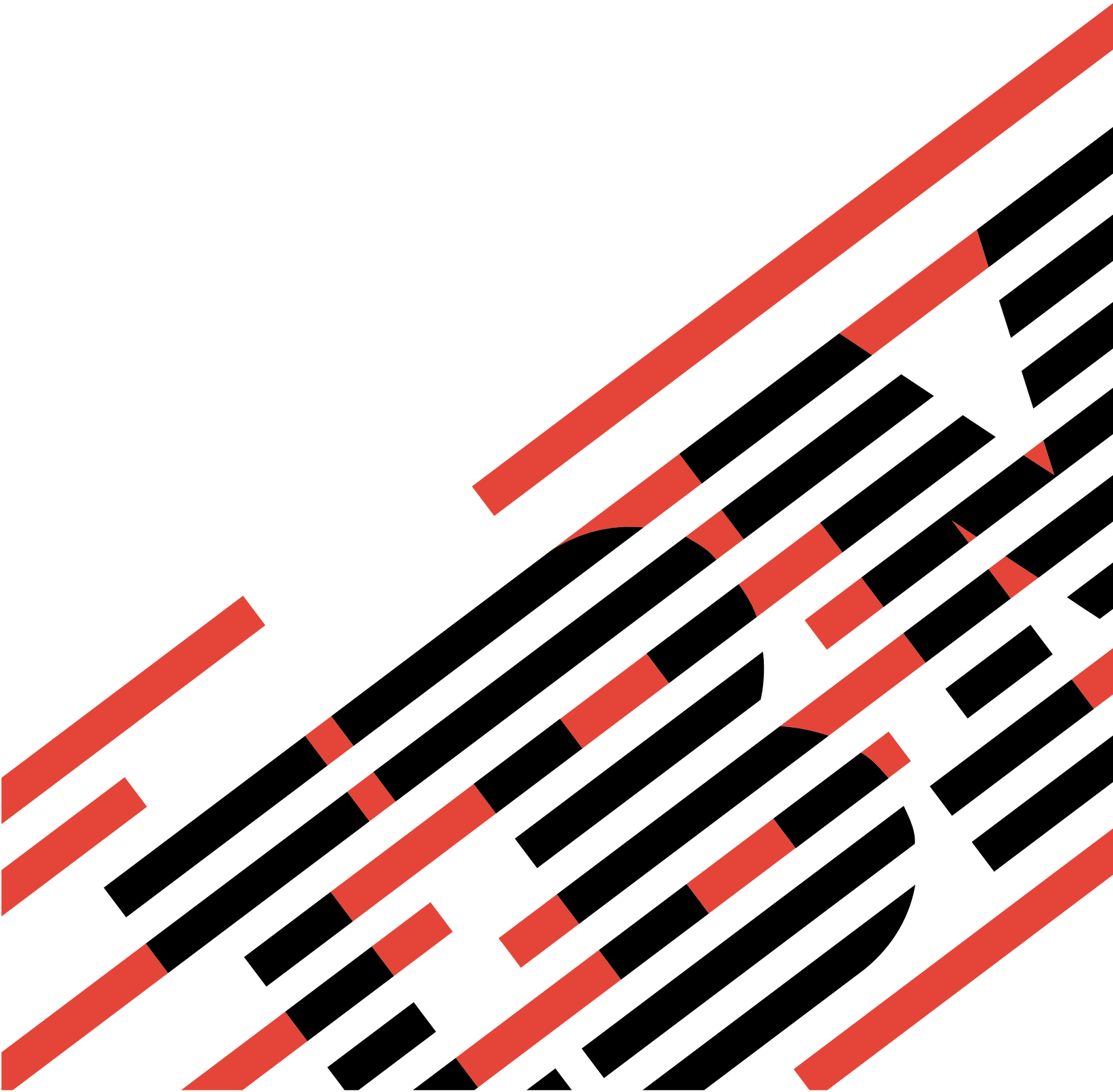
IBM

eserver

iSeries

확약 제어



IBM

@server

iSeries

확약 제어

목차

확약 제어

확약 제어는 데이터 무결성을 보장하는 기능입니다. 확약 제어를 사용하여 데이터베이스 파일이나 테이블과 같은 자원에 대한 변경 그룹을 트랜잭션으로 정의 및 처리할 수 있습니다. 확약 제어는 참여한 모든 시스템에서 개별 변경이 전체적으로 발생하거나 아무런 변경도 발생하지 않도록 합니다. iSeries용 DB2 Universal Database™는 확약 제어 기능을 사용하여 *NONE(확약하지 않음)이외의 분리 레벨을 사용하여 실행 중인 데이터베이스 트랜잭션을 확약 및 롤백합니다.

작업이나 작업 내의 활성 그룹 또는 시스템이 비정상적으로 종료하는 경우 시스템이 어플리케이션을 다시 시작할 수 있도록 확약 제어를 사용하여 어플리케이션을 설계할 수 있습니다. 확약 제어를 사용할 경우 어플리케이션이 다시 시작되었을 때 이전에 발생한 장애에 의한 불완전 트랜잭션으로 인해 데이터베이스에 부분적 갱신이 이루어지는 일은 없습니다.

다음 정보를 참조하여 iSeries 서버에서 확약 제어를 가동 및 실행하십시오.

V5R2의 새로운 사항
이 주제에서는 확약 제어에 대한 새로운 정보를 설명합니다.

이 주제 인쇄
이 정보 전체를 인쇄합니다.

확약 제어 개념
확약 제어가 작동하는 방식을 이해하려면 이 정보를 읽으십시오.

확약 제어 시작
확약 제어 시작에 대한 정보를 얻으려면 이 정보를 읽으십시오.

확약 제어 종료
확약 제어를 종료하기 위해 필요한 필수 조건 및 확약 제어를 종료하는 방법을 이해하려면 이 정보를 읽으십시오.

시스템에 의한 확약 제어 종료
시스템이 확약 제어 종료를 시작할 때 수행해야 할 태스크를 설명합니다.

트랜잭션 및 확약 제어 관리
확약 제어를 사용하여 시스템을 관리하기 위해 수행해야 하는 태스크를 설명합니다.

시나리오 및 예제: 확약 제어
한 회사에서 확약 제어를 설정하는 방법에 대한 시나리오 및 예제를 설명합니다. 확약 제어를 사용하는 프로그램의 코드 예제를 살펴 보십시오.

트랜잭션 및 확약 제어의 문제점 해결
확약 제어의 문제점을 해결해야 할 때 이 정보를 읽으십시오.

확약 제어 관련 정보
확약 제어와 관련된 주제, 매뉴얼, IBM 레드북 및 외부 웹 사이트를 참조하십시오.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

V5R2의 새로운 사항

V5R2의 경우 확약 제어에 대해 개선된 사항 및 추가된 사항이 많습니다. 다음은 이러한 개선된 사항 및 추가된 사항을 요약한 것입니다.

- **확약 제어 및 독립 디스크 풀(pool)**
  독립 디스크 풀(pool)이 가능한 새로운 라이브러리로 확약 제어를 지원합니다.
- **확약 제어에 대한 XA 트랜잭션 지원**
  확약 제어에 대한 XA 트랜잭션 지원으로 보다 향상된 XA 스펙 준수를 제공합니다.
- **확약 제어에 대한 iSeries Navigator 지원**
  iSeries Navigator는 확약 제어에 대한 지원을 제공합니다. 다음 주제에서 자세한 내용을 다룹니다.
  - 확약 제어 정보 표시
  - 확약 제어에 대한 XA 트랜잭션 지원
  - 통신 장애 후 트랜잭션 회복
  - 확약 및 롤백 강제 실행 시점 및 재동기화 취소 시점

이 릴리스의 새로운 사항이나 변경된 사항에 대한 다른 정보를 찾으려면 사용자 메모 를 참조하십시오.

이 주제 인쇄

PDF 버전을 보거나 다운로드하려면 확약 제어를 선택하십시오(약 600 KB 또는 112 페이지).

PDF 파일 저장

보거나 인쇄하기 위해 워크스테이션에 PDF를 저장하려면 다음과 같이 하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 버튼으로 클릭하십시오(위의 링크를 마우스 오른쪽 버튼으로 클릭).
2. **다른 이름으로 대상 저장...**을 클릭하십시오.
3. PDF를 저장하려는 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

Adobe Acrobat Reader 다운로드

이 PDF를 보거나 인쇄하기 위해 Adobe Acrobat Reader가 필요한 경우 Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html) 에서 사본을 다운로드할 수 있습니다.

확약 제어 개념

이 페이지에서는 확약 제어가 작동하는 방식, 확약 제어가 시스템과 대화하는 방식 및 네트워크의 다른 시스템과 대화하는 방식에 대해 이해하는 데 도움이 되는 정보를 제공합니다.

- 확약 제어 작동 방식
- 확약 및 롤백 조작 작동 방식
- 확약 정의
- 확약 제어 하의 자원
- 확약 제어 및 독립 디스크 풀(pool)
- 확약 제어를 위한 고려사항 및 제한사항
- 일괄처리 어플리케이션을 위한 확약 제어
- 2단계 확약 제어
- 확약 제어를 위한 XA 트랜잭션 지원
- SQL 서버 모드 및 스레드 범위의 트랜잭션

확약 제어 작동 방식

확약 제어는 데이터베이스 파일 또는 표와 같은 자원에 대한 변경 그룹을 트랜잭션으로 정의 및 처리할 수 있도록 하는 기능입니다. 확약 제어를 통해 개별 변경의 전체 그룹이 모든 참여 시스템에서 발생하도록 하거나 변경이 전혀 발생하지 않도록 할 수 있습니다. 예를 들어 자금을 저축에서 수표 계정으로 전송하는 경우 하나 이상의 변경이 그룹으로 발생합니다. 이러한 전송은 사용자에게는 단일 변경으로 보입니다. 그러나 저축 및 수표 계정이 둘 다 갱신되므로 둘 이상의 변경이 데이터베이스에 발생하게 됩니다. 두 가지 계정 모두 정확하게 갱신되도록 하려면 수표 및 예금 계정에 변경이 모두 발생하거나 변경이 전혀 발생하지 않아야 합니다.

확약 제어를 통해 다음을 수행할 수 있습니다.

- 트랜잭션 내 모든 변경이 영향을 받는 모든 자원에 대해 완료됩니다.
- 처리가 인터럽트되면 트랜잭션 내 모든 변경이 제거됩니다.
- 어플리케이션에서 트랜잭션에 오류가 발생했다고 판단되면 트랜잭션 중 변경된 내용은 모두 제거됩니다.

작업, 작업 내 활성 그룹 또는 시스템이 비정상적으로 종료하는 경우 확약 제어가 어플리케이션을 다시 시작할 수 있도록 어플리케이션을 설계할 수 있습니다. 확약 제어를 사용하면 어플리케이션이 다시 시작될 때 이전의 실패에서의 불완전한 트랜잭션으로 인해 데이터베이스에 부분적인 갱신이 일어나지 않도록 할 수 있습니다.

트랜잭션

트랜잭션은 사용자에게는 단일 아토믹(atomic) 변경으로 보여야 하는 시스템 상의 오브젝트에 대한 개별 변경의 그룹입니다.

주: iSeries Navigator에서는 트랜잭션이라는 용어를 사용하지만 문자 기반의 인터페이스에서는 논리 작업 단위(LUW)라는 용어를 사용합니다. 이 두 가지 용어는 서로 대신해서 사용할 수 있습니다. 이 주제에서는 특별히 문자 기반의 인터페이스를 언급하는 경우가 아니라면 트랜잭션이라는 용어를 사용합니다.

트랜잭션은 다음 중 하나가 될 수 있습니다.

- 데이터베이스 파일 변경이 발생하지 않는 조회.
- 하나의 데이터베이스 파일을 변경하는 간단한 트랜잭션.
- 하나 이상의 데이터베이스 파일을 변경하는 복잡한 트랜잭션.
- 하나 이상의 데이터베이스 파일을 변경하지만 이 변경이 트랜잭션 논리 그룹의 일부인 복잡한 트랜잭션.
- 둘 이상의 위치에 있는 데이터베이스 파일에 대한 간단한 또는 복잡한 트랜잭션. 이러한 데이터베이스 파일은 다음과 같을 수 있습니다.
  - 하나의 리모트 시스템에 상주.
  - 로컬 시스템 및 하나 이상의 리모트 시스템에 상주.
  - 로컬 시스템의 하나 이상의 저널에 지정. 각 저널은 **로컬 위치**로 간주됩니다.
- 데이터베이스 파일 이외의 오브젝트가 관련되는 로컬 시스템에서의 간단한 또는 복잡한 트랜잭션.

확약 및 롤백 조작 작동 방식

다음의 두 가지 조작이 확약 제어 하에 수행된 변경에 영향을 미칩니다.

- 확약 조작
  확약 조작은 이전 확약 또는 롤백 조작 이후 확약 제어 하에 발생한 모든 변경을 영구적으로 만듭니다. 시스템은 트랜잭션과 관련된 모든 잠금을 해제합니다.
- 롤백 조작
  롤백 조작은 이전 확약 또는 롤백 조작 이후로 변경된 내용을 모두 제거합니다. 시스템은 트랜잭션과 관련된 모든 잠금을 해제합니다.

다음과 같은 프로그래밍 언어 및 API가 확약 및 롤백 조작을 지원합니다.

언어 또는 **API**	확약	롤백
CL	COMMIT 명령	ROLLBACK 명령
ILE RPG/400	COMIT 조작 코드	ROLBK 조작 코드
ILE COBOL/400(R)	COMMIT 술어	ROLLBACK 술어
ILE C/400(R)	_Rcommit 함수	_Rrollbck 함수
PL/I	PLICOMMIT 서브루틴	PLIROLLBACK 서브루틴
SQL	COMMIT문	ROLLBACK문
SQL 호출 레벨 인터페이스 (CLI)	SQLTransact() 함수(트랜잭션의 확약 및 롤백에 사용)	
XA API	db2xa_commit() API	db2xa_rollback() API

다음 링크에서 이들 프로그래밍 언어 및 API에 대한 자세한 정보를 제공합니다.

- COBOL/400 User's Guide
- RPG/400 User's Guide

- ILE C/C++ Programmer's Guide
- CL 프로그래밍
- System API Programming
- iSeries용 DB2 UDB SQL 호출 레벨 인터페이스(ODBC)
- iSeries용 DB2 UDB 프로그래밍 개념

확약 조작

확약 조작은 이전 확약 또는 롤백 조작 이후 확약 제어 하에 발생한 모든 변경을 영구적으로 만듭니다. 시스템은 트랜잭션과 관련된 모든 잠금을 해제합니다.

시스템은 확약 요구를 수신하면 다음 단계를 수행합니다.

- 시스템은 확약 식별이 제공되는 경우 회복시 사용할 목적으로 저장합니다.
- 다음 두 가지 조건이 모두 참인 경우 시스템은 확약 조작을 수행하기 전에 파일에 레코드를 기록합니다.
  - 레코드가 확약 제어 하에서 로컬 또는 리모트 데이터베이스 파일에 추가되었습니다.
  - 파일이 열렸을 때 SEQONLY(*YES)가 지정되어 블록된 I/O 피드백이 시스템에 의해 사용되고 부분적인 레코드 블록이 존재합니다.

  그렇지 않으면 I/O 피드백 영역 및 I/O 버퍼는 변경되지 않습니다.
- 시스템은 확약 정의에 있는 각각의 API 확약 자원에 대하여 종료 프로그램을 확약 및 롤백하도록 호출합니다. 한 위치에 등록된 종료 프로그램이 둘 이상 있는 경우 시스템은 등록된 순서대로 해당 위치에 대한 종료 프로그램을 호출합니다.
- 저널에 할당된 자원에 대하여 레코드 변경이 이루어진 경우 시스템은 확약 정의와 연관된 모든 로컬 저널에 C CM 저널 항목을 기록합니다. 확약 제어 하의 저널 항목 순서에서는 확약 정의가 활성 상태인 동안 일반적으로 기록되는 항목을 보여줍니다.
- 시스템에서는 지연 중인 오브젝트 레벨 변경을 영구적으로 만듭니다.
- 시스템은 확약 제어를 목적으로 확보되고 유지되었던 레코드 및 오브젝트 잠금을 해제합니다. 이 자원들은 다른 사용자들이 사용할 수 있습니다.
- 시스템은 확약 정의의 정보를 변경하여 현재 트랜잭션이 종료되었음을 보여줍니다.

확약 조작이 성공적으로 수행되려면 시스템이 앞의 단계를 모두 올바르게 수행해야 합니다.

롤백 조작

롤백 조작은 이전 확약 또는 롤백 조작 이후로 변경된 내용을 모두 제거합니다. 시스템은 트랜잭션과 관련된 모든 잠금을 해제합니다. 시스템은 롤백 요구를 수신하면 다음 단계를 수행합니다.

- 시스템은 다음 두 가지가 모두 참인 경우 I/O 버퍼에서 레코드를 지웁니다.
  - 레코드가 확약 제어 하에서 로컬 또는 리모트 데이터베이스 파일에 추가되었습니다.

– 파일이 열렸을 때 SEQONLY(*YES)가 지정되어 블록된 I/O 피드백이 시스템에 의해 사용되고 아직 데이터베이스에 기록되지 않은 부분적인 레코드 블록이 존재합니다.

그렇지 않으면 I/O 피드백 영역 및 I/O 버퍼는 변경되지 않습니다.

- 시스템은 확약 정의에 있는 각각의 API 확약 자원에 대하여 종료 프로그램을 확약 및 롤백하도록 호출합니다. 한 위치에 등록된 종료 프로그램이 둘 이상 있는 경우 시스템은 등록된 순서와는 반대의 순서로 해당 위치에 대한 종료 프로그램을 호출합니다.
- 파일에서 레코드가 삭제되었으면 시스템은 레코드를 다시 파일에 추가합니다.
- 시스템은 이 트랜잭션 중에 레코드에 변경된 내용을 제거하고 원래의 레코드(사전 이미지)를 파일에 다시 위치시킵니다.
- 이 트랜잭션 중에 파일에 추가된 레코드가 있다면 이들 레코드는 파일에 삭제된 레코드로 남아 있게 됩니다.
- 트랜잭션 중에 저널에 지정된 자원에 레코드 변경이 있었다면 시스템은 롤백 조작이 발생했음을 나타내는 저널 항목 C RB를 저널에 추가합니다. 저널에는 롤백된 레코드 변경 사항의 이미지도 포함됩니다. 롤백 조작이 요구되기 전에 변경된 레코드의 사전 이미지와 사후 이미지가 저널에 들어가게 됩니다. 시스템은 또한 확약 가능한 자원이 그 저널에 지정되는 경우 디폴트 저널에 C RB 항목을 기록합니다.
- 시스템은 다음 위치 중 하나에 확약 제어 하에서 열린 파일을 위치시킵니다.
  – 이전 트랜잭션에서 액세스된 마지막 레코드
  – 이 확약 정의를 사용하여 파일에 대한 확약 조작이 수행되지 않은 경우 열기 위치에서

  순차 처리를 하는 경우 다음 고려사항이 특히 중요합니다.
- 시스템은 데이터베이스 파일에 대한 확약할 수 없는 변경은 롤백하지 않습니다. 예를 들어 열린 파일은 닫히지 않고 지워진 파일은 복원되지 않습니다. 시스템은 이 트랜잭션 동안 닫힌 파일은 다시 열거나 다시 위치시키지 않습니다.
- 시스템은 확약 제어를 목적으로 획득한 레코드 잠금을 해제하고 다른 사용자들이 그 레코드를 사용할 수 있게 됩니다.
- 현재 시스템에 의해 저장된 확약 식별은 동일한 확약 정의에 대한 마지막 확약 조작으로 제공된 확약 식별과 동일합니다.
- 시스템은 이 트랜잭션 중에 변경된 오브젝트 레벨의 확약 가능 변경 내용을 거꾸로 되돌리거나 롤백합니다.
- 확약 제어를 목적으로 획득된 오브젝트 잠금은 해제되고 다른 사용자들이 그 오브젝트를 사용할 수 있습니다.
- 시스템은 이전 확약 경계를 현재의 확약 경계로 설정합니다.
- 시스템은 확약 정의의 정보를 변경하여 현재 트랜잭션이 종료되었음을 보여줍니다.

롤백 조작이 성공적으로 수행되려면 시스템이 앞의 단계를 모두 올바르게 수행해야 합니다.

확약 정의

STRCMTCTL(확약 제어 시작) 명령을 사용하여 시스템에서 확약 제어를 시작할 때 확약 정의를 작성합니다. 또한 분리 레벨이 확약하지 않음 이외의 값인 경우 iSeries용 DB2 UDB는 자동으로 확약 정의를 작성합니다. 확약 정의에는 해당 작업 내에서 확약 제어 하에 변경되는 자원과 관련된 정보가 들어 있습니다. 시스템은 확약 정의가 종료할 때까지 확약 자원이 변경됨에 따라 확약 정의의 확약 제어 정보를 유지보수합니다. 시스템의 각 활동 트랜잭션은 확약 정의로 표현됩니다. 후속 트랜잭션은 각 활동 트랜잭션의 확약 또는 롤백 후에 확약 정의를 재사용할 수 있습니다.

확약 정의에는 일반적으로 다음이 포함됩니다.

- STRCMTCTL 명령 매개변수
- 확약 정의의 현재 상태
- 현재 트랜잭션 중에 변경된 내용이 포함된 데이터베이스 파일 및 기타 확약 가능한 자원에 대한 정보

작업 범위 잠금의 확약 정의의 경우 확약 제어를 시작한 작업만이 확약 정의를 알고 있습니다. 다른 작업은 해당 확약 정의를 알지 못합니다.

프로그램은 복수의 확약 정의를 시작 및 사용할 수 있습니다. 작업에 대한 각 확약 정의는 연관된 확약 가능한 자원이 있는 별도의 트랜잭션을 식별합니다. 이 트랜잭션은 해당 작업에 대해 시작된 다른 확약 정의와 연관된 트랜잭션과 별도로 확약 또는 롤백됩니다.

다음은 확약 정의에 대한 추가 정보를 제공합니다.

- 확약 정의 범위
- 확약 정의명
- 예: 작업 및 확약 정의

확약 정의 및 독립 디스크 풀(pool)에 대한 규칙은 확약 제어 및 독립 디스크 풀(pool)을 참조하십시오.

확약 정의 범위

확약 정의의 **범위**는 어떤 프로그램이 확약 정의를 사용하고 트랜잭션 중에 획득한 잠금 범위에 대한 내용을 결정합니다. 확약 정의를 시작하는 인터페이스는 확약 정의의 범위를 결정합니다. 확약 정의에는 네 가지 범위가 가능한데, 이들은 크게 두 가지 범주로 나뉩니다.

작업 범위 잠금을 수행하는 확약 정의

- 활성 그룹 레벨의 확약 정의
- 작업 레벨의 확약 정의
- 명시적으로 명명된 확약 정의

트랜잭션 범위의 잠금을 수행하는 확약 정의

- 트랜잭션 범위의 확약 정의

작업 범위의 잠금을 수행하는 확약 정의는 확약 정의를 시작한 작업에서 실행되는 프로그램만이 사용할 수 있습니다. 그에 비해 트랜잭션 범위의 잠금을 수행하는 확약 정의는 둘 이상의 작업이 사용할 수 있습니다.

일반적으로 어플리케이션은 활성 그룹 레벨 또는 작업 레벨의 확약 정의를 사용합니다. 이러한 확약 정의는 STRCMTCTL(확약 제어 시작) 명령을 사용하여 명시적으로 작성되거나 *NONE이 아닌 분리 레벨을 지정하여 SQL 어플리케이션이 실행할 때 시스템에 의해 내재적으로 작성됩니다.

활성 그룹 레벨 확약 정의

가장 많이 사용되는 범위는 활성 그룹 범위입니다. 활성 그룹 레벨의 확약 정의는 STRCMTCTL 명령이 확약 정의를 명시적으로 시작할 때 또는 확약 하지 않음 이외의 분리 레벨을 사용하여 실행되는 SQL 어플리케이션이 확약 정의를 시작할 때 디폴트 범위가 됩니다. 활성 그룹 내에서 실행되는 프로그램만이 해당 확약 정의를 사용합니다. 한 번에 하나의 작업에 대하여 많은 활성 그룹 레벨의 확약 정의가 사용될 수 있습니다. 그러나 각 활성 그룹 레벨 확약 정의는 하나의 활성 그룹과만 연관될 수 있습니다. 활성 그룹 내에서 실행되는 프로그램은 자신의 확약 가능 변경을 활성 그룹 레벨 확약 정의와만 연관시킬 수 있습니다.

iSeries Navigator, WRKCMTDFN(확약 정의에 대한 작업) 명령, DSPJOB(작업 표시) 명령 또는 WRKJOB(작업에 대한 작업) 명령이 활성 그룹 레벨의 확약 정의를 표시할 때 이 필드에는 다음과 같은 내용이 표시됩니다.

- 확약 정의 필드에는 활성 그룹의 이름이 표시됩니다. *DFTACTGRP 특수 값은 디폴트 활성 그룹을 나타냅니다.
- 활성 그룹 필드에는 활성 그룹 번호가 표시됩니다.
- 작업 필드에는 확약 정의를 시작한 작업이 표시됩니다.
- 스레드 필드에는 *NONE이 표시됩니다.

작업 레벨 확약 정의

STRCMTCTL CMTSCOPE(*JOB)를 발행하여 확약 정의를 작업 범위로 지정할 수 있습니다. 활성 그룹 레벨의 확약 정의가 시작되지 않은 활성 그룹에서 실행되는 프로그램이 그 작업에 대해 다른 프로그램에서 이미 시작하지 않았다면 작업 레벨의 확약 정의를 사용합니다. 하나의 작업에 대하여 하나의 작업 레벨 확약 정의만을 시작할 수 있습니다.

iSeries Navigator, WRKCMTDFN(확약 정의에 대한 작업) 명령, DSPJOB(작업 표시) 명령 또는 WRKJOB(작업에 대한 작업) 명령이 작업 레벨의 확약 정의를 표시할 때 이 필드에는 다음과 같은 내용이 표시됩니다.

- 확약 정의 필드에는 특수 값 *JOB이 표시됩니다.
- 활성 그룹 필드는 공백입니다.
- 작업 필드에는 확약 정의를 시작한 작업이 표시됩니다.
- 스레드 필드에는 *NONE이 표시됩니다.

주어진 활성 그룹의 경우 그 활성 그룹 내에서 실행되는 프로그램은 단 하나의 확약 정의를 사용할 수 있습니다. 따라서 활성 그룹 내에서 실행되는 프로그램은 작업 레벨의 확약 정의나 활성 그룹 레벨의 확약 정의 중

어느 하나를 사용할 수 있지만 동시에 둘 다를 사용할 수는 없습니다. SQL 서버 모드를 사용하지 않는 다중 스레드 작업에서 프로그램에 대한 트랜잭션 작업의 범위는 그것을 수행하는 스레드와는 관계없이 프로그램의 활성 그룹에 따라 적절한 확약 정의로 정해집니다. 여러 스레드가 하나의 동일한 활성 그룹을 사용하는 경우 서로 협조하여 트랜잭션 작업을 수행하고 올바른 시점에 확약 및 롤백이 일어나도록 해야 합니다.

작업 레벨의 확약 정의가 그 작업에 대해 사용 중이라고 하더라도 그 활성 그룹 내에서 확약 제어 요구나 작업 레벨 확약 정의에 대한 조작을 수행하는 프로그램이 없는 경우 프로그램은 활성 그룹 레벨의 확약 정의를 시작할 수 있습니다. 아니면 활성 그룹 레벨의 확약 정의를 시작하기 전에 먼저 작업 레벨 확약 정의를 종료해야 합니다. 활성 그룹 레벨의 확약 정의가 시작되지 못하도록 하는 확약 제어 요구 또는 작업 레벨 확약 정의에 대한 조치에는 다음과 같은 것들이 있습니다.

- 확약 제어 하에서 데이터베이스 열기(전체 또는 공유)
- QTNADDCR(확약 자원 추가) API를 사용하여 API 확약 자원 추가
- 트랜잭션 확약
- 트랜잭션 롤백
- 확약 제어 하에서 리모트 자원 추가
- QTNCHGCO(확약 옵션 변경) API를 사용하여 확약 옵션 변경
- QTNRBRQD(롤백 필수) API를 사용하여 확약 정의를 롤백 필수 상태로 만들기
- 확약 주기 ID 포함 매개변수에 QJOSJRNE(저널 항목 송신) API를 사용하여 현재의 확약 주기 ID를 포함하는 사용자 저널 항목 송신

마찬가지로 활성 그룹 내 프로그램이 현재 활성 그룹 레벨의 확약 정의를 사용하고 있는 경우 같은 활성 그룹 내에서 실행되는 프로그램이 작업 레벨 확약 정의를 사용할 수 있으려면 먼저 확약 정의를 종료해야 합니다.

데이터베이스 파일을 열 때 열린 파일에 대한 열기 범위는 활성 그룹일 수도 있고 작업 범위일 수도 있는데 한 가지 제한사항이 있습니다. 즉 확약 제어 하에서 프로그램이 파일을 열고 그 파일이 작업 범위로 지정된 경우 열기 요구를 하는 프로그램은 작업 레벨 확약 정의를 사용해야 합니다.

명시적으로 명명된 확약 정의

명시적으로 명명된 확약 정의는 어플리케이션에서 사용하는 트랜잭션에 영향을 미치지 않고 시스템에서 자체 확약 제어 트랜잭션을 수행해야 할 때 시스템에서 시작됩니다. 이러한 유형의 확약 정의를 시작하는 기능의 예는 문제점 기록부입니다. 어플리케이션에서는 명시적으로 명명된 확약 정의를 시작할 수 없습니다.

iSeries Navigator, WRKCMTDFN(확약 정의에 대한 작업) 명령, DSPJOB(작업 표시) 명령 또는 WRKJOB(작업에 대한 작업) 명령이 명시적으로 명명된 확약 정의를 표시할 때 이 필드에는 다음과 같은 내용이 표시됩니다.

- 확약 정의 필드에는 시스템에서 지정한 이름이 표시됩니다.
- 활성 그룹 필드는 공백입니다.
- 작업 필드에는 확약 정의를 시작한 작업이 표시됩니다.
- 스레드 필드에는 *NONE이 표시됩니다.

트랜잭션 범위의 확약 정의

트랜잭션 범위의 확약 정의는 트랜잭션 범위의 잠금에 대한 XA API를 사용하여 시작됩니다.

이 API는 활성 그룹 기반이 아닌 스레드 기반 또는 SQL 기반의 확약 제어 프로토콜을 사용합니다. 즉 이 API는 트랜잭션 작업이 수행되는 동안 확약 정의를 특정 스레드나 SQL 연결과 연관시키고 트랜잭션을 확약 또는 롤백하는 데 사용됩니다. 시스템은 이러한 확약 정의를 API 프로토콜에 따라 트랜잭션 작업을 수행하는 스레드에 첨부합니다. 이 확약 정의는 다른 작업의 스레드에서도 사용할 수 있습니다.

iSeries Navigator, WRKCMTDFN(확약 정의에 대한 작업) 명령, DSPJOB(작업 표시) 명령 또는 WRKJOB(작업에 대한 작업) 명령이 트랜잭션 범위의 확약 정의를 표시할 때 이 필드에는 다음과 같은 내용이 표시됩니다.

- 확약 정의 필드에는 특수 값 *TNSOBJ가 표시됩니다.
- 활성 그룹 필드는 공백입니다.
- 작업 필드에는 확약 정의를 시작한 작업이 표시됩니다.
- 스레드 필드에는 확약 정의가 첨부되는 스레드가 표시됩니다(또는 현재 어떤 스레드에도 확약 정의가 첨부되어 있지 않다면 *NONE).

확약 정의명

시스템은 작업에 대해 시작된 모든 확약 정의에 이름을 부여합니다. 다음 표에서는 다양한 확약 정의 및 특정 작업에 대한 연관된 이름을 보여줍니다.

활성 그룹	확약 범위	확약 정의명
모두	작업	*JOB
디폴트 활성 그룹	활성 그룹	*DFTACTGRP
사용자가 명명한 활성 그룹	활성 그룹	활성 그룹 이름(예: PAYROLL)
시스템에서 명명한 활성 그룹	활성 그룹	활성 그룹 번호(예: 0000000145)
없음	명시적으로 명명됨	QDIR001(시스템 전용으로 시스템에서 정의하는 확약 정의의 예) 시스템에서 정의하는 확약 정의명은 Q로 시작합니다.
없음	트랜잭션	*TNSOBJ

통합 언어 환경(ILE)으로 컴파일된 프로그램만이 디폴트 활성 그룹 이외의 활성 그룹에 대한 확약 제어를 시작할 수 있습니다. 그러므로 작업이 하나 이상의 ILE 컴파일 프로그램을 실행하고 있는 경우에만 복수의 확약 정의를 사용할 수 있습니다. 통합 언어 환경(R)에 대한 자세한 내용은 ILE 개념 을 참조하십시오.

기본 프로그램 모델(OPM) 프로그램은 디폴트 활성 그룹에서 실행되고 디폴트로 *DFTACTGRP 확약 정의를 사용합니다. OPM 및 ILE 혼합 환경에서 모든 프로그램에 의해 이루어지는 모든 확약 가능 변경이 함께 확약되거나 롤백되어야 하는 경우 작업은 작업 레벨 확약 정의를 사용해야 합니다.

활성 그룹 범위로 열려 있는 데이터베이스 파일은 활성 그룹 레벨의 확약 정의나 작업 레벨의 확약 정의와 연관될 수 있습니다. 작업 범위로 열려 있는 데이터베이스 파일은 작업 레벨의 확약 정의와만 연관될 수 있습니다. 그러므로 OPM 또는 ILE 중 작업 범위의 확약 제어 하에 데이터베이스 파일을 여는 프로그램은 모두 작업 레벨 확약 정의를 사용해야 합니다.

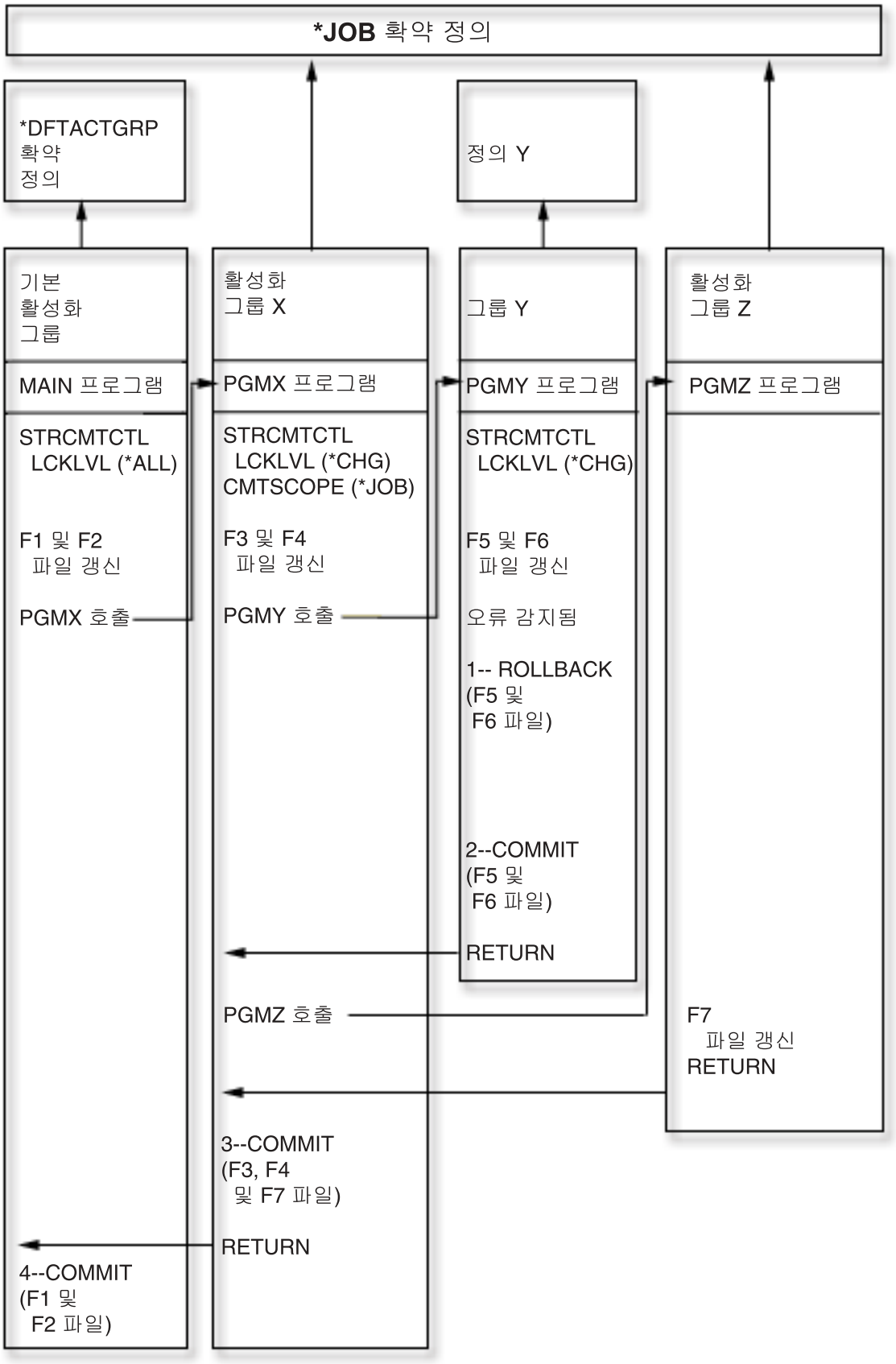
어플리케이션 프로그램에서는 확약 제어를 요구할 때 확약 정의명을 사용하여 특정 확약 정의를 식별하지 않습니다. 확약 정의명은 작업에 대한 특정 확약 정의를 식별하기 위해 주로 메세지에서 사용됩니다.

