄호로 묶은 DBCS 자료와 그래픽(괄호로 묶지 않은) DBCS 자료의 일반적인 두 가지 유형이 있습니다.

괄호로 묶은 DBCS 자료 앞에는 DBCS SO 문자가 오고 뒤에는 DBCS SI 문자가 옵니다. 그래픽 DBCS 자료는 SO 및 SI 문자로 묶여지지 않습니다. 따라서 어플리케이션 프로그램은 그래픽 DBCS 자료에 대해 요구되지 않는 괄호로 묶인 DBCS 자료를 처리하기 위한 특수 처리가 요구될 수 있습니다.

(DDS 코딩 양식의 35열에 지정된) 특정 DBCS 자료 유형은 다음과 같습니다.

항목 의미

O DBCS 개방(DBCS-open): 1바이트 자료와 괄호로 묶인 2바이트 자료가 모두 들어 있는 스트링

E DBCS 선택(DBCS-either): 모두 1바이트 자료이거나 모두 괄호로 묶인 2바이트 자료가 들어 있는 스트링

J DBCS 전용(DBCS-only): 괄호로 묶인 2바이트 자료만 들어 있는 스트링

G DBCS 그래픽(DBCS-graphic): 괄호로 묶지 않은 2바이트 자료만 들어 있는 스트링

주: DBCS 자료 유형이 포함된 파일은 1바이트 문자 세트(SBCS) 시스템에서 작성될 수 있습니다. DBCS 자료 유형이 들어 있는 파일은 SBCS 시스템에서 열려 사용할 수 있으나 시스템이 DBCS나 혼용 CCSID에서 SBCS CCSID로 변환을 시도할 때 코드화 문자 세트 ID(CCSID) 변환 오류가 발생할 수 있습니다. 작업 CCSID가 65535인 경우 이러한 오류는 발생하지 않습니다.

DBCS 상수

상수는 사용할 실제 문자 스트링을 식별합니다. 문자 스트링은 어포스트로피로 묶고 DBCS 문자 스트링은 DBCS SO 및 SI 문자로 묶습입니다(아래 예에서는 문자 <와 > 로 표시됨). DBCS 그래픽 상수 앞에는 문자 *G*가 옵니다.

DBCS 상수 유형은 다음과 같습니다.

유형 예

DBCS 전용
‘<A1A2A3>’

DBCS 개방
‘<A1A2A3>BCD’

DBCS 그래픽
G‘<A1A2A3>’

DBCS 필드 맵핑 고려사항

이 주제에서는 2바이트 문자 세트(DBCS) 필드에 대한 실제 파일과 논리 파일 사이에서의 유효한 자료 맵핑의 유형을 보여줍니다.

실제 파일 자료 유형	논리 파일 자료 유형								
	문자	16진	DBCS 개방	DBCS 전용	DBCS 전용	DBCS 그래픽	UCS2 그래픽	UTF-8	UTF-16
문자	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효함
16진	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 개방	유효하지 않음	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효함
DBCS 전용	유효하지 않음	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음[1]	유효하지 않음[1]	유효하지 않음[1]
DBCS 전용	유효함	유효함	유효함	유효함	유효함	유효함	유효하지 않음[1]	유효하지 않음[1]	유효하지 않음[1]
DBCS 그래픽	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효함	유효함
UCS2 그래픽	유효함	유효하지 않음	유효함	유효하지 않음[1]	유효하지 않음[1]	유효함	유효함	유효함	유효함
UTF-8	유효함	유효하지 않음	유효함	유효하지 않음[1]	유효하지 않음[1]	유효함	유효함	유효함	유효함
UTF-16	유효함	유효하지 않음	유효함	유효하지 않음[1]	유효하지 않음[1]	유효함	유효함	유효함	유효함

주: 표에서 [1]은 변환 후 나타나는 대체 문자의 사용 가능성 때문에 이러한 맵핑이 지원되지 않음을 표시합니다.

DBCS 필드 연결

필드가 연결될 때 그 자료 형태가 변경될 수 있습니다. (결과 자료 형태는 시스템에 의해 자동적으로 결정됩니다.) 2바이트 문자 세트(DBCS) 필드가 연결될 경우 이러한 지침을 따르는 것이 좋습니다.