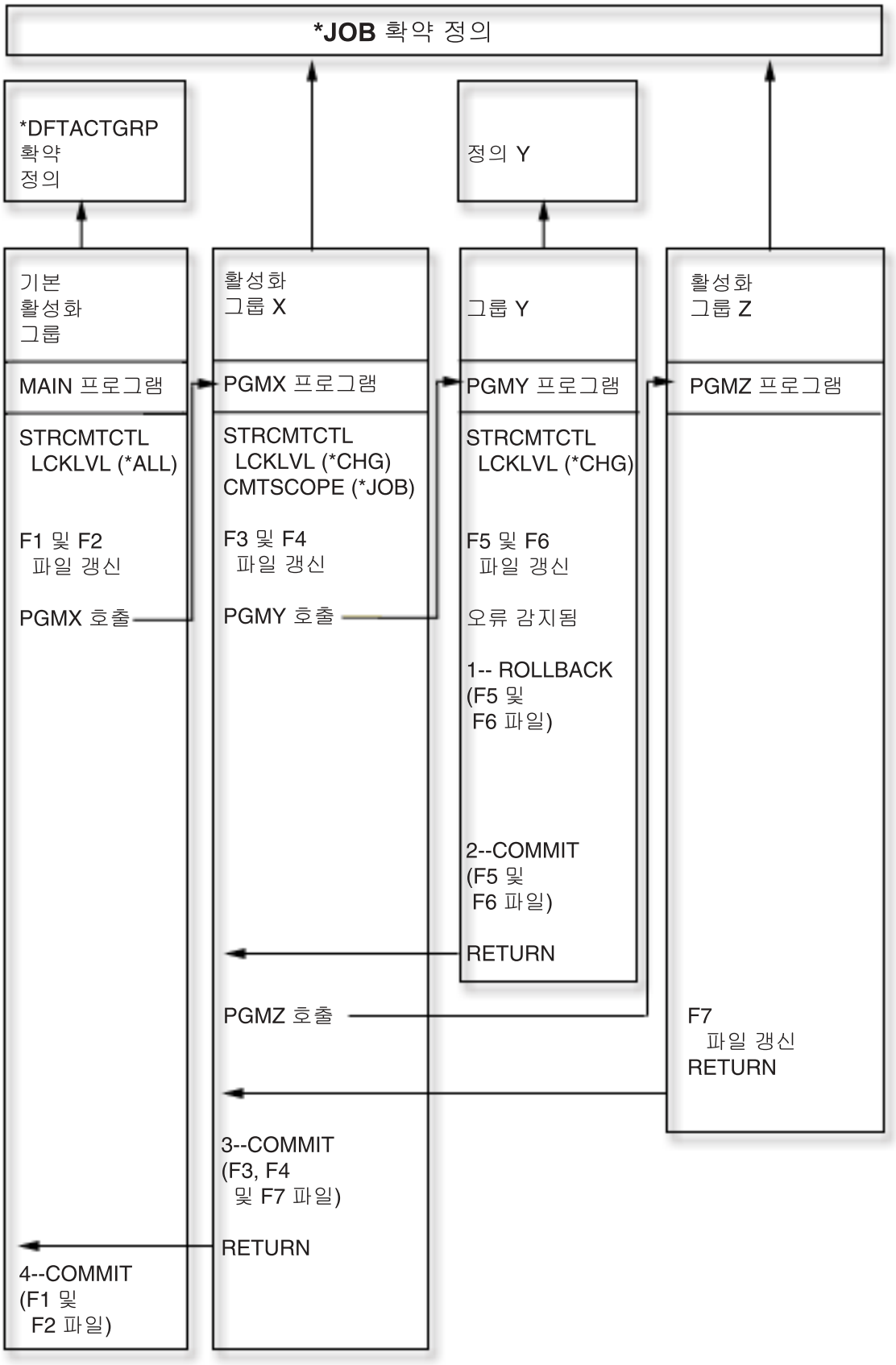
활성 그룹 레벨의 확약 정의의 경우 시스템은 요구한 프로그램이 어떤 활성 그룹에서 실행되고 있는지에 근거하여 어떤 확약 정의를 사용할 것인지 결정합니다. 어떤 위치에서든 활성 그룹 내에서 실행되는 프로그램은 단일 확약 정의를 사용할 수 있으므로 가능합니다.

트랜잭션 범위의 잠금을 실행하는 트랜잭션의 경우 XA API 및 CLI에 추가되는 트랜잭션 관련 속성으로 호출 스레드가 사용하는 확약 정의가 무엇인지 판별합니다.

예: 작업 및 확약 정의

다음 그림은 복수 확약 정의를 사용하는 작업의 예를 보여줍니다. 각 활성 그룹 레벨에서 어떤 파일 갱신이 확약 또는 롤백되는지를 보여줍니다. 이 예는 모든 프로그램에 의한 데이터베이스 파일에 대한 갱신은 모두 확약 제어 하에서 이루어지는 것으로 간주됩니다.


*JOB 확약 정의
*DFTACTGRP
확약
정의
정의 Y
기본
활성화
그룹
활성화
그룹 X
그룹 Y
활성화
그룹 Z
MAIN 프로그램
PGMX 프로그램
PGMY 프로그램
PGMZ 프로그램
STRCMTCTL
LCKLVL (*ALL)
F1 및 F2
파일 갱신
PGMX 호출
STRCMTCTL
LCKLVL (*CHG)
CMTSCOPE (*JOB)
F3 및 F4
파일 갱신
PGMY 호출
STRCMTCTL
LCKLVL (*CHG)
F5 및 F6
파일 갱신
오류 감지됨
1-- ROLLBACK
(F5 및
F6 파일)
2--COMMIT
(F5 및
F6 파일)
RETURN
PGMZ 호출
F7
파일 갱신
RETURN
3--COMMIT
(F3, F4
및 F7 파일)
RETURN
4--COMMIT
(F1 및
F2 파일)


*JOB 확약 정의
*DFTACTGRP
확약
정의
정의 Y
기본
활성화
그룹
활성화
그룹 X
그룹 Y
활성화
그룹 Z
MAIN 프로그램
PGMX 프로그램
PGMY 프로그램
PGMZ 프로그램
STRCMTCTL
LCKLVL (*ALL)
F1 및 F2
파일 갱신
PGMX 호출
STRCMTCTL
LCKLVL (*CHG)
CMTSCOPE (*JOB)
F3 및 F4
파일 갱신
PGMY 호출
STRCMTCTL
LCKLVL (*CHG)
F5 및 F6
파일 갱신
오류 감지됨
1-- ROLLBACK
(F5 및
F6 파일)
2--COMMIT
(F5 및
F6 파일)
RETURN
PGMZ 호출
F7
파일 갱신
RETURN
3--COMMIT
(F3, F4
및 F7 파일)
RETURN
4--COMMIT
(F1 및
F2 파일)

*

다음 표에서는 이전 그림의 시나리오가 변경되는 경우 파일이 확약 또는 롤백되는 방식을 보여줍니다.

작업의 복수 확약 정의에 대한 추가 예

시나리오 변경	이 파일 변경 시 미치는 영향			
	F1 및 F2	**F3 및 F4**	**F5 및 F6**	**F7**
PGMX는 확약 조작 대신 롤백 조작을 수행합니다(3= =COMMIT이 ROLLBACK이 됨).	아직 지연 중	롤백됨	이미 확약됨	롤백됨
PGMZ는 PGMX로 돌아오기 전에 확약 조작을 수행합니다.	아직 지연 중	PGMZ에 의해 확약됨	이미 확약됨	확약됨
PGMZ는 F7 파일을 갱신한 후에 CMTSCOPE(*ACTGRP)를 지정하여 확약 제어를 시작하려 합니다. 작업 레벨 확약 정의를 사용하여 변경이 지연되므로 시도는 실패합니다.	아직 지연 중	아직 지연 중	이미 확약됨	아직 지연 중
PGMX는 COMMIT(*YES)를 사용하여 확약 제어를 시작하지 않고 F3 및 F4 파일을 열지 않습니다. PGMZ는 COMMIT(*YES)를 사용하여 F7 파일을 열려고 시도합니다.	아직 지연 중	확약 제어 하에 있지 않음	이미 확약됨	*JOB 확약 정의가 존재하지 않기 때문에 (PGMX가 작성하지 않았음) F7 파일을 열 수 없습니다.

확약 제어가 오브젝트와 작동하는 방식

확약 제어 하에 오브젝트를 두면 오브젝트는 확약 가능 자원이 됩니다. 이는 확약 정의를 사용하여 등록되고 해당 확약 정의에 대해 발생하는 각 확약 조작 및 롤백 조작에 참여합니다.

다음 주제에서는 확약 가능 자원의 이러한 속성을 설명합니다.

- 자원 유형
- 위치
- 확약 프로토콜
- 액세스 목적

다음 링크에는 확약 제어 하의 자원에 대한 자세한 정보가 제공됩니다.

- 확약 가능 자원의 유형
- 로컬 및 리모트 확약 가능 자원
- 확약 가능 자원의 액세스 목적

- 확약 가능 자원의 확약 프로토콜
- 저널된 파일 및 확약 제어
- 확약 제어 하의 저널 항목 순서
- 확약 주기 ID
- 레코드 잠금

확약 가능 자원의 유형

아래 표에서는 다음과 같은 내용을 보여줍니다.

- 확약 가능 자원의 유형
- 확약 제어 하에 놓이는 방식
- 확약 제어에서 제거되는 방식
- 자원 유형에 적용되는 제한사항

자원 유형	확약 제어 하에 놓이는 방식	확약 제어에서 제거하는 방식	확약 가능한 변경의 종류	제한사항
FILE- 로컬 데이터베이스 파일	확약 제어 하에서 열기[1]	지연된 변경이 없으면 파일 닫기 파일을 닫을 때 변경이 지연되면 다음 확약 또는 롤백 조작을 수행한 후	레코드 레벨 변경	단일 트랜잭션에 대하여 최대 500 000 000개의 레코드[2]
DDL- 로컬 SQL 표 및 XQL 콜렉션에 대한 오브젝트 레벨 변경	확약 제어 하에서 SQL 실행	오브젝트 레벨 변경 후 확약 또는 롤백 조작 수행	다음과 같은 오브젝트 레벨 변경: • SQL 패키지 작성 • SQL 표 작성 • SQL 표 드롭	SQL을 사용한 오브젝트 레벨 변경만 확약 제어 하에 놓입니다.
DDM- 리모트 분산 자료 관리 (DDM) 파일	확약 제어 하에서 열기. DDM에 대한 확약 제어 지원에 확약 제어 및 분산 자료 관리에 대한 자세한 정보가 들어 있습니다.	지연된 변경이 없으면 파일 닫기 파일을 닫을 때 변경이 지연되면 다음 확약 또는 롤백 조작을 수행한 후	레코드 레벨 변경	
LU 6.2- 보호 대화	대화 시작[3]	대화 종료		
DRDA(R)- 분산 관계형 데이터베이스	SQL CONNECT문 사용	연결 종료		

자원 유형	확약 제어 하에 놓이는 방식	확약 제어에서 제거하는 방식	확약 가능한 변경의 종류	제한사항
API- 로컬 API 확약 자원	QTNADDCR(확약 자원 추가) API	QTNRMVCR(확약 자원 제거) API	사용자 프로그램이 판별합니다. 이러한 변경 내용 추적을 지원하기 위해 QJOSJRNE(저널 항목 송신) API를 사용하여 사용자 프로그램에서 저널 항목을 기록할 수 있습니다.	어플리케이션은 확약, 롤백 또는 재동기화 조작 중에 종료 프로그램이 호출되도록 해야 합니다.
TCP-TCP/IP 연결	TCP/IP 연결을 사용하도록 정의된 RDB에 SQL CONNECT문을 사용하거나 TCP/IP 위치를 사용하여 정의된 DDM 파일 열기	지연된 변경이 없는 경우 SQL 연결을 종료하거나 DDM 파일을 닫기 DDM 파일에 지연된 변경이 있는데 닫는 경우 다음 확약 또는 롤백 조작을 수행한 후에 연결이 닫힙니다.		

주:

[1]확약 제어 하에 데이터베이스 파일을 위치시키는 방법에 대한 자세한 내용은 적절한 언어 참조서를 참조하십시오. 확약 제어를 위한 관련 정보는 사용자가 사용할 수 있는 언어 매뉴얼로 링크됩니다.

[2]QAQQINI 파일을 사용하여 500 000 000 한계를 줄일 수 있습니다. 이에 대한 지침은 트랜잭션 크기 관리를 참조하십시오.

[3]DDM 연결이 시작되면 DDM 파일은 PTCCNV(*YES)를 지정하고 DDM 파일이 SNA 리모트 위치를 사용하여 정의되며 LU6.2 자원은 DDM 자원과 함께 추가됩니다.

DRDA 연결이 시작되면 다음 내용이 모두 참인 경우 LU6.2 자원이 DRDA 자원과 함께 추가됩니다.

- 프로그램이 분산 작업 단위 연결 프로토콜을 사용하고 있습니다.
- SNA 리모트 위치를 사용하여 정의된 RDB에 대해 연결되어 있습니다. 보호 대화 시작에 대한 자세한 내용은 APPC Programming 을 참조하십시오.

로컬 및 리모트 확약 가능 자원

확약 가능 자원은 로컬 자원일 수도 있고 리모트 자원일 수도 있습니다.

로컬 확약 가능 자원

로컬 확약 가능 자원은 어플리케이션과 동일한 시스템에 상주합니다. 확약 제어 하의 자원과 연관되는 각 저널은 로컬 위치로 간주될 수 있습니다. 저널하지 않고 등록된 모든 자원(DDL 자원 및 API 자원에 대해서 선택적)은 별도의 로컬 위치로 간주될 수 있습니다.

확약 가능 자원이 독립 디스크 풀에 있는 경우 확약 정의가 다른 디스크 풀에 있으면 그 자원은 로컬로 간주되지 않습니다. 확약 가능 자원 및 독립 디스크 풀에 대한 자세한 내용은 확약 제어 및 독립 디스크 풀(pool)을 참조하십시오.

리모트 확약 가능 자원

리모트 확약 가능 자원은 어플리케이션과는 다른 시스템에 상주합니다. 리모트 위치는 리모트 시스템에 대한 각각의 고유한 대화에 대하여 존재합니다. 확약 정의는 하나 이상의 리모트 시스템 상의 하나 이상의 리모트 위치를 가질 수 있습니다.

시스템 디스크 풀 또는 독립 디스크 풀에 대하여 확약 제어 하에 로컬 자원을 둘 때 DRDA를 사용하여 다른 독립 디스크 풀에 있는 확약 제어 하의 자원에 액세스해야 합니다.

다음은 확약 가능 자원 유형 및 이들의 위치를 보여줍니다.

자원 유형	위치
FILE	로컬
DDL	로컬
API	로컬
DDM	리모트
LU62	리모트
DRDA	로컬 또는 리모트
TCP	리모트

확약 가능한 자원의 액세스 목적

자원이 확약 제어 하에 놓일 때 자원 관리자는 자원이 어떻게 액세스될 것인지를 표시합니다.

- 갱신
- 읽기 전용
- 미결정

액세스 목적은 자원이 트랜잭션에 어떻게 함께 참여하는지를 판별합니다. 다음 표에서는 특정 유형의 자원에 어떤 액세스 목적이 가능하며 등록 시 시스템이 자원에 대한 액세스 목적을 판별하는 방법을 보여줍니다.

자원 유형	가능한 액세스 목적	액세스 목적 판별 방법
FILE	갱신, 읽기 전용	파일이 열린 방식에 근거
DDL	갱신	항상 갱신
API	갱신	항상 갱신
DDM	갱신, 읽기 전용	파일이 열린 방식에 근거
LU62	미결정	항상 미결정
DRDA	갱신, 읽기 전용, 미결정	DRDA 레벨 1의 경우 다른 리모트 자원이 등록되지 않은 경우 액세스 목적이 갱신됩니다. 그렇지 않은 경우 액세스 목적은 읽기 전용입니다. DRDA 레벨 2의 경우 액세스 목적은 항상 미결정입니다.
TCP	미결정	항상 미결정

이미 등록되어 있는 자원의 액세스 목적은 새 자원이 등록될 수 있는지를 판별하는 것입니다. 다음과 같은 규칙이 적용됩니다.

- 액세스 목적이 갱신인 1단계 자원은 다음 중 어느 하나가 참일 때 등록할 수 없습니다.

- 액세스 목적이 갱신인 자원이 다른 위치에서 이미 등록되었습니다.
- 액세스 목적이 미결정인 자원이 다른 위치에서 이미 등록되었습니다.
- 액세스 목적이 미결정인 자원은 동일한 위치에서 이미 등록되었고 자원은 현재 트랜잭션 중에 변경되었습니다.

- 액세스 목적이 갱신인 2단계 확약 자원은 액세스 목적이 갱신인 1단계 확약이 이미 등록되어 있을 때 등록할 수 없습니다.

확약 가능 자원의 확약 프로토콜

확약 프로토콜은 1단계 또는 2단계 확약 처리에 자원이 참여해야 하는 기능입니다. API 확약 가능 자원을 제외한 로컬 자원은 항상 2단계 자원입니다.

확약 가능 자원이 독립 디스크 풀에 있는 경우 확약 정의가 다른 디스크 풀에 있으면 그 자원은 로컬 자원 또는 2단계 자원으로 간주되지 않습니다. 확약 가능 자원 및 독립 디스크 풀에 대한 자세한 내용은 확약 제어 및 독립 디스크 풀(pool)을 참조하십시오.

2단계 자원은 **보호 자원**이라고도 합니다. 리모트 자원 및 API 확약 가능 자원은 확약 제어 하에 놓일 때 1단계 자원 또는 2단계 자원으로 등록되어야 합니다. 다음 표에서는 어떤 유형의 확약 가능 자원이 확약 정의에서 1단계 자원과 공존할 수 있는지 보여줍니다.

자원 유형	공존 가능한 자원
1단계 API 자원	다른 로컬 자원. 리모트 자원은 안됨.
1단계 리모트 자원	같은 위치에 있는 다른 1단계 자원. 로컬 자원은 안됨.

저널된 파일 및 확약 제어

데이터베이스 파일(자원 유형 FILE 또는 DDM)이 확약 제어 하에서 출력을 위해 열리거나 확약하지 않음 이외의 분리 레벨을 사용하는 SQL 어플리케이션에 의해 참조되는 경우 그 데이터베이스 파일을 저널(기록)해야 합니다. 파일은 확약 제어 하에서 입력만을 위해서 열려고 하는 경우에는 저널할 필요가 없습니다. 다음과 같은 경우 오류가 발생합니다.

- 확약 제어 하에서 출력을 위해 데이터베이스 파일을 열려는 시도가 있었으나 파일이 현재 저널되지 않았습니다.
- 확약 제어 하에서 열린 파일에서 사용할 수 있는 확약 정의가 시작되지 않았습니다.

그 파일만 확약 제어 하에 열려 있을 때 데이터베이스 파일에 대해 사후 이미지만 저널되는 경우 시스템은 자동으로 사전 및 사후 이미지를 모두 저널링하기 시작합니다. 사전 이미지는 확약 제어 하에서 발생한 파일의 변경에 대해서만 기록됩니다. 확약 제어 하에 있지 않은 다른 변경 사항이 동시에 파일에 발생하는 경우 그러한 변경에 대해서는 사후 이미지만 기록됩니다.

시스템은 레코드 레벨의 확약 가능 변경 및 오브젝트 레벨의 확약 가능 변경을 저널에 자동으로 기록합니다. 레코드 레벨 변경의 경우 시스템은 필요에 따라 회복을 목적으로 저널 항목을 사용합니다. 회복을 목적으로 하는 경우 오브젝트 레벨의 확약 가능 변경의 항목을 사용하지 않습니다. 또한 시스템은 API 확약 자원에 대한

저널 항목을 자동으로 쓰지 않습니다. 그러나 API 자원에 대한 종료 프로그램은 QJOSJRNE(저널 항목 송신) API를 사용하여 감사 추적을 제공하거나 회복을 지원하기 위해 저널 항목을 기록합니다. 이 항목의 내용은 사용자 종료 프로그램에 의해 제어됩니다.

시스템은 저널 이외의 메카니즘을 사용하여 오브젝트 레벨의 확약 자원에 대한 회복을 수행합니다. 각각의 특정 API 확약 자원과 연관된 확약 및 롤백 종료 프로그램을 호출하여 API 확약 자원에 대한 회복이 수행됩니다. 종료 프로그램은 이 상황에서 필요한 실제 회복을 수행할 책임을 갖습니다.

저널링에 대한 자세한 내용은 저널 관리를 참조하십시오.

확약 제어 하의 저널 항목 순서

다음 표에서는 확약 정의가 활동 중인 동안 일반적으로 기록되는 항목 순서를 보여줍니다. 저널 코드 파인더를 사용하여 저널 항목의 내용에 대한 더 많은 정보를 얻을 수 있습니다.

확약 제어 항목은 최소한 다음 중 하나가 참인 경우에만 저널(로컬 또는 리모트)에 기록됩니다.

- 저널은 STRCMTCTL(확약 제어 시작) 명령에서 디폴트 저널로 지정됩니다.
- 저널에 지정된 최소 하나의 파일이 확약 제어 하에서 열립니다.
- 저널과 연관된 최소한 하나의 API 확약 자원이 확약 제어 하에 등록됩니다.

항목 유형	설명	기록되는 위치	기록되는 시점
C BC	확약 제어 시작	STRCMTCTL 명령에 지정된 경우 디폴트 저널	STRCMTCTL 명령이 발행될 때
		각 로컬 위치에 대한 저널	저널에 지정된 첫 번째 파일이 열리거나 API 자원이 저널에 등록될 때
C SC	확약 주기 시작	각 로컬 위치에 대한 저널	이 저널[1]에 지정된 파일에 대한 트랜잭션에 대하여 첫 번째 레코드 변경이 발생할 때
		API 자원에 대한 저널	*확약 주기 ID 포함* 키를 사용하여 QJOSJRNE API가 맨 처음 사용될 때
저널 코드 D 및 F	DDL 오브젝트 레벨 항목	갱신되는 오브젝트와 연관된 저널. 확약 주기 ID를 포함하는 저널 항목만이 트랜잭션의 일부인 DDL 오브젝트 레벨 변경을 나타냅니다.	갱신이 발생할 때
저널 코드 R	레코드 레벨 항목	갱신되는 파일과 연관된 저널.	갱신이 발생할 때
저널 코드 U	사용자 작성 항목	API 자원과 연관된 저널	어플리케이션 프로그램이 SNDJRNE 명령이나 QJOSJRNE API를 사용하는 경우
C CM	확약	각 위치에 대한 저널	확약이 정상적으로 완료되었을 때
		디폴트 저널	확약 가능 자원이 저널과 연관될 때

항목 유형	설명	기록되는 위치	기록되는 시점
C RB	롤백	각 로컬 위치에 대한 저널	롤백 조작이 완료된 후
		디폴트 저널	확약 가능 자원이 저널과 연관될 때
C LW	트랜잭션 종료	STRCMTCTL 명령에 지정된 경우 디폴트 저널. 시스템이 LW 헤더 레코드와 하나 이상의 상세 레코드를 기록합니다. 이 항목은 OMTJRNE(*NONE)이 STRCMTCTL 명령에 지정되거나 시스템 오류가 발생한 경우에만 작성됩니다.	확약 또는 롤백 조작이 완료될 때
C EC	확약 제어 종료	각 로컬 위치에 대한 저널	ENDCMTCTL(확약 제어 완료) 명령이 완료될 때
		디폴트 저널이 아닌 로컬 저널	해당 저널과 연관된 모든 확약 가능 자원이 확약 제어에서 제거된 후 확약 경계가 설정될 때
주: CRTJRN(저널 작성) 또는 CHGJRN(저널 변경) 명령의 FIXLENDTA(고정 길이 자료) 매개변수에 *LUW(논리 작업 단위) 값을 지정하여 저널 항목의 고정 길이 부분에 트랜잭션 정보를 포함시키도록 지정할 수 있습니다. FIXLENDTA (*LUW) 매개변수를 지정함으로써 각 C SC 저널 항목의 고정 길이 부분에는 현재 트랜잭션의 논리 작업 단위 ID(LUWID)가 포함됩니다. XA 트랜잭션에서와 마찬가지로 FIXLENDTA(*LUW) 매개변수를 지정하면 각 C SC 저널 항목의 고정 길이 부분에는 현재 트랜잭션의 XID가 포함됩니다. LUWID 또는 XID는 특정 트랜잭션에 복수의 저널 또는 시스템이 관련되었을 때 그 트랜잭션에 대한 모든 확약 주기를 찾는 데 도움이 됩니다.			

확약 주기 ID

확약 주기는 하나의 확약 경계에서 다음 확약 경계로 가는 데 걸리는 시간입니다. 시스템에서는 특정 확약 주기에 대한 모든 저널 항목을 함께 연관시키기 위해 **확약 주기 ID**를 지정합니다. 트랜잭션에 참여하는 각 저널은 각각의 확약 주기 및 확약 주기 ID를 가지고 있습니다.

확약 주기 ID는 확약 주기에 대하여 기록되는 C SC 저널 항목의 저널 순번입니다. 확약 주기 ID는 확약 주기 동안 기록되는 각각의 저널 항목에 위치하게 됩니다. 확약 주기 동안 둘 이상의 저널이 사용되는 경우 각 저널에 대한 확약 주기 ID는 달라집니다.

CRTJRN(저널 작성) 또는 CHGJRN(저널 변경) 명령의 FIXLENDTA(고정 길이 자료) 매개변수에 *LUW(논리 작업 단위) 값을 지정하여 저널 항목의 고정 길이 부분에 트랜잭션 정보를 포함시키도록 지정할 수 있습니다. FIXLENDTA (*LUW) 매개변수를 지정함으로써 각 C SC 저널 항목의 고정 길이 부분에는 현재 트랜잭션의 논리 작업 단위 ID(LUWID)가 포함됩니다. XA 트랜잭션에서와 마찬가지로 FIXLENDTA(*LUW) 매개변수를 지정하면 각 C SC 저널 항목의 고정 길이 부분에는 현재 트랜잭션의 XID가 포함됩니다. LUWID 또는 XID는 특정 트랜잭션에 복수의 저널 또는 시스템이 관련되었을 때 그 트랜잭션에 대한 모든 확약 주기를 찾는 데 도움이 됩니다.

QJOSJRNE(저널 항목 송신) API를 사용하여 API 자원에 대한 저널 항목을 작성할 수 있습니다. 옵션으로 이 저널 항목에 대한 확약 주기 ID를 포함시킬 수 있습니다.

확약 주기 ID를 사용하여 APYJRNCHG(저널된 변경 적용) 명령이나 RMVJRNCHG(저널된 변경 제거) 명령을 통해 저널된 변경을 확약 경계에 적용 또는 제거할 수 있습니다. 이러한 제한사항은 다음과 같이 적용됩니다.

- 확약 제어 하에서 이루어지는 대부분의 오브젝트 레벨 변경은 저널에 기록되지만 APYJRNCHG 및 RMVJRNCHG 명령을 사용하여 적용되거나 제거되지는 않습니다.
- QJOSJRNE API는 저널 코드 U를 갖는 사용자 작성 저널 항목을 기록합니다. 이 항목은 APYJRNCHG 및 RMVJRNCHG 명령을 사용하여 적용하거나 제거할 수 없습니다. 이 항목은 사용자 작성 프로그램을 사용하여 적용하거나 제거해야 합니다.

레코드 잠금

작업이 레코드 잠금을 확보하고 있고 다른 작업에서 갱신을 위해 그 레코드를 검색하려고 할 때 요구 작업은 대기하고 다음 중 하나가 발생할 때까지 활동 처리로부터 제거됩니다.

- 레코드 잠금이 해제됩니다.
- 지정된 대기 시간이 종료합니다.

하나 이상의 작업이 다른 작업에 의해 잠긴 레코드를 요구할 수 있습니다. 레코드 잠금이 해제되면 레코드를 요구한 첫 번째 작업이 그 레코드를 수신합니다. 잠긴 레코드를 기다릴 때 다음과 같은 작성, 변경 또는 대체 명령의 WAITRCD 매개변수에 대기 시간을 지정하십시오.

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

대기 시간을 지정할 때 다음을 고려하십시오.

- 값을 지정하지 않으면 프로그램은 처리에 대해 디폴트 대기 시간을 기다립니다.
- 트랜잭션 범위의 잠금만 실행하는 확약 정의의 경우 작업 디폴트 대기 시간은 다음에 지정할 수 있는 트랜잭션 잠금 대기 시간으로 대체됩니다.
  - xa_open API
  - JDBC 또는 JTA 인터페이스. 분산 트랜잭션은 이러한 API를 나열합니다.
- 지정된 시간 내에 레코드가 할당되지 못하면 통지 메세지가 고급 언어 프로그램으로 송신됩니다.
- 레코드의 대기 시간을 초과하면 작업 기록부로 송신된 메세지에 요구 작업을 기다리게 한 잠금 상태의 레코들 확보하고 있는 작업 이름이 표시됩니다. 레코드 잠금 예외가 발생하면 작업 기록부를 사용하여 어떤 프로그램을 변경할 것인지 판별하여 긴 시간 동안 레코드를 잠그지 않도록 합니다.

프로그램은 다음과 같은 이유로 긴 시간 동안 레코드 잠금을 유지하게 됩니다.

- 워크스테이션 사용자가 변경을 고려하는 동안 레코드는 잠금 상태에 있습니다.
- 레코드 잠금은 긴 확약 트랜잭션의 일부입니다. 작은 트랜잭션을 만들어 확약 조작이 더 자주 수행될 수 있도록 해보십시오.
- 바람직하지 않은 잠금이 발생했습니다. 예를 들어 파일이 고유 키를 사용하는 갱신 파일로 정의되고 프로그램은 추가 레코드를 파일에 갱신 및 추가한다고 가정합시다. 워크스테이션 사용자가 파일에 레코드를 추가하려고 하는 경우 프로그램은 키가 이미 있는지 판별하기 위해 레코드에 액세스를 시도할 수도 있습니다. 이러한 경우 프로그램은 워크스테이션 사용자에게 요구가 유효하지 않다고 알려줍니다. 레코드를 파일에서 검색할 때 레코드는 동일한 파일에 대하여 다른 읽기 조작에 의해 내재적으로 잠금이 해제되거나 명시적으로 해제될 때까지 잠금 상태로 있게 됩니다.

  주: 고급 언어 인터페이스를 사용하여 레코드 잠금을 해제하는 방법에 대한 정보는 해당 고급 언어 참조 매뉴얼을 참조하십시오. 확약 제어에 대한 관련 정보에 확약 제어와 함께 사용할 수 있는 고급 언어 매뉴얼에 대한 링크가 나와 있습니다.

  파일에서 검색된 레코드가 다음 확약 또는 롤백 조작까지 잠금 상태에 있으므로 LCKLVL(*ALL)을 지정하면 잠금 지속 시간이 훨씬 더 길어집니다. 다른 읽기 조작에 의해 내재적으로 해제되지 않으며 명시적으로 해제할 수 없습니다.

파일에 잠금을 수행할 수 있는 다른 기능은 활동 중 보관 기능입니다. 활동 중인 동안 서버 보관에서 활동 중 보관 기능에 대한 자세한 내용을 다룹니다.

확약 제어 및 독립 디스크 풀(pool)

독립 디스크 풀 및 독립 디스크 풀 그룹은 각기 별도의 OS/400 SQL 데이터베이스를 가질 수 있습니다. 이 데이터베이스로 확약 제어를 사용할 수 있습니다. 그러나 독립 디스크 풀 또는 독립 디스크 풀 그룹이 각각 별도의 SQL 데이터베이스를 갖고 있으므로 다음과 같은 고려사항을 따라야 합니다.

확약 정의에 대한 고려사항

확약 제어를 시작할 때 확약 정의는 QRECOVERY 라이브러리에 작성됩니다. 독립 디스크 풀 또는 독립 디스크 풀 그룹은 각각의 QRECOVERY 라이브러리 버전을 가지고 있습니다. 독립 디스크 풀에서 QRECOVERY 라이브러리의 이름은 QRCYxxxxx이고 여기에서 xxxxx는 독립 디스크 풀의 번호입니다. 예를 들어 독립 디스크 풀 39의 QRECOVERY 라이브러리명은 QRCY00039입니다. 또한 독립 디스크 풀이 디스크 풀 그룹의 일부인 경우 1차 디스크 풀에 QRCYxxxxx 라이브러리가 포함됩니다.

확약 제어를 시작하면 확약 정의가 해당 작업과 연관된 독립 디스크 풀의 QRECOVERY 라이브러리에 작성되며, 독립 디스크 풀에서 확약 제어를 활성화시킵니다.

확약 정의에 대한 다른 고려사항은 다음과 같습니다.

- 독립 디스크 풀에서 확약 제어가 사용 중인 동안 SETASPGRP(ASP 그룹 설정) 명령을 사용하면 다음과 같은 영향을 미치게 됩니다.
  - 독립 디스크 풀에서 전환하고 디스크 풀에서 자원이 확약 제어에 등록되면 SETASPGRP 명령은 실패하고 메세지 CPDB8EC, 이유 코드 2, ″스레드에 미확약 트랜잭션이 있습니다″가 발생합니다. 이 메세지 다음에는 메세지 CPFB8E9가 발생합니다.

  - 독립 디스크 풀에서 전환하고 확약 제어에 자원이 등록되지 않은 경우 확약 정의는 전환하려는 독립 디스크 풀로 이동됩니다.
  - 시스템 디스크 풀(ASP 그룹 *NONE)에서 전환하는 경우 확약 제어는 영향을 받지 않습니다. 확약 정의는 시스템 디스크 풀에 남아 있습니다.
  - 통지 오브젝트를 사용하는 경우 통지 오브젝트는 확약 정의와 동일한 독립 디스크 풀 또는 독립 디스크 풀 그룹에 있어야 합니다. 확약 정의를 다른 독립 디스크 풀이나 독립 디스크 풀 그룹으로 이동시키는 경우 통지 오브젝트 역시 이동된 다른 독립 디스크 풀이나 독립 디스크 풀 그룹에 있어야 합니다. 확약 정의가 비정상적으로 종료하면 다른 독립 디스크 풀이나 독립 디스크 풀 그룹의 통지 오브젝트가 갱신됩니다. 통지 오브젝트를 다른 독립 디스크 풀이나 독립 디스크 풀 그룹에서 찾을 수 없는 경우 갱신은 실패하고 메세지 CPF8358이 발생합니다.
- 독립 디스크 풀에서의 확약 정의 회복은 독립 디스크 풀 연결변환 처리 중에 수행되고 IPL 회복과 유사합니다.
- 독립 디스크 풀의 확약 정의는 시스템 IPL시에 회복되지 않습니다.
- 독립 디스크 풀의 단절변환은 확약 정의에 다음과 같은 영향을 미칩니다.
  - 독립 디스크 풀과 연관된 작업이 종료합니다.
  - 새로운 확약 정의를 독립 디스크 풀에서 작성할 수 없습니다.
  - 독립 디스크 풀에 상주하는 확약 정의를 사용할 수 없게 됩니다.
  - 작업에 첨부되어 있지는 않지만 독립 디스크 풀에 상주하는 확약 정의는 트랜잭션 범위의 잠금을 해제합니다.
- 독립 디스크 풀 데이터베이스에서 리모트 데이터베이스로 연결하기 위해 LU6.2 SNA 연결(보호 대화 또는 분산 작업 단위(DUW))을 사용할 수 없습니다. 독립 디스크 풀 데이터베이스에서 리모트 데이터베이스로 연결하기 위해 보호되지 않는 SNA 대화를 사용할 수 있습니다.

작업이나 스레드에 대해 확약 제어가 활동 중인 경우 확약 정의가 속해 있는 독립 디스크 풀이나 디스크 풀 그룹 외부의 자료에 대한 액세스는 마치 다른 시스템의 자료처럼 리모트로만 가능합니다. SQL CONNECT문을 발행하여 독립된 디스크 풀에 있는 관계형 데이터베이스(RDB)에 연결할 때 시스템은 리모트 연결을 설정합니다.

시스템 디스크 풀 및 기본 디스크 풀은 별도의 디스크 풀의 자료에 대한 읽기 전용 액세스를 위해 리모트 연결이 필요하지 않습니다. 마찬가지로 독립 디스크 풀에서도 시스템 디스크 풀이나 기본 디스크 풀의 자료에 대한 읽기 전용 액세스를 위해 리모트 연결이 필요하지 않습니다.

XA 트랜잭션 고려사항

XA 환경에서 각 데이터베이스는 별도의 자원 관리자로 간주됩니다. 트랜잭션 관리자가 동일한 트랜잭션 하의 두 데이터베이스에 액세스하려는 경우 XA 프로토콜 및 두 자원 관리자를 사용하여 2단계 확약을 수행해야 합니다.

각 독립 디스크 풀은 별도의 SQL 데이터베이스이므로 XA 환경에서 각 독립 디스크 풀 역시 별도의 자원 관리자로 간주됩니다. 어플리케이션 서버가 두 개의 서로 다른 독립 디스크 풀을 대상으로 트랜잭션을 수행하려면 트랜잭션 관리자 역시 2단계 확약 프로토콜을 사용해야 합니다.

독립 디스크 풀에 대한 자세한 내용은 독립 디스크 풀(pool)을 참조하십시오.

확약 제어의 고려사항 및 제한사항

다음은 확약 제어에 대한 여러가지 고려사항 및 제한사항입니다.

데이터베이스 파일 고려사항

- 확약 제어 하에서 공유 파일을 열도록 지정하면 그 이후 그 파일에 대한 사용은 확약 제어 하에서 열려야 합니다.
- LCKLVL(*ALL)을 사용하여(내재적으로 또는 고급 언어 프로그램에 의해 또는 OVRDBF(데이터베이스 파일로 대체) 명령을 사용하여) 읽기 전용으로 열린 파일에 SEQONLY(*YES)가 지정된 경우 SEQONLY(*YES)가 무시되고 SEQONLY(*NO)가 사용됩니다.
- 확약 제어 하의 레코드 레벨 변경은 저널에 기록됩니다. APYJRNCHG(저널된 변경 적용) 명령 또는 RMVJRNCHG(저널된 변경 제거) 명령을 사용하여 이러한 변경을 데이터베이스에 적용하거나 데이터베이스에서 제거할 수 있습니다.
- 확약 제어 하에서 파일의 사전 이미지와 사후 이미지가 모두 저널됩니다. 저널에 파일의 사후 이미지만 저널하도록 지정한 경우 시스템은 자동으로 확약 제어 하에 발생한 파일 변경의 사전 이미지를 저널합니다. 그러나 파일에 대한 모든 변경사항에 대하여 사전 이미지가 캡처되지 않으므로 이 파일에 대해서는 RMVJRNCHG 명령을 사용할 수 없습니다.

오브젝트 및 레코드 레벨 변경에 대한 고려사항

- 확약 제어 하에서 이루어지는 대부분의 오브젝트 레벨 변경은 저널에 기록되지만 APYJRNCHG 및 RMVJRNCHG 명령을 사용하여 적용되거나 제거되지는 않습니다. 그러나 QJOSJRNE(저널 항목 송신) API를 사용하여 다른 이벤트에 대해 저널 항목을 송신할 수 있습니다. 회복 중에 이러한 항목을 검색하고 사용자 작성 프로그램을 처리할 수 있습니다.
- SQL을 사용한 확약 제어 하의 오브젝트 레벨 및 레코드 레벨 변경은 요구 프로그램이 실행 중인 활성 그룹에 대해 현재 활동 중인 확약 정의를 사용합니다. 작업 레벨 또는 활성 그룹 레벨 확약 정의 중 어느 것도 활동 상태가 아니면 SQL은 활성 그룹 레벨의 확약 정의를 시작합니다. SQL을 사용하여 확약 제어 하에서 이루어진 변경에 대한 더 자세한 정보는 iSeries용 DB2 UDB SQL 프로그래밍 개념을 참조하십시오.

1단계 및 2단계 확약 고려사항

- 1단계 리모트 대화 또는 연결이 설정되는 동안 다른 위치로의 리모트 대화 또는 연결은 허용되지 않습니다. 확약 경계가 설정되고 모든 자원이 제거되면 위치가 변경될 수 있습니다.
- 2단계 확약을 사용하고 있는 경우 SBMRMTCMD(리모트 명령 제출) 명령을 사용하여 리모트 위치에서 확약 제어를 시작하거나 다른 확약 제어 조작을 수행할 필요가 없습니다. 시스템이 대신 이러한 기능을 수행합니다.

- 1단계 리모트 위치의 경우 SQL이 호출 스택에 있고 리모트 관계형 데이터베이스가 시스템에 없을 때 COMMIT 및 ROLLBACK CL 명령은 실패합니다. SQL이 호출 스택에 없으면 COMMIT 및 ROLLBACK 명령은 실패하지 않습니다.
- 1단계 리모트 위치의 경우 확약 제어는 리모트 자원에 대한 확약 가능 변경을 수행하기 전에 소스 시스템에서 시작되어야 합니다. SQL 프로그램이 *NONE이 아닌 확약 제어 옵션을 사용하여 실행되는 경우 시스템은 연결 시에 소스 시스템에서 분산 데이터베이스 SQL에 대한 확약 제어를 시작합니다. 첫 번째 리모트 자원이 확약 제어 하에 놓이면 시스템은 목표 시스템에서 확약 제어를 시작합니다.

저장 고려사항

저장을 수행하는 작업에 다음과 같은 유형의 확약 가능 변경을 갖는 하나 이상의 활동 확약 정의가 있는 경우 저장 조작이 금지됩니다.

- 저장 중인 라이브러리에 있는 파일에 대한 레코드 변경. 논리 파일의 경우 모든 관련된 실제 파일(PF)이 검사됩니다.
- 저장 중인 라이브러리 내의 모든 오브젝트 레벨 변경.
- QTNADDCR(확약 자원 추가) API를 사용하여 추가되었고 일반적인 저장 처리 허용 필드를 디폴트 값 N으로 설정한 모든 API 자원.

이것은 저장 조작이 부분적인 트랜잭션으로 인한 변경을 저장 매체에 저장하지 않도록 합니다.

오브젝트 잠금 및 레코드 잠금을 사용하여 확약 정의의 지연 변경이 저장 매체에 저장되지 않도록 할 수 있습니다. 이것은 오브젝트나 API 확약 자원과 연관된 오브젝트에 변경이 이루어질 때 잠금이 확보된 경우 API 확약 자원에 대해서만 참입니다.

기타 고려사항 및 제한사항

- 새로운 릴리스로 시스템을 업그레이드하기 전에 지연된 재동기화를 모두 완료하거나 취소해야 합니다. 자세한 내용은 소프트웨어 설치 전에 2단계 확약 무결성 확인을 참조하십시오.
- COMMIT 및 ROLLBACK 값은 확약 또는 롤백 중에 WRKACTJOB 기능 필드에 표시됩니다. 장시간 동안 기능이 COMMIT 또는 ROLLBACK으로 남아 있는 경우 다음과 같은 상황 중 하나가 발생했을 가능성이 있습니다.
  - 확약 또는 롤백 중에 자원 실패가 발생하면 재동기화가 필요합니다. 재동기화가 완료되거나 취소될 때까지 어플리케이션으로 제어가 리턴되지 않습니다.
  - 이 시스템은 확약 중에 읽기 전용으로 인가되었습니다. 제어는 확약을 시작한 시스템이 이 시스템으로 자료를 보낼 때까지 어플리케이션으로 리턴되지 않습니다.
  - 이 시스템은 확약 중에 생략되도록 인가되었습니다. 제어는 확약을 시작한 시스템이 이 시스템으로 자료를 보낼 때까지 어플리케이션으로 리턴되지 않습니다.

일괄처리 어플리케이션에 대한 확약 제어

일괄처리 어플리케이션에는 확약 제어가 필요할 수도 또는 필요하지 않을 수도 있습니다. 어떤 경우 일괄처리 어플리케이션이 입력 파일을 읽고 마스터 파일을 갱신하는 단일 기능을 수행할 수 있습니다. 그러나 비정상 종료 후 다시 시작하는 것이 중요하다면 이러한 종류의 어플리케이션에 확약 제어를 사용할 수 있습니다.

입력 파일은 레코드가 처리되었음을 나타내는 코드가 레코드에 들어 있는 갱신 파일입니다. 이 파일 및 갱신된 다른 파일은 모두 확약 제어 하에 놓이게 됩니다. 입력 파일에 이 코드가 있으면 이는 완료된 트랜잭션임을 나타냅니다. 프로그램은 입력 파일을 읽고 완료된 코드를 가지고 있는 레코드는 통과시킵니다. 이렇게 하여 정상 조건 및 재시작 조건에 동일한 프로그램 논리를 사용할 수 있습니다.

일괄처리 어플리케이션에 서로에게 종속적인 입력 레코드가 들어 있거나 스위치 또는 총계가 포함되는 경우 재시작에 대한 정보를 제공하는 데 통지 오브젝트를 사용할 수 있습니다. 통지 오브젝트에 들어 있는 값은 입력 파일 내에서 마지막으로 확약된 트랜잭션으로부터 다시 처리를 시작하는 데 사용됩니다.

입력 레코드가 서로에게 종속적인 경우 입력 레코드를 트랜잭션으로 처리할 수 있습니다. 일괄처리 작업은 최대 500 000 000개의 레코드를 잠글 수 있습니다. QAQQINI(옵션 파일 조회)를 사용하여 이 한계를 줄일 수 있습니다. CHGQRYA(조회 속성 변경) 명령의 QRYOPTLIB 매개변수를 사용하여 사용할 작업에 대한 조회 옵션 파일을 지정할 수 있습니다. 옵션 파일 조회의 COMMITMENT_CONTROL_LOCK_LEVEL 값을 작업에 대한 잠금 한계로 사용하십시오.

2000개의 잠금을 초과하는 확약 주기의 경우 시스템 성능이 눈에 띄게 저하될 것입니다. 그렇지 않은 경우 대화식 어플리케이션과 동일한 잠금 고려사항이 존재하지만 레코드가 일괄처리 어플리케이션에서 잠금 상태로 있는 시간 길이는 대화식 어플리케이션에서보다 덜 중요합니다.

2단계 확약 제어

2단계 확약 제어를 통해 여러 시스템 상의 확약 가능한 자원이 동기화될 수 있습니다. OS/400은 SNA LU 6.2 구조에 따라 2단계 확약을 지원합니다. 시스템에 의해 2단계 확약에 사용되는 내부 프로토콜에 대한 자세한 정보는 *SNA Transaction Programmer's Reference for LU Type 6.2, GC30-3084-05*를 참조하십시오. OS/400의 모든 지원되는 릴리스는 SNA LU 6.2의 Presumed Nothing 프로토콜 및 SNA LU 6.2의 Presumed Abort 프로토콜을 지원합니다.

2단계 확약은 분산 작업 단위(DUW) DRDA 프로토콜로서 TCP/IP를 사용하여 지원될 수도 있습니다. TCP/IP DUW 연결을 사용하려면 모든 시스템(어플리케이션 리퀘스터 및 어플리케이션 서버 모두)이 V5R1M0 이상이어야 합니다. DRDA에 대한 자세한 내용은 Open Group web site의 Open Group Technical Standard, *DRDA V2 Vol. 1: Distributed Relational Database Architecture*를 참조하십시오.

2단계 확약 하에서 시스템은 두 가지 웨이브로 확약 조작을 수행합니다.

- **준비 웨이브** 동안 자원 관리자는 트랜잭션 관리자에게 확약 요구를 발행합니다. 트랜잭션 관리자는 관리하는 다른 자원 및 다른 트랜잭션 관리자에게 트랜잭션이 확약될 준비가 되었음을 알립니다. 모든 자원 관리자는 확약될 준비가 되었음을 응답해야 합니다. 이것을 **인가**라고 합니다.

- **확약 웨이브** 동안 확약 요구를 시작하는 트랜잭션 관리자는 준비 웨이브 결과에 따라 무었을 할 것인지 결정합니다. 준비 웨이브가 성공적으로 완료되고 모든 참가자가 준비되었음으로 인가되는 경우 트랜잭션 관리자는 자신이 관리하는 모든 자원 및 다른 트랜잭션 관리자에게 트랜잭션을 확약하도록 지시합니다. 준비 웨이브가 성공적으로 완료되지 못한 경우 모든 트랜잭션 관리자 및 자원 관리자는 트랜잭션을 롤백하라는 지시를 받게 됩니다.

리모트 자원을 사용한 확약 및 롤백 조작

리모트 자원이 확약 제어 하에 있을 때 개시자는 모든 리모트 에이전트로 확약 요구를 보냅니다. 요구는 트랜잭션 프로그램 네트워크를 통해 보내집니다. 각 에이전트는 확약 조작 결과를 가지고 응답합니다.

준비 웨이브 중에 오류가 발생하면 개시자는 모든 에이전트로 롤백 요구를 송신합니다. 확약된 웨이브 중에 오류가 발생하면 시스템은 가능한 많은 위치를 확약 상태로 만들려고 시도합니다. 이러한 시도로 인해 발견적(heuristic) 혼합 상태가 될 수 있습니다. 가능한 상태에 대한 자세한 정보는 2단계 확약 제어에 대한 트랜잭션 상태를 참조하십시오.

모든 오류는 사용자에게 신호된 곳에서 개시자에게 다시 보내집니다. 디폴트 저널이 STRCMTCTL(확약 제어 시작) 명령에 지정되면 C LW 항목이 기록됩니다. 오류가 발생하면 OMTJRNE(*LUWID)가 지정되었어도 이 항목이 기록됩니다. 확약 가능한 자원을 수동으로 동기화하기 위해 오류 메세지 및 확약 정의에 대한 상태 정보와 함께 이들 항목을 사용할 수 있습니다.

리모트 자원이 확약 제어 하에 있을 때 개시자는 모든 리모트 에이전트로 롤백 요구를 보냅니다. 요구는 트랜잭션 프로그램 네트워크를 통해 보내집니다. 각 에이전트는 롤백 조작 결과를 가지고 응답합니다.

자세한 내용은 다음을 참조하십시오.

- 확약 처리에서의 역할
- 2단계 확약 제어에 대한 트랜잭션 상태
- 2단계 확약 제어에 대한 확약 정의

확약 처리에서의 역할

트랜잭션 확약에 하나 이상의 자원 관리자가 포함된 경우 각 자원 관리자는 트랜잭션에서 역할을 수행합니다. 자원 관리자는 트랜잭션 중에 변경된 내용을 확약 또는 롤백하는 책임을 갖게 됩니다. 자원 유형별 자원 관리자는 다음과 같습니다.

FILE
데이터베이스 관리자

DDM
데이터베이스 관리자

DDL
데이터베이스 관리자

DRDA
통신 트랜잭션 프로그램

LU62
통신 트랜잭션 프로그램

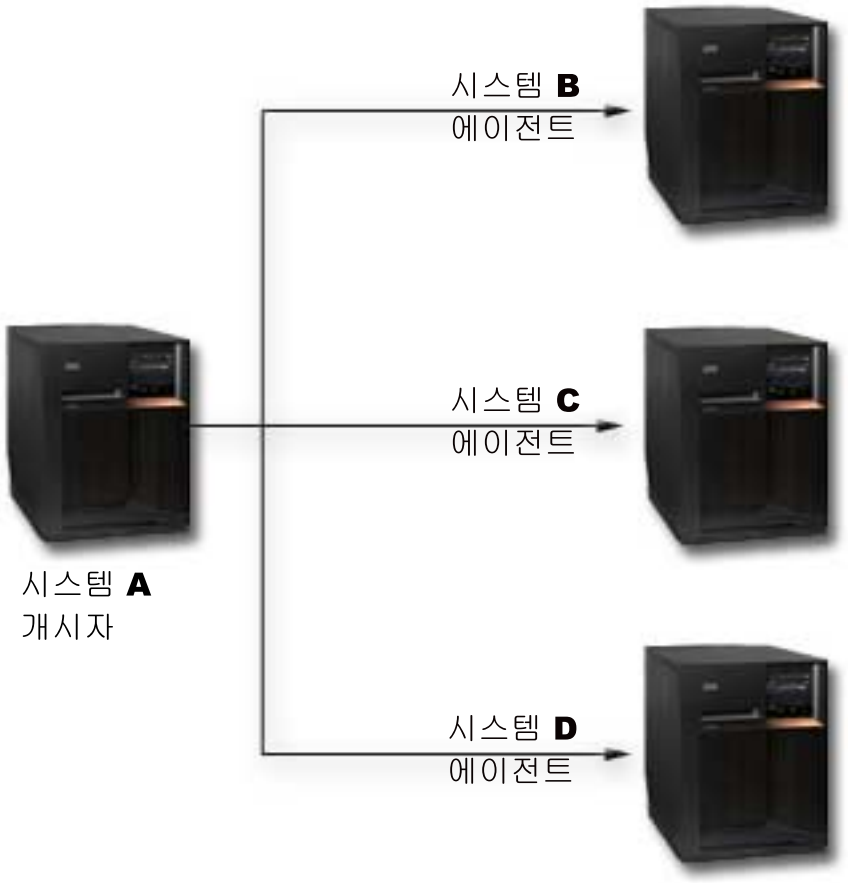
API
API 종료 프로그램

다음 그림에서는 트랜잭션에서의 기본 역할을 보여줍니다. 그림의 구조는 **트랜잭션 프로그램 네트워크**라고 합니다. 구조는 단일 레벨 트리일 수도 있고 다중 레벨 트리일 수도 있습니다.

2단계 확약 처리에서의 역할: 단일 레벨 트리

시스템 A의 어플리케이션이 확약 요구를 발행하면 시스템 A의 자원 관리자가 **개시자**가 됩니다. TCP/IP를 통한 DRDA 분산 작업 단위의 경우 개시자를 **조정 담당자**라고 합니다.

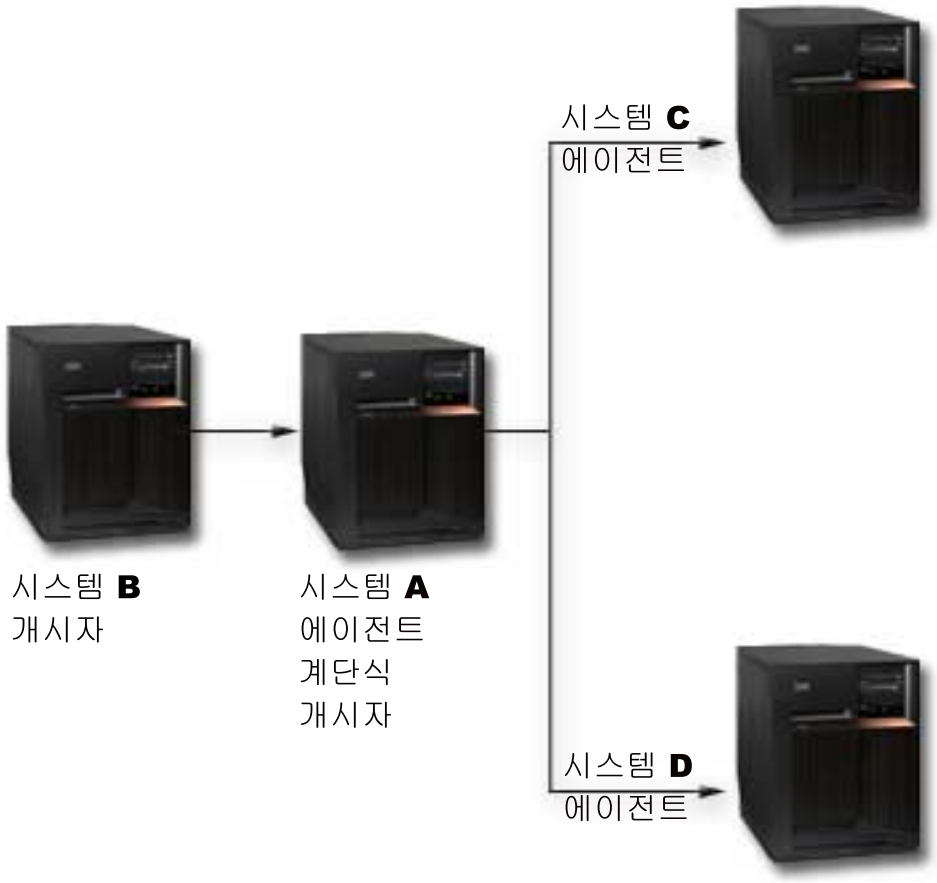
다른 세 시스템(B, C 및 D)의 자원 관리자는 이 트랜잭션에 대한 **에이전트**가 됩니다. TCP/IP를 통한 DRDA 분산 작업 단위의 경우 에이전트는 **참여자**라고도 합니다.



2단계 확약 처리에서의 역할: 다중 레벨 트리

어플리케이션이 APPC 통신을 사용하여 2단계 확약을 수행하는 경우 시스템간 관계는 하나의 트랜잭션에서 다음 트랜잭션으로 변경될 수 있습니다. 다음 그림에서는 시스템 B의 어플리케이션이 확약 요구를 발행할 때의 시스템을 보여줍니다. 이 구성은 다중 레벨 트리입니다.

다중 레벨 트랜잭션 트리가 지원되지 않으므로 이 그림에서의 역할은 TCP/IP를 통한 DRDA 분산 작업 단위에 적용되지 않습니다.



시스템 B가 시스템 C 및 시스템 D와 직접 통신하지 않으므로 트랜잭션 프로그램 네트워크에는 또 따른 레벨이 있습니다. 시스템 A의 자원 관리자는 이제 에이전트 및 **직렬 개시자**의 역할을 갖습니다.

LU 6.2 2단계 트랜잭션의 성능을 향상시키기 위해 개시자는 에이전트 중 하나에 **최종 에이전트**의 역할을 지정할 수도 있습니다. 최종 에이전트는 준비 웨이브에 참여하지 않습니다. 확약된 웨이브에서 최종 에이전트는 가장 먼저 확약합니다. 최종 에이전트의 확약이 정상적으로 완료되지 않으면 개시자가 다른 에이전트에게 롤백을 지시합니다.

TCP/IP를 통한 DRDA 분산 작업 단위의 경우 조정 담당자는 **재동기 서버**의 역할을 참여자에게 지정할 수도 있습니다. 재동기 서버는 조정 담당자와의 통신 오류가 발생하거나 조정 담당자에 시스템 장애가 발생하는 경우 다른 참여자를 재동기화시키는 책임을 갖고 있습니다.

2단계 확약 제어에 대한 트랜잭션 상태

확약 정의는 트랜잭션 프로그램 네트워크의 일부인 각 위치에서 설정됩니다. 각 확약 정의의 경우 시스템은 현재 및 이전 트랜잭션의 상태를 추적합니다. 시스템은 이 상태를 사용하여 통신 또는 시스템 장애로 인해 트랜잭션이 인터럽트되는 경우 확약 또는 롤백 여부를 결정합니다. 복수의 위치가 트랜잭션에 참여하고 있는 경우 각 위치의 트랜잭션 상태를 비교하여 올바른 조치(확약 또는 롤백)를 결정합니다. 이렇게 올바른 조치를 판별하기 위해 위치 간에 통신하는 프로세스를 **재동기화**라고 합니다.

아래 표에서는 다음과 같은 내용을 보여줍니다.

- 트랜잭션 중에 발생할 수 있는 기본 상태
- 발생 가능한 추가적인 상태
- 통신 또는 시스템 장애로 인해 트랜잭션이 인터럽트되는 경우 재동기화를 필요로 하는 상태인지 여부. 가능한 값은 다음과 같습니다.

  필요하지 않음
  각 위치는 독자적으로 올바른 결정을 내릴 수 있습니다.

  필요할 수 있음
  각 위치에서 올바른 결정을 내릴 수 있지만 개시자에게 그 결정을 알려 주어야 합니다.

  필요함
  올바른 결정이 이루어지려면 각 위치의 상태를 판별해야 합니다.
- 통신 또는 시스템 장애에 의해 취해진 조치

상태명	설명	트랜잭션이 인터럽트되는 경우 재동기화	통신 또는 시스템 장애에 의해 취해진 조치
2단계 확약 처리 중의 기본 상태:			
재설정(**RST**)	확약 경계로부터 프로그램이 확약 또는 롤백 요구를 발행할 때까지.	필요하지 않음.	지연 변경은 롤백됩니다.
준비가 진행 중(**PIP**)	개시자가 준비 웨이브를 시작했습니다. 아직 모든 위치가 인가되지 않았습니다.	필요할 수 있음.	지연 변경은 롤백됩니다.
준비됨(**PRP**)	이 위치 및 트랜잭션 프로그램 네트워크에서 훨씬 아래에 있는 위치까지 모두 확약되도록 인가했습니다. 이 위치는 아직 개시자로부터 확약하라는 통지를 받지 못했습니다.	필요함.	확신할 수 없음. 재동기화 프로세스 결과에 따라 다릅니다.
확약이 진행 중(**CIP**)	모든 위치에서 확약을 인가했습니다. 개시자가 확약된 웨이브를 시작했습니다.	필요함.	지연 변경이 확약됩니다. 모든 위치가 확약되도록 재동기화가 수행됩니다. 다른 위치에 의해 발견적(heuristic) 롤백이 보고되면 오류가 보고됩니다.
확약됨(**CMT**)	모든 에이전트는 확약되고 이 노드에 응답을 리턴했습니다.	필요할 수 있음.	없음.
2단계 확약 처리 중 추가적인 상태:			

상태명	설명	트랜잭션이 인터럽트되는 경우 재동기화	통신 또는 시스템 장애에 의해 취해진 조치
최종 에이전트 지연(**LAP**)	마지막 에이전트가 선택된 경우 이 상태는 PIP 상태와 CIP 상태 사이의 개시자에서 발생합니다. 개시자는 최종 에이전트에게 확약하도록 지시했으나 아직 응답을 받지 못했습니다.	필요함	확신할 수 없음. 재동기화 프로세스 결과에 따라 다릅니다.
읽기 전용 인가(**VRO**)	이 에이전트는 지연 변경이 없음을 표시함으로써 준비 웨이브에 응답합니다. 읽기 전용 인가 상태가 허용되는 경우 이 에이전트는 확약된 웨이브에 포함되지 않습니다.	필요할 수 있음.	없음.
롤백이 필요함(**RBR**)	다음 중 하나가 발생했습니다. • 에이전트가 확약 조작 전에 롤백 요구를 발행했습니다. • 트랜잭션 장애가 발생했습니다. • QTNRBRQD API를 사용하여 트랜잭션이 롤백이 필요함 상태가 되었습니다. 확약 제어 하에서 추가적인 변경을 수행하기 위한 트랜잭션 프로그램이 허용되지 않습니다.	필요할 수 있음.	지연 변경은 롤백됩니다.
오퍼레이터 조치 또는 오류로 인해 발생하는 조건:			
강제 롤백	최종 에이전트를 제외하고 이 위치 및 트랜잭션 프로그램 네트워크에서 휠씬 아래의 위치까지 오퍼레이터가 개입하여 롤백되었습니다.	필요할 수 있음	지연 변경은 이미 롤백되었습니다.
강제 확약	최종 에이전트를 제외하고 이 위치 및 트랜잭션 프로그램 네트워크에서 휠씬 아래의 위치까지 오퍼레이터가 개입하여 확약되었습니다.	필요할 수 있음	지연 변경은 이미 확약되었습니다.
발견적(heuristic) 혼합(**HRM**)	일부 자원 관리자가 확약되고 일부는 롤백되었습니다. 오퍼레이터가 개입했거나 시스템 오류가 발생했습니다. 발견적 혼합은 확약 정의 화면에 상태로 표시되지 않습니다. 통지 메세지가 오퍼레이터에게 보내집니다.	필요할 수 있음	데이터베이스를 일관성 있는 상태로 만들려면 오퍼레이터는 모든 참여 위치에서 조작을 복원해야 합니다.

2단계 확약 제어에 대한 확약 정의

확약 제어를 시작한 후 QTNCHGCO(확약 옵션 변경) API를 사용하여 트랜잭션에 대한 확약 옵션을 변경하십시오. 환경 및 어플리케이션에 따라 확약 옵션을 변경함으로써 시스템 성능을 향상시킬 수 있습니다.

다음 링크에서는 확약 옵션 및 그 옵션을 사용해야 하는 이유에 대해 설명합니다.

- 읽기 전용 인가 허용 출력을 기다리지 않음
- 생략하려는 경우 확인 표시
- 최종 에이전트를 선택하지 않음
- 신뢰할 수 있는 인가

TCP/IP 연결을 통해 DRDA 분산 작업 단위를 사용하고 있는 경우 적용할 수 있는 유일한 옵션은 읽기 전용 인가 허용입니다.

2단계 확약에 대한 확약 정의: 읽기 전용 인가 허용: 일반적으로 트랜잭션 관리자는 확약 처리 단계 모두에 참여합니다. 확약 처리 성능을 개선하기 위해 트랜잭션에서 일부 또는 모든 위치를 설정하여 트랜잭션 관리자가 읽기 전용으로 인가하도록 할 수 있습니다. 해당 위치에 트랜잭션 동안 확약 가능한 변경이 없다면 트랜잭션 관리자는 준비 웨이브 동안 읽기 전용으로 인가합니다. 위치는 확약된 웨이브에 참여하지 않습니다. 일반적으로 확약된 웨이브 중에 발생하는 통신 흐름은 하나 이상의 리모트 위치에서 갱신이 발생하지 않는 트랜잭션 동안에 감소되므로 전반적인 성능이 향상됩니다.

확약 제어를 시작한 후에 QTNCHGCO(확약 변경 옵션) API를 사용하여 읽기 전용 인가 허용 옵션을 Y로 변경하십시오. 다음 내용이 참이면 이를 수행하고자 할것입니다.

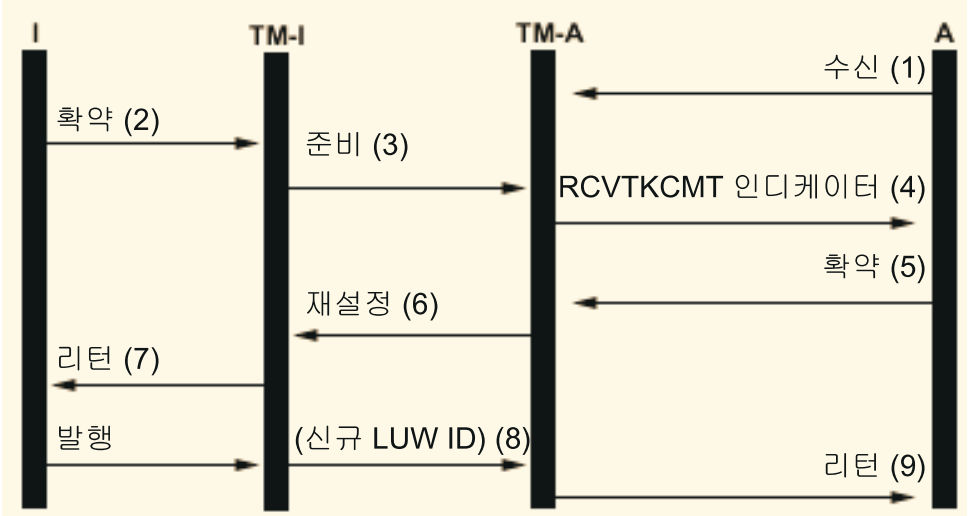
- 하나 또는 그 이상의 리모트 시스템에는 종종 트랜잭션에 대한 확약 가능한 변경이 없습니다.
- 이전 트랜잭션에 의해 파일 커서(다음 레코드)가 설정된 위치에 트랜잭션이 종속되지 않습니다. 위치가 읽기 전용을 인가하면 어플리케이션은 트랜잭션이 롤백되었는지 여부를 통지받지 못합니다. 이 위치는 데이터베이스 파일에 대한 읽기 조작을 확약하고 커서 위치를 이동했습니다. 파일 커서의 위치는 순차 처리를 하는 경우에만 중요합니다.

확약 정의가 설정되어 읽기 전용 인가가 허용되는 경우 어플리케이션은 다른 위치로부터의 다음 메세지 흐름을 기다립니다.

읽기 전용 인가 허용 옵션은 클라이언트/서버의 성격을 지닌 어플리케이션을 위한 것입니다. 프로그램 A의 목적이 프로그램 I로부터의 요구를 충족시키는 것 뿐이고 다른 독립적인 작업은 수행하지 않는 경우 프로그램 A에게 읽기 전용 인가 옵션을 허용하는 것이 적절합니다.

에이전트가 읽기 전용을 인가할 때 최종 에이전트 최적화를 하지 않는 확약 처리 흐름

다음 그림에서는 에이전트가 읽기 전용을 인가할 때 최종 에이전트 최적화를 하지 않고 어플리케이션 프로그램이 확약 명령을 발행하는 경우 어플리케이션 프로그램과 트랜잭션 관리자 사이의 메세지 흐름을 보여줍니다. 개시자 어플리케이션 프로그램 및 에이전트 어플리케이션 프로그램은 모두 2단계 확약 처리를 인식하지 못합니다. 그림에서 괄호 안의 숫자는 뒤에 나오는 설명의 번호 항목에 해당합니다.



범례

I = 개시자 (확약 요청을 초기화하는 어플리케이션)
TM-I = 개시자용 트랜잭션 관리자
A = 에이전트 (확약 요청을 수신하는 어플리케이션)
TM-A = 에이전트용 트랜잭션 관리자

다음은 에이전트가 읽기 전용을 인가할 때 최종 에이전트 최적화를 하지 않는 일반적인 처리에 대한 이벤트의 설명입니다. 이것은 기본적인 흐름을 설명합니다. 트랜잭션 프로그램 네트워크에 여러 레벨이 있거나 오류가 발생한 경우 이벤트 순서는 더 복잡해질 수 있습니다.

1. 어플리케이션 프로그램 A는 수신 요구를 함으로써 프로그램 I로부터 요구를 수신할 준비가 되었음을 나타냅니다.
2. 개시자 어플리케이션(I)은 확약 명령을 발행합니다.
3. 이 개시자에 대한 트랜잭션 관리자(TM-I)가 이 트랜잭션에 대한 개시자 역할을 맡습니다. 즉 준비 메세지를 트랜잭션에 참여하고 있는 다른 모든 위치로 송신함으로써 준비 웨이브를 시작합니다.
4. 다른 모든 위치에 대한 트랜잭션 관리자는 에이전트(TM-A)의 역할을 맡습니다. 어플리케이션 프로그램 A는 확약 요구가 수신된 TM-A에 의해 통지를 받습니다. ICF 파일의 경우에는 RCVTKCMT(확약 조치 수신) ICF 인디케이터가 켜집니다.
5. 어플리케이션 프로그램 A는 확약 지침(또는 롤백 지침)을 발행하여 응답합니다. 이것은 어플리케이션 프로그램의 인가입니다.
6. 어플리케이션 프로그램 A에서 확약 변경 옵션(QTNCHGCO)을 사용하여 읽기 전용 인가가 허용된 확약 옵션을 Y로 설정하고 트랜잭션 중에 에이전트에 변경된 내용이 없는 경우, 에이전트(TM-A)는 재설정 메세지를 사용하여 개시자(TM-I)에 응답합니다. 에이전트에 대한 확약된 웨이브는 없습니다.
7. 트랜잭션이 에이전트 TM-A에서 완료되었다는 리턴이 어플리케이션 프로그램 A로 송신됩니다.
8. 다음번 개시자(TM-I)가 자료 흐름 또는 확약 명령 중 어느 하나를 에이전트(TM-A)로 발행할 때 TM-I는 현재 트랜잭션 ID가 해당 메세지와 함께 송신되도록 합니다. 그 이유는 확약 조작 중에 TM-I와 다른 시스템간에 통신 장애가 발생했을 경우 새로운 트랜잭션 ID가 TM-I에서 생성되었기 때문입니다.

9. 트랜잭션이 에이전트 TM-A에서 완료되었다는 리턴이 어플리케이션 프로그램 A로 송신됩니다. 어플리케이션 A에 의해 다음 트랜잭션이 시작되기 전에 TM-I로부터 새로운 트랜잭션 ID를 수신해야 하므로 다음 메세지가 수신된 후까지 리턴이 지연됩니다.

2단계 확약에 대한 자세한 정보는 확약 처리에서의 역할 및 2단계 확약 제어에 대한 트랜잭션 상태를 참조하십시오.

2단계 확약에 대한 확약 정의: 출력을 기다리지 않음: 재동기화가 필요할 정도의 통신 또는 시스템 장애가 확약 조작 중에 발생한 경우 디폴트는 확약 조작이 완료되기 전에 재동기화가 완료될 때까지 기다리는 것입니다.

주: TCP/IP 연결을 통해 DRDA 분산 작업 단위를 사용하고 있는 경우는 출력을 기다리지 않음 옵션이 적용되지 않습니다. TCP/IP 연결을 통한 DRDA 분산 작업 단위에서는 출력을 기다리지 않습니다.

다음과 같은 조건이 참인 경우 이 작동을 변경할 것을 고려해 보아야 합니다.

- 참여하고 있는 어플리케이션이 서로 독립적입니다.
- 데이터베이스 파일이 동기화되어 있는지 확인하기 위해 프로그램 논리에서 이전 트랜잭션 결과를 필요로 하지 않습니다.

확약 제어를 시작한 후에 QTNCHGCO(확약 옵션 변경) API를 사용하여 재동기화 출력을 기다리지 않는 확약 정의를 지정할 수 있습니다. 출력 대기 옵션에 N(아니오)을 지정하면 시스템은 데이터베이스 서버 작업(QDBSRVnn)을 사용하여 재동기화를 비동기식으로 처리합니다.

주: 이러한 데이터베이스 서버 작업은 IPL 프로세스 중에 시작됩니다. 확약 제어 옵션을 변경하는 경우 시스템이 시작하는 작업 수에는 영향을 미치지 않습니다.

이 주제에서는 출력 대기 옵션의 두 가지 값인 Y(예)와 N(아니오)만을 언급합니다. 실제로는 지정할 수 있는 값이 두 가지가 더 있습니다. 즉 L(예 또는 개시자로부터 상속)과 U(아니오 또는 개시자로부터 상속)입니다. 이 값을 사용하면 각 확약 조작 중에 사용된 실제 값이 시스템에 의해 예 또는 아니오로 해석됩니다. QTNCHGCO(확약 옵션 변경) API에서 이 값에 대한 자세한 내용을 참조하십시오.

주: 개시자와 에이전트가 모두 추정 중단을 지원하는 경우 개시자 값을 에이전트가 상속하는 것만 가능합니다.

출력 대기(WFO) 옵션은 정상적이고 오류 없는 확약 처리에는 영향을 미치지 않습니다. 오류가 발생하면 WFO 옵션은 다음과 같은 조건을 사용하여 어플리케이션이 재동기화를 기다릴 것인지 여부를 판별합니다.

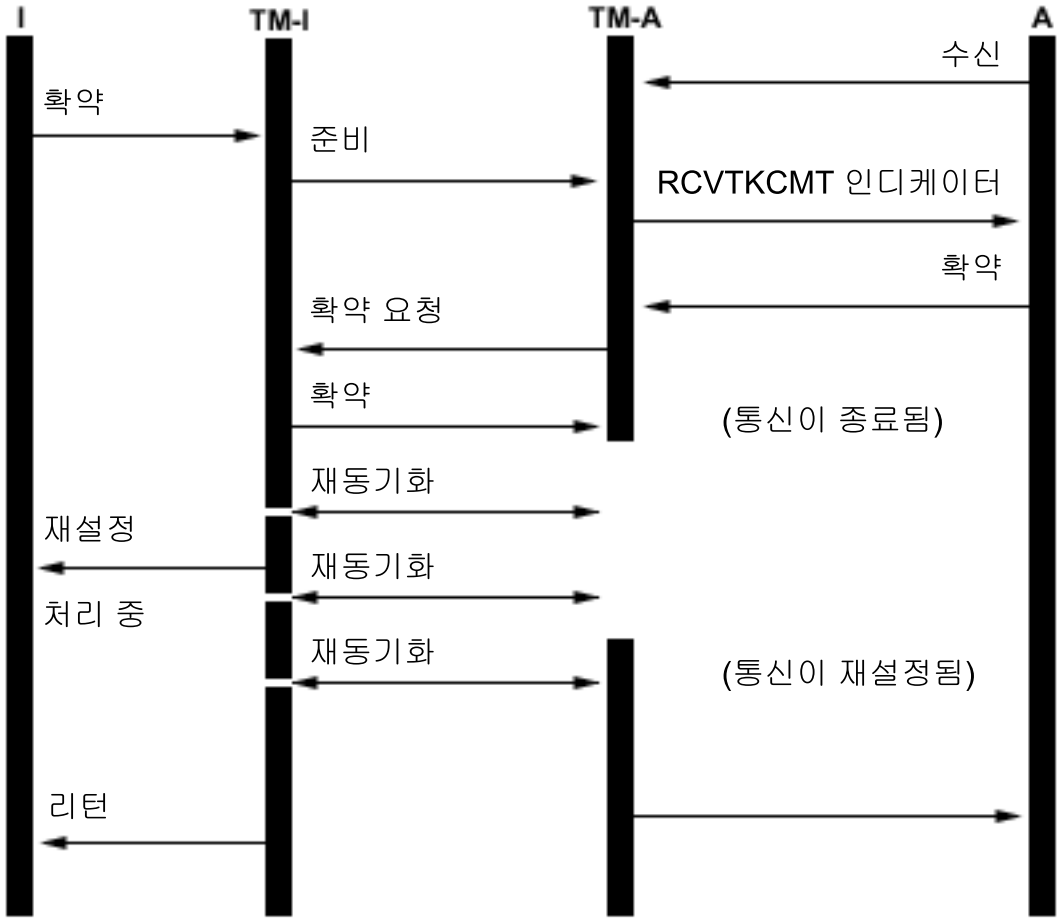
- 분석된 WFO 옵션이 Y(예)이면 어플리케이션은 재동기화 결과를 기다립니다.
- 분석된 WFO 옵션이 N(아니오)이면 추정 중단 프로토콜을 지원하는 위치의 롤백이나 준비 웨이브 중에 장애가 발생하여 재동기화는 수행되지 않고 확약 정의는 롤백됩니다.
- 확약 정의가 확실하지 않다면(트랜잭션 상태가 준비 또는 최종 에이전트 지연) 어플리케이션은 분석된 WFO 값에 관계없이 재동기화 결과를 기다립니다. 확실하지 않은 확약 정의에 대한 자세한 내용은 2단계 확약 제어에 대한 트랜잭션 상태를 참조하십시오.