- 오퍼레이팅 시스템은 연결 중인 필드의 자료 유형에 따라 자료 유형을 할당합니다. 연결에 DBCS 필드가 포함되어 있으면 다음과 같은 일반 규칙이 적용됩니다.
  - 연결에 하나 이상의 16진(H) 필드가 있으면, 결과의 자료 형태는 16진(H)이 됩니다.
  - 연결에 모든 필드가 DBCS 전용(J)이면, 결과의 자료 유형은 DBCS 전용(J)이 됩니다.
  - 연결에 하나 이상의 DBCS(O, E, J) 필드가 포함되었으나 16진(H) 필드가 없으면, 결과의 자료 유형은 DBCS 개방(O)입니다.
  - 연결에 둘 이상의 DBCS 개방(O) 필드가 포함되어 있으면, 결과의 자료 유형은 가변 길이 DBCS 개방(O) 필드입니다.
  - 연결에 자료 유형 중 하나 이상의 가변 길이 필드가 포함되어 있으면, 결과의 자료 유형은 가변 길이입니다.
  - DBCS 그래픽(G) 필드는 또 다른 DBCS 그래픽 필드에만 연결될 수 있습니다. 결과의 자료 유형은 DBCS 그래픽(G)입니다.

- UCS2 그래픽(G) 필드는 또 다른 UCS2 그래픽 필드, UTF-8 문자 필드 또는 UTF-16 그래픽 필드에 연결될 수 있습니다. 결과 자료 유형은 피연산자 중 하나가 UTF-16일 경우 UTF-16이고, 피연산자 중 하나가 UTF-8이고 피연산자가 UTF-16이 아닐 경우에는 UTF-8이며 그 밖의 경우에는 UCS-2입니다.
- UTF-8 문자(A) 필드는 다른 UTF-8 필드, UTF-16 필드 또는 UCS-2 필드와 연결될 수 있습니다. 결과 자료 유형은 피연산자 중 하나가 UTF-16일 경우 UTF-16이고, 피연산자 중 하나가 UTF-8이고 피연산자가 UTF-16이 아닐 경우에는 UTF-8이며 그 밖의 경우에는 UCS-2입니다.
- UTF-16 그래픽(G) 필드는 또 다른 UTF-16 필드, UTF-8 필드 또는 UCS-2 필드에 연결될 수 있습니다. 결과 자료 유형은 피연산자 중 하나가 UTF-16일 경우 UTF-16이고, 피연산자 중 하나가 UTF-8이고 피연산자가 UTF-16이 아닐 경우에는 UTF-8이며 그 밖의 경우에는 UCS-2입니다.

• 연결 필드의 최대 길이는 연결 필드의 자료 유형과 연결되고 있는 필드의 길이에 따라 다릅니다. 연결 필드가 존 십진(S)이면, 총 길이가 31바이트를 초과할 수 없습니다. 연결 필드가 문자(A), DBCS 개방(O) 또는 DBCS 전용(J)이면, 총 길이가 32,766바이트(필드가 가변 길이일 경우에는 32,740바이트)를 초과할 수 없습니다.

  DBCS 그래픽(G) 필드의 길이는 2바이트 *문자* 수로 표시됩니다.

• 결합 논리 파일의 경우 연결될 필드는 동일한 실제 파일에서 온 필드여야 합니다. CONCAT 키워드에 지정된 첫 번째 필드로 사용될 실제 파일이 식별됩니다. 그러므로 첫 번째 필드는 논리 파일의 기초가 되는 실제 파일들 사이에서 고유해야 하거나 사용될 실제 파일을 지정하기 위해서는 JREF 키워드를 지정해야 합니다.
• 연결 필드의 용도는 I(입력 전용)여야 합니다.
• 자료 유형이 O 또는 J인 연결 필드에는 REFSHIFT가 할당될 수 없습니다.

주:

1. 괄호로 묶은 DBCS 필드가 연결되면 한 필드 끝에 있는 SI 문자와 다음 필드의 처음에 있는 SO 문자가 제거됩니다. 연결에 하나 이상의 16진 필드가 들어 있으면, SI 및 SO 문자쌍은 첫 번째 16진 필드 앞의 DBCS 필드에 대해서만 제거됩니다.
2. DBCS 필드가 들어 있는 연결 필드는 입력 전용 필드여야 합니다.
3. 연결 DBCS 필드의 결과 자료 유형은 OPNQRYF(조회 파일 열기) 명령을 사용할 때 달라질 수 있습니다.