- 분석된 WFO 옵션이 N이고 조건 2나 3 중 어느 것도 참이 아닌 경우 시스템은 재동기화를 한 번 시도합니다. 성공하지 못한 경우 시스템은 STATUS 메세지 CPF83E6을 어플리케이션으로 신호하여 재동기화가 진행 중임을 알립니다.

  CPF83E6이 STATUS 메세지이므로 어플리케이션이 모니터하는 경우에만 영향을 미칩니다. 일반적으로 어플리케이션에서는 이 메세지를 정보용 메세지로 취급합니다. 트랜잭션에 참여하고 있는 시스템은 장애가 보수될 때까지 트랜잭션을 재동기화하려고 시도합니다. 이러한 후속적인 재동기화 시도는 데이터베이스 서버 작업에서 수행됩니다. 데이터베이스 서버 작업에서 수행되는 후속적인 재동기화 시도가 실패하면 메세지 CPI83D0가 QSYSOPR로 송신됩니다.

출력 대기-예

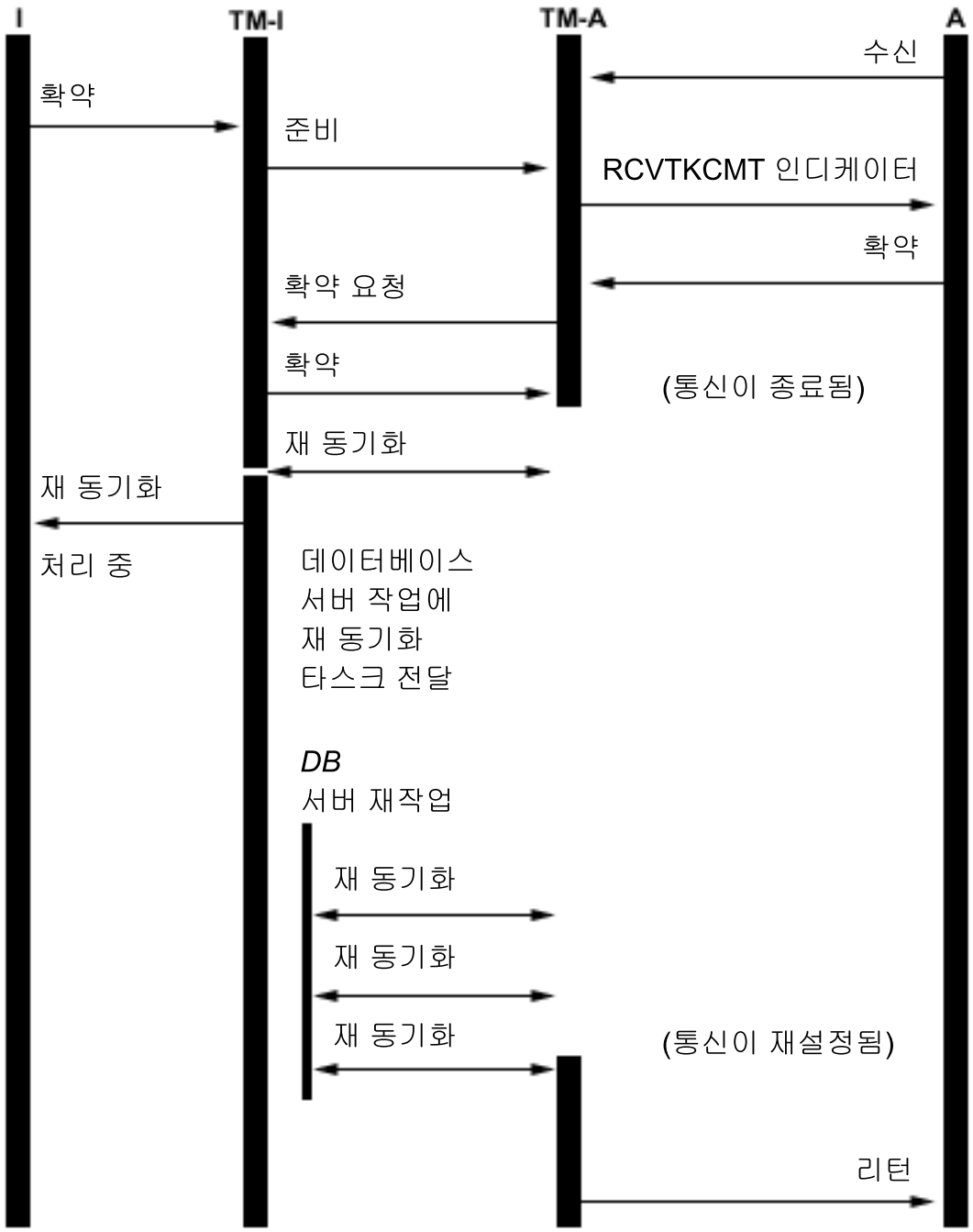
다음 그림에서 개시자(I)에 대한 확약 정의가 출력 대기 옵션에 디폴트 값 Y(예)를 사용합니다. TM-I와 TM-A 사이의 통신이 유실되면 어플리케이션 A와 어플리케이션 I는 모두 트랜잭션이 재동기화될 때까지 기다립니다.



출력 대기-아니오

다음 그림에서 개시자에 대한 확약 정의에서 분석된 WFO가 N(아니오)으로 설정되어 있습니다. 앞의 목록에서 TM-A가 조건 3을 충족시키고 TM-I는 조건 4를 충족시킵니다. TM-A와의 재동기화를 한 번 시도한 후 제어는 어플리케이션 I로 돌아옵니다. 데이터베이스 서버 작업은 재동기화를 시도합니다. 어플리케이션 I는 확약 요구가 성공적으로 완료되면 리턴 인디케이터를 수신하지 못합니다. 통신이 다시 설정될 때까지 제어도 에이전트 어플리케이션(A)으로 리턴되지 않습니다. 이것은 장애 시점에 따라 다릅니다. 이 경우 통신 장애는 확약 메세지가 개시자로부터 수신되기 전에 발생하고 TM-A를 확약할 것인지 롤백할 것인지 확실하지 않은 상태로 남게 됩니다. 트랜잭션 관리자가 확실하지 않은 경우 그 시스템에서 분석된 WFO 값에 상관없이 재동기화가 완료될 때까지 제어를 가지고 있습니다.

재동기화가 완료되기 전에 모든 시스템의 어플리케이션이 다시 계속되도록 하려면 모든 시스템에서 분석된 WFO 옵션을 N(아니오)으로 변경하거나 개시자를 N으로 바꾸고 나머지 시스템을 U(아니오 또는 개시자로부터 상속)로 설정하십시오. 그러나 트랜잭션 관리자가 확약을 할 것인지 롤백을 할 것인지에 대해 확실치 않은 경우 분석된 WFO 옵션은 무시되고 항상 제어를 리턴하기 전에 재동기화가 완료될 때까지 기다립니다.



리모트 관계형 데이터베이스에 연결되고 보호된 대화가 이미 시작된 경우 시스템은 출력 대기 값을 N으로 내재적으로 변경합니다. 이러한 이유는 출력 대기 값이 N이면 확약 조작의 성능이 향상되고 리모트 시스템에서 추정 중단을 지원하기 때문입니다. 출력 대기 값에 대한 내재적 변경은 DRDA 및 DDM 어플리케이션에 대해서만 수행됩니다. APPC 어플리케이션은 변경하기 위해 QTNCHGCO API를 호출하지 않는 한 디폴트 출력 대기 값 Y를 사용합니다.

2단계 확약에 대한 확약 정의: 생략 확인 표시: 일반적으로 트랜잭션 프로그램 네트워크의 모든 위치에 있는 트랜잭션 관리자는 모든 확약 또는 롤백 조작에 참여합니다. 성능 향상을 위해 트랜잭션에서 위치의 일부 또는 모두에 대하여 트랜잭션 관리자가 생략 확인을 표시하도록 설정할 수 있습니다.

주: 생략 확인 표시 옵션은 TCP/IP 연결을 통해 DRDA 분산 작업 단위를 사용하는 경우에는 적용되지 않습니다.

통신 흐름이 트랜잭션 중에 해당 위치로 송신되지 않으면 그 위치는 확약 또는 롤백 조작이 수행될 때 생략됩니다. 이것은 일반적으로 확약 또는 롤백 중에 발생하는 통신 흐름이 하나 이상의 리모트 위치로 아무런 자료가 송신되지 않는 트랜잭션 동안 감소되므로 전반적인 성능이 향상됩니다.

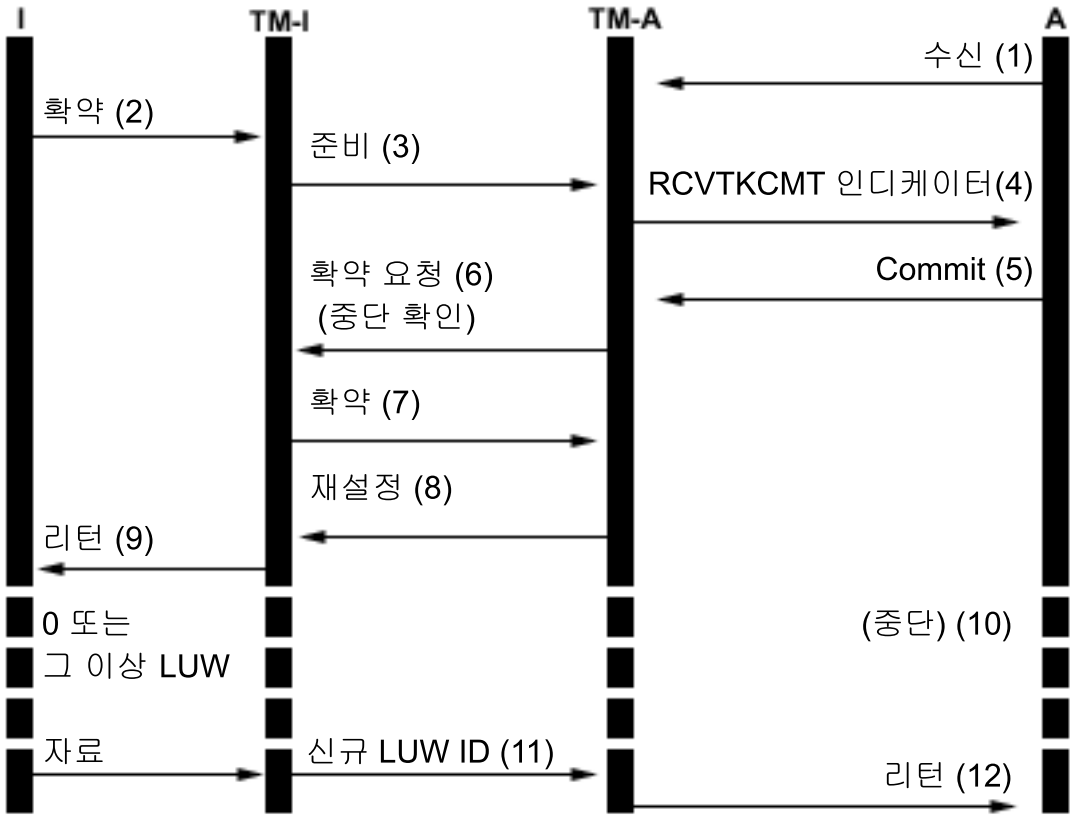
확약 제어를 시작한 후에 QTNCHGO(확약 옵션 변경) API를 사용하여 생략 확인 옵션을 Y(예)로 변경할 수 있습니다. 하나 이상의 시스템이 종종 트랜잭션에 참여하지 않는다면 이를 수행해 볼 수 있습니다.

생략 확인을 표시하도록 확약 정의를 설정하면 어플리케이션은 다른 위치로부터의 다음 메세지 흐름을 기다립니다.

생략 확인 옵션은 클라이언트/서버 성격을 띠는 어플리케이션을 위한 것입니다. 프로그램 A의 목적이 프로그램 I로부터의 요구를 충족시키는 것 뿐이고 다른 독립적인 작업은 수행하지 않는 경우 프로그램 A에게 생략 확인 옵션을 허용하는 것이 적절합니다.

에이전트가 생략 확인을 인가할 때 마지막 에이전트 최적화를 수행하지 않는 확약 처리 흐름

다음 그림에서는 어플리케이션 프로그램에서 에이전트가 생략 확인을 표시할 때 최종 에이전트 최적화를 수행하지 않는 확약 지시를 발행할 때 어플리케이션 프로그램과 트랜잭션 관리자 간의 메세지의 흐름을 보여줍니다. 개시자 어플리케이션 프로그램 및 에이전트 어플리케이션 프로그램은 모두 2단계 확약 처리를 인식하지 못합니다. 그림에서 괄호 안의 숫자는 뒤에 나오는 설명의 번호 항목에 해당합니다.



범례

I = 개시자 (확약 요청을 초기화하는 어플리케이션)

TM-I = 개시자용 트랜잭션 관리자

A

TM-A = 에이전트용 트랜잭션 관리자

다음은 에이전트가 생략 확인을 인가할 때 최종 에이전트 최적화를 수행하지 않는 일반적인 처리에 대한 이벤트의 설명입니다. 이것은 기본적인 흐름을 설명합니다. 트랜잭션 프로그램 네트워크에 여러 레벨이 있거나 오류가 발생한 경우 이벤트 순서는 더 복잡해질 수 있습니다.

1. 어플리케이션 프로그램 A는 수신 요구를 함으로써 프로그램 I로부터 요구를 수신할 준비가 되었음을 나타냅니다.
2. 개시자 어플리케이션(I)은 확약 명령을 발행합니다.
3. 이 개시자에 대한 트랜잭션 관리자(TM-I)가 이 트랜잭션에 대한 개시자 역할을 맡습니다. 즉 준비 메세지를 트랜잭션에 참여하고 있는 다른 모든 위치로 송신함으로써 준비 웨이브를 시작합니다.
4. 다른 모든 위치에 대한 트랜잭션 관리자는 에이전트(TM-A)의 역할을 맡습니다. 어플리케이션 프로그램 A는 확약 요구가 수신된 TM-A에 의해 통지를 받습니다. ICF 파일의 경우에는 RCVTKCMT(확약 조치 수신) ICF 인디케이터가 켜집니다.
5. 어플리케이션 프로그램 A는 확약 지침(또는 롤백 지침)을 발행하여 응답합니다. 이것은 어플리케이션 프로그램의 인가입니다.

6. 어플리케이션 프로그램 A에서 확약 변경 옵션(QTNCHGCO)을 사용하여 확약 생략 확인 옵션을 Y로 설정하면 에이전트(TM-A)는 요구 확약 메세지를 사용하여 개시자(TM-I)에 응답할 때 인디케이터가 송신됩니다.

   주: 확약 생략 확인 옵션에 대한 변경은 정상적으로 완료된 다음 확약 조작 때까지 유효하지 않습니다.
7. 개시자(TM-I)가 모든 인가를 수신하면 TM-I는 확약 메세지를 송신합니다. 이것은 확약된 웨이브를 시작합니다.
8. 각 에이전트(TM-A)는 확약하고 재설정 메세지로 응답합니다.
9. 트랜잭션이 개시자에서 완료되었다는 것을 나타내는 리턴이 어플리케이션 프로그램(I)으로 송신됩니다.
10. TM-I에서 발생할 수 있는 트랜잭션의 수에는 제한이 없지만 TM-A 또는 TM-A의 자료에 대한 변경이 필요하지는 않습니다. TM-A는 이 트랜잭션에 포함되지 않습니다.
11. 다음 번에 개시자(TM-I)가 에이전트(A)로 메세지를 발행할 때 새로운 트랜잭션 ID가 메세지와 함께 송신됩니다. 개시자가 에이전트로 메세지를 보내기 전에 확약 또는 롤백 조작을 수행하는 경우 이러한 조작 중에는 어떠한 메세지도 에이전트로 송신되지 않습니다(이 에이전트는 확약이나 롤백에서 '생략'되었습니다). 에이전트가 생략된 동안 하나 이상의 트랜잭션이 개시자에서 확약되었거나 롤백되었을 것이므로 개시자는 다음 메세지가 에이전트로 보내지면 현재의 트랜잭션 ID를 전달해야 합니다.
12. 원래의 확약이 완료되었으며 현재 트랜잭션에 참여하고 있음을 알리는 리턴이 어플리케이션 프로그램(A)로 송신됩니다.

2단계 확약에 대한 확약 정의: 최종 에이전트를 선택하지 않음: 디폴트로 개시자에 대한 트랜잭션 관리자는 확약 조작 중에 최종 에이전트로 어떤 에이전트든 선택할 수 있습니다.

주: TCP/IP 연결을 통해 DRDA 분산 작업 단위를 사용하고 있는 경우는 최종 에이전트를 선택하지 않음 옵션이 적용되지 않습니다.

다중 레벨 트리의 경우 개시자에 의해 최종 에이전트로 선택된 에이전트 역시 자신의 최종 에이전트를 선택할 수 있습니다. 개시자와 최종 에이전트 간의 두 개의 통신 흐름이 없어지므로(이 시스템들에 대한 준비 단계가 없어짐) 최종 에이전트를 확약 조작 중에 선택하면 성능이 개선됩니다.

그러나 일단 개시자가 최종 에이전트에 확약 요구를 송신하면 최종 에이전트의 인가를 받을 때까지 기다려야 합니다. 이것은 확약 정의의 출력대기 값과 상관 없습니다. 정상적이고 오류 없는 확약 처리 중에는 이것은 문제가 되지 않습니다. 그러나 이 창에 있을 동안 오류가 발생하면 개시자는 재동기화가 완료될 때까지 작동할 수 없습니다. 개시자 어플리케이션이 단말기에서 사용자로부터의 요구를 처리하고 있다면 이 문제는 유용성과 관련하여 고려해 볼 수 있습니다.

이러한 오류가 발생할 때 정상적인 확약 조작 중의 성능이 유용성에 미치는 영향보다 더 중요한지 여부는 사용자가 판단해야 합니다. 요구 확약이 최종 에이전트로 송신되기 전에 오류가 발생한 경우 LUW가 즉시 롤백되고 개시자는 기다리지 않습니다. 따라서 오류로 인해 개시자가 기다리게 될 때 그 창은 매우 작으며 그러한 오류는 드물게 발생해야 하는 것입니다.

유용성이 성능 개선보다 덜 중요하다고 판단된다면 확약 정의를 변경하여 최종 에이전트를 선택하지 않도록 하십시오. 확약 제어를 시작한 후에는 QTNCHGCO(확약 옵션 변경) API를 사용하여 허용되는 최종 에이전트 옵션을 N으로 변경하십시오.

확약 처리 흐름에 영향을 미치는 신뢰할 수 있는 인가: 신뢰할 수 있는 인가는 확약 조작 후에 개시자 어플리케이션으로 일찍 리턴하고 확약 조작 중에 하나의 메세지를 제거함으로써 성능을 향상시키는 최적화 조치입니다. TCP/IP를 통한 DRDA 분산 작업 단위에 대해서는 명시적인 신뢰할 수 있는 인가 최적화 조치가 없습니다. 그러나 OS/400은 TCP/IP 연결에 대한 확인 재설정(잊기)을 요구하지 않습니다. 따라서 재설정(잊기)은 항상 TCP/IP 연결을 전제합니다.

확약 제어를 시작한 후 QTNCHGCO(확약 변경 옵션) API를 사용하여 신뢰할 수 있는 인가 승인 옵션을 Y로 변경하십시오.

신뢰할 수 있는 인가는 에이전트가 확실하지 않을 때 통신 오류가 발생하면 발견적(heuristic) 결정을 해당 에이전트에서 내리지 않을 것이라는 개시자에 대한 에이전트의 약속으로 생각할 수 있습니다. 신뢰할 수 있는 인가 최적화를 사용하는 에이전트는 확약의 준비 웨이브 동안 인디케이터를 개시자에게 보냅니다. 개시자 역시 신뢰할 수 있는 인가 최적화를 사용하고 있는 경우 개시자는 확약 메세지에 대한 응답으로 재설정이 필요하지 않다는 인디케이터를 에이전트에 보냅니다. 이것은 재설정 메세지를 제거하고 따라서 확약 메세지가 보내지자마자 개시자에서 어플리케이션으로 트랜잭션 메니저가 리턴할 수 있습니다.

다음과 같은 조건이 참인 경우 신뢰할 수 있는 인가 최적화 사용을 고려해야 합니다.

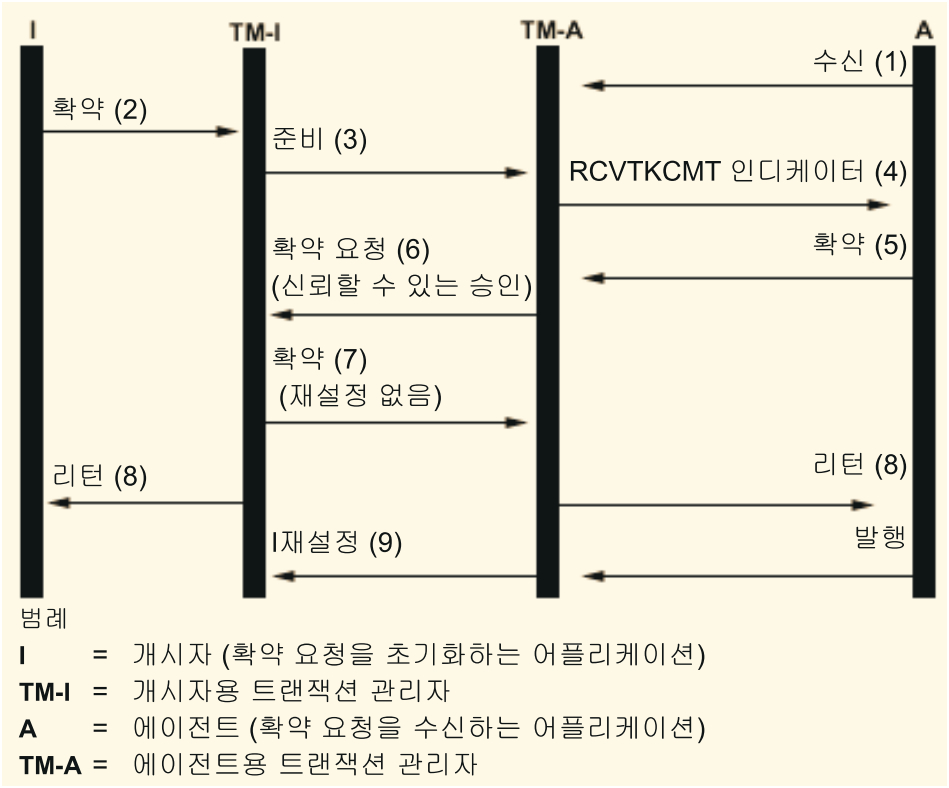
- 장애를 복구할 수 없는 경우가 아니라면 시스템 또는 통신 장애가 발생한 경우 확실하지 않은 에이전트에서 발견적(heuristic) 결정이 내려질 가능성이 없습니다.
- 프로그램 논리에서 데이터베이스 파일이 동기화 되어 있다고 확신하기 위해 이전 트랜잭션 결과가 필요하지 않습니다.

다음과 같은 조건이 모두 참이면 신뢰할 수 있는 인가 최적화가 OS/400에 의해 사용됩니다.

- 개시자 및 에이전트 위치에서 확약 제어의 추정 중단 레벨을 지원합니다.
- 개시자 위치에서 에이전트로부터의 신뢰할 수 있는 인가 표시를 승인합니다. OS/400 개시자에서 이것은 두 가지 확약 옵션 값에 따라 달라집니다.
  - 출력 대기 확약 옵션 값은 아니오여야 합니다(예가 디폴트).
  - 신뢰할 수 있는 인가 확약 승인 옵션은 예여야 합니다(예가 디폴트).
- 에이전트 위치는 준비 웨이브 중에 신뢰할 수 있는 인가를 수행합니다. OS/400 에이전트는 항상 신뢰할 수 있는 인가를 수행합니다. 이는 발견적(heuristic) 결정을 내리는 것에 대한 부정적인 부작용의 가능성을 경고하는 수동 프로시듀어를 통해서만 발견적(heuristic) 결정을 내릴 수 있기 때문입니다.

신뢰할 수 있는 인가 최적화를 사용하는 확약 처리 흐름

다음 그림에서는 신뢰할 수 있는 인가 최적화가 사용될 때 어플리케이션 프로그램과 트랜잭션 관리자 사이의 메세지 흐름을 보여줍니다. 개시자 어플리케이션 프로그램 및 에이전트 어플리케이션 프로그램은 모두 2단계 확약 처리를 인식하지 못합니다. 그림에서 괄호 안의 번호는 그 다음에 나오는 설명의 번호 항목에 해당합니다.



다음은 에이전트가 신뢰할 수 있는 인가를 수행할 때 최종 에이전트 최적화를 하지 않는 일반적인 처리에 대한 이벤트의 설명입니다. 이것은 기본적인 흐름을 설명합니다. 트랜잭션 프로그램 네트워크에 여러 레벨이 있거나 오류가 발생한 경우 이벤트 순서는 더 복잡해질 수 있습니다.

1. 어플리케이션 프로그램 A는 수신 요구를 함으로써 프로그램 I로부터 요구를 수신할 준비가 되었음을 나타냅니다.
2. 개시자 어플리케이션(I)은 확약 명령을 발행합니다.
3. 이 개시자에 대한 트랜잭션 관리자(TM-I)가 이 트랜잭션에 대한 개시자 역할을 맡습니다. 즉 준비 메세지를 트랜잭션에 참여하고 있는 다른 모든 위치로 송신함으로써 준비 웨이브를 시작합니다.
4. 다른 모든 위치에 대한 트랜잭션 관리자는 에이전트(TM-A)의 역할을 맡습니다. 어플리케이션 프로그램 A는 확약 요구가 수신된 TM-A에 의해 통지를 받습니다. ICF 파일의 경우에는 RCVTKCMT(확약 조치 수신) ICF 인디케이터가 켜집니다.
5. 어플리케이션 프로그램 A는 확약 지침(또는 롤백 지침)을 발행하여 응답합니다. 이것은 어플리케이션 프로그램의 인가입니다.

6. 에이전트(TM-A)는 요구 확약 메세지로 개시자(TM-I)에 응답합니다. OS/400 시스템은 요구 확약과 함께 신뢰할 수 있는 인가 요구 인디케이터를 송신합니다.
7. 개시자(TM-I)가 모든 인가를 수신하면 TM-I는 확약 메세지를 송신합니다. 출력 대기 확약 옵션이 N(아니오)이고 신뢰할 수 있는 인가 승인 확약 옵션이 Y(예)이면 재설정 인디케이터가 확약 메세지와 함께 송신되지 않습니다. 이는 에이전트에게 확약에 대한 응답으로 재설정 메세지가 필요하지 않다고 알려 줍니다.
8. 트랜잭션이 완료됩니다. 리턴은 어플리케이션 프로그램(I와 A)으로 보내집니다. 이 리턴은 확약 조작이 성공적이었음을 나타냅니다. 확약된 메세지를 수신하기 전에 수행된 발견적(heuristic) 결정으로 인해 시스템 A에 발견적(heuristic) 손상이 발생하는 경우 어플리케이션 I는 통지를 받지 못합니다. 대신 메세지가 QSYSOPR 메세지 대기행렬로 들어갑니다. 그러나 어플리케이션 A는 발견적(heuristic) 손상 표시를 **수신합니다**.
9. 다음 번 에이전트(TM-A)가 자료 흐름이든 확약 명령어이든 개시자(TM-I)로 메세지를 보낼 때 내재된 재설정 인디케이터가 전송되어 TM-A가 확약을 성공적으로 완료했음을 TM-I에 알립니다. 그 이유는 TM-A가 7단계에서 정상적으로 확약 메세지를 수신했음을 확인할 때까지 TM-I가 완료된 트랜잭션에 대한 정보를 가지고 있어야 하기 때문입니다.

확약 제어에 대한 XA 트랜잭션 지원

iSeries용 DB2 UDB는 X/Open 글로벌 트랜잭션에 참여할 수 있습니다. Open Group에서는 관련없는 자원에 대한 변경사항이 단일 글로벌 트랜잭션의 일부가 되도록 허용하는 산업 표준 모델을 정의했습니다. 이것의 예가 별개의 업체에서 제공하는 데이터베이스에 대한 변경입니다. 이 모델을 X/Open 분산 트랜잭션 처리 모델이라고 합니다. 다음에서 X/Open 분산 트랜잭션 처리 모델에 대해 자세히 설명됩니다.

- X/Open Guide, February 1996, Distributed Transaction Processing: Reference Model, Version 3 (ISBN:1-85912-170-5, G504), The Open Group.
- X/Open CAE Specification, December 1991, Distributed Transaction Processing: The XA Specification (ISBN:1-872630-24-3, C193 or XO/CAE/91/300), The Open Group.
- X/Open CAE Specification, April 1995, Distributed Transaction Processing: The TX (Transaction Demarcation) Specification (ISBN:1-85912-094-6, C504), The Open Group.

iSeries용 DB2 UDB에서 제공되는 XA 트랜잭션 지원을 사용하려면 먼저 이 책들의 내용 특히 XA Specification을 잘 알고 있어야 합니다. 이 책들은 Open Group web site에서 찾을 수 있습니다.

DTP 모델에는 다섯 가지 구성요소가 있습니다.

어플리케이션 프로그램(AP)
데이터베이스와 같은 자원이 관계된 일련의 조작을 지정하여 일반 사용자가 원하는 기능을 구현합니다. 글로벌 트랜잭션의 시작과 끝을 정의하고 트랜잭션 경계 안에서 자원에 액세스하며 일반적으로 각 트랜잭션을 확약할 것인지 또는 롤백할 것인지에 대한 결정을 내립니다.

트랜잭션 관리자(TM)

글로벌 트랜잭션을 관리하고 이 트랜잭션에 대한 시작 결정을 조정하며 아토믹(atomic) 트랜잭션의 완료를 위해 트랜잭션을 확약 또는 롤백합니다. TM은 또한 구성요소가 실패한 후 RM과 함께 회복 활동을 조정합니다.

자원 관리자(RM)
데이터베이스 관리 시스템과 같은 컴퓨터 공유 자원의 정의된 부분을 관리합니다. AP는 RM이 제공한 인터페이스에 대해 각 RM이 정의한 인터페이스를 사용하여 트랜잭션을 완료합니다.

통신 자원 관리자(CRM)
현재 TM 정의역 내부 또는 외부에서 모델의 인스턴스가 다른 인스턴스에 액세스할 수 있도록 합니다. CRM은 iSeries용 DB2 UDB의 범위를 벗어나는 것으로 여기에서는 논의되지 않습니다.

통신 프로토콜
서로 통신하기 위해 CRM에 의해 사용되는 프로토콜. 이것은 iSeries용 DB2 UDB의 범위를 벗어나는 것으로 여기에서는 논의되지 않습니다.

XA 스펙은 DTP 모델의 TM 및 RM 구성요소에 의해 사용되는 인터페이스 세트를 설명하는 DTP 모델의 일부입니다. iSeries용 DB2 UDB는 이 인터페이스를 UNIX 양식의 API 및 종료 프로그램 세트로 구현합니다. 이들 API 및 iSeries용 DB2 UDB를 RM으로 사용하는 방법에 대한 자세한 설명은 XA API를 참조하십시오.

iSeries Navigator 및 XA 트랜잭션

iSeries Navigator는 XA 트랜잭션 관리를 **글로벌 트랜잭션**으로 지원합니다.

글로벌 트랜잭션에는 iSeries용 DB2 UDB의 외부 및 내부에서 발생한 변경이 포함됩니다. 글로벌 트랜잭션은 Open Group XA 구조 또는 이와 유사한 다른 구조를 사용하여 외부 트랜잭션 관리자에 의해 조정됩니다. 어플리케이션은 트랜잭션 관리자에 의해 제공되는 인터페이스를 사용하여 글로벌 트랜잭션을 확약 또는 롤백합니다. 트랜잭션 관리자는 XA 구조 또는 다른 구조에 의해 정의된 확약 프로토콜을 사용하여 트랜잭션을 완료합니다. iSeries용 DB2 UDB가 글로벌 트랜잭션에 참여하는 경우 XA 자원 관리자로서 작동합니다. 글로벌 트랜잭션에는 두 종류가 있습니다.

- **트랜잭션 범위의 잠금:** 트랜잭션을 위해 획득된 잠금이 트랜잭션 범주에서 이루어집니다. 트랜잭션은 하나의 작업이나 스레드에서 다른 작업이나 스레드로 이동할 수 있습니다.
- **작업 범위의 잠금:** 트랜잭션을 위해 획득된 잠금이 작업 범주에서 이루어집니다. 트랜잭션을 시작한 작업에서 트랜잭션을 이동할 수 없습니다.

로컬 시스템에 있는 데이터베이스에 대해 XA 트랜잭션을 실행하는 경우 트랜잭션 범위의 잠금을 위한 XA API를 사용해야 합니다. 이 API는 작업 범위의 잠금을 위한 XA API에 비해 제한이 적고 다음과 같은 상황에서 더 나은 성능을 제공합니다.

- 단일 XA 트랜잭션 분기에서 작동하기 위해 복수의 SQL 연결이 사용되는 경우.
- 복수의 동시 XA 트랜잭션 분기에서 작동하기 위해 단일 SQL 연결이 사용되는 경우.

이러한 상황에서는 작업 범위의 잠금을 위한 XA API를 사용할 때 XA 트랜잭션 분기를 실행하려면 별도의 작업이 시작되어야 합니다.

리모트 시스템에 있는 데이터베이스에 대하여 실행하려면 작업 범위의 잠금을 위한 XA API를 사용해야 합니다.

XA 트랜잭션의 고려사항

iSeries용 DB2 UDB를 RM으로 사용하려면 다음과 같은 고려사항 및 제한사항을 이해해야 합니다. ″스레드″라는 용어는 스레드가 가능하지 않은 작업을 의미하거나 아니면 스레드 가능 작업 내의 단일 스레드를 의미합니다.

달리 언급되지 않는 한 다음과 같은 고려사항이 트랜잭션 범위의 잠금을 수행하는 트랜잭션 및 작업 범위의 잠금을 수행하는 트랜잭션 모두에 적용됩니다.

iSeries용 DB2 UDB 고려사항

- XA 트랜잭션은 SQL 서버 모드에서 실행되는 작업에서만 수행할 수 있습니다. 이것이 미치는 영향 중 하나는 XA 트랜잭션 중에 iSeries용 DB2 UDB를 변경할 때 어플리케이션이 SQL 인터페이스로 제한된다는 점입니다. 이미 SQL 서버 모드에서 실행되고 있지 않은 작업에서 db2xa_open() API가 사용되는 경우 SQL 서버 모드는 내재적으로 시작됩니다. 확약 제어를 위한 SQL 서버 모드 및 스레드 범위의 트랜잭션을 참조하십시오.
- XA API 호출 시 iSeries용 DB2 UDB에 의해 감지된 오류는 XA 스펙에 따라 리턴 코드를 통해 보고됩니다. 리턴 코드만으로 오류의 의미가 명확하지 않을 때는 진단 메세지가 작업 기록부에 남게 됩니다.

삽입된 SQL 고려사항

- XA 트랜잭션에 대하여 SQL(구조화 조회 언어)을 사용하려면 SQL 연결이 이루어지기 전에 db2xa_open() 어플리케이션 프로그램 인터페이스를 사용해야 합니다. 연결될 관계형 데이터베이스는 Xainfo 매개변수에 의해 db2xa_open() API로 전달되어야 합니다. 연결이 라우트되는 작업에서 사용될 사용자 프로파일 및 암호는 db2xa_open() API로 전달됩니다. 전달되지 않는 경우 프로파일에는 연결을 시도할 때 지정되었거나 디폴트로 지정된 값이 사용됩니다.
- XA 트랜잭션 수행 시 삽입된 SQL이 사용되는 경우 동일한 스래드에서 연결이 이루어진다고 해도 각 연결에 대하여 수행되는 작업은 다른 작업으로 라우트됩니다. 이것은 단일 스레드의 모든 연결에 대해 수행된 작업이 동일한 작업으로 라우트되는 XA 없는 SQL 서버 모드와는 다릅니다. 이는 XA 스펙에서는 각 자원 관리자 인스턴스에 대하여 별도의 준비, 확약 또는 롤백 호출이 필요하기 때문입니다.

  주: 다음 고려사항은 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다.

- XA 트랜잭션을 수행하는 데 삽입된 SQL이 사용되는 경우 관계형 데이터베이스 당 하나의 연결만이 각 스레드에 대해 작성됩니다. 스레드가 활발하게 트랜잭션 분기와 연관되지 않을 때마다 스레드 연결 중 하나를 통해 요구된 작업으로 인해 RM은 TM의 ax_reg() 종료 프로그램을 사용하여 작업이 트랜잭션 분기를 시작, 재개 또는 결합할지 여부를 판별합니다.