관련 개념

342 페이지의『OPNQRYF(조회 파일 열기) 명령을 통해 DBCS 필드와의 연결 사용』
OPNQRYF(조회 파일 열기) 연결 기능을 사용할 경우 i5/OS 라이센스 프로그램은 연결되는 필드의 자료 유형을 기초로 하여 결과 자료 유형을 할당합니다. 연결에 2바이트 문자 세트(DBCS) 필드가 포함되면 결과 자료 유형은 일반적으로 논리 파일 내의 연결 필드와 동일하지만 약간의 변환이 있습니다.

DBCS 필드 서브스트링 조작

서브스트링 조작을 통해 필드의 일부나 논리 파일 내의 상수를 사용할 수 있습니다.

괄호로 묶은 DBCS 자료 유형의 경우 서브스트링의 시작 위치와 길이는 *바이트* 수로 표시됩니다. DBCS 그래픽(G) 자료 유형의 경우 서브스트링의 시작 위치와 길이는 *문자* 수로 표시됩니다.

논리 파일의 DBCS 필드 비교

두 개의 필드를 비교하거나 한 개의 필드를 상수와 비교할 때 유형이 호환되면 고정 길이 필드를 가변 길이 필드와 비교할 수 있습니다.

표 51에서는 논리 파일의 2바이트 문자 세트(DBCS)에 대한 유효한 비교에 대해 설명합니다.

표 51. 논리 파일의 DBCS 필드에 대한 유효한 비교

	모든 숫자	**문자**	**16진**	**DBCS 개방**	**DBCS 선택**	**DBCS 전용**	**DBCS 그래픽**	**UCS2 그래픽**	**UTF-8**	**UTF-16**	**날짜**	**시간**	**시간소인**
모든 숫자	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
문자	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
16진	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 개방	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 선택	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 전용	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 그래픽	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
UCS2 그래픽	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
UTF-8	유효하지 않음	유효함	유효하지 않음	유효함	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
UTF-16	유효하지 않음	유효함	유효하지 않음	유효함	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
날짜	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음	유효하지 않음
시간	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음
시간소인	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함

OPNQRYF(조회 파일 열기) 명령에서 DBCS 필드 사용

이러한 주제에서는 OPNQRYF(조회 파일 열기) 명령 기능(와일드 카드, 연결 및 정렬 순서 기능)에 대한 2바이트 문자 세트(DBCS) 필드 사용에 대해 설명합니다. 표에는 OPNQRYF 명령을 통해 DBCS 필드에 대해 수행할 수 있는 유효한 비교를 표시합니다.

DBCS 필드와 함께 와일드 카드 기능 사용

2바이트 문자 세트(DBCS) 필드가 있는 와일드 카드(%WLDCRD) 기능은 괄호로 묶은 DBCS 필드와 사용되는지 또는 DBCS 그래픽 필드와 사용되는지에 따라 달라집니다.

괄호로 묶은 DBCS 필드와 함께 와일드 카드 기능을 사용할 때는 1바이트 및 2바이트 와일드 카드 값(별표와 밑줄)을 모두 사용할 수 있습니다. 다음과 같은 특수 규칙이 적용됩니다.

- 1바이트 밑줄은 하나의 EBCDIC 문자를 의미하며, 2바이트 밑줄은 하나의 2바이트 문자를 의미합니다.
- 1바이트 또는 2바이트의 별표는 임의 유형의 문자 수를 의미합니다.

DBCS 그래픽 필드와 함께 와일드 카드 기능을 사용할 때는 2바이트 와일드 카드(별표와 밑줄)만 허용됩니다. 다음과 같은 특수 규칙이 적용됩니다.

- 2바이트 밑줄은 하나의 2바이트 문자를 의미합니다.
- 2바이트 별표는 2바이트 문자 수를 의미합니다.

OPNQRYF(조회 파일 열기) 명령을 통한 DBCS 필드 비교

두 필드나 상수를 비교할 경우 유형에 호환성이 있으면 고정 길이 필드를 가변 길이 필드와 비교할 수 있습니다.

표 52에서는 OPNQRYF 명령을 통해 2바이트 문자 세트(DBCS) 필드에 대해 수행할 수 있는 유효한 비교에 대해 설명합니다.