  작업이 트랜잭션 분기를 시작해야 하는 경우 이는 해당 관계형 데이터베이스에 대한 스레드 연결을 통해 수행됩니다.

  작업에서 트랜잭션 분기를 결합해야 하는 경우 이는 트랜잭션 분기를 시작한 스레드에서 이루어진 해당 관계형 데이터베이스에 대한 연결을 통해 다시 라우트됩니다. 시스템에서는 이 연결에 대한 사용자 프로파일과 결합하는 스레드의 연결에 대한 사용자 프로파일이 동일하도록 강요하지 않습니다. 이것이 보안과 관련된 문제가 아닌지 여부는 TM의 책임입니다. 일반적으로 TM은 모든 연결에 동일한 사용자 프로파일을 사용합니다. 이 사용자 프로파일에는 TM에서 관리하는 모든 자료에 대한 권한이 부여됩니다. 이 자료에 대한 보다 복잡한 액세스 보안은 표준 iSeries 보안 메카니즘을 사용하는 대신 TM이나 AP에서 관리합니다.
- 작업이 트랜잭션 분기를 재개해야 하는 경우 사용되는 연결은 일시 중단된 분기 연관이 트랜잭션 분기를 시작해서 설정된 것인지 또는 결합해서 설정된 것인지 여부에 따라 달라집니다.

  후속 작업은 해당 트랜잭션 분기와의 스레드 연관을 일시중단하거나 종료하는 데 db2xa_end() API가 사용될 때까지 동일한 연결을 통해 수행됩니다.

CLI 고려사항

- CLI가 XA 트랜잭션 수행에 사용되는 경우 db2xa_open() API가 사용된 후 동일한 스레드에 연결이 둘 이상 설정될 수 있습니다. 다른 스레드에서 동일한 Xainfo 매개변수 값을 사용하여 먼저 db2xa_open() API를 사용하는 경우 XA 트랜잭션을 수행하기 위해 그 다른 스레드에서 해당 연결을 사용할 수 있습니다.

  주: 다음 고려사항은 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다.
- CLI가 XA 트랜잭션을 수행하는 데 사용되는 경우 트랜잭션 분기를 시작하는 데 사용되는 연결은 해당 트랜잭션 분기에서 모든 작업에 대해 사용되어야 합니다. 다른 스레드가 트랜잭션 분기에 결합되는 경우 트랜잭션 분기를 시작하는 데 사용되는 연결에 대한 연결 핸들은 동일한 연결을 통해 작업을 수행할 수 있도록 결합 스레드로 전달되어야 합니다. 마찬가지로 스레드가 트랜잭션 분기를 재개해야 하는 경우 동일한 연결을 사용해야 합니다.

  주: 다음은 트랜잭션 범위의 잠금 및 작업 범위의 잠금을 수행하는 트랜잭션에 적용됩니다.

  CLI 연결 핸들은 다른 작업에서 사용할 수 없으므로 결합 기능은 CLI가 사용될 때 트랜잭션 분기를 시작한 동일한 작업에서 실행되는 스레드에 제한됩니다.

리모트 관계형 데이터베이스 고려사항

주: 리모트 관계형 데이터베이스에 대한 이 고려사항은 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다.

- 리모트 관계형 데이터베이스에 대한 XA 연결은 관계형 데이터베이스가 분산 작업 단위(DUW) DRDA 연결을 지원하는 시스템에 있는 경우에만 지원됩니다. 여기에는 SNA LU6.2 대화를 통해 DRDA를 실행하는 시스템이 포함됩니다. 또한 TCP/IP 연결을 사용하여 DRDA를 실행할 때 V5R1을 사용하는 시스템도 포함됩니다.

- XA 결합 기능을 사용하기 전에 결합 스레드에 db2xa_open() API를 사용해야 합니다. 트랜잭션 분기를 시작한 스레드와 결합 스레드 모두에서 동일한 관계형 데이터베이스명 및 RMID가 db2xa_open() API에 지정되어야 합니다. 결합이 시도될 때 트랜잭션 분기가 활동 상태이면 결합 스레드는 블록됩니다. 결합 스레드는 활동 스레드가 트랜잭션 분기와의 연관을 일시 중단하거나 종료시킬 때까지 블록 상태로 남아 있습니다.

회복 고려사항

- 모든 확약 정의에 대하여 제공되는 수동 발견적(heuristic) 확약 및 롤백 지원은 트랜잭션 분기가 준비된 상태에 있는 동안 확약 또는 롤백을 강제로 실행해야 할 필요가 있을 때 사용할 수 있습니다. 자세한 내용은 확약 및 롤백 강제 실행 시점 및 재동기화 취소 시점을 참조하십시오.

트랜잭션 분기 고려사항

- XA 트랜잭션 분기에 대한 정보는 iSeries Navigator 및 WRKJOB(작업에 대한 작업), DSPJOB(작업 표시) 및 WRKCMTDFN(확약 정의에 대한 작업) 명령을 사용하여 확약 제어 정보의 일부로 표시됩니다. TM명, 트랜잭션 분기 상태, 트랜잭션 ID 및 분기 규정자가 모두 표시됩니다. `WRKCMTDFN JOB(*ALL) STATUS(*XOPEN)` 명령을 사용하거나 iSeries Navigator에서 **글로벌 트랜잭션**을 표시함으로써 현재 활동 중인 모든 XA 트랜잭션과 관련된 확약 정의를 표시할 수 있습니다.

  주: 다음 항목은 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다.

- db2xa_end() API를 사용하여 스레드와 기존 트랜잭션 분기 간의 연관이 일시중단되거나 종료된 경우 그 스레드는 새로운 트랜잭션 분기를 시작할 수 있습니다. 새로운 트랜잭션 분기를 시작하는 데 사용된 연결이 다른 트랜잭션 분기를 시작하기 위해 이전에 사용되었던 것이고 그 트랜잭션 분기와의 연관이 db2xa_end() API에 의해 종료되었거나 일시중단되었다면 새로운 SQL 서버 작업이 시작될 수 있습니다. 첫 번째 트랜잭션 분기가 db2xa_commit() 또는 db2xa_rollback() API에 의해 아직 완료되지 않은 경우에만 새로운 SQL 서버 작업이 필요합니다. 이 경우 연결이 설정되었을 때 원래의 SQL 서버 작업이 식별되었던 것처럼 새로운 SQL 서버 작업을 식별하는 다른 완료 메세지 SQL7908이 작업 기록부로 송신됩니다. 새로운 트랜잭션 분기에 대한 모든 SQL 요구는 새로운 SQL 서버 작업으로 라우트됩니다. 트랜잭션 분기가 db2xa_commit() 또는 db2xa_rollback() API에 의해 완료되면 새로운 SQL 서버 작업이 재순환되어 사전 시작 작업 풀로 리턴됩니다.
- 다음과 같은 상황이 발생하면 트랜잭션 분기는 시스템에 의해 롤백 전용으로 표시됩니다.
  - 트랜잭션 분기와 연관되어 있을 때 스레드가 종료하는 경우.
  - 트랜잭션 분기와 사용 중인 연관을 가지고 있는 스레드에서 db2xa_close() API가 사용된 경우.
- 다음과 같은 상황 중 하나가 발생하면 트랜잭션 분기는 시스템에 의해 롤백됩니다.
  - 트랜잭션 분기와 관련된 연결이 종료한 경우.
  - 트랜잭션 분기를 시작한 작업이 종료한 경우.
  - 시스템 장애가 발생한 경우.
- 연관된 스레드가 있는지 여부와 관계없이 시스템에 의해 트랜잭션 분기가 롤백되는 상황이 하나 있습니다. 이는 연결 작업이 라우트되는 SQL 서버 작업이 종료될 때 발생합니다. 이는 해당 작업에 대하여 ENDJOB(작업 종료) CL 명령이 사용되는 경우에만 발생할 수 있습니다.

주: 다음 항목은 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다.

- 다음과 같은 상황 중 어느 하나가 발생할 때 사용 중인 연관을 가지고 있는 스레드가 없으면 트랜잭션 분기는 영향을 받지 않습니다. TM은 db2xa_open() API 및 트랜잭션 분기를 시작한 스레드에 지정된 동일한 Xainfo 매개변수 값을 사용한 스레드로부터의 트랜잭션 분기를 확약 또는 롤백할 수 있습니다.
  - 트랜잭션 분기와 관련된 연결이 종료한 경우.
  - 트랜잭션 분기에 대한 작업을 수행했으나 더 이상 그에 대한 사용 중인 연관을 가지고 있지 않은 스레드나 작업에서 db2xa_close() API를 사용하는 경우.
  - 트랜잭션 분기에 대한 작업을 수행했으나 더 이상 그에 대한 사용 중인 연관을 가지고 있지 않은 스레드나 작업에서 db2xa_close() API를 사용하는 경우.
  - 시스템 장애가 발생한 경우. 이 경우 트랜잭션 분기가 준비된 상태에 있는 경우에만 영향을 받지 않습니다. 유휴 상태인 경우 시스템은 롤백합니다.

확약 제어를 위한 SQL 서버 모드 및 스레드 범위의 트랜잭션

작업 범위의 잠금을 사용하는 확약 정의는 일반적으로 활성 그룹 범위에서 적용됩니다. 작업이 다중 스레드인 경우 작업 내 모든 스레드는 확약 정의에 액세스할 수 있으며 특정 트랜잭션에 대한 변경은 여러 스레드에 걸쳐 확장될 수 있습니다. 즉 프로그램이 동일한 활성 그룹에서 실행되는 모든 스레드는 단일 트랜잭션에 참여합니다.

트랜잭션 작업이 활성 그룹보다는 스레드 범위로 이루어지는 것이 바람직한 경우가 있습니다. 즉 각 스레드는 자신의 확약 정의를 가지고 각 확약 정의에 대한 트랜잭션 작업은 다른 스레드에서 수행되는 작업과는 독립적으로 수행됩니다.

SQL 서버 모드에서 실행되도록 작업을 변경하는 것은 QWTCHGJB(작업 변경) API를 사용하여 iSeries용 DB2 UDB에 의해 지원됩니다. SQL 연결이 SQL 서버 모드에서 요구되면 이는 별도의 작업으로 라우트됩니다. 그 연결에 대하여 수행되는 모든 후속 SQL 조작도 그 작업으로 라우트됩니다. 연결이 이루어지면 SQL 요구가 어떤 작업으로 라우트되는지를 나타내는 완료 메세지 SQL7908이 SQL 서버 모드 작업의 작업 기록부로 보내집니다. 확약 정의는 이 메세지에 나오는 작업이 소유합니다. 오류가 발생하면 SQL문을 수행한 작업에서는 실제 아무런 작업도 이루어지지 않으므로 문제의 원인을 이해하기 위해서는 두 작업 모두의 작업 기록부를 살펴 보아야 할 것입니다.

SQL 서버 모드에서 실행될 때 확약 제어 하에서 작업을 수행하기 위해서는 SQL 인터페이스만 사용될 수 있습니다. 삽입된 SQL 또는 호출 레벨 인터페이스(CLI)를 사용할 수 있습니다. 하나의 스레드에서 삽입된 SQL을 통해 설정된 모든 연결이 동일한 백엔드 작업으로 라우트됩니다. 이것을 사용하여 SQL 서버 모드에서 실행되지 않는 작업에서 실행되듯 단일 확약 요구가 모든 연결에 대한 작업을 확약하도록 할 수 있습니다. CLI를 통해 이루어지는 각 연결은 별도의 작업으로 라우트됩니다. CLI는 독립적으로 확약되거나 롤백될 각 연결에 대하여 수행되는 작업을 필요로 합니다.

SQL 서버 모드에서 실행될 때 확약 제어 하에서 다음과 같은 조작을 수행할 수 없습니다.

- SQL 인터페이스가 아닌 인터페이스를 사용한 레코드 변경

- DDM 파일 변경
- API 확약 자원 변경

SQL 서버 모드에서 실행되는 작업에서 확약 제어를 직접 시작할 수 없습니다. SQL 서버 모드에 대한 자세한 내용은 데이터베이스 주제의 다음 페이지를 참조하십시오.

- SQL 서버 모드에서 DB2 CLI를 실행하는 이유
- SQL 서버 모드에서 DB2 CLI 시작
- 서버 모드에서 DB2 CLI 실행을 위한 제한사항

확약 제어 시작

확약 제어를 시작하려면 STRCMTCTL (확약 제어 시작) 명령을 사용하십시오.

주: 확약 제어는 SQL 어플리케이션에서 시작할 필요가 없습니다. SQL은 SQL 분리 레벨이 *NONE이 아닐 때 연결 시 확약 제어를 내재적으로 시작합니다.

STRCMTCTL 명령을 사용할 때 다음을 지정할 수 있습니다.

이 매개변수들은 다음 주제에서 설명됩니다.

확약 잠금 레벨
STRCMTCTL 명령에서 LCKLVL 매개변수를 사용하여 잠금 레벨을 지정하십시오. 사용자가 지정한 레벨은 확약 정의에 대한 확약 제어 하에 열리고 위치하는 데이터베이스 파일에 대한 레코드 잠금의 디폴트 레벨이 됩니다. 자세한 정보는 확약 잠금 레벨을 참조하십시오.

통지 오브젝트 확약
NTFY 매개변수를 사용하여 통지 오브젝트를 지정하십시오. 통지 오브젝트는 특정 확약 정의가 정상적으로 종료하지 못한 경우 그 확약 정의에 대해 마지막으로 성공한 트랜잭션을 식별하는 정보가 들어 있는 메세지 대기행렬, 자료 영역 또는 데이터베이스 파일입니다. 자세한 내용은 통지 오브젝트 확약을 참조하십시오.

확약 범위 매개변수
CMTSCOPE 매개변수를 사용하여 확약 범위를 지정하십시오. 확약 제어가 시작되면 시스템은 확약 정의를 작성합니다. 확약 범위 매개변수는 확약 정의에 대한 범위를 식별합니다. 디폴트 확약 정의 범위는 확약 제어 요구를 시작하는 프로그램의 활성 그룹 범위입니다. 그에 대한 대체 범위는 작업 범위입니다.

디폴트 저널 매개변수
확약 제어를 시작할 때 디폴트 저널을 지정할 수 있습니다. 다음과 같은 이유로 디폴트 저널을 사용할 수 있습니다.

- 트랜잭션 저널 항목을 캡처하고자 합니다. 이 항목을 사용하여 어떤 자원이 트랜잭션과 연관되었는지에 관한 이력을 분석하는 데 도움이 됩니다. 저널된 변경을 적용하고 제거하는 데 사용되지 않습니다. OMTJRNE(저널 항목 생략) 매개변수는 시스템이 트랜잭션 항목을 작성하는지 여부를 판별합니다.

- 라우팅 단계 안에서 파일을 닫았다가 다시 여는 작업의 성능을 개선시키고자 합니다. 디폴트 저널이 아닌 저널에 지정된 모든 파일을 닫은 경우 저널에 대한 모든 시스템 정보가 라우팅 단계에서 제거됩니다. 그 저널에 지정된 파일이 나중에 열리는 경우 그 저널에 대한 모든 정보를 다시 작성해야 합니다. 시스템은 저널에 지정된 자원이 활동 상태인지 여부와는 상관없이 확약 정의를 사용하는 디폴트 저널에 대한 정보를 보관합니다.

텍스트 확약 매개변수
TEXT 텍스트 매개변수를 사용하여 작업에 대해 시작된 확약 정의에 대한 정보를 표시할 때 확약 정의와 연관된 특정 텍스트를 식별합니다. 텍스트가 지정되지 않으면 시스템은 디폴트 텍스트 설명을 제공합니다.

저널 항목 생략 매개변수
성능 개선을 위해 디폴트 저널을 지정하는 경우 OMTJRNE 매개변수를 사용하여 시스템이 트랜잭션 저널 항목을 작성하는 것을 방지할 수 있습니다. 시스템이 트랜잭션 항목을 작성하도록 하면 저널 리시버의 크기가 많이 증가하며 확약 및 롤백 조작 중에 성능을 저하시킵니다.

사용자의 확약 제어 환경 또는 새로운 어플리케이션을 설정하고 테스트할 때 트랜잭션 항목은 유용합니다.

트랜잭션 항목은 다음과 같은 조건 하에서 OMTJRNE 매개변수 값과 상관없이 디폴트 저널에 쓰여집니다.

- 확약 또는 롤백 조작 중에 시스템 오류가 발생합니다.
- 트랜잭션에 참여한 자원을 수동 변경하고 이 변경이 발견적(heuristic) 혼합 조건을 생성했습니다. 발견적(heuristic) 혼합 조건에 대한 설명은 2단계 확약 제어에 대한 트랜잭션 상태를 참조하십시오. 이러한 유형의 수동 변경을 발견적(heuristic) 결정이라고 합니다.

이 상황에서 어떤 조치를 판별하기 위해 트랜잭션에 어떤 자원이 참여했는지에 대한 정보를 사용할 수 있습니다.

저널 항목의 가변 길이 부분의 표 15에서 표 21까지는 트랜잭션 저널 항목에 대한 항목별 자료를 보여줍니다.

다음에서는 확약 제어 시작에 대한 자세한 정보를 제공합니다.

- 통지 오브젝트 확약
- 확약 잠금 레벨

통지 오브젝트 확약

통지 오브젝트는 특정 확약 정의가 정상적으로 종료하지 못한 경우 그 확약 정의에 대해 마지막으로 성공한 트랜잭션을 식별하는 정보가 들어 있는 메세지 대기행렬, 자료 영역 또는 데이터베이스 파일입니다. 확약 정의에 대해 마지막으로 성공한 트랜잭션을 식별하는 데 사용되는 정보는 확약 조작을 특정 확약 가능 자원 변경과 연관시키는 **확약 식별**에 의해 주어집니다.

확약 정의에 대하여 마지막으로 성공한 트랜잭션의 확약 식별은 확약 정의가 정상적으로 종료하지 못한 경우에만 통지 오브젝트에 들어갑니다. 이 정보는 어플리케이션에 대한 처리가 종료되어 어플리케이션이 다시 시작될 수 있는 지점을 판별하는 데 사용될 수 있습니다.

독립 디스크 풀의 경우 통지 오브젝트는 확약 정의와 동일한 독립 디스크 풀 또는 독립 디스크 풀 그룹에 있어야 합니다. 확약 정의를 다른 독립 디스크 풀이나 독립 디스크 풀 그룹으로 이동시키는 경우 통지 오브젝트 역시 이동된 다른 독립 디스크 풀이나 독립 디스크 풀 그룹에 있어야 합니다. 확약 정의가 비정상적으로 종료하면 다른 독립 디스크 풀이나 독립 디스크 풀 그룹의 통지 오브젝트가 갱신됩니다. 통지 오브젝트를 다른 독립 디스크 풀이나 독립 디스크 풀 그룹에서 찾을 수 없는 경우 갱신은 실패하고 메세지 CPF8358이 발생합니다.

저널된 자원이 현재 트랜잭션에 참여하고 확약 조작이 확약 식별을 사용하여 수행되는 경우 확약 식별은 특정 트랜잭션을 확약 중인 것으로 식별하는 확약 저널 항목(저널 코드 및 저널 유형이 C CM)에 위치하게 됩니다. 확약 정의가 포함된 확약 저널 항목은 트랜잭션에 참여하는 자원과 연관된 각 저널로 보내집니다.

다음 표에서는 확약 식별을 지정하는 방법 및 최대 *크기*를 보여줍니다. 확약 식별이 최대 *크기*를 초과하면 통지 오브젝트에 기록될 때 절단됩니다.

언어	조작	확약 조작의 최대 문자
CL	COMMIT 명령	3000[1]
ILE RPG*	COMIT 조작 코드	4000[1]
PLI.	PLICOMMIT 서브루틴	4000[1]
ILE C*	_Rcommit 함수	4000[1]
ILE COBOL*	COMMIT 술어	지원되지 않음
SQL	COMMIT문	지원되지 않음
주: [1]통지 오브젝트가 자료 영역인 경우 최대 *크기*는 2000자입니다.		

통지 오브젝트가 확약 식별을 사용하여 갱신될 때 다음과 같이 갱신됩니다.

데이터베이스 파일
데이터베이스 파일이 통지 오브젝트로 사용되면 확약 식별은 파일 끝에 추가됩니다. 기존 레코드가 파일에 남게 됩니다. 동시에 여러 사용자나 작업이 레코드를 변경할 수 있으므로 파일의 각 확약 식별에는 해당 자료를 작업 및 실패한 확약 정의와 연관시킬 고유 정보가 들어 있어야 합니다. 해당 파일은 저널될 수 있습니다.

자료 영역
자료 영역이 통지 오브젝트로 사용되면 확약 식별이 자료 영역에 위치할 때 자료 영역의 전체 내용이 대체됩니다. 둘 이상의 사용자나 작업이 동일 프로그램을 사용하는 경우 정상적으로 종료하지 못한 마지막 확약 정의로부터의 확약 식별만이 자료 영역에 들어갑니다. 따라서 하나의 자료 영역 통지 오브젝트로는

어플리케이션 프로그램을 다시 시작하기에 올바른 정보를 제공하기에 부족할 수도 있습니다. 이 문제를 해결하려면 각 워크스테이션 사용자 또는 작업에 대한 각 확약 정의에 대해 별도의 자료 영역을 사용하십시오.

메세지 대기행렬
메세지 대기행렬이 통지 오브젝트로 사용되는 경우 메세지 CPI8399가 메세지 대기행렬로 송신됩니다. 확약 식별은 메세지 CPI8399에 대한 2차 레벨 텍스트에 들어갑니다. 통지 오브젝트에 대한 데이터베이스 파일을 사용할 때 각 확약 식별의 내용은 작업에 대한 특정 확약 정의를 고유하게 식별하여 어플리케이션 프로그램이 다시 시작될 수 있도록 합니다.

통지 오브젝트 사용 예는 예: 통지 오브젝트를 사용하여 어플리케이션 시작을 참조하십시오.

확약 잠금 레벨

STRCMTCTL(확약 제어 시작) 명령의 LCKLVL 매개변수에 지정하는 값이 확약 정의를 위한 확약 제어 하에 열리고 위치하는 데이터베이스 파일에 대한 레코드 잠금의 디폴트 레벨이 됩니다. 레코드 잠금의 디폴트 레벨은 로컬 데이터베이스 파일을 열 때 대체할 수 없습니다. 그러나 SQL이 액세스하는 데이터베이스 파일은 첫 번째 SQL문을 발행했을 때 유효한 현재의 SQL 분리 레벨을 사용합니다. 확약 제어 고려사항 및 제한사항에서 오브젝트 및 레코드 레벨 변경에 대한 고려사항에 대해 설명합니다.

사용자의 필요에 따라 잠금 레벨을 지정해야 하며 허용되는 대기 시간 및 릴리스 프로시듀어가 가장 자주 사용됩니다.

다음 설명은 확약 제어 하에 열리는 파일에만 적용됩니다.

***CHG 잠금 레벨**
변경된 레코드를 동시에 실행 중인 다른 작업이 변경할 수 없도록 하려면 이 값을 사용하십시오. 확약 제어 하에 열린 파일의 경우 잠금은 트랜잭션이 수행되는 동안 유지됩니다. 확약 제어 하에서 열리지 않는 파일의 경우 레코드에 대한 잠금은 레코드가 읽히는 시점부터 갱신 조작이 완료될 때까지 유지됩니다.

***CS 잠금 레벨**
변경 및 검색된 레코드를 동시에 실행 중인 다른 작업이 변경할 수 없도록 하려면 이 값을 사용하십시오. 변경되지 않은 검색된 레코드는 해제되거나 다른 레코드가 검색될 때까지만 보호됩니다.

*CS 잠금 레벨은 이 작업이 읽은 레코드를 다른 작업이 갱신을 위해 읽을 수 없도록 합니다. 또한 프로그램은 다른 작업에서 레코드 잠금 유형 *UPDATE로 잠금된 레코드를 그 작업이 다른 레코드에 액세스할 때까지 갱신하기 위해 읽을 수 없습니다.

***ALL 잠금 레벨**
이 값을 사용하여 확약 제어 하에 변경된 레코드 및 검색된 레코드를 동시에 확약 제어 하에 실행되는 작업이 변경하지 못하도록 보호할 수 있습니다. 검색되었거나 변경된 레코드는 다음 확약 또는 롤백 조작이 수행될 때까지 보호됩니다.

*ALL 잠금 레벨을 사용하면 이 작업이 읽은 레코드를 다른 작업이 갱신하기 위해 액세스할 수 없습니다. 이것은 일반적인 잠금 프로토콜과는 다릅니다. 잠금 레벨이 *ALL로 지정되면 다른 작업에서 *UPDATE 레코드 잠금 유형을 사용하여 잠금된 경우 갱신을 위해 읽히지 않은 레코드도 액세스할 수 없습니다.

다음 표에서는 확약 제어 하에 있는 파일 및 확약 제어 하에 있지 않은 파일에 대한 레코드 잠금 지속 시간을 보여줍니다.

요구	**LCKLVL 매개변수**	잠금 지속 시간	잠금 유형
읽기 전용	확약 제어 하지 않음	잠금 없음	없음
	*CHG	잠금 없음	없음
	*CS	읽기에서 다음 읽기, 확약 또는 롤백까지	*READ
	*ALL	읽기에서 확약 또는 롤백까지	*READ
갱신을 위해 읽은 후 갱신 또는 삭제[1]	확약 제어 하지 않음	읽기에서 갱신 또는 삭제까지	*UPDATE
	*CHG	읽기에서 갱신 또는 삭제까지	*UPDATE
		갱신 또는 삭제에서 다음 확약 또는 롤백까지[2]	*UPDATE
	*CS	읽기에서 갱신 또는 삭제까지	*UPDATE
		갱신 또는 삭제에서 다음 확약 또는 롤백까지[2]	*UPDATE
	*ALL	읽기에서 갱신 또는 삭제까지	*UPDATE
		갱신 또는 삭제에서 다음 확약 또는 롤백까지[2]	
갱신을 위해 읽은 후 해제[1]	확약 제어 하지 않음	읽기에서 해제까지	*UPDATE
	*CHG	읽기에서 해제까지	*UPDATE
	*CS	읽기에서 해제, 확약 또는 롤백까지	*UPDATE
		그리고나서 해제에서 다음 읽기, 확약 또는 롤백까지	*UPDATE
	읽기에서 해제, 확약 또는 롤백까지	*UPDATE	그리고나서 해제에서 다음 확약 또는 롤백까지
	*UPDATE		
추가	확약 제어 하지 않음	잠금 없음	None
	*CHG	추가에서 확약 또는 롤백까지	*UPDATE
	*CS	추가에서 확약 또는 롤백까지	*UPDATE
	*ALL	추가에서 확약 또는 롤백까지	*UPDATE
직접 쓰기	확약 제어 하지 않음	직접 쓰기 지속 시간 동안	*UPDATE
	*CHG	직접 쓰기에서 확약 또는 롤백까지	*UPDATE
	*CS	직접 쓰기에서 확약 또는 롤백까지	*UPDATE
	*ALL	직접 쓰기에서 확약 또는 롤백까지	*UPDATE

요구	LCKLVL 매개변수	잠금 지속 시간	잠금 유형
주: [1]확약 또는 롤백 조작이 갱신을 위한 읽기 조작 후 및 그 레코드가 갱신, 삭제 또는 해제 되기 전에 수행되는 경우 그 레코드는 확약 또는 롤백 조작 동안 잠금 해제가 됩니다. 레코드에 대한 보호는 확약 또는 롤백이 완료되자 마자 해제됩니다. [2]레코드가 삭제되고 해당 트랜잭션에 대한 확약 또는 롤백은 아직 발행되지 않은 경우 삭제된 레코드는 잠금 상태로 남아 있지 않습니다. 동일하거나 다른 작업이 삭제된 레코드를 키로 읽으려고 시도하는 경우 그 작업은 레코드를 찾을 수 없음 표시를 수신합니다. 그러나 그 파일에 대한 고유 키 액세스 경로가 존재하는 경우 트랜잭션이 확약될 때까지 다른 작업은 삭제된 레코드와 동일한 고유 키 값을 갖는 레코드를 삽입하거나 갱신할 수 없습니다. 레코드가 삭제되었으나 해당 트랜잭션에 대해 아직 확약 또는 롤백이 발행되지 않은 경우 삭제된 레코드는 잠금 상태로 남아 있지 않습니다. 동일하거나 다른 작업이 삭제된 레코드를 키로 읽으려고 시도하는 경우 그 작업은 레코드를 찾을 수 없음 표시를 수신합니다. 그러나 그 파일에 대한 고유 키 액세스 경로가 존재하는 경우 트랜잭션이 확약될 때까지 다른 작업은 삭제된 레코드와 동일한 고유 키 값을 갖는 레코드를 삽입하거나 갱신할 수 없습니다.			

레코드 잠금 유형 *READ는 잠금 레벨이 *CS 또는 *ALL일 때 갱신을 위해 읽히지 않은 레코드에 대하여 사용합니다. 이 유형의 잠금은 다른 작업이 갱신을 위해 레코드를 읽을 수 없도록 방지하지만 읽기 전용 조작으로 레코드를 액세스하는 것은 방지하지 않습니다.

레코드 잠금 유형 *UPDATE는 갱신을 위해 갱신, 삭제, 추가 또는 읽혀지는 레코드에 대하여 사용합니다. 이 유형의 잠금은 다른 작업이 갱신을 위해 레코드를 읽을 수 없도록 하고 레코드 잠금 레벨이 *CS 또는 *ALL인 확약 제어 하에서 실행되는 작업이 읽기 전용 조작을 위한 것이더라도 레코드에 액세스할 수 없도록 합니다.

확약 제어를 사용하지 않는 프로그램은 다른 작업에 의해 잠금 상태가 된 레코드를 읽을 수는 있지만 LCKLVL 매개변수에 지정된 값과 상관없이 갱신하기 위해서는 레코드를 읽을 수 없습니다.

활성 그룹이나 작업에 대해 확약 제어가 시작될 때 확약 정의에 지정된 잠금 레벨은 해당 특정 확약 정의와 연관되어 열려 있는 파일에만 적용됩니다.

주: *CS 및 *ALL 잠금 레벨 값은 현재 지연 변경을 가지고 있는 레코드를 다른 작업에서 검색할 수 없도록 보호합니다. 그러나 *CS 및 *ALL 잠금 레벨 값은 현재 지연 변경을 가지고 있는 하나의 활성 그룹에서 실행되고 있는 프로그램을 사용하여 동일한 작업 내 다른 활성 그룹에서 실행되는 프로그램으로부터 레코드를 검색할 수 없도록 보호하지는 *않습니다*.

동일한 작업 내에서 동일한 확약 정의를 사용하여 해당 레코드를 액세스하는 한, 프로그램은 현재 트랜잭션 내에서 이미 변경된 레코드를 변경할 수 있습니다. 작업 레벨 확약 정의를 사용할 때 변경된 레코드에 대한 액세스는 작업 레벨의 확약 정의를 사용하는 활성 그룹 내에서 실행되는 프로그램으로부터 이루어질 수 있습니다.

확약 제어 종료

ENDCMTCTL(확약 제어 종료) 명령을 사용하여 작업 레벨의 확약 정의 또는 활성 그룹 레벨의 확약 정의에 대한 확약 제어를 종료할 수 있습니다. ENDCMTCTL 명령을 발행하는 것은 요구하는 프로그램에서 사용하는 확약 정의를 종료해야 한다고 시스템에 알리는 것입니다. ENDCMTCTL 명령은 해당 작업에 대한 단 하나의 확약 정의를 종료하고 해당 작업에 대한 나머지 다른 확약 정의는 변경되지 않은 채 남아 있습니다.

활성 그룹 레벨의 확약 정의가 종료되면 그 작업에 대해 이미 작업 레벨의 확약 정의가 시작되어 있지 않은 한 그 활성 그룹 내에서 실행 중인 프로그램은 더 이상 확약 제어 하에서 변경할 수 없습니다. 작업 레벨의 확약 정의가 활성 상태이면 방금 확약 제어를 종료한 활성 그룹 내에서 실행되는 프로그램에 의해 즉시 사용될 수 있습니다.

작업 레벨 확약 정의가 종료된 경우 작업 레벨 확약 정의를 사용하던 작업 내에서 실행 중인 프로그램은 STRCMTCTL 명령을 사용하여 먼저 확약 제어를 시작해야지만 확약 제어 하에서 변경을 수행할 수 있습니다.

ENDCMTCTL 명령을 발행하기 전에 종료될 확약 정의에 대하여 다음이 충족되어야 합니다.

- 종료될 확약 정의에 대한 확약 제어 하에 열린 모든 파일을 먼저 닫아야 합니다. 작업 레벨 확약 정의를 종료하는 경우 여기에는 작업 레벨 확약 정의를 사용하는 모든 활성 그룹에서 실행되는 모든 프로그램에 의해 열린 확약 제어 하의 파일이 모두 포함됩니다.
- QTNRMVCR API를 사용하여 종료될 확약 정의에 대한 모든 API 확약 자원을 먼저 제거해야 합니다. 작업 레벨 확약 정의를 종료할 때 여기에는 작업 레벨 확약 정의를 사용하는 모든 활성 그룹에서 실행되는 프로그램에 의해 추가된 API 확약 자원이 모두 포함됩니다.
- 종료될 확약 정의와 연관된 리모트 데이터베이스와의 연결을 단절해야 합니다.
- 확약 정의와 연관된 모든 보호 대화는 올바른 동기화 레벨을 사용하여 정상적으로 종료해야 합니다.

확약 제어가 대화식 작업에서 종료되거나 확약 정의와 연관된 하나 이상의 확약 가능 자원이 지연 변경을 가지고 있는 경우, 지연 변경을 확약할 것인지, 롤백할 것인지 또는 EMDCMTCTL 요구를 취소할 것인지 묻는 조회 메세지 CPA8350이 사용자에게 보내집니다.

확약 제어가 일괄처리 작업에서 종료되고 종료될 확약 정의와 연관된 하나 이상의 닫힌 파일이 아직 지연 변경을 가지고 있는 경우, 변경은 롤백되고 다음과 같은 메세지가 송신됩니다.

- 로컬 자원만 등록되어 있는 경우 CPF8356
- 리모트 자원만 등록되어 있는 경우 CPF835C
- 로컬 및 리모트 자원이 모두 등록되어 있는 경우 CPF83E4

종료되는 확약 정의에 대해 통지 오브젝트가 정의되어 있는 경우 통지 오브젝트는 갱신될 수 있습니다. 시스템에 의한 통지 오브젝트 갱신에 대한 자세한 내용은 4-13 페이지의 ″통지 오브젝트 갱신″을 참조하십시오.

최종 에이전트로 등록된 API를 가지고 있는 활성 그룹이 종료할 때 그 API의 종료 프로그램이 호출되어 확약 또는 롤백 결정을 수신합니다. 이 경우 활성 그룹이 정상적으로 종료하더라도 API 종료 프로그램에서 롤백 요구가 리턴됩니다. 그러므로 내재적 확약 조작이 수행되지 않을 수도 있습니다.

확약 정의가 성공적으로 종료되고 나면 필요한 회복 조치가 모두 수행됩니다. 방금 종료한 확약 종료와 연관된 확약 자원에 대해서는 별도의 회복 조치가 수행되지 않습니다.

확약 정의가 종료된 후 작업 레벨 또는 활성 그룹 레벨의 확약 정의가 활성 그룹 내에서 실행되는 프로그램에 대하여 다시 시작될 수 있습니다. 작업 레벨 확약 정의는 그 작업에 대해 이미 시작하지 않은 경우에만 시작될 수 있습니다.

확약 정의는 활성 그룹 내에서 실행되는 프로그램에 의해 여러번 시작되고 종료될 수 있지만 반복되는 시작 및 종료 조치에 필요한 시스템 자원의 양은 작업 성능 및 전반적인 시스템 성능을 저하시킬 수 있습니다. 그러므로 나중에 호출될 프로그램이 그 확약 정의를 사용할 경우 활성 상태로 두는 것이 바람직합니다.

시스템에 의한 확약 제어 종료

시스템에서 확약 제어를 종료하거나 내재적 확약 또는 롤백 조작을 수행할 수 있습니다. 때로는 시스템에 의한 확약 제어 종료가 정상적인 경우도 있습니다. 그렇지 않은 경우 확약 제어는 비정상적인 시스템 또는 작업 종료로 종료됩니다.

다음 페이지에서는 시스템이 내재적으로 확약 제어를 종료시키는 상황 및 사용자가 어떤 조치를 취해야 하는지에 대해 설명합니다.

- 활성 그룹 종료 중 확약 제어
- 내재적 확약 및 롤백 조작
- 정상적인 라우팅 단계 종료 시 확약 제어
- 비정상적인 시스템 또는 작업 종료 시 확약 제어
- 확약 제어 종료 후 통지 오브젝트 갱신
- 초기 프로그램 로드(IPL) 시 확약 제어 회복

활성 그룹 종료 중 확약 제어

활성 그룹이 종료하면 시스템은 자동으로 활성 그룹 레벨의 확약 정의를 종료합니다. 활성 그룹 레벨의 확약 정의에 대해 지연된 변경 사항이 있고 활성 그룹이 정상적으로 종료하는 경우 시스템은 활성 그룹이 종료하기 전에 확약 정의에 대한 내재적 확약 조작을 수행합니다. 활성 그룹이 비정상적으로 종료하거나 활성 그룹 범주의 확약 제어 하에 열린 파일을 닫을 때 시스템에 오류가 발생한 경우 활성 그룹 레벨의 확약 정의에 대해 내재적 롤백 조작이 수행됩니다.

주: 내재적 확약 또는 롤백 조작은 활성 그룹이 *JOB 또는 *DFTACTGRP 확약 정의에 대한 처리를 종료하는 동안에는 절대로 수행되지 *않습니다*. 활성 그룹을 종료해도 *JOB 및 *DFTACTGRP 확약 정의는 절대 종료되지 않기 때문입니다. 대신 이러한 확약 정의는 ENDCMTCTL을 사용하여 명시적으로 종료하거나 작업 종료 시 시스템에 의해 종료됩니다.

활성 그룹이 종료하면 시스템은 자동으로 활성 그룹 범위의 파일을 모두 닫습니다. 여기에는 확약 제어 하에서 열린 활성 그룹 범위의 모든 데이터베이스 파일도 포함됩니다. 이러한 파일에 대한 닫기는 활성 그룹 레벨의 확약 정의에 대해 수행될 수 있는 내재적 확약 조작 수행 전에 발생합니다. 그러므로 내재적 확약 조작이 수행되기 전에 I/O 버퍼에 상주하는 레코드가 먼저 데이터베이스에 자동으로 저장됩니다.