표 52. OPNQRYF 명령을 통해 DBCS 필드에 대해 수행할 수 있는 유효한 비교

	모든 숫자	문자	16진	DBCS 개방	DBCS 선택	DBCS 전용	DBCS 그래픽	UCS2 그래픽	날짜	시간	시간소인
모든 숫자	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음
문자	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함
16진	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효함	유효함	유효함
DBCS 개방	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효함	유효함	유효함	유효함
DBCS 선택	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효함	유효함	유효함	유효함
DBCS 전용	유효하지 않음	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효함	유효하지 않음	유효하지 않음	유효하지 않음
DBCS 그래픽	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
UCS2 그래픽	유효하지 않음	유효함	유효하지 않음	유효함	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음
날짜	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음	유효하지 않음
시간	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함	유효하지 않음
시간소인	유효하지 않음	유효함	유효함	유효함	유효함	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효하지 않음	유효함

OPNQRYF(조회 파일 열기) 명령을 통해 DBCS 필드와의 연결 사용

OPNQRYF(조회 파일 열기) 연결 기능을 사용할 경우 i5/OS 라이센스 프로그램은 연결되는 필드의 자료 유형을 기초로 하여 결과 자료 유형을 할당합니다. 연결에 2바이트 문자 세트(DBCS) 필드가 포함되면 결과 자료 유형은 일반적으로 논리 파일 내의 연결 필드와 동일하지만 약간의 변환이 있습니다.

다음은 OPNQRYF 명령을 통해 DBCS 필드와 연결을 사용하는 경우 적용되는 규칙입니다.

- 연결에 하나 이상의 16진(H) 필드가 있으면, 결과의 자료 형태는 16진(H)이 됩니다.
- 연결에 하나 이상의 UCS2 그래픽 필드가 포함되어 있으면, 결과의 자료 유형은 UCS2 그래픽입니다.
- 연결에 모든 필드가 DBCS 전용(J)이면, 결과의 자료 유형은 가변 길이 DBCS 전용(J)이 됩니다.
- 연결에 하나 이상의 DBCS(O, E, J) 필드가 포함되었으나 16진(H) 필드나 UCS2 그래픽 필드가 없으면, 결과의 자료 유형은 가변 길이 DBCS 개방(O)입니다.

- 연결에 자료 유형 중 하나 이상의 가변 길이 필드가 포함되어 있으면, 결과의 자료 유형은 가변 길이입니다.
- DBCS 그래픽(G) 필드가 다른 DBCS 그래픽(G) 필드에 연결되어 있을 경우 결과의 자료 유형은 DBCS 그래픽(G)입니다.

관련 개념

339 페이지의『DBCS 필드 연결』

필드가 연결될 때 그 자료 형태가 변경될 수 있습니다. (결과 자료 형태는 시스템에 의해 자동적으로 결정됩니다.) 2바이트 문자 세트(DBCS) 필드가 연결될 경우 이러한 지침을 따르는 것이 좋습니다.

DBCS와 함께 정렬 순서 사용

정렬 순서가 지정되면 2바이트 문자 세트(DBCS) 자료의 변환은 이루어지지 않습니다. DBCS 선택이나 DBCS 개방 필드에 있는 SBCS 자료만이 변환됩니다. UCS2 자료는 변환됩니다.

데이터베이스 프로그래밍 관련 정보

다음은 데이터베이스 프로그래밍 주제와 관련되는 iSeries 서적 및 정보 센터 주제를 나열한 것입니다. PDF를 보거나 인쇄할 수 있습니다.

매뉴얼

- 백업 및 회복 이 책은 iSeries 서버의 복구 및 가용성 옵션에 대한 일반 정보를 제공합니다.
- IDDU 사용 이 책은 행정 비서, 비즈니스 전문가 또는 프로그래머가시스템에 대한 자료 사전, 파일 및 레코드를 설명할 수 있도록 대화식 자료 정의 유틸리티(DDU)의 사용에 관한 정보를 제공합니다.
- iSeries 사용에 대한 조회 이 보충 매뉴얼에서는 행정 비서, 비즈니스 전문가 또는 프로그래머에게 데이터베이스 파일에서 자료를 얻기 위해 iSeries용 IBM 조회을 사용하는 것에 관한 정보를 제공합니다. 또한 조회를 사인 온하는 방법, 선택된 자료에 관한 보고서를 작성하기 위해 조회를 정의하고 수행하는 방법 등에 대해 설명합니다.
- 보안 참조 이 책은 해당 권한이 없는 사용자가 시스템이나 자료를 사용하지 못하도록 하고, 의도적이거나 우발적인 손상 또는 파괴로부터 자료를 보호하고, 최신의 보안 정보를 유지하며, 시스템에 보안을 설정하기 위해 어떠한 시스템 보안 지원이 사용되어야 하는지에 관해 설명합니다.

기타 정보

- iSeries SQL 참조서용 DB2 UDB 정보 이 주제는 어플리케이션 프로그래머, 프로그래머 또는 데이터베이스 관리자에게 구조화 조회 언어 명령문과 그 매개변수에 관한 자세한 정보를 제공합니다.
- API(Application programming interfaces) 이 주제는 어플리케이션 프로그래머들에게 어플리케이션 프로그래밍 인터페이스를 사용하여 시스템 레벨 및 기타 i5/OS 어플리케이션을 개발하는 데 필요한 정보를 제공합니다.

- 백업 및 회복 이 주제에는 백업 및 회복 전략을 계획하는 방법, 자료를 위해 디스크 보호를 설정하는 방법, 시스템을 백업하는 방법, 장애 발생 시 시스템 종료를 제어하는 방법에 관한 정보가 들어 있습니다.
- 확약 제어 이 주제에는 데이터베이스 변경 내용 동기화를 보장하기 위해 확약 제어를 사용하는 것에 대한 정보가 들어 있습니다.
- 제어 언어(CL) 이 주제는 어플리케이션 프로그래머 및 시스템 프로그래머에게 제어 언어(CL) 및 명령에 대한 자세한 정보를 제공합니다.
- 데이터베이스 파일 관리 이 주제는 어플리케이션 프로그래머에게 어플리케이션 프로그램에서 파일 사용에 대한 정보를 제공합니다. 여기에는 파일 CPYF(복사) 명령과 대체 명령에 관한 주제들도 포함되어 있습니다.
- DDS 개념 이 주제는 어플리케이션 프로그래머에게 데이터베이스 파일(논리 및 실제 파일 모두) 및 특정 장치 파일(사용자 프로그램 외부에 대한 표시장치, 프린터 및 시스템간 통신 기능(ICF))을 설명하는 데 필요한 항목 및 키워드에 대한 설명을 제공합니다.
- 디스크 관리 이 주제는 정보를 지속적으로 가용한 상태로 유지하기 위해 디스크 장치 및 디스크 풀을 관리하고 보호할 수 있도록 도움을 줍니다.
- 분산 자료 관리 이 주제는 어플리케이션 프로그래머에게 리모트 파일 처리에 대한 정보를 제공합니다. 즉, i5/OS 분산 자료 관리(DDM)에 리모트 파일을 정의하는 방법, DDM 파일을 작성하는 방법, 어느 파일 유틸리티가 DDM을 통해 지원되는지와 다른 시스템과 관련이 된 경우에는 i5/OS DDM의 요구사항에 대해서도 설명합니다.
- i5/OS 국제화 이 주제는 자료 처리 관리자, 시스템 오퍼레이터 및 관리자, 어플리케이션 프로그래머, 일반 사용자 및 시스템 엔지니어에게 iSeries 시스템의 자국어 지원 기능을 이해하고 사용하기 위한 정보를 제공합니다. 그리고 iSeries 글로벌화 및 다국어 시스템의 계획, 설치, 구성 및 사용에 관한 정보도 제공합니다. 또한 다국어 자료의 데이터베이스 관리에 대한 설명과 다국어 시스템에 대한 어플리케이션 고려사항도 제공합니다.
- 저널 관리 이 주제는 시스템 관리 액세스 경로 보호(SMAPP), 로컬 저널 및 리모트 저널의 설정, 관리 및 문제 해결에 대한 정보를 제공합니다.
- 성능 이 주제는 시스템 조정에 관한 설명, 수집될 자료의 내용과 레코드 형식에 대한 정보가 포함된 성능 자료의 수집에 관한 정보, 시스템의 전반적인 조작을 제어하거나 변경할 시스템 값에 대한 작업 설명, 어느 사용자가 시스템을 사용 중인지 또는 어느 자원을 사용 중인지 등을 판별하기 위한 자료의 수집 방법에 관한 설명을 제공합니다.
- SQL 프로그래밍 이 주제는 어플리케이션 프로그래머, 프로그래머 또는 데이터베이스 관리자에게 구조화 조회 언어(SQL) 명령문을 설계, 기록, 실행 및 테스트하는 방법에 대한 개요를 제공합니다. 대화식 SQL에 대해서도 설명합니다.
- 작업 관리 이 주제는 프로그래머에게 작업 관리 환경의 작성 및 변경 방법에 대한 정보를 제공합니다.