수행될 내재적 확약 또는 롤백 조작의 일부로 활성 그룹 레벨 확약 정의와 연관된 각 API 확약 자원에 대한 API 확약 및 롤백 종료 프로그램이 호출됩니다. 종료 프로그램은 5분 이내에 처리를 완료해야 합니다. API 확약 및 롤백 종료 프로그램이 호출된 후 시스템은 자동으로 API 확약 자원을 제거합니다.

활성 그룹 종료로 인해 종료되는 확약 정의에 대해 내재적 롤백 조작이 수행되는 경우 그 확약 정의에 통지 오브젝트가 정의되어 있으면 통지 오브젝트가 갱신될 수 있습니다. 시스템의 통지 오브젝트 갱신에 대한 자세한 내용은 통지 오브젝트 갱신을 참조하십시오.

내재적 확약 및 롤백 조작

일반적으로 확약 또는 롤백 조작은 확약 제어를 지원하는 사용 가능한 프로그래밍 언어 중 하나를 사용하여 어플리케이션 프로그램에서 시작됩니다. 이러한 유형의 확약 및 롤백 조작을 **명시적 확약 및 롤백 요구**라고 합니다. 그러나 어떤 경우에는 시스템이 확약 정의에 대하여 확약 또는 롤백 조작을 시작합니다. 시스템에 의해 시작된 확약 및 롤백 조작을 **내재적 확약 및 롤백 요구**라고 합니다.

다음의 두 표에서는 지연 변경을 가지고 있는 확약 정의와 관련하여 특정 이벤트가 발생할 때 시스템이 수행하는 일을 보여줍니다. 다음 중 하나가 참이면 확약 정의는 지연 변경을 갖게 됩니다.

- 확약 가능한 자원이 갱신된 경우.
- 파일을 읽음으로써 파일 위치가 변경되었기 때문에 확약 제어 하에 열린 데이터베이스 파일이 읽힌 경우.
- 확약 정의가 API 자원을 가지고 있는 경우. 사용자 프로그램에 의해 API 자원 변경이 이루어지기 때문에 시스템은 모든 API 자원이 지연 변경을 가지고 있는 것으로 전제해야 합니다.

C CM(확약 조작) 저널 항목 및 C RB(롤백 조작) 저널 항목은 이 조작이 명시적인지 또는 암시적인지를 나타냅니다.

아래 표에서는 다음에 근거하여 작업이 정상적으로 또는 비정상적으로 종료할 때 시스템이 취하는 조치를 보여줍니다.

- 트랜잭션 상태
- 확약 정의에 대한 action-if-end 작업 값
- API 자원이 최종 에이전트인지 여부

상태	최종 에이전트 API	Action if Endjob[1] 옵션	확약 또는 롤백 조작
RST	N/A	N/A	확약 정의가 X/Open 글로벌 트랜잭션과 연관되지 않은 경우 내재적 롤백이 수행됩니다. 확약 정의가 X/Open 글로벌 트랜잭션과 연관된 경우 다음이 발생합니다. • 트랜잭션 분기 상태가 활동(S1) 상태가 아니면 아무런 조치도 수행되지 않으며 트랜잭션 분기는 동일한 상태로 남게 됩니다. • 트랜잭션 분기 상태가 활동(S1) 상태이면 내재적 롤백이 수행됩니다.

상태	최종 에이전트 API	Action if Endjob[1] 옵션	확약 또는 롤백 조작
PIP	N/A	N/A	확약 정의가 X/Open 글로벌 트랜잭션과 연관되지 않은 경우 내재적 롤백이 수행됩니다. 확약 정의가 X/Open 글로벌 트랜잭션과 연관된 경우 트랜잭션 분기는 유휴(S2) 상태가 되며 유휴(S2) 상태로 남아 있게 됩니다.
PRP	N/A	WAIT	확약 정의가 X/Open[2] 글로벌 트랜잭션과 연관되지 않은 경우 다음이 발생합니다. • 확약 조작 개시자로부터 결정을 수신하기 위해 재동기화가 시작됩니다. • 확약 또는 롤백을 하기 위한 리턴된 결정이 수행됩니다. 이는 명시적 조작으로 간주됩니다.
PRP	N/A	C	확약 정의가 X/Open[2] 글로벌 트랜잭션과 연관되지 않은 경우 내재적 확약 조작이 수행됩니다.
		R	확약 정의가 X/Open 글로벌 트랜잭션과 연관되지 않은 경우 내재적 롤백 조작이 수행됩니다. 확약 정의가 X/Open 글로벌 트랜잭션과 연관된 경우 다음이 발생합니다. • 트랜잭션을 시작한 작업이 종료하는 경우 트랜잭션은 XA TM이 확약하거나 롤백할 때까지 준비 상태로 남아 있습니다. XA 트랜잭션 분기 상태는 이 경우 준비(S3) 상태로 남아 있게 됩니다. • 트랜잭션 작업이 라우트되는 SQL 서버 작업이 종료하면 강제 롤백이 내재적으로 수행됩니다. XA 트랜잭션 분기 상태는 발견적 완료(S5) 상태로 변경됩니다.
CIP	N/A	N/A	명시적 확약 조작이 수행됩니다.

상태	최종 에이전트 API	Action if Endjob[1] 옵션	확약 또는 롤백 조작
LAP	NO	WAIT	1. 최종 에이전트에 대한 재동기화는 확약 또는 롤백 결정을 검색하는 데 사용됩니다.
			2. 확약 또는 롤백을 하기 위한 리턴된 결정이 수행됩니다. 이는 명시적 조작으로 간주됩니다.
LAP	YES	WAIT	1. 최종 에이전트 API가 확약 또는 롤백 결정을 검색하기 위해 호출됩니다.
			2. 확약 또는 롤백 조작이 수행됩니다. 이는 명시적 조작으로 간주됩니다.
LAP	N/A	C	내재적 확약 조작이 수행됩니다.
		R	내재적 롤백 조작이 수행됩니다.
CMT	N/A	N/A	이 확약 정의 및 다운스트림 위치에 대하여 확약 조작이 이미 완료되었습니다. 확약 조작이 완료되었습니다.
VRO	N/A	N/A	로컬 및 리모트 에이전트가 읽기 전용으로 인가되었습니다. 모든 다운스트림 에이전트도 읽기 전용으로 인가되어야 합니다. 아무런 조치도 필요하지 않습니다.
RBR	N/A	N/A	롤백 조작이 필요합니다. 명시적 롤백 조작이 수행됩니다.

주:

[1] QTNCHGCO(확약 옵션 변경) API를 사용하여 `Action if Endjob` 옵션을 변경할 수 있습니다.

[2]확약 정의가 X/Open 글로벌 트랜잭션과 연관된 경우 다음이 발생합니다.

- 트랜잭션을 시작한 작업이 종료하는 경우 트랜잭션은 XA TM이 확약하거나 롤백할 때까지 준비 상태로 남아 있습니다. XA 트랜잭션 분기 상태는 이 경우 준비(S3) 상태로 남아 있게 됩니다.
- 트랜잭션 범위의 잠금에 한해 트랜잭션 작업이 라우트되는 SQL 서버 작업이 종료하는 경우 강제 롤백이 내재적으로 수행됩니다. XA 트랜잭션 분기 상태는 발견적 완료(S5) 상태로 변경됩니다.

다음 표에서는 활성 그룹이 종료할 때 시스템이 취하는 조치를 보여주며, 이 표는 작업 범위의 잠금을 수행하는 트랜잭션에만 적용됩니다. 시스템 조치는 다음에 근거하여 수행됩니다.

- 트랜잭션 상태(활성 그룹이 종료하면 항상 재설정(RST)됩니다).
- 활성 그룹이 정상적 또는 비정상적으로 종료하는 방식
- API 자원이 최종 에이전트인지 여부

주: API 자원이 최종 에이전트로 등록되면 확약 또는 롤백 결정에 대한 제어는 최종 에이전트로 넘어 갑니다. 결정의 결과는 명시적 조작으로 간주됩니다.

상태	최종 에이전트 API	종료 유형	확약 또는 롤백 조작
RST	아니오	정상	내재적 확약 조작이 수행됩니다. 보호 대화가 존재하는 경우 확약 정의는 확약 조작의 루트 개시자가 됩니다.
RST	아니오	비정상	내재적 롤백이 수행됩니다.
RST	예	정상	API 종료 프로그램이 호출됩니다. 확약 또는 롤백 조작이 API에 의해 결정됩니다.
RST	예	비정상	API 종료 프로그램이 호출됩니다. 확약 또는 롤백 조작이 API에 의해 결정됩니다.

정상적인 라우팅 단계 종료 시 확약 제어

라우팅 단계가 정상적으로 종료되면 시스템은 작업에 대한 모든 확약 정의를 종료합니다.

주: 다음은 작업 범위의 잠금을 사용하는 확약 정의에만 적용됩니다.

다음 중 하나에 의해 라우팅 단계가 정상적으로 종료합니다.

- 일괄처리 작업에 대한 정상적인 종료.
- 대화식 작업에 대한 정상적인 사인 오프.
- RRTJOB(작업 재라우트), TFRJOB(작업 전송) 또는 TFRBCHJOB(일괄처리 작업 전송) 명령은 현재의 라우팅 단계를 종료하고 새로운 라우팅 단계를 시작합니다.

그 외의 다른 라우팅 단계의 종료는 비정상 종료로 간주되고 작업 기록부의 작업 완료 메세지 CPF1164에 0이 아닌 완료 코드로 표시하여 알 수 있습니다.

라우팅 단계 종료 시 확약 정의를 종료하기 전에 시스템은 확약 정의가 지연 변경을 가지고 있는 경우 내재적인 롤백 조작을 수행합니다. 여기에는 그 확약 정의와 연관된 각 API 확약 자원에 대한 API 확약 및 롤백 종료 프로그램 호출이 포함됩니다. 종료 프로그램은 5분 이내에 처리를 완료해야 합니다. API 확약 및 롤백 종료 프로그램이 호출된 후 시스템은 자동으로 API 확약 자원을 제거합니다.

확약 정의에 통지 오브젝트가 정의되면 통지 오브젝트가 갱신될 수 있습니다. 시스템에 의한 통지 오브젝트 갱신에 대한 자세한 내용은 통지 오브젝트 갱신을 참조하십시오.

비정상적인 시스템 또는 작업 종료 시 확약 제어

이 주제는 작업 범위의 잠금이 적용된 확약 정의에만 해당됩니다. 작업이 비정상적으로 종료되면 시스템은 모든 확약 정의를 종료합니다. 이 확약 정의는 작업 종료 처리 중에 종료됩니다. 시스템이 비정상적으로 종료하

면 시스템은 시스템의 비정상 종료 시 활동 중이던 모든 작업이 시작 및 사용하고 있던 확약 정의를 모두 종료합니다. 이 확약 정의는 비정상적인 시스템 종료 후 다음 IPL 시 수행되는 데이터베이스 회복 처리의 일부로 종료됩니다.

주의:
확약 정의 회복은 전원 장애, 하드웨어 장애 또는 운영 체제나 사용권 내부 코드의 오류로 인해 발생한 시스템이나 작업의 비정상 종료를 의미합니다. 작업을 강제로 비정상적으로 종료시키기 위해 비정상 작업 종료(ENDJOBABN) 명령을 사용해서는 안됩니다. 이러한 비정상 종료로 인해 종료시키려는 작업의 활동 트랜잭션에 대한 지연된 변경 사항이 일부만 확약되거나 롤백될 수 있습니다. 다음 IPL시 ENDJOBABN 명령으로 종료된 작업의 부분적인 트랜잭션에 대해 회복을 시도할 수도 있습니다.

ENDJOBABN 명령을 사용하여 종료시킨 작업에 대해 IPL 시 시스템이 수행하는 확약 제어 회복 결과는 확실하지 않습니다. 작업이 비정상적으로 종료할 때 확약 자원에 대한 모든 잠금이 해제되기 때문입니다. 부분적인 트랜잭션으로 인해 지연된 변경 사항을 다른 작업에서 사용할 수 있습니다. 이렇게 지연된 변경사항으로 인해 다른 어플리케이션 프로그램에서 데이터베이스를 잘못 변경할 수 있습니다. 마찬가지로 나중에 수행되는 IPL 회복 작업은 작업이 비정상적으로 종료된 후 어플리케이션에 의해 변경된 부분에 부정적인 영향을 미칠 수 있습니다. 예를 들어 SQL 표는 지연된 표 작성에 대한 롤백 조치로 수행된 IPL 회복 시에 삭제될 수 있습니다. 그러나 작업이 비정상적으로 종료된 후에 다른 어플리케이션이 이미 표에 몇 행을 삽입했을 수도 있습니다.

시스템은 비정상 작업 종료시 또는 비정상 시스템 종료 후 다음 IPL 시에 종료된 확약 정의에 대하여 다음을 수행합니다.

- 확약 조작 중간에 확약 정의 처리가 인터럽트되지 않는 한, 확약 정의에 지연된 변경 사항이 있는 경우 시스템은 확약 정의가 종료되기 전에 내재적인 롤백 조작을 수행합니다. 확약 조작이 중간에 종료되는 경우 그 상태에 따라 트랜잭션은 롤백, 재동기화 또는 확약될 수 있습니다. 내재적 확약 및 롤백 조작을 참조하십시오. 내재적 롤백 조작을 수행하거나 확약 조작을 완료하기 위한 처리에는 확약 정의와 연관된 각 API 확약 자원에 대한 API 확약 및 롤백 종료 프로그램 호출이 포함됩니다. API 확약 및 롤백 종료 프로그램이 호출된 후 시스템은 자동으로 API 확약 자원을 제거합니다.

  주의:
  트랜잭션이 확실하지 않은 상태(트랜잭션 상태가 LAP 또는 PRP)일 때 작업을 종료하면 데이터베이스의 무결성이 손상될 수 있습니다(하나 이상의 시스템에서 변경 사항이 확약되고 다른 시스템에서 롤백될 수 있음).
  - *작업 종료 시 조치* 확약 옵션이 COMMIT인 경우 트랜잭션에 참여하는 다른 시스템에서의 변경 사항이 확약되거나 롤백되는 것과는 관계없이 작업이 종료되면 이 시스템에서의 변경 사항은 확약됩니다.
  - *작업 종료 시 조치* 확약 옵션이 ROLLBACK인 경우 트랜잭션에 참여하는 다른 시스템에서의 변경 사항이 확약되거나 롤백되는 것과는 관계없이 작업이 종료되면 이 시스템에서의 변경 사항은 롤백됩니다.

– *작업 종료 시 조치* 확약 옵션이 WAIT인 경우 확약 또는 롤백 결정을 소유한 시스템에 대한 재동기화가 완료될 때까지 작업이 종료되지 않습니다. 재동기화가 완료되기 전에 작업이 종료되도록 하려면 발견적(heuristic) 결정을 내리고 재동기화를 취소해야 합니다.

장기 수행 롤백 중에 시스템을 비정상 종료하는 것은 권장되지 **않습니다**. 장기 수행 롤백 중에 시스템을 비정상 종료하면 작업이 종료될 때(또는 시스템이 종료된 후 다음 IPL 시에)또 다른 롤백이 일어날 수 있습니다. 뒤에 수행되는 롤백은 처음의 롤백이 수행했던 작업을 반복하므로 시간이 훨씬 더 오래 걸립니다.

- 확약 정의에 통지 오브젝트가 정의되면 통지 오브젝트가 갱신될 수 있습니다. 시스템의 통지 오브젝트 갱신에 대한 자세한 내용은 통지 오브젝트 갱신을 참조하십시오.

확약 제어 종료 전에 프로세스가 종료하고 보호 대화가 아직 활동 중이면 확약 또는 롤백에 확약 정의가 필요할 수도 있습니다. 이 조치는 확약 정의의 상태 옵션 및 작업 종료 시 조치 옵션에 따라 달라집니다.

통지 오브젝트 갱신

통지 오브젝트의 목적 상 다음이 미확약 변경으로 간주됩니다.

- 확약 제어 하에 이루어진 레코드 갱신
- 확약 제어 하에서 삭제된 레코드
- 확약 제어 하에서 로컬 DDL 오브젝트에 대해 수행된 오브젝트 레벨 변경
- 확약 제어 하에서 열린 데이터베이스 파일에 대한 읽기 조작. 이는 롤백 조작이 수행될 때 파일 위치가 마지막 확약 경계로 되돌려졌기 때문입니다. 확약 제어 하에서 읽기 조작을 수행하는 경우 파일 위치가 변경되므로 확약 정의에 대한 미확약 변경이 존재합니다.
- 다음과 같은 자원 중 하나가 추가된 확약 정의는 항상 미확약 변경을 포함하는 것으로 간주됩니다.
  – API 확약 자원
  – 리모트 분산 관계형 데이터베이스 구조(DRDA*) 자원
  – 분산 데이터베이스 관리 구조(DDM) 자원
  – LU 6.2 자원

  이것은 이러한 유형의 자원과 연관된 오브젝트가 실제 언제 변경되었는지 모르기 때문입니다. 확약 가능 자원 유형에 이러한 유형의 자원을 추가하고 이에 대한 작업을 수행하는 방법에 대해 자세히 나와 있습니다.

시스템은 통지 오브젝트를 갱신하고 다음과 같은 방법에 기초하여 확약 정의를 종료합니다.

- 작업이 정상적으로 종료하고 미확약 변경이 존재하는 경우 시스템은 마지막으로 성공한 확약 조작의 확약 식별을 통지 오브젝트에 넣지 *않습니다*.
- 활성 그룹이 종료할 때 내재적 확약 조작이 활성 그룹 레벨의 확약 정의에 대하여 수행되는 경우 시스템은 마지막으로 성공한 확약 조작의 확약 식별을 통지 오브젝트에 넣지 *않습니다*.

  주: 내재적 확약 조작은 *DFTACTGRO 또는 *JOB 확약 정의에 대해 수행되지 *않습니다*.
- 확약 정의에 대한 첫 번째 성공적인 확약 조작이 이루어지기 전에 시스템, 작업 또는 활성 그룹이 비정상적으로 종료한 경우 마지막 확약 식별이 없으므로 시스템은 통지 오브젝트를 갱신하지 *않습니다*. 이 조건과

정상적인 프로그램 완료를 구분하려면 사용자 프로그램이 확약 정의에 대한 첫 번째 성공적인 확약 조작을 완료하기 전에 특정 항목을 사용하여 통지 오브젝트를 갱신해야 합니다.

- 최소한 하나의 성공적인 확약 조작이 수행된 후에 작업이나 시스템의 비정상 종료가 발생한 경우 시스템은 그 확약 조작의 확약 식별을 통지 오브젝트에 넣습니다. 마지막으로 성공한 확약 조작이 확약 식별을 지정하지 않은 경우 통지 오브젝트는 갱신되지 않습니다. 작업의 비정상 종료의 경우 이 통지 오브젝트 처리는 그 작업에 대해 사용 중이던 각 확약 정의에 대해 수행됩니다. 시스템의 비정상 종료의 경우 이 통지 오브젝트 처리는 시스템 상의 모든 작업에 대해 사용 중이던 각 확약 정의에 대해 수행됩니다.
- 시스템은 다음 조건이 모두 발생하는 경우 해당 확약 정의에 대하여 마지막으로 성공한 확약 조작의 확약 식별을 사용하여 통지 오브젝트를 갱신합니다.
  - 디폴트가 아닌 활성 그룹이 종료합니다.
  - 활성 그룹 레벨의 확약 정의에 대하여 내재적 롤백 조작이 수행됩니다.
  - 최소한 하나의 성공적인 확약 조작이 확약 정의에 대해 수행되었습니다.

  마지막으로 성공한 확약 조작이 확약 식별을 지정하지 않은 경우 통지 오브젝트는 갱신되지 않습니다. 활성 그룹이 비정상적으로 종료하거나 활성 그룹 범주의 확약 제어 하에서 열린 파일을 닫을 때 시스템에 오류가 발생한 경우 활성 그룹 레벨의 확약 정의에 대해 내재적 롤백 조작이 수행됩니다. 활성 그룹에 대한 데이터베이스 파일의 범위에 대한 자세한 정보 및 활성 그룹이 종료되는 방법은 사용 중인 ILE 언어의 참조서를 참조하십시오.
- 작업이 정상적으로 종료할 때 미확약 변경이 존재하고 성공적으로 수행된 확약 조작이 하나 이상 있는 경우 마지막으로 성공한 확약 조작의 확약 식별이 통지 오브젝트에 들어가고 미확약 변경은 롤백됩니다. 마지막으로 성공한 확약 조작이 확약 식별을 지정하지 않은 경우 통지 오브젝트는 갱신되지 않습니다. 이 통지 오브젝트 처리는 작업 종료 시 그 작업에 대해 활동 상태였던 각 확약 정의에 대해 수행됩니다. 작업의 정상 종료 중에 수행되는 기능에 대한 자세한 정보는 정상적인 라우팅 단계 종료 시 확약 제어를 참조하십시오.
- ENDCMTCTL 명령이 실행될 때 미확약 변경이 존재하면 마지막으로 성공한 확약 조작이 확약 식별을 지정한 경우에만 통지 오브젝트가 갱신됩니다.
  - 일괄처리 작업의 경우 미확약 변경은 롤백되고 마지막으로 성공한 확약 조작의 확약 식별이 통지 오브젝트에 들어가게 됩니다.
  - 대화식 작업의 경우 조회 메세지 CPA8350에 대한 응답이 변경 롤백이면 미확약 변경은 롤백되고 마지막으로 성공한 확약 조작의 확약 식별이 통지 오브젝트에 들어가게 됩니다.
  - 대화식 작업의 경우 조회 메세지 CPA8350에 대한 응답이 변경 확약이면 시스템은 사용할 확약 식별을 입력하도록 프롬트하고 변경이 확약됩니다. 프롬트 화면에서 입력된 확약 식별은 통지 오브젝트에 위치합니다.
  - 대화식 작업의 경우 조회 메세지 CPA8350에 대한 응답이 ENDCMTCTL 요구를 취소하라는 것인 경우 지연 변경은 남아 있고 통지 오브젝트는 *갱신되지 않습니다.*

비정상 종료 후 초기 프로그램 로드(IPL) 시 확약 제어 회복

시스템이 비정상적으로 종료한 후 초기 프로그램 로드(IPL)를 수행할 때 시스템은 시스템 종료 시 활동 중이던 모든 확약 정의를 회복하려고 시도합니다. 마찬가지로 독립 디스크 풀(pool)을 연결변환할 때 시스템은 단

절변환 또는 비정상 종료될 때 활동 중이던 독립 디스크 풀(pool)과 관련된 모든 확약 정의를 회복하려고 시도합니다. 회복은 IPL시 시스템에 의해 시작된 데이터베이스 서버 작업에 의해 수행됩니다. 다른 작업에서 수행할 수 없거나 수행하면 안되는 작업을 처리하기 위해 시스템은 데이터베이스 서버 작업을 시작합니다.

데이터베이스 서버 작업의 이름은 QDBSRVnn으로 nn은 두 자리 숫자입니다. 데이터베이스 서버 작업의 수는 시스템 크기에 따라 다릅니다. 마찬가지로 독립 디스크 풀(pool) 또는 독립 디스크 풀(pool) 그룹에 대한 데이터베이스 서버 작업의 이름은 QDBSxxxVnn이고 여기에서 xxx는 독립 디스크 풀(pool) 번호이며 nn은 두 자리 숫자입니다. 예를 들어 QDBS035V02는 독립 디스크 풀(pool) 35에 대한 데이터베이스 서버 작업의 이름이 될 수 있습니다.

2단계 확약 제어에 대한 트랜잭션 상태에서는 실패가 발생했을 때 트랜잭션 상태에 따라 시스템이 취하는 조치를 보여줍니다. PRP와 LAP의 두 가지 상태의 경우 시스템 조치는 확실하지 않습니다.

주:

- 다음은 작업 범위의 잠금을 사용하는 확약 정의에만 적용됩니다.
- 트랜잭션 관리자는 이 주제에서 설명된 재동기화 프로세스가 아닌 XA API를 사용하여 XA 트랜잭션과 연관된 확약 정의를 회복합니다(잠금이 작업 범위든 트랜잭션 범위든 상관 없이).

시스템은 트랜잭션에 참여한 다른 위치와의 재동기화를 수행할 때까지 무엇을 해야 할지 결정할 수 없습니다. 이 재동기화는 IPL 또는 연결변환 조작이 완료된 후에 수행됩니다.

시스템은 데이터베이스 서버 작업을 사용하여 이 재동기화를 수행합니다. 회복될 필요가 있는 확약 정의는 데이터베이스 서버 작업과 연관됩니다. IPL 동안에 시스템은 시스템이 종료되기 전에 확약 정의가 보유하고 있던 모든 레코드 잠금 및 다른 오브젝트 잠금을 확보합니다. 이러한 잠금은 재동기화가 완료되고 자원이 확약 또는 롤백될 때까지 로컬 확약 자원을 보호하기 위해 필요합니다.

리모트 위치와의 재동기화 상태를 나타내기 위해 데이터베이스 서버 작업의 작업 로그로 메세지가 송신됩니다. 트랜잭션이 확실하지 않은 경우 로컬 자원이 확약되거나 롤백되려면 먼저 해당 트랜잭션에 대한 결정을 소유한 위치와의 재동기화가 완료되어야 합니다.

트랜잭션에 대한 결정이 내려지면 다음과 같은 메세지가 데이터베이스 서버 작업에 대한 작업 기록부로 송신됩니다.

CPI8351
&1 지연 변경이 롤백되는 중입니다.

CPC8355
&19/&18/&17 작업에 대한 확약 정의 &8의 IPL 이후 회복이 완료되었습니다.

CPD835F
&19/&18/&17 작업에 대한 확약 정의 &8의 IPL 회복이 실패했습니다.

회복 관련 기타 메세지도 송신될 수 있습니다. 이 메세지들은 이력(QHST) 기록부로 송신됩니다. 오류가 발생하면 QSYSOPR 메세지 대기행렬로도 메세지가 송신됩니다.

iSeries Navigator를 사용하거나 데이터베이스 서버 작업의 작업 로그를 표시하거나 WRKCMTDFN(확약 정의에 대한 작업) 명령을 사용하여 회복 진행 상황을 판별할 수 있습니다. iSeries Navigator 및 확약 정의에 대한 작업 화면을 통해서도 시스템에 의한 강제 확약 또는 롤백을 수행할 수 있지만 이것은 마지막 수단으로서만 사용해야 합니다. 트랜잭션에 참여한 모든 위치가 결국 조작으로 리턴될 것으로 예상된다면 시스템이 스스로 재동기화하도록 허용해야 합니다. 그럼으로써 데이터베이스 무결성이 보장될 수 있습니다.

트랜잭션 및 확약 제어 관리

다음은 확약 제어 관리를 위해 수행할 수 있는 타스크입니다.

확약 제어 정보 표시
여기에서는 시스템 상의 모든 트랜잭션에 대한 정보를 표시하고 그 트랜잭션과 연관된 작업에 대한 정보를 표시하기 위해 할 수 있는 타스크를 설명합니다.

확약 제어 성능 최적화
여기에서는 확약 제어가 시스템 성능에 미치는 영향을 최소화하기 위해 할 수 있는 타스크를 설명합니다.

확약 제어 정보 표시

iSeries Navigator를 사용하여 시스템의 모든 트랜잭션(논리적 작업 단위)에 대한 정보를 표시할 수 있습니다. 또는 트랜잭션과 연관된 작업이 있는 경우 그 작업에 대한 정보도 볼 수 있습니다.

주: 이 표시 조치는 SQL 어플리케이션에 대한 분리 레벨을 표시하지 않습니다.

정보를 표시하려면 다음과 같이 하십시오.

1. **iSeries Navigator** 창에서 사용하려는 서버를 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업하려는 시스템을 확장하십시오.
4. **트랜잭션**을 확장하십시오.

   주: X/Open 글로벌 트랜잭션과 연관된 트랜잭션을 보려면 **글로벌 트랜잭션**을 확장하십시오. DB2 UDB 관리 트랜잭션을 보려면 **데이터베이스 트랜잭션**을 확장하십시오.
5. **글로벌 트랜잭션**또는 **데이터베이스 트랜잭션**을 확장하십시오.

이 화면에서는 다음을 보여줍니다.

- 작업 단위 ID
- 작업 단위 상태
- 작업
- 사용자
- 번호
- 진행 중인 재동기화
- 확약 정의

온라인 도움말에서 각 화면의 모든 상태 표시 및 필드에 대한 정보를 제공합니다.

또한 iSeries Navigator를 사용하여 다음과 같은 정보를 표시합니다.

- 트랜잭션에 대해 잠긴 오브젝트 표시
- 트랜잭션과 연관된 작업 표시
- 트랜잭션의 자원 상태 표시
- 트랜잭션 등록 정보 표시

트랜잭션에 대해 잠긴 오브젝트 표시

트랜잭션 범위의 잠금을 실행하는 글로벌 트랜잭션에 대해서만 잠긴 오브젝트를 표시할 수 있습니다.

트랜잭션에 대해 잠긴 오브젝트를 표시하려면 다음과 같이 하십시오.

1. **iSeries Navigator** 창에서 사용하려는 서버를 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업하려는 시스템을 확장하십시오.
4. **트랜잭션**을 확장하십시오.
5. **글로벌 트랜잭션**을 확장하십시오.
6. 작업하려는 트랜잭션을 마우스 오른쪽 버튼으로 클릭하고 **잠긴 오브젝트**를 선택하십시오.

트랜잭션과 연관된 작업 표시

트랜잭션과 연관된 작업을 표시하려면 다음과 같이 하십시오.

1. **iSeries Navigator** 창에서 사용하려는 서버를 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업하려는 시스템을 확장하십시오.
4. **트랜잭션**을 확장하십시오.
5. **글로벌 트랜잭션** 또는 **데이터베이스 트랜잭션**을 확장하십시오.
6. 작업하려는 트랜잭션을 마우스 오른쪽 버튼으로 클릭하고 **작업**을 선택하십시오.

작업 범위의 잠금을 실행하는 데이터베이스 트랜잭션 및 글로벌 트랜잭션의 경우 그 트랜잭션과 연관된 작업 리스트가 표시됩니다.

트랜잭션 범위의 잠금을 실행하는 글로벌 트랜잭션의 경우 이 트랜잭션 오브젝트가 첨부된 작업 또는 이 트랜잭션 오브젝트가 첨부되기를 기다리는 작업의 리스트가 표시됩니다.

트랜잭션의 자원 상태 표시

트랜잭션의 자원 상태를 표시하려면 다음과 같이 하십시오.

1. **iSeries Navigator** 창에서 사용하려는 서버를 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업하려는 시스템을 확장하십시오.

4. **트랜잭션**을 확장하십시오.
5. **글로벌 트랜잭션** 또는 **데이터베이스 트랜잭션**을 확장하십시오.
6. 작업하려는 트랜잭션을 마우스 오른쪽 버튼으로 클릭하고 **자원 상태**를 선택하십시오.

트랜잭션 등록 정보 표시

트랜잭션 등록 정보를 표시하려면 다음과 같이 하십시오.

1. **iSeries Navigator** 창에서 사용하려는 서버를 확장하십시오.
2. **데이터베이스**를 확장하십시오.
3. 작업하려는 시스템을 확장하십시오.
4. **트랜잭션**을 확장하십시오.
5. **글로벌 트랜잭션** 또는 **데이터베이스 트랜잭션**을 확장하십시오.
6. 작업하려는 트랜잭션을 마우스 오른쪽 버튼으로 클릭하고 **등록 정보**를 선택하십시오.

확약 제어 성능 최적화

확약 제어를 사용하는 경우 시스템 성능에 영향을 미칠 수 있는 자원이 필요합니다. 확약 제어와 관련하여 몇 가지 요소가 시스템 성능에 영향을 미칩니다. 다음은 성능에 영향을 미치지 않는 요소, 성능을 저하시키는 요소 및 성능을 향상시키는 요소입니다.

성능에 영향을 미치지 않는 요소

파일 열기

확약 열기 옵션을 지정하지 않고 파일을 여는 경우 확약 정의가 시작되었다고 하더라도 추가적인 시스템 자원이 사용되지 않습니다. 확약 열기 옵션을 지정하는 것에 대한 자세한 내용은 적절한 고급 언어 참조 매뉴얼을 참조하십시오.

성능을 저하시키는 요소

저널링

파일을 저널링하려면 시스템 자원이 필요합니다. 그러나 대부분의 경우 저널링은 확약 제어 없이 사용하는 것보다 확약 제어와 함께 사용하는 것이 더 뛰어난 성능을 발휘합니다. 사후 이미지만을 지정하는 경우 확약 제어는 확약 제어가 유효한 동안 이것을 사전 이미지와 사후 이미지 모두로 변경합니다. 일반적으로 이것은 성능이라기보다는 공간과 관련된 고려사항입니다. 저널링에 대한 자세한 내용은 저널 관리를 참조하십시오.

확약 조작

트랜잭션 중에 저널된 자원이 변경되는 경우 각 트랜잭션 확약 시 이들 자원과 관련된 각 저널에 두 개의 항목을 추가합니다. 추가된 항목의 수는 작은 트랜잭션의 수를 크게 증가시킬 수 있습니다. 저널 리시버를 저널과 별도의 디스크 풀(pool)에 두고자 할 수 있습니다.

롤백 조작

확약 제어는 데이터베이스에 기록되어 있는 지연 변경을 롤백해야 하므로 롤백이 발생할 때마다 추가적인 시스템 자원이 필요합니다. 또한 레코드 변경이 지연된 경우 롤백 조작을 수행하면 저널에 항목이 추가됩니다.

STRCMTCTL(확약 제어 시작) 및 ENDCMTCTL(확약 제어 종료) 명령

STRCMTCTL 및 ENDCMTCTL 명령을 사용하면 확약 정의가 시작되고 종료될 때마다 시스템에 의해 추가적인 오버헤드가 발생합니다. 각 트랜잭션에 STRCMTCTL 및 ENDCMTCTL 명령을 사용하는 것은 피하도록 하십시오. 필요할 때만 사용하십시오. 대화식 작업이 시작될 때 확약 정의를 설정할 수 있고 그 작업이 지속되는 동안 사용하십시오.

확약 제어 트랜잭션에 둘 이상의 저널 사용

2단계 확약을 사용하여 확약 제어 하에 열려 있는 파일이 두 개 이상의 저널을 저널할 수 있습니다. 그러나 둘 이상의 저널을 사용하면 확약 정의를 관리하기 위하여 추가적인 시스템 자원이 사용됩니다. 또한 둘 이상의 저널을 사용하면 회복이 복잡해집니다.

레코드 잠금

레코드 잠금은 다른 어플리케이션에 영향을 미칠 수 있습니다. 특정 작업 내에서 잠긴 레코드의 수는 그 작업에 사용되는 전반적인 시스템 자원을 증가시킵니다. 동일한 레코드에 액세스해야 하는 어플리케이션은 그 트랜잭션이 종료할 때까지 기다려야 합니다.

SEQONLY(*YES) 요구

(OVRDBF 명령을 사용하여) 사용자가 SEQONLY(*YES) 옵션을 요구하거나 (어플리케이션 프로그램에서 내재적으로 SEQONLY(*YES) 옵션을 사용하려고 하고) 파일이 LCKLVL(*ALL)의 확약 제어 하에서 입력용으로만 열리는 경우 옵션은 SEQONLY(*NO)로 변경됩니다. 이 옵션은 레코드가 블록되지 않으므로 입력 파일의 성능에 영향을 미칠 수 있습니다.

활동 중 보관 처리가 사용 중일 때 데이터베이스에 대한 레코드 레벨 변경 요구

확약 정의가 확약 경계에 있고 활동 중 보관 조작이 다른 작업에서 실행 중인 경우 확약 제어 하에서 데이터베이스 파일에 대한 레코드 레벨의 변경을 수행하는 것이 지연될 수도 있습니다. 이것은 보관 요구에 포함된 일부 오브젝트와 동일한 저널에 파일이 저널될 때 발생할 수 있습니다.

주: WRKACTJOB(활동 작업에 대한 작업) 명령에서 이 상태 열에는 활동 중 보관 체크 포인트로 인해 작업이 보류 중일 때 CMTW(확약 대기)를 표시합니다.

활동 중 보관 처리가 사용 중일 때 변경에 대한 확약 또는 롤백

다른 작업에서 활동 중 보관 조작이 실행되고 있으면 확약 또는 롤백 조작이 확약 경계에서 지연될 수 있습니다. API 자원이 QTNADDCR API를 사용하여 추가되지 않았고 *정상적인 보관 처리 허용* 필드 값이 Y 가 아닌 경우 API 확약 자원이 이전에 확약 정의에 추가되었을 때 일어날 수 있습니다.

작업이 확약 또는 롤백 요구 동안 보류되고 확약 또는 롤백 요구가 한 번에 하나의 확약 정의에 대해서만 수행될 수 있기 때문에 API 자원이 추가된 둘 이상의 확약 정의를 갖는 작업에서는 항상 활동 중 보관 조작이 완료되지 못합니다.

활동 중 보관 처리가 사용 중일 때 오브젝트 레벨 변경 요구

확약 정의가 확약 경계에 있고 활동 중 보관 조작이 다른 작업에서 실행 중인 경우 확약 제어 하에서 오브젝트 레벨의 변경을 수행하는 것이 지연될 수도 있습니다. 이것은 그 오브젝트가 들어 있는 라이브러리에 대하여 활동 중 보관 조작이 실행되는 동안 오브젝트 레벨 변경이 이루어질 때 발생할 수 있습니다. 예를 들어 활동 중 보관 조작이 MYSQLLIB 라이브러리에 대하여 실행될 때 MYSQLLIB 라이브러리의 MYTBL 테이블에 대한 확약 제어 하의 SQL 테이블 작성 조작이 지연될 수 있습니다.

주: 대기 시간이 60초를 초과하면 사용자에게 계속 기다릴 것인지 조작을 취소할 것인지를 묻는 조회 메세지 CPA8351이 보내집니다.

QTNADDCR API를 사용하여 API 자원 추가

모든 확약 정의가 확약 경계에 있고 활동 중 보관 조작이 다른 작업에서 실행 중인 경우 QTNADDCR API를 사용하여 API 확약 자원을 추가하려는 요구는 지연될 수도 있습니다.

주:

1. 대기 시간이 60초를 초과하면 사용자에게 계속 기다릴 것인지 조작을 취소할 것인지를 묻는 조회 메세지 CPA8351이 보내집니다.
2. 이는 *정상적인 보관 처리 허용* 필드 값이 Y인 경우 QTNADDCR API를 사용하여 추가된 API 자원에는 적용되지 않습니다.

성능을 향상시키는 요소

디폴트 저널 사용

디폴트 저널을 사용하면 확약 정의가 활동 중인 동안 확약 제어 하에서 모든 파일을 닫고 다시 여는 경우 성능 향상에 도움이 될 수 있습니다. 그러나 OMTJRNE(*NONE)을 지정하여 디폴트 저널을 사용하는 경우 확약 및 롤백 조작 성능을 저하시킵니다.

최종 에이전트 선택

확약 조작 동안에 최종 에이전트와 시스템 사이에 상호작용이 거의 필요 없으므로 최종 에이전트를 선택하면 성능이 개선됩니다. 그러나 확약 조작 중에 통신 장애가 발생하면 출력 대기 옵션 값에 상관없이 재동기화가 완료될 때까지 확약 조작이 완료되지 않습니다. 이러한 장애는 드문 일이지만 이 옵션을 통해 어플리케이션 작성자는 장애가 발생할 때 사용자가 재동기화가 완료될 때까지 한 없이 기다릴 수도 있는 부정적인 상황을 고려해 볼 수 있습니다. 성공적인 확약 조작 동안의 최종 에이전트 최적화에 의해 제공되는 성능 향상 측면에서 중요한 고려사항입니다. 이 고려사항은 일반적으로 일괄처리 작업에서보다는 대화식 작업에서 더 중요합니다.

디폴트는 시스템에 의해 선택되도록 허용된 최종 에이전트지만 QTNCHGCO API를 사용하여 사용자는 이 값을 수정할 수 있습니다.

출력 대기 옵션 사용하지 않음

리모트 자원이 확약 제어 하에 있는 경우 성능은 출력 대기 옵션이 N(아니오)으로 설정되고 모든 리모트 시스템이 추정 중단을 지원할 때 개선됩니다. 출력 대기 옵션은 첫 번째 연결이 리모트 시스템으로 연결

될 때 DRDA 및 DDM 어플리케이션에 대해 시스템에 의해 설정됩니다. APPC 어플리케이션은 명시적으로 출력 대기 옵션을 설정하거나 디폴트 값 Y를 사용해야 합니다.

생략 확인 옵션 선택
생략 확인 옵션이 선택되면 성능이 개선됩니다. 이 옵션에 대한 자세한 정보는 2단계 확약의 확약 정의: 생략 확인 표시를 참조하십시오.

읽기 전용 인가 옵션 선택
읽기 전용 인가 옵션을 선택하면 성능이 개선됩니다. 이 옵션에 대한 자세한 정보는 2단계 확약에 대한 확약 정의: 읽기 전용 인가 허용을 참조하십시오.

성능 향상을 위해 할 수 있는 타스크는 다음과 같습니다.

- 잠금 최소화
- 트랜잭션 크기 관리

잠금 최소화

레코드 잠금을 최소화하는 일반적인 방법은 레코드 잠금을 해제하는 것입니다. (LCKLVL(*ALL)이 지정되어 있는 경우 이 방법은 작동하지 않습니다.) 예를 들어 하나의 파일 유지보수 어플리케이션은 일반적으로 다음과 같은 작업을 수행합니다.

- 변경될 레코드 ID를 입력하도록 프롬트를 표시합니다.
- 요구된 레코드를 검색합니다.
- 레코드를 표시합니다.
- 워크스테이션 사용자가 변경하도록 합니다.
- 레코드를 갱신합니다.

대부분의 경우 레코드는 갱신으로 인해 요구된 레코드의 액세스가 잠기게 됩니다. 그 레코드를 기다리는 다른 작업의 경우 레코드 대기 시간을 초과하는 경우도 있습니다. 워크스테이션 사용자가 변경을 고려하는 동안 레코드 잠금이 발생하지 않도록 하려면 데이터베이스에서 검색된 후(레코드 표시 화면이 표시되기 전에) 레코드를 해제하십시오. 그러면 갱신하기 전에 다시 레코드에 액세스해야 합니다. 해제되었다가 다시 액세스 되는 사이에 레코드가 변경되는 경우 워크스테이션 사용자에게 알려야 합니다. 프로그램은 원래 레코드의 필드를 하나 이상 저장하여 검색된 후 동일한 레코드의 필드와 비교함으로써 다음과 같이 레코드 변경 여부를 판별할 수 있습니다.

- 레코드에서 갱신 계수 필드를 사용하고 갱신 직전에 이 필드에 1을 추가합니다. 프로그램은 원래 값을 저장하고 이 값을 레코드가 다시 검색되면 필드의 값과 비교합니다. 변경이 발생하면 워크스테이션 사용자에게 통지하고 레코드가 다시 표시됩니다. 갱신 계수 필드는 갱신이 발생한 경우에만 변경됩니다. 워크스테이션 사용자가 변경을 고려하는 동안 레코드는 해제됩니다. 이 방법은 파일을 갱신하는 모든 프로그램에서 사용해야 합니다.
- 전체 자료 레코드의 내용을 저장하고 이를 다음번 검색 시 레코드와 비교합니다.

위의 두 가지 경우 모두에서 동일한 필드명이 마스터 레코드와 화면 파일에 사용되는 RPG의 외부 서술 자료의 사용을 피했습니다. 레코드가 다시 검색될 때 워크스테이션 사용자가 변경한 내용이 오버레이되므로(RPG에서는) 동일한 필드명을 사용하지 않습니다.

레코드 자료를 자료 구조로 이동시켜서 이 문제를 해결하거나 DDS 키워드 RTNDTA를 사용하는 경우 외부 서술 자료를 계속 사용할 수 있습니다. RTNDTA 키워드를 사용하여 오퍼레이팅 시스템이 자료를 화면에서 프로그램으로 이동시키지 않고 화면에서 자료를 다시 읽을 수 있습니다. 이를 통해 프로그램은 다음을 수행할 수 있습니다.

1. 레코드 ID를 입력하도록 프롬트합니다.
2. 데이터베이스에서 요구된 레코드를 검색합니다.
3. 레코드를 해제합니다.
4. 레코드 변경 여부를 판별하는 데 사용될 필드를 저장합니다.
5. 레코드를 표시하고 워크스테이션 사용자가 응답하기를 기다립니다.

화면에서 워크스테이션 사용자가 레코드를 변경하면 프로그램은 다음과 같은 순서의 작업을 수행합니다.

1. 다시 데이터베이스에서 레코드를 검색합니다.
2. 저장된 필드를 비교하여 데이터베이스 레코드가 변경되었는지 판별합니다. 변경되었다면 프로그램은 레코드를 해제하고 레코드가 나타날 때 메세지를 보냅니다.
3. RTNDTA 키워드를 사용하여 읽기 조작을 실행함으로써 화면에서 레코드를 검색하고 데이터베이스 레코드의 레코드를 갱신합니다.
4. 워크스테이션 사용자가 요구를 취소하면 더 이상 해제할 추가 레코드가 없으므로 다음 논리 프롬트로 진행합니다.

LCKLVL(*CHG) 및 LCKLVL(*CS)가 이 상황에서 사용됩니다. LCKLVL(*ALL)이 사용되면 확약 또는 롤백 조작을 사용하여 레코드 잠금을 해제해야 합니다.

잠금에 대한 자세한 내용은 교착 상태 감지를 참조하십시오.

트랜잭션 크기 관리

이 논의에서 트랜잭션은 대화식을 의미합니다. (확약 제어는 일괄처리 어플리케이션에 대해서도 사용될 수 있는데, 이는 종종 일련의 트랜잭션으로 간주됩니다. 동일한 다수의 고려사항이 일괄처리 어플리케이션에도 적용되며 이는 일괄처리 어플리케이션의 확약 제어에서 논의됩니다.)

트랜잭션과 연관된 각 저널에 대하여 최대 500 000 000개의 레코드를 잠글 수 있습니다. QAQQINI(옵션 파일 조회)를 사용하여 이 한계를 줄일 수 있습니다. CHGQRYA(조회 속성 변경) 명령의 QRYOPTLIB 매개변수를 사용하여 사용할 작업에 대한 옵션 파일 조회를 지정하십시오. 옵션 파일 조회의 COMMITMENT_CONTROL_LOCK_LEVEL 값을 작업에 대한 잠금 한계로 사용하십시오.

레코드에 대한 잠금 레벨을 선택할 때 트랜잭션 *크기*를 고려하십시오. 트랜잭션이 종료하기 전에 얼마나 오래 레코드를 잠그는지는 *크기*로 결정해야 합니다. 확약 제어에 대한 확약 또는 롤백 조작이 Enter 키를 한 번 사용하는 것으로 제한되는지 또는 트랜잭션이 Enter 키를 여러번 사용하는 것으로 구성될 것인지를 결정해야 합니다.

주: 트랜잭션이 짧을수록 활동 중 보관 체크 포인트 처리를 시작하기 위해 기다리는 작업이 보다 일찍 계속하고 완료할 수 있습니다.

예를 들어 주문 입력 어플리케이션의 경우 고객은 한 번의 주문으로 여러 품목을 주문할 수 있고 이는 주문된 모든 품목에 대하여 주문 상세 레코드 및 재고 마스터 레코드 갱신을 필요로 합니다. 트랜잭션이 전체 주문으로 정의되고 Enter 키를 매번 사용하는 것이 품목을 주문하는 것으로 간주되면 주문의 모든 레코드는 전체 주문 시간 동안 잠금 상태가 됩니다. 따라서 자주 사용되는 레코드(재고 마스터 레코드 등)는 오랜 시간 동안 잠금 상태에 있을 수 있고 다른 작업의 진행을 막을 수 있습니다. 서브 파일을 사용하여 한 번의 Enter 키로 모든 품목이 입력되는 경우 전체 주문에 대한 잠금 지속 시간은 최소화됩니다.

일반적으로 잠금의 횟수와 지속 시간이 최소화되어야 여러 워크스테이션 사용자가 오래 기다리지 않고 동일한 자료에 액세스할 수 있습니다. 사용자가 화면에서 자료를 입력하는 동안 잠금을 실행하지 않음으로써 이것이 가능해집니다. 일부 어플리케이션에서는 둘 이상의 워크스테이션 사용자가 동일한 자료에 액세스하는 일이 필요 없을 수도 있습니다. 예를 들어 고객 한 사람당 열려 있는 품목 레코드 수가 많은 현금 송금 어플리케이션에서 일반적인 방법은 모든 레코드를 잠그고 워크스테이션 사용자가 현금 송금을 완료하고 영수증을 받을 때까지 지연시키는 것입니다.

워크스테이션 사용자가 트랜잭션에 대하여 Enter 키를 여러번 누르면 트랜잭션을 여러 세그먼트로 수행할 수 있습니다. 예를 들면 다음과 같습니다.

- 첫 번째 세그먼트는 워크스테이션 사용자가 정보를 요구하는 조회입니다.
- 두 번째 세그먼트는 전체 트랜잭션을 완료하기 위해 워크스테이션 사용자의 목적을 확인하는 것입니다.
- 세 번째 세그먼트는 영향을 받는 검색 및 갱신입니다.

이 접근 방식을 통해 레코드 잠금이 한 번의 Enter 키 사용으로 제한됩니다.

이러한 조회 우선 접근방식은 표시된 정보로부터 결정을 내려야 하는 어플리케이션에 주로 사용됩니다. 예를 들어 비행기 예약 어플리케이션에서 고객은 어떤 항공편을 이용할 것인지 결정하기 전에 시간, 연결 항공편 및 좌석 배치 등을 확인하고자 할 수 있습니다. 일단 고객이 결정을 하면 트랜잭션이 입력됩니다. 트랜잭션이 실패하면(항공편에 자리가 없음) 롤백 기능이 사용되고 다른 요구가 입력됩니다. 결정을 내릴 때까지 첫 번째 조회에서 레코드가 잠겨 있다면 다른 예약 업무 담당자는 해당 트랜잭션이 끝날 때까지 기다려야 할 것입니다.

시나리오 및 예제: 확약 제어

다음은 확약 제어의 시나리오 및 예제입니다. 이 시나리오에서는 JKL Toy Company에서 로컬 데이터베이스의 트랜잭션을 추적하기 위해 확약 제어를 구현한 방식을 보여줍니다.

예제는 확약 제어에 대한 샘플 코드를 제공합니다. 연습 문제는 RPG 프로그램으로 확약 제어를 구현한 것입니다. 여기에는 각 단계에서 발생하는 것을 보여주는 논리 흐름이 포함됩니다.

다음의 세 가지 예에서는 시스템의 비정상 종료 후 어플리케이션을 다시 시작하기 위해 확약 제어를 사용하는 예를 보여줍니다.

시나리오

- 시나리오: 확약 제어

예

- 확약 제어의 연습 문제
- 확약 제어의 연습 문제에 대한 논리 흐름
- 예: 트랜잭션 기록 파일을 사용하여 어플리케이션 시작
- 예: 통지 오브젝트를 사용하여 어플리케이션 시작
- 예: 표준 처리 프로그램을 사용하여 어플리케이션 시작

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

시나리오: 확약 제어

JKL Toy Company에서는 확약 제어를 사용하여 제조 및 재고용 데이터베이스 레코드를 보호합니다. 이 시나리오에서는 JKL Toy Company에서 부품이 재고 관리 부서에서 제조 부서로 전송될 때 확약 제어를 사용하는 방법을 보여줍니다.

JKL Toy Company의 네트워크 환경에 대한 설명은 시나리오: 저널 관리를 참조하십시오. 다음의 시나리오는 이 회사의 생산 서버인 JKLPROD에서 확약 제어가 작동하는 방식을 보여줍니다.

이 시나리오는 두 가지 예에서 확약 제어 사용 시 장점을 보여줍니다. 첫 번째 예에서는 이 회사의 재고 관리 프로그램인 Program A가 확약 제어를 사용하지 않고 실행되는 경우와 이 때 발생할 수 있는 문제점을 보여줍니다. 두 번째 예에서는 이 프로그램이 확약 제어를 사용하여 실행되는 경우를 보여줍니다.

JKL Toy Company는 재고 관리 어플리케이션 프로그램인 Program A를 서버 JKLPROD에서 사용합니다. Program A는 두 개의 레코드를 사용합니다. 한 레코드는 창고에 보관되어 있는 항목을 추적합니다. 다른 레코드는 창고에서 나와 생산에 사용되는 항목을 추적합니다.

확약 제어를 하지 않는 Program A

다음 어플리케이션 프로그램에서는 확약 제어를 사용하지 않는다고 가정합니다. 시스템은 갱신을 위해 읽은 레코드에 잠금을 수행합니다. 다음 단계에서는 응용프로그램이 다이오드가 창고에서 나와 수표 계정으로 전송될 때까지 추적하는 방법에 대해 설명합니다.

- Program A는 창고 레코드를 잠그고 검색합니다. (다른 프로그램에 의해 레코드가 잠겨 있는 경우 잠시 기다려야 합니다.)

- Program A는 생산 레코드를 잠그고 검색합니다. (역시 잠시 기다려야 할 수 있습니다.) Program A에서 두 가지 레코드에 잠금을 수행했으므로 다른 프로그램이 변경할 수 없습니다.
- Program A는 창고 레코드를 갱신합니다. 그러면 다른 프로그램에서 갱신을 위해 읽을 수 있도록 레코드가 해제됩니다.
- Program A는 생산 레코드를 갱신하고 그러면 다른 프로그램에서 갱신을 위해 읽을 수 있도록 레코드가 해제됩니다.

확약 제어를 사용하지 않는 경우 이 프로그램이 모든 상황에서 올바르게 작동하도록 문제를 해결해야 합니다. 예를 들어 작업 또는 시스템 상의 장애로 인해 Program A에서 두 가지 레코드를 모두 갱신하지 않는 경우 문제가 발생합니다. 이 경우, 두 파일이 일치하지 않습니다. 즉 다이오드가 창고 레코드에서 제거되었지만 생산 레코드에 추가되지 않았습니다. 확약 제어를 사용하면 트랜잭션에서 일어난 변경이 모두 완료되거나 또는 트랜잭션 처리가 인터럽트되는 경우 파일들이 원래의 상태로 돌아갑니다.

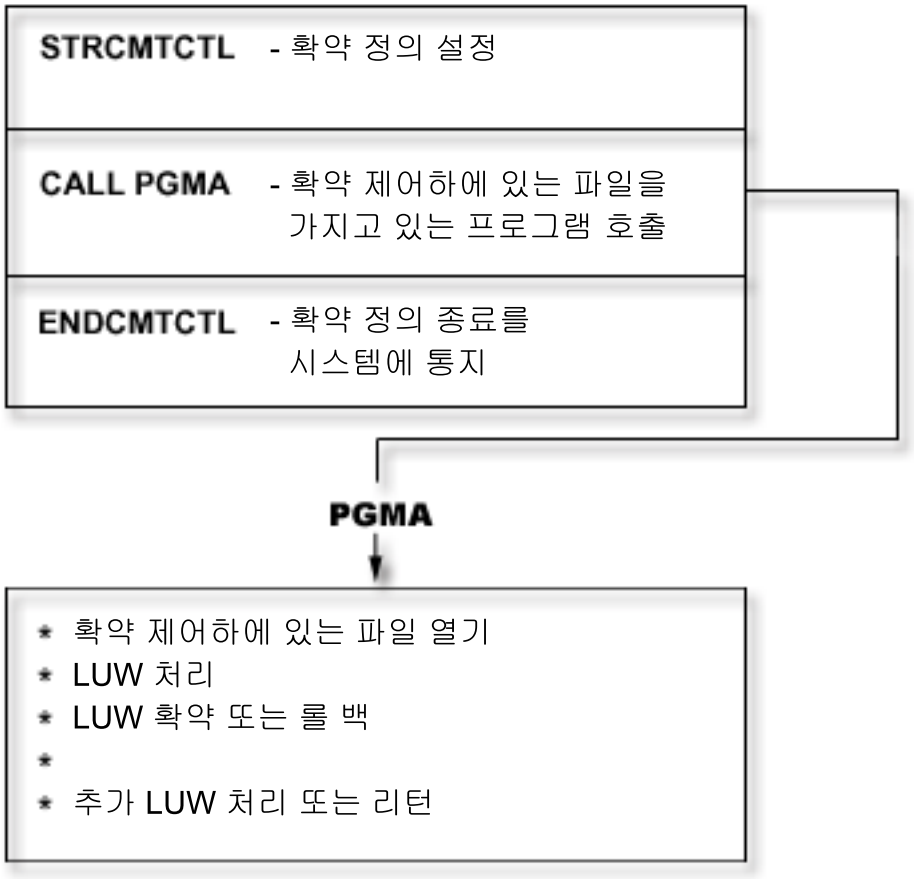
확약 제어를 사용하는 Program A

확약 제어를 사용하는 경우 앞의 예는 다음과 같이 바뀝니다.

1. 확약 제어가 시작됩니다.
2. Program A는 창고 레코드를 잠그고 검색합니다. (다른 프로그램에 의해 레코드가 잠겨 있는 경우 잠시 기다려야 합니다.)
3. Program A는 생산 레코드를 잠그고 검색합니다. (역시 잠시 기다려야 할 수 있습니다.) Program A에서 두 가지 레코드에 잠금을 수행했으므로 다른 프로그램이 변경할 수 없습니다.
4. Program A는 창고 레코드를 갱신하고 확약 제어는 계속 레코드를 잠금 상태에 둡니다.
5. Program A는 생산 레코드를 갱신하고 확약 제어는 계속 레코드를 잠금 상태에 둡니다.
6. Program A는 트랜잭션을 확약합니다. 창고 레코드 및 생산 레코드는 파일에서 영구적으로 변경됩니다. 변경은 저널에 기록되고 이는 디스크에 나타날 것입니다. 확약 제어는 두 가지 레코드의 잠금을 해제합니다. 이제 다른 프로그램에서 갱신을 위해 읽을 수 있도록 레코드가 해제됩니다.

트랜잭션이 확약될 때까지 확약 제어에 의해 두 레코드에 대한 잠금 상태가 유지되므로 하나의 레코드가 갱신되고 다른 하나는 갱신되지 않는 상황이 발생할 수 없습니다. 트랜잭션이 확약되기 전에 라우팅 단계 또는 시스템 장애가 발생하는 경우 시스템은 변경된 내용을 제거(롤백)하여 파일을 마지막 트랜잭션이 확약된 지점까지 갱신되도록 합니다.

파일이 확약 제어 하에 있어야 하는 각 라우팅 단계의 경우 다음 그림에 보인 단계가 발생합니다.



확약 제어 하에서 수행되는 조작은 저널에 저널됩니다. 확약 제어 저널 시작 항목은 확약 제어 하의 첫 번째 파일 열기 항목 후에 나타납니다. 이는 첫 번째 파일 열기 항목이 확약 제어에 사용될 저널을 결정하기 때문입니다. 첫 번째 열기 조작에서 결정된 저널 항목을 사용하여 후속 열기 조작을 점검하여 모든 파일이 동일한 저널을 사용하도록 합니다.

작업 실패 또는 시스템 실패가 발생하면 확약 제어 하의 자원은 확약 경계로 갱신됩니다. 트랜잭션이 시작되었으나 라우팅 단계가 종료하기 전에 완료되지 않는 경우 시스템은 그 트랜잭션을 롤백하고 트랜잭션은 라우팅 단계가 끝난 후 파일에 표시되지 않습니다. 트랜잭션이 완료되기 전에 시스템이 비정상적으로 종료하면 시스템은 해당 트랜잭션을 롤백하고 그 뒤에 발생하는 사용권 내부 코드의 정상적인 초기 프로그램 로드(IPL) 후 파일 해당 트랜잭션이 나타나지 않습니다. 롤백이 발생하면 언제든지 반전 항목이 저널에 위치하게 됩니다.

예를 들어 JKL Company는 재고로 100개의 다이오드를 가지고 있다고 가정합니다. 제조 시 창고에서 20개를 꺼냈으므로 80개가 남았습니다. 데이터베이스가 갱신되면 사전 이미지(before-image) 100과 사후 이미지(after-image) 80이 발생합니다.

항목을 저널링한 후 그러나 확약점 또는 롤백점에 도달하기 전에 시스템이 비정상적으로 종료했다고 가정합니다. IPL 후 시스템은 저널 항목을 읽고 해당 데이터베이스 레코드를 갱신합니다. 이 갱신에서 갱신을 반전시키는 두 개의 저널 항목을 만들어냅니다. 첫 번째 항목은 사전 이미지(80)이고 두 번째 항목은 사후 이미지(100)입니다.

비정상 종료 후 IPL이 정상적으로 완료되면 시스템은 확약되지 않은 데이터베이스 변경을 제거(롤백)합니다. 앞의 예에서 확약 조작이 해당 트랜잭션에 대한 저널에 있지 않았기 때문에 시스템은 창고 레코드에서 변경을 제거합니다. 이 경우 창고 레코드의 사전 이미지는 파일에 들어갑니다. 저널에는 롤백된 변경 및 롤백 조작이 발생했다는 표시가 들어 있습니다.

확약 제어에 대한 연습 문제

이 연습 문제는 사용자가 확약 제어 및 그 요구사항을 이해하는 데 도움이 됩니다. 다음 단계는 사용자가 OS/400 사용권 프로그램 및 자료 파일 유틸리티(DFU)를 잘 알고 있으며 이 주제를 읽었음을 전제로 합니다. 논리 흐름은 사용자가 확약 제어에 대한 이 연습 프로그램을 더 깊이 이해하는 데 도움이 될 것입니다.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

이 문제를 시작하기 전에 다음을 수행하십시오.

- 이 연습 문제를 위한 별도의 라이브러리를 작성하십시오. 지시사항에서 라이브러리 이름은 CMTLIB입니다. CMTLIB를 사용자의 라이브러리 이름으로 대체하십시오.
- 소스 파일 및 작업 설명을 작성하십시오.

다음 단계를 수행하십시오.

1. ITMP(품목 마스터 파일)라는 이름의 실제 파일(PF)을 작성하십시오. 이 파일의 자료 서술 스펙(DDS)은 다음과 같습니다.

```
10    A    R ITMR
20    A      ITEM     2
30    A      ONHAND   5   0
40    A    K ITEM
```

2. TRNP(트랜잭션 파일)라는 이름의 실제 파일(PF)을 작성하십시오. 이 파일은 트랜잭션 기록부 파일로 사용됩니다. 이 파일의 DDS는 다음과 같습니다.

```
10    A    R TRNR
20    A      QTY     5  0
30    A      ITEM    2
40    A      USER    10
```

3. TRNL(트랜잭션 논리 파일)이라는 이름의 논리 파일을 작성하십시오. 이 파일은 어플리케이션 재시작을 지원하는 데 사용됩니다. *USER* 필드는 LIFO 유형입니다. 이 파일의 DDS는 다음과 같습니다.

```
10                       LIFO
20    A     R TRNR       PFILE (TRNP)
30    A     K USER
```

4. STRDFU 명령을 입력하고 ITMP 파일에 대한 ITMU라는 이름의 DFR 어플리케이션을 작성하십시오. 어플리케이션 정의 시 DFU에서 제공되는 디폴트를 그대로 사용하십시오.

5. CHGDTA ITMU를 입력하고 ITMP 파일에 다음 레코드를 입력하십시오.

품목	사용 가능 수량
AA	450
BB	375
CC	4000

6. F3을 사용하여 프로그램을 종료하십시오. 이 항목은 이 프로그램이 조작할 일부 데이터를 제공합니다.
7. 다음과 같이 CL 프로그램 ITMPCSC(품목 처리)를 작성하십시오.

```
PGM
DCL &USER *CHAR LEN(10)
RTVJOBA USER(&USER)
CALL ITMPCS PARM(&USER)
ENDPGM
```

이것은 ITMPCS 프로그램을 호출하는 제어 프로그램입니다. 이것은 사용자명을 검색하여 처리 프로그램으로 전달합니다. 이 어플리케이션은 고유한 사용자명이 사용된다고 가정합니다.

8. 다음과 같이 DDS에서 ITMPCSD라는 이름의 화면 파일을 작성하십시오.

형식에는 두 가지가 있는데 첫 번째는 기본적인 프롬트 화면이고 두 번째는 오퍼레이터가 입력된 마지막 트랜잭션을 검토하는 것입니다. 이 화면 파일은 IMPC 프로그램이 사용합니다.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A          R PROMPT
  2.00      A                                       CA03(93 'End of program')
  3.00      A                                       CA04(94 'Review last')
  4.00      A                                       SETOFF(64 'No rcd to rvw')
  5.00      A                                    1  2'INVENTORY TRANSACTIONS'
  6.00      A                                    3  2'Quantity'
  7.00      A            QTY            5  0I    +1
  8.00      A  61                                   ERRMSG('Invalid +
  9.00      A                                       quantity' 61)
 10.00      A                                   +5'ITEM'
 11.00      A            ITEM           2    I   +1
 12.00      A  62                                   ERRMSG('Invalid +
 13.00      A                                       Item number' 62)
 14.00      A  63                                   ERRMSG('Rollback +
 15.00      A                                       occurred' 63)
 16.00      A  64                               24  2'CF4 was pressed and +
 17.00      A                                       there are no +
 18.00      A                                       transactions for +
 19.00      A                                       this user'
 20.00      A                                       DSPATR(HI)
 21.00      A                                   23  2'CF4 Review last +
 22.00      A                                       transaction'
 23.00      A          R REVW
 24.00      A                                    1  2'INVENTORY TRANSACTIONS'
 25.00      A                                   +5'REVIEW LAST TRANSACTION'
 26.00      A                                    3  2'Quantity'
 27.00      A            QTY            5  0    +1EDTCDE(Z)
 28.00      A                                   +5'Item'
 29.00      A            ITEM           2       +1
```

9. 확약 제어를 위한 연습 프로그램의 논리 흐름에 나와 있는 논리 흐름을 살펴 보십시오.

10. STRSEU 명령을 입력한 후 다음과 같이 소스를 입력하십시오.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 1.00      FITMP    UF  E           K        DISK
 2.00      F*                                          KCOMIT
 3.00      FTRNP    O   E                    DISK
 4.00      F*                                          KCOMIT
 5.00      FTRNL    IF  E           K        DISK
 6.00      F            TRNR                           KRENAMETRNR1
 7.00      FITMPCSD  CF E                    WORKSTN
 8.00      C* Enter parameter with User name for -TRNP- file
 9.00      C           *ENTRY    PLIST
10.00      C                     PARM           USER   10
11.00      C           LOOP      TAG
12.00      C                     EXFMTPROMPT
13.00      C* Check for CF3 for end of program
14.00      C   93                DO                              End of Pgm
15.00      C                     SETON                      LR
16.00      C                     RETRN
17.00      C                     END
18.00      C* Check for CF4 for review last transaction
19.00      C   94                DO                              Review last
20.00      C* Check for existence of a record for this user in -TRNL- file
21.00      C           USER      CHAINTRNR1                 64    Not found
22.00      C   64                GOTO LOOP
23.00      C                     EXFMTREVW
24.00      C                     GOTO LOOP
25.00      C                     END
26.00      C* Access Item record
27.00      C           ITEM      CHAINITMR                  62    Not found
28.00      C* Handle -not found- Condition
29.00      C   62                GOTO LOOP
30.00      C* Does sufficient quantity exist
31.00      C           ONHAND    SUB  QTY       TEST    50   61  Minus
32.00      C* Handle insufficient quantity
33.00      C    61               DO
34.00      C* Release Item record which was locked by the CHAIN for update
35.00      C                     EXCPTRLSITM
36.00      C                     GOTO LOOP
37.00      C                     END
38.00      C* Change ONHAND and update the Item record
39.00      C                     Z-ADDTEST      ONHAND
40.00      C                     UPDATITMR
41.00      C* Test for Special Simulation Conditions
42.00      C           ITEM      IFEQ 'CC'
43.00      C*      Simulate program need for rollback
44.00      C           QTY       IFEQ 100
45.00      C                     SETON                      63    Simult Rlbck
46.00      C*                    ROLBK
47.00      C                     GOTO LOOP
48.00      C                     END
49.00      C*      Simulate an abnormal program cancellation by Div by zero
50.00      C*        Operator Should respond -C- to inquiry message
51.00      C           QTY       IFEQ 101
52.00      C                     Z-ADD0         ZERO    30
53.00      C           TESTZ     DIV  ZERO      TESTZ   30       Msg occurs
54.00      C                     END
55.00      C*      Simulate an abnormal job cancellation by DSPLY.
```

```
56.00      C*        Operator Should System Request to another job
57.00      C*         and cancel this one with OPTION(*IMMED)
58.00      C           QTY       IFEQ 102
59.00      C           'CC=102'  DSPLY                           Msg occurs
60.00      C                     END
61 00      C                     END                           ITEM=CC
62.00      C* Write the -TRNP- file
63 00      C                     WRITETRNR
64.00      C* Commit the update to -ITMP- and write to -TRNP-
65.00      C*                    COMIT
66.00      C                     GOTO LOOP
67.00      OITMR    E                RLSITM
```

11. CRTRPGPGM 명령을 입력하여 이전 단계에서 입력한 소스로부터 ITMPCS를 작성하십시오.
12. CALL ITMPCSC 명령을 입력한 후 Enter를 누르고 F4를 누르십시오. 이 오퍼레이터에 대한 항목이 없다는 메세지가 표시되어야 합니다.
13. 다음 데이터를 입력하여 프로그램이 올바르게 작동하는지 살펴 보십시오.

수량	품목
3	AA
4	BB

14. F4를 누르십시오. 마지막으로 입력된 BB 품목과 함께 검토 화면이 표시되어야 합니다. 다음 자료를 입력하십시오.

수량	품목
5	FF(유효하지 않은 품목 번호 메세지가 발생해야 합니다.)
9000	BB(수량 부족 오류 메세지가 발생해야 합니다.)
100	CC(롤백 메세지가 발생해야 합니다.)
102	CC(RPG DSPLY 조작이 발생해야 합니다. Enter키를 누르십시오.)
101	CC(INQMSGRPY 작업 속성 설정에 따라 프로그램은 0으로 나누기 조건이 발생했거나 종료했음을 나타내는 조회 메세지를 표시해야 합니다. 조회 메세지가 표시되면 C를 눌러 RPG 프로그램을 취소한 후 후속 조회에서 CL 프로그램을 취소하십시오. 그러면 예기치 않은 오류 조건이 시뮬레이트됩니다.)

15. 자료 표시 명령인 DSPDTA ITMP를 입력하십시오.

    레코드 AA 및 BB가 올바르게 갱신되었는지 확인하십시오. 값은 AA = 447, BB = 371 및 CC = 3697 이어야 합니다. 주의할 점은 CC에서 수량이 빠졌지만 트랜잭션 레코드는 기록되지 않았다는 점입니다.
16. 확약 제어를 위한 저널 리시버를 작성하십시오. CRTJRNRCV(저널 리시버 작성) 명령을 사용하여 CMRLIB 라이브러리에 RCVR1이라는 저널 리시버를 작성하십시오. 최소 5000KB 이상의 임계값을 지정하십시오. 너무 자주 발생하는 저널 변경이 성능에 미치는 영향을 최소화하기 위해 새로운 저널 리시버 생성 간격 시간을 최대화할 수 있을만큼 시스템 공간이 충분하다면 임계값은 클수록 좋습니다.
17. 확약 제어를 위한 저널을 작성하십시오. 저널 작성(CRTJRN) 명령을 사용하여 CMTLIB 라이브러리에 JRNTEST라는 저널을 작성하십시오. 이 저널은 확약 제어용으로만 사용되므로 MNGRCV(*SYSTEM) DLTRCV(*YES)라고 지정하십시오. JRNRCV 매개변수에는 16(80See page)단계에서 작성한 저널 리시버를 지정하십시오.
18. STRJRNPF(실제 파일 저널 시작) 명령에 FILE(CMTLIB/ITMP CMTLIB/TRNP) JRN(CMTLIB/JRNTEST) 매개변수를 지정하여 확약 제어에 사용될 파일을 저널하십시오.

IMAGES 매개변수에는 디폴트 값인 *AFTER을 사용하는데 이것은 저널에 레코드의 사후 이미지만 나타남을 의미합니다. ITMP 및 TRNP 파일이 이제 저널링을 시작합니다.

일반적으로 사용자는 저널링을 시작한 후 파일을 저장할 것입니다. 저널 항목과 동일한 JID를 갖지 않는 복원된 파일에는 저널된 변경 사항을 적용할 수 없습니다. 이 연습 문제에서는 저널된 변경 사항을 적용할 필요가 없으므로 저널된 파일을 저장하는 것은 생략할 수 있습니다.

19. CALL ITMPCSC 명령을 입력하고 다음 트랜잭션을 입력하십시오.

수량	품목
5	AA
6	BB

F3을 눌러 프로그램을 종료하십시오.

20. 저널 표시 명령을 입력하십시오: DSPJRN CMTLIB/JRNTEST.

저널에 표시된 항목을 주의깊게 살펴 보십시오. 항목이 프로그램에 의해 수행되었던 것과 같은 순서(R UP = update of ITMP 다음에 R PT = record added to TRNP가 나옴)로 저널에 발생합니다. 이것은 논리 파일이 실제 파일인 TRNP 위에 정의되고 시스템이 RPG 디폴트를 대체하기 때문입니다. 논리 파일이 없다면 SEQONLY(*YES)라는 RPG 가정이 사용되고 PT 항목 블록이 표시되는데 이는 블록이 가득 찰 때까지 레코드가 RPG 버퍼에 보관되기 때문입니다.

21. 다음과 같이 ITMPCSC CL 프로그램을 변경하십시오(새로 나온 명령문은 별표(*)로 표시됩니다).

```
     PGM
     DCL &USER *CHAR LEN(10)
     RTVJOBA USER(&USER)
*    STRCMTCTL LCKLVL(*CHG)
     CALL ITMPCS PARM(&USER)
*    MONMSG MSGID(RPG9001) EXEC(ROLLBACK)
*    ENDCMTCTL
     ENDPGM
```

STRCMTCTL 명령은 확약 제어 환경을 설정합니다. LCKLVL은 갱신을 위해 읽혔지만 갱신되지 않은 레코드가 트랜잭션 중에 해제될 수 있음을 지정합니다. MONMSG 명령은 RPG 이탈 메세지를 처리하고 RPG 프로그램이 비정상적으로 종료한 경우 ROLLBACK을 수행합니다. ENDCMTCTL 명령은 확약 제어 환경을 종료합니다.

22. 기존의 ITMPCSC 프로그램을 삭제하고 다시 작성하십시오.
23. 명령문 2.00, 4.00, 46.00 및 65.00에서 주석 기호를 없애십시오. 이제 소스를 확약 제어와 함께 사용할 수 있습니다.
24. 기존의 ITMPCS 프로그램을 삭제하고 다시 작성하십시오. 이제 이 프로그램을 확약 제어 하에서 실행할 준비가 되었습니다.
25. CALL ITMPCSC 명령 및 다음 트랜잭션을 입력하십시오.

수량	품목
7	AA
8	BB

26. 시스템 요구를 사용하여 현재 작업 표시 옵션을 요구하십시오. 작업 표시 화면이 표시되면 옵션 16을 선택하여 확약 제어 상태 표시를 요구하십시오.

    화면에 표시된 값을 주의깊게 살펴 보십시오. 프로그램에서 두 개의 확약 명령문이 실행되었으므로 두 번의 확약이 있어야 합니다.

27. F9를 눌러 확약 제어 하의 파일 목록 및 각 파일에 대한 활동량을 살펴 보십시오.
28. 프로그램으로 돌아와 F3을 눌러 프로그램을 종료하십시오.
29. DSPJRN CMTLIB/JRNTEST를 입력하여 확약 제어에 대한 파일 항목 및 특수 저널 항목을 살펴 보십시오.

C BC	STRCMTCTL 명령이 발생했습니다.
C SC	확약 주기를 시작하십시오. 이것은 트랜잭션의 첫 번째 데이터베이스 조작으로 인해 레코드가 확약 제어의 일부로 삽입, 갱신 또는 삭제될 때마다 발생합니다.
C CM	확약 조작이 발생했습니다.
C EC	ENDCMTCTL 명령이 발생했습니다.