PDF 파일 저장

워크스테이션에서 보거나 인쇄할 PDF를 저장하려면 다음을 수행하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 단추로 클릭하십시오(위의 링크를 마우스 오른쪽 단추로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.

3. PDF를 저장할 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Reader 다운로드

이 PDF를 보거나 인쇄하려면 Adobe Reader를 시스템에 설치해야 합니다. Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html) 에서 사본을 무료로 다운로드할 수 있습니다.

코드 라이센스 및 면책사항 정보

IBM은 사용자의 특정 요구에 맞게 유사한 기능을 생성할 수 있도록 모든 프로그래밍 코드 예제를 사용할 수 있는 비독점적인 저작권 라이센스를 부여합니다.

강행 법규에 규정된 보증 조항의 적용을 제외하고, IBM은 해당 프로그램 또는 기술 지원에 대한 상품성, 특정 목적에의 적합성 및 타인의 권리 비침해에 대한 묵시적 보증을 포함한(단, 이에 한하지 않음) 일체의 묵시적 또는 명시적인 보증이나 주장도 제공하지 않습니다.

IBM, IBM 프로그램 개발자 또는 공급자는, 손해 발생의 가능성을 통지 받은 경우를 포함한 어떠한 경우에도 다음에 대하여 책임 지지 않습니다.

1. 데이터의 손실 또는 손상
2. 직접적인, 특별한, 우연에 의한 또는 간접적인 손상 또는 이에 따른 경제적 손실 또는
3. 기대했던 이익, 사업, 수익, 영업권 또는 비용 절감이 실현되지 못함으로 인해 발생하는 손해

일부 관할권에서는 부수적 또는 결과적 손해의 제외사항이나 제한사항을 허용하지 않으므로, 상기 제외사항이나 제한사항이 귀하에게 적용되지 않을 수도 있습니다.

부록. 주의사항

이 정보는 미국에서 제공되는 제품과 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 일체의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(1) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용료 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 라이센스가 있는 프로그램 및 이 프로그램에 대해 사용 가능한 모든 라이센스가 있는 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA), 시스템 코드용 IBM 라이센스 계약 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 레벨 상태의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한, 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 문서의 사용자는 해당 데이터를 사용자의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM의 향후 방향 또는 의도에 관한 모든 언급은 별도의 통지없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원시 언어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 그러므로 IBM은 이 프로그램들의 신뢰성, 서비스 및 기능을 보장할 수 없습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 그 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (귀하의 회사명) (연도). 이 코드의 일부는 IBM Corp. 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. Copyright IBM Corp. _연도_. All rights reserved.All rights reserved.

이 정보를 소프트카피로 보는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

프로그래밍 인터페이스 정보

이 데이터베이스 프로그래밍 서적 문서는 사용자가 IBM i5/OS의 서비스를 확보하기 위해 프로그램을 작성할 수 있도록 프로그래밍 인터페이스를 제공합니다.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표입니다.

- AS/400
- C/400
- COBOL/400
- DB2
- DB2 Universal Database
- DRDA
- e(로고)서버
- i5/OS
- IBM
- IBM(로고)
- Integrated Language Environment
- iSeries
- RPG/400
- System/36
- System/38
- WebSphere

Microsoft, Windows, Windows NT 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

조건

다음 조건에 따라 본 발행물을 사용할 수 있습니다.

개인적 사용: 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의없이 본 발행물 또는 그 일부를 배포 또는 게시하거나 이에 대한 2차적 저작물을 만들 수 없습니다.

상업적 사용: 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 귀하 사업장 내에서만 복제, 배포 및 게시할 수 있습니다. 귀하의 사업장 외에서는 IBM의 명시적 동의없이 본 발행물의 2차적 저작물을 만들거나 본 발행물 또는 그 일부를 복제, 배포 또는 게시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 본 발행물이나 본 발행물에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대해서는 어떠한 허가나 라이센스 또는 권리도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 발행물의 사용이 IBM의 이익을 해친다고 판단하거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 것을 조건으로 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 본 발행물의 내용에 대해 어떠한 보증도 하지 않습니다. IBM은 상품성 및 특정 목적에의 적합성에 대한 보증을 포함하여 명시적이든 묵시적이든 일체의 보증없이 ″현상태대로″ 본 발행물을 제공합니다.