    저널에 대해 IMAGES(*AFTER)를 요구했다고 하더라도 확약 제어 사전 이미지 및 사후 이미지(R UB 및 R UP 유형)가 자동으로 발생합니다.

30. CALL ITMPCSC 명령 및 다음 트랜잭션을 입력하십시오.

수량	품목
12	AA
100	CC(이것은 어플리케이션이 롤백을 사용할 필요가 있는지를 시뮬레이트하는 조건입니다. RPG 명령문 40.00에 의해 갱신되는 ITMP 파일의 CC 레코드가 롤백됩니다.)

31. F4를 눌러 입력된 최종 트랜잭션을 판별하십시오.

    마지막으로 확약된 트랜잭션은 AA 품목에 대한 항목입니다.

32. 시스템 요구를 사용하여 현재 작업 표시 옵션을 요구하십시오. 작업 표시 화면이 표시되면 확약 제어 상태 표시를 요구하십시오.

    화면에 표시된 값과 이 값들이 롤백에 의해 어떻게 변경되었는지 살펴 보십시오.

33. 프로그램으로 돌아오십시오.
34. 기본 프롬트 화면으로 돌아와 F3을 눌러 프로그램을 종료하십시오.
35. DSPJRN CMTLIB/JRNTEST 명령을 입력하십시오.

    롤백 항목(C RB 항목)에 사용할 저널에 표시되는 추가 항목을 주의깊게 살펴 보십시오. ITMP 레코드가 롤백되면 세 개의 항목이 저널에 들어갑니다. 이는 확약 제어 하에서 데이터베이스를 변경하면 사전(R BR) 및 사후(R UR) 항목이 생성되기 때문입니다.

36. 저널 코드가 R이고 이들 항목 유형이 UB, UP, BR 및 UR인 항목을 표시하십시오. 옵션 5를 사용하여 전체 항목을 표시하십시오. 수량 필드는 팩 십진 필드이므로 F11을 사용하여 16진수로 표시하십시오. 다음에 주의하십시오.
    - UB 레코드의 ITMP 레코드에 대한 사용 가능 값
    - UP 레코드에 의해 사용 가능 값이 감소하는 방식
    - BR 레코드가 UP 레코드와 동일하게 되는 방식

- 처음에 UB 레코드에 대해 표시된 값을 UR 레코드가 리턴하는 방식

마지막 항목은 롤백 종료를 위한 RB 항목입니다.

37. CALL ITMPCSC 명령을 입력한 후 Enter 키를 누르고 F4를 누르십시오. 입력된 마지막 트랜잭션을 주의깊게 살펴 보십시오.
38. 다음 트랜잭션을 입력하십시오.

수량	품목
13	AA
101	CC(이것은 프로그램을 종료시키는 예기치 않은 오류를 시뮬레이트하는 조건입니다. 이 시뮬레이션은 필드를 0으로 나누어 발생합니다. 프로그램은 INQMSGRPY 작업 속성 설정에 따라 조회 메세지를 표시하거나 종료합니다. 조회 메세지가 표시되면 C를 눌러 프로그램을 종료하십시오. RPG 프로그램 오류를 모니터하도록 CL 프로그램을 변경했으므로 두 번째 조회는 발생하지 않습니다.)

39. DSPJRN CMTLIB/JRNTEST 명령을 입력하십시오.

    동일한 유형의 롤백 처리가 발생했지만 이번에는 RPG 프로그램이 아닌 CL 프로그램의 MONMSG 명령의 EXEC 매개변수에 의해 롤백이 발생했습니다. 두 개의 RB 항목을 표시하여 어떤 프로그램이 발생시킨 것인지 확인하십시오.
40. WRKJOB을 입력하고 나중에 사용할 완전 규정된 작업명을 기록하십시오.
41. CALL ITMPCSC 명령을 입력하고 다음 트랜잭션을 입력하십시오.

수량	품목
14	AA
102	CC(RPG DSPLY 조작이 외부 메세지 대기행렬에 발생해야 합니다. 시스템 요구 키를 사용하여 시스템 요구 메뉴에서 옵션 1을 선택하여 두 번째 작업으로 전송하십시오.)

42. 두 번째 작업으로 사인 온하여 환경을 다시 설정하십시오.
43. ENDJOB 명령을 입력하고 앞에서 식별된 완전 규정된 작업명 및 OPTION(*IMMED)를 지정하십시오. 이것은 비정상적인 작업 또는 시스템 종료를 시뮬레이트합니다.
44. 약 30초 동안 기다린 후 CALL ITMPCSC를 입력하고 F4를 누르십시오.

    마지막으로 확약된 트랜잭션을 주의깊게 살펴 보십시오. 앞서 입력한 AA 품목이어야 합니다.
45. 기본 프롬트 화면으로 돌아와 F3을 눌러 프로그램을 종료하십시오.
46. DSPJRN CMTLIB/JRNTEST 명령을 입력하십시오.

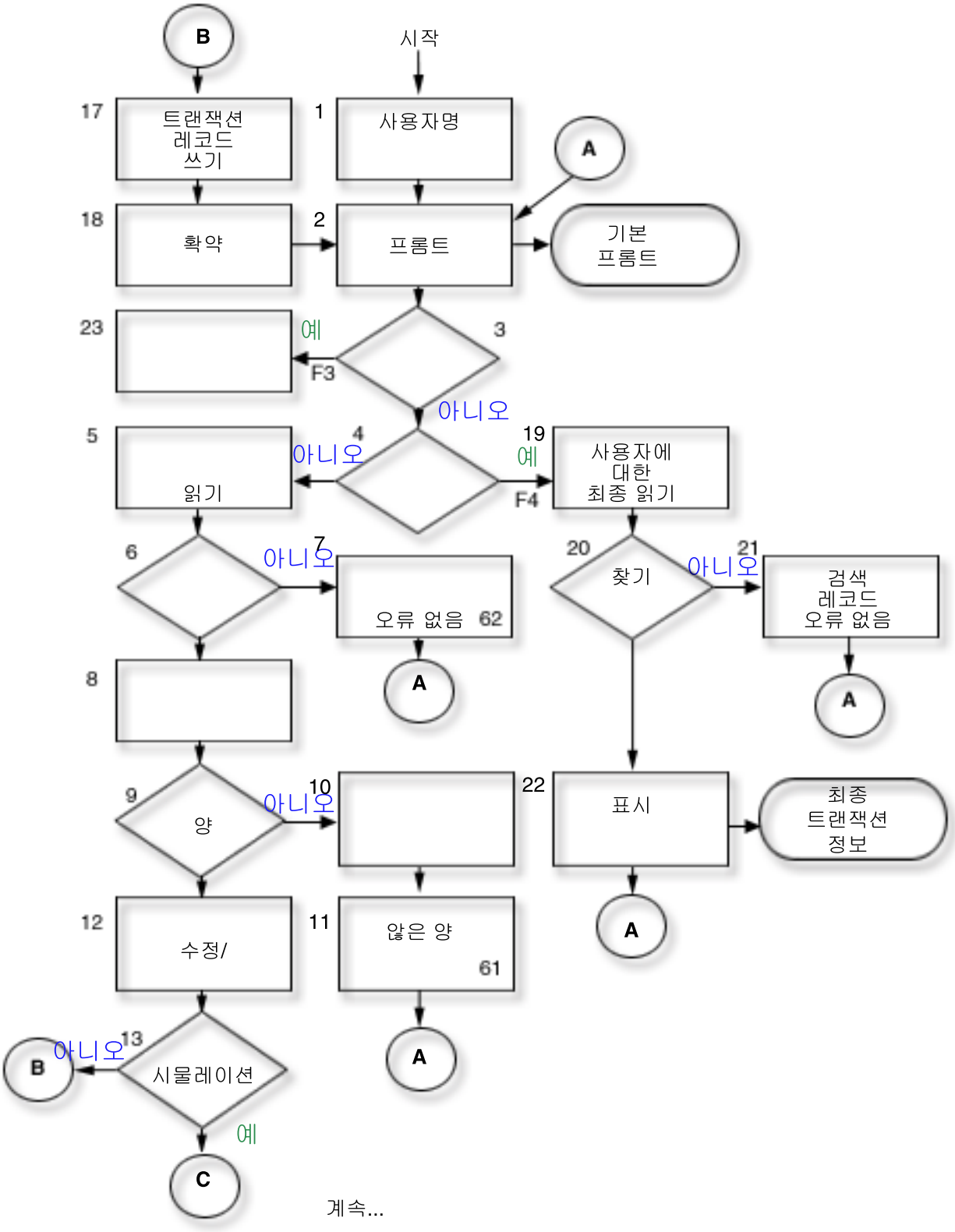
    동일한 유형의 롤백 처리가 발생했지만 이번에는 프로그램 중 하나가 아닌 시스템에 의해 롤백이 발생했습니다. RB 항목은 작업 관리 비정상 종료 프로그램인 QWTPITPP 프로그램에 의해 작성된 것입니다.

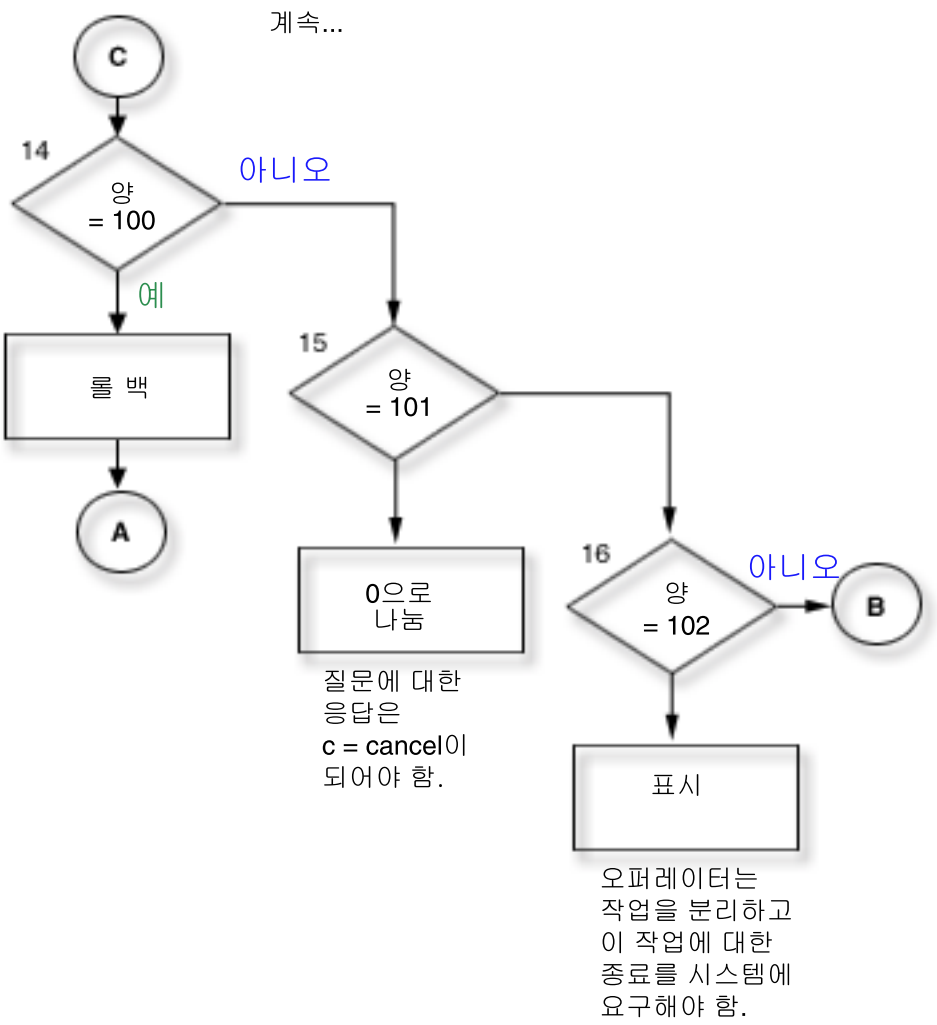
여기에서는 확약 제어의 기본 기능을 사용했습니다. 어플리케이션에서 계속 확약 제어를 사용하거나 다음과 같은 다른 기능을 사용해 볼 수도 있습니다.

- 통지 오브젝트 사용
- LCKLVL(*ALL)을 사용하여 읽기 전용인 레코드 잠금
- LCKLVL(*ALL)을 사용하여 동일한 파일에서 복수 레코드 잠금

연습 문제의 논리 흐름

다음 이미지에서는 확약 제어에 대한 연습 문제의 흐름을 보여줍니다. 흐름이 진행됨에 따라 논리 흐름과 연관된 각 단계를 보려면 번호가 매겨진 모양들을 클릭하십시오. 또한 단계 전체를 보려면 연습 문제의 논리 흐름과 연관된 단계로 갈 수도 있습니다.


B
시작
17
트랜잭션
레코드
쓰기
1
사용자명
A
18
확약
2
프롬트
기본
프롬트
23
예
3
F3
아니오
5
읽기
4
아니오
19
예
사용자에
대한
최종 읽기
F4
6
아니오
7
오류 없음 62
A
20
찾기
아니오
21
검색
레코드
오류 없음
A
8
9
양
아니오
10
22
표시
최종
트랜잭션
정보
12
수정/
11
않은 양
61
A
A
B
아니오
13
시물레이션
예
C
계속...



연습 프로그램의 논리 흐름과 연관된 단계

다음 단계는 연습 문제의 논리 흐름과 연관된 것입니다.

1. 매개변수로 전달된 사용자명을 검색합니다. 이것은 TRNP 파일에 기록하는 데 사용되고 각 오퍼레이터가 입력한 마지막 트랜잭션을 검색하는 데에도 사용됩니다. 이 어플리케이션은 오퍼레이터에 고유한 사용자명이 사용된다고 가정합니다.
2. 형식 이름 PROMPT를 사용하여 기본 화면에 대한 프롬트를 표시합니다.
3. F3을 누르면 프로그램 기능 종료를 시작합니다.
4. F4를 누르면 오퍼레이터가 입력한 마지막 트랜잭션에 액세스하는 루틴을 시작합니다.
5. *ITEM* 필드를 사용하여 항목 레코드를 읽습니다. 파일이 갱신 파일이므로 이 요구는 레코드에 대하여 잠금을 실행합니다.
6. ITMP 파일에서 찾을 수 없음 조건을 검사합니다.
7. ITMP 레코드가 없으면 인디케이터 62를 켜서 오류 메세지를 발생시키고 2단계(86See page)로 돌아갑니다.

8. 사용 가능 수량(ONHAND)에서 요구 수량(QTY)을 빼서 작업 영역에 그 값을 넣습니다.
9. 요구를 충족시킬 수 있도록 수량이 충분한지 검사합니다.
10. 수량이 부족하면 ITMP 파일의 해당 레코드에 대한 잠금을 해제합니다. 수량 부족으로 인해 이 단계가 필요합니다.
11. 인디케이터 61을 설정하여 부족 수량 화면 오류 메세지를 신호하고 2단계(86See page)로 돌아옵니다.
12. ONHAND 필드 값을 새로운 잔고량으로 변경하고 ITMR 레코드를 갱신하십시오.
13. 롤백이 필요한 조건을 시뮬레이션하는 데 사용할 수 있는 특별한 항목이 ITEM 필드에 있는지 검사합니다.
14. QTY=100인지 검사합니다. 롤백 조작을 발행합니다. 이것은 프로그램에 의한 롤백 필요 감지 조건을 시뮬레이션합니다.
15. QTY=101인지 검사합니다. 프로그램에서 예외를 발생시켜 조회 메세지를 생성합니다. 이 기능에 대하여 0으로 나누기를 사용하십시오. 작업 설명 INQMSGRPH 옵션이 자동 응답을 제공하지 않는 한 오퍼레이터는 C를 입력하여 프로그램을 취소해야 합니다. 이것은 예기치 않은 오류가 발생하여 오퍼레이터가 프로그램을 취소하는 조건을 시뮬레이션합니다.
16. QTY=102인지 검사합니다. 조회 조작을 사용하여 화면을 발행합니다. 그러면 이 단계에서 프로그램이 중단되고 시스템 요구 키를 사용하여 다른 작업으로 들어갈 수 있습니다. 작업 갱신을 취소합니다. 이것은 확약 경계 가운데에서 작업 또는 시스템의 비정상 종료가 발생한 조건을 시뮬레이션합니다.
17. 트랜잭션 레코드를 TRNP에 기록합니다.
18. 그 트랜잭션에 대한 레코드를 확약하고 2단계(86See page)로 리턴합니다.
19. USER를 키로 사용하여 TRNL 파일에 대한 액세스 경로에 있는 첫 번째 레코드를 읽습니다. 이 파일은 LIFO 순서로 되어 있으므로 이것은 이 사용자가 입력한 마지막 트랜잭션 레코드입니다.
20. 파일에 이 사용자에 대한 항목이 들어 있지 않은 경우 발생하는 레코드를 찾을 수 없음 조건을 TRNL 파일에서 검사합니다.
21. 이 사용자에 대한 레코드가 없는 경우 인디케이터 64를 설정하여 오류 메세지를 발생시키고 2단계(86See page)로 리턴합니다.
22. 이 사용자가 입력한 마지막 트랜잭션을 표시합니다. 이 정보는 오퍼레이터가 이전에 입력한 것이 무엇이었는지 또는 트랜잭션이 언제 재시작되었는지 잊었을 때 사용할 수 있습니다. 오퍼레이터가 응답하면 2단계(86See page)로 리턴합니다.
23. 프로그램 기능 종료를 수행합니다.

예: 트랜잭션 기록 파일을 사용하여 어플리케이션 시작

이 예에서는 트랜잭션 기록 파일을 사용하여 비정상 종료 후에 어플리케이션을 시작하는 방법에 대한 지시사항 및 샘플 코드를 제공합니다.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

트랜잭션 기록 파일은 통지 오브젝트가 사용되지 않는 경우 시스템 또는 작업 실패 후 어플리케이션을 다시 시작할 때 사용됩니다. 트랜잭션 기록 파일은 종종 대화식 어플리케이션에서 트랜잭션이 미치는 영향을 요약하기 위해 사용됩니다.

예를 들어 주문 입력 어플리케이션에서 주문된 각 품목에 대한 레코드는 일반적으로 트랜잭션 기록 파일에 기록됩니다. 레코드에는 주문 품목, 수량 및 가격이 들어갑니다. 미지급금 계정 어플리케이션에서는 대금을 받아야할 각 계정 번호에 대한 트랜잭션 기록 파일에 레코드가 기록됩니다. 일반적으로 이 레코드에는 계정 번호, 대금 액수 및 업체 등의 정보가 포함됩니다.

트랜잭션 기록 파일이 이미 존재하는 많은 어플리케이션에서는 워크스테이션 사용자가 입력된 최종 트랜잭션에 대한 정보를 요구할 수 있습니다. 트랜잭션 기록 파일이 이미 존재하는 어플리케이션에 확약 제어를 추가함으로써 다음을 수행할 수 있습니다.

- 데이터베이스 파일을 확약 경계로 갱신할 수 있습니다.
- 트랜잭션을 다시 시작하는 것을 단순화할 수 있습니다.

확약 제어 하에서 어플리케이션을 다시 시작하기 위해 트랜잭션 기록 파일을 사용하는 경우 고유하게 워크스테이션 사용자를 식별할 수 있어야 합니다. 시스템에서 고유한 사용자 프로파일명이 사용되는 경우 해당 프로파일명을 트랜잭션 기록 레코드의 필드에 넣을 수 있습니다. 이 필드는 파일에 대한 키로 사용할 수 있습니다.

다음 예에서는 주문 명세 파일을 사용하여 트랜잭션을 수행하며 트랜잭션 기록 파일이 이미 존재한다고 가정합니다. 프로그램은 다음을 수행합니다.

1. 워크스테이션 사용자가 수량 및 품목 번호를 입력하도록 프롬트합니다.
2. 기간계 마스터 파일(PRDMSTP)의 수량을 갱신합니다.
3. 트랜잭션 기록 파일(ISSLOGL)에 레코드를 기록합니다.

사용 가능한 재고 수량이 부족한 경우 프로그램은 트랜잭션을 거부합니다. 품목 번호, 설명, 수량, 사용자명 및 날짜가 트랜잭션 기록 파일에 기록되어 있으므로 워크스테이션 사용자는 데이터 입력이 어디에서 인터럽트되었는지 프로그램에 물어볼 수 있습니다.

실제 파일 PRDMSTP의 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDMSTR                   TEXT('Master record')
  2.00      A            PRODCT         3         COLHDG('Product' 'Number')
  3.00      A            DESCRP        20         COLHDG('Description')
  4.00      A            ONHAND         5  0      COLHDG('On Hand' 'Amount')
  5.00      A                                      EDTCDE(Z)
  6.00      A          K PRODCT
```

ISSLOGP에 의해 사용되는 실제 파일 ISSLOGP의 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R ISSLOGR                   TEXT('Product log record')
  2.00      A            PRODCT         3         COLHDG('Product' 'Number')
  3.00      A            DESCRP        20         COLHDG('Description')
```

```
  4.00      A            QTY             3  0       COLHDG('Quantity')
  5.00      A                                        EDTCDE(Z)
  6.00      A            USER           10          COLHDG('User' 'Name')
  7.00      A            DATE            6  0       EDTCDE(Y)
  8.00      A                                        COLHDG('Date')
```

논리 파일 **ISSLOGL**의 **DDS**

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A                                        LIFO
  2.00      A          R ISSLOGR                     PFILE(ISSLOGP)
  3.00      A          K USER
```

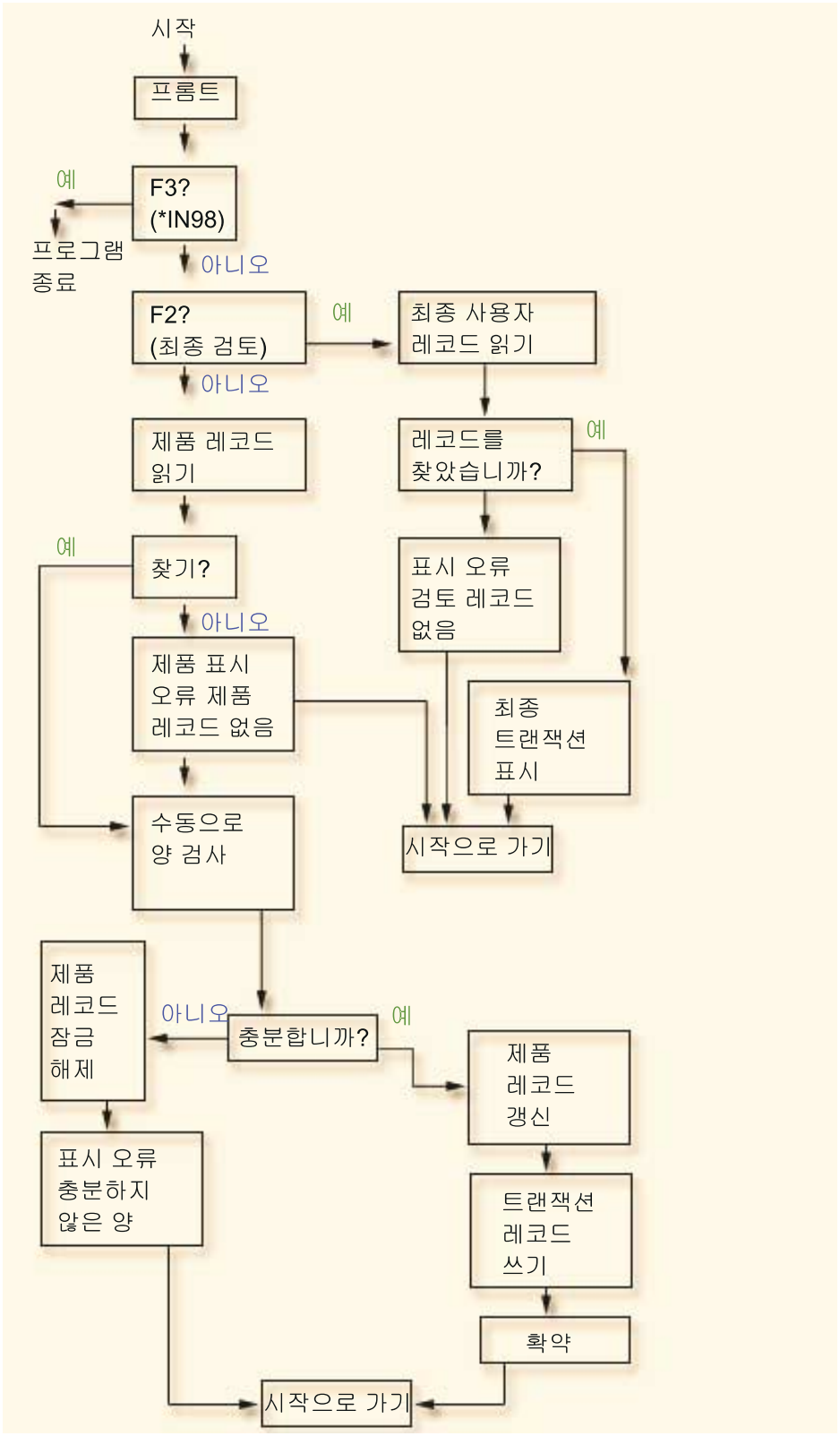
프로그램에서 사용되는 화면 파일 **PRDISSD**의 **DDS**

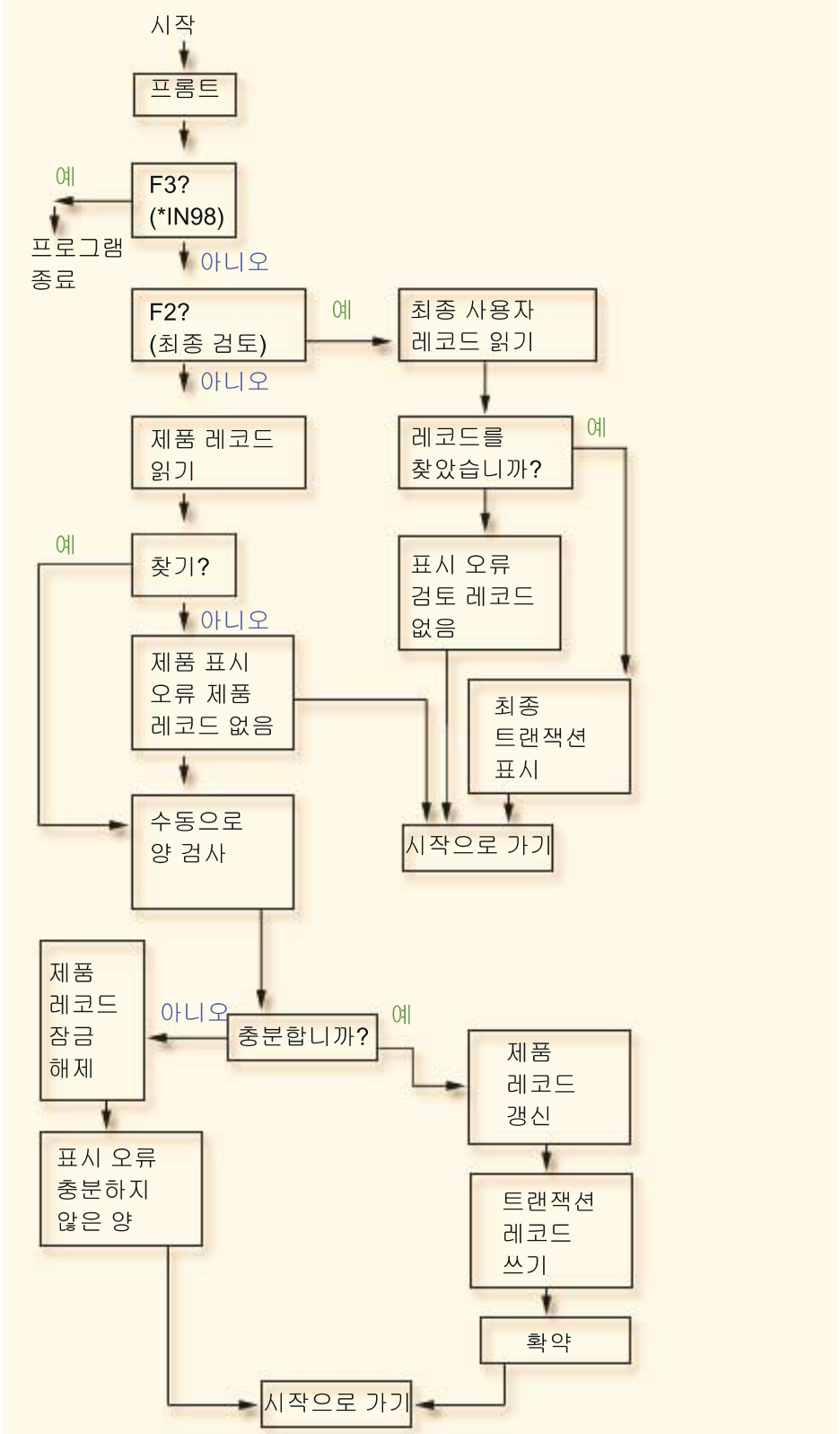
```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 1.00       A                                        REF(ISSLOGP)
 2.00       A          R PROMPT
 3.00       A                                        CA03(98 'End of program')
 4.00       A                                        CA02(97 'Where am I')
 5.00       A                                     1 20'ISSUES PROCESSING'
 6.00       A                                     3  2'Quantity'
 7.00       A            QTY       R        I    +1
 8.00       A  62                                    ERRMSG('Not enough +
 9.00       A                                        Qty' 62)
 10.00      A                                    +6'Product'
 11.00      A            PRODCT    R        I    +1
 12.00      A  61                                    ERRMSG('No Product +
 13.00      A                                        record found' 62)
 14.00      A  55                                15  2'No Previous record exists'
 15.00      A                                    24  2'CF2 Last transaction'
 16.00      A          R RESTART
 17.00      A                                     1 20'LAST TRANSACTION +
 18.00      A                                        INFORMATION'
 19.00      A                                     5  2'Product'
 20.00      A            PRODCT    R             +1
 21.00      A                                     7  2'Description'
 22.00      A            DESCRP    R             +1
 23.00      A                                     9  2'Qty'
 24.00      A            QTY       R             +1REFFLD(QTY)
```

이 프로세스는 프로그램 흐름에서 설명됩니다.

프로그램 흐름


시작
프롬트
예
F3?
(*IN98)
프로그램
종료
아니오
F2?
(최종 검토)
예
최종 사용자
레코드 읽기
아니오
제품 레코드
읽기
레코드를
찾았습니까?
예
예
찾기?
표시 오류
검토 레코드
없음
아니오
제품 표시
오류 제품
레코드 없음
최종
트랜잭션
표시
수동으로
양 검사
시작으로 가기
제품
레코드
잠금
해제
아니오
충분합니까?
예
제품
레코드
갱신
표시 오류
충분하지
않은 양
트랜잭션
레코드
쓰기
확약
시작으로 가기


시작
프롬트
예
F3?
(*IN98)
프로그램
종료
아니오
F2?
(최종 검토)
예
최종 사용자
레코드 읽기
아니오
제품 레코드
읽기
레코드를
찾았습니까?
예
예
찾기?
표시 오류
검토 레코드
없음
아니오
제품 표시
오류 제품
레코드 없음
최종
트랜잭션
표시
수동으로
양 검사
시작으로 가기
제품
레코드
잠금
해제
아니오
충분합니까?
예
제품
레코드
갱신
표시 오류
충분하지
않은 양
트랜잭션
레코드
쓰기
확약
시작으로 가기

*

RPG COMMIT 연산 코드는 PRDMSTP 파일이 갱신되고 레코드가 트랜잭션 기록 파일에 기록된 후에 지정됩니다. 오퍼레이터에게 표시되는 각 프롬트는 새로운 트랜잭션에 대한 경계를 나타내므로 트랜잭션은 단일 입력 트랜잭션으로 간주됩니다.

사용자명은 프로그램이 호출될 때 프로그램으로 전달됩니다. 트랜잭션 기록 파일에 대한 액세스 경로는 후입선출(LIFO) 순서로 정의되므로 프로그램은 마지막으로 입력된 레코드에 쉽게 액세스할 수 있습니다.

워크스테이션 사용자는 자료 입력이 중단된 지점을 식별하는 기능을 사용하여 시스템 또는 작업 실패 후 프로그램을 다시 시작할 수 있습니다. 프로그램에 코드를 추가할 필요는 없습니다. 현재 트랜잭션 기록 파일을 사용하고 있지만 현재 위치를 찾기 위해 사용하지는 않은 경우 사용자명을 트랜잭션 기록 파일에 추가하고(사용자명이 고유하다고 전제) 이 접근 방식을 프로그램에서 사용하십시오.

다음은 사용된 RPG 프로그램을 보여줍니다. 확약 제어에 필요한 명령문은 화살표(==>)로 표시되어 있습니다.

RPG 프로그램

```
 SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... .. 7 ..
 =>1.00      FPRDMSTP UP  E           K        DISK        KCOMIT
 =>2.00      FISSLOGL IF  E           K        DISK        KCOMIT
   3.00      PRDISSD  CP  E                    WORKSTN
   4.00                 *ENTRY    PLIST
   5.00                           PARM           USER    10
   6.00      C*
   7.00      C*  Initialize fields used in Trans Log Rcd
   8.00      C*
   9.00      C                    MOVE UDATE     DATE
  10.00      C*
  11.00      C*  Basic processing loop
  12.00      C*
  13.00      C           LOOP     TAG
  14.00      C                    EXFMTPROMPT
  15.00      C   98               GOTO END                       End of pgm
  16.00      C   97               DO                             Where am I
  17.00      C                    EXSR WHERE
  18.00      C                    GOTO LOOP
  19.00      C                    END
  20.00      C           PRODCT   CHAINPRDMSTR              61     Not found
  21.00      C   61               GOTO LOOP
  22.00      C           ONHAND   SUB  QTY      TEST    50    62   Less than
  23.00      C   62               DO                             Not enough
  24.00      C                    EXCPTRLSMST                    Release lock
  25.00      C                    GOTO LOOP
  26.00      C                    END
  27.00      C*
  28.00      C*  Update master record and output the Transaction Log Record
  29.00      C*
  30.00      C                    Z-ADDTEST      ONHAND
  31.00      C                    UPDATPRDMSTR
  32.00      C                    WRITEISSLOGR
=>33.00      C                    COMIT
  34.00      C                    GOTO LOOP
  35.00      C*
  36.00      C*  End of program processing
  37.00      C*
```

```
38.00      C           END       TAG
39.00      C                     SETON                     LR
40.00      C*
41.00      C* WHERE subroutine for "Where am I" requests
42.00      C*
43.00      C           WHERE     BEGSR
44.00      C           USER      CHAINISSLOGL              55    Not found
45.00      C  N55                EXFMTRESTART
46.00      C                     ENDSR
47.00      OPRDMSTR E                RLSMST
```

RPG 프로그램 PRDISS 호출에 사용되는 CL 프로그램

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00            PGM
  2.00            DCL          &USER *CHAR LEN(10)
  3.00            STRCMTCTL    LCKLVL(*CHG)
  4.00            RTVJOBA      USER(&USER)
  5.00            CALL         PRDISS PARM(&USER)
  6.00            MONMSG       MSGID(RPG9001) EXEC(ROLLBACK)
  7.00            ENDCMTCTL
  8.00            ENDPGM
```

이 프로그램에서 확약 제어를 사용하려면 일반적으로 잠금 레벨 *CHG를 지정합니다. 확약 조작이 실행될 때까지 레코드는 변경에 의해 잠금 상태를 유지합니다. 재고 수량이 부족한 경우 레코드는 명시적으로 해제됩니다. (프로그램에서 레코드가 명시적으로 해제되지 않았으면 파일에서 갱신을 위해 다음 레코드를 읽을 때 해제됩니다.)

이 예에서는 잠금 레벨 *ALL을 사용할 때 추가적인 장점은 없습니다. *ALL이 사용되면 수량이 부족할 때 롤백이나 확약 조작을 사용하여 레코드를 해제해야 합니다.

앞의 코드는 RPG 프로그램 PRDISS를 호출하는 CL 프로그램입니다. STRCMTCTL/ENDCMTCTL 명령을 사용하는 방식에 주목하십시오. 고유한 사용자명이 검색되어(RTVJOBA 명령) 프로그램으로 전달됩니다. MONMSG 명령을 사용하여 롤백을 수행하는 것에 대한 내용은 예: 표준 처리 프로그램을 사용하여 어플리케이션 시작에 나와 있습니다.

예: 통지 오브젝트를 사용하여 어플리케이션 시작

비정상 종료 후 프로그램이 시작될 때 통지 오브젝트에서 항목을 찾을 수 있습니다. 항목이 존재하면 프로그램은 트랜잭션을 다시 시작할 수 있습니다. 트랜잭션이 다시 시작된 후 프로그램은 통지 오브젝트를 지워서 다른 때 동일한 트랜잭션이 시작되지 않도록 합니다.

다음은 통지 오브젝트를 사용할 수 있는 방법입니다.

- 데이터베이스 파일에 확약 식별이 위치하는 경우 이 파일을 조회하여 각 어플리케이션 또는 워크스테이션 작업을 다시 시작할 위치를 판별합니다.
- 확약 식별이 특정 워크스테이션에 대한 메세지 대기행렬에 위치한 경우 워크스테이션 사용자가 사인 온할 때 메세지를 사용자에게 보내 확약된 마지막 트랜잭션을 알려줍니다.

- 확약 식별이 키나 사용자명을 가지고 있는 데이터베이스 파일에 위치하는 경우 프로그램은 시작할 때 이 파일을 읽을 수 있습니다. 파일에 레코드가 존재하면 프로그램을 다시 시작합니다. 프로그램은 확약된 마지막 트랜잭션을 식별하는 메세지를 워크스테이션 사용자에게 보낼 수 있습니다. 어떤 회복이든 프로그램에 의해 수행됩니다. 데이터베이스 파일에 레코드가 존재하면 프로그램은 프로그램 끝에서 그 레코드를 삭제합니다.
- 일괄처리 어플리케이션의 경우 총계, 스위치 설정 및 어플리케이션을 다시 시작하는 데 필요한 상태 정보 등을 포함하는 데이터 영역에 확약 식별이 있을 수 있습니다. 어플리케이션이 시작되면 데이터 영역에 액세스하여 저장되어 있는 값을 확인합니다. 어플리케이션이 정상적으로 종료하면 데이터 영역은 다음 실행을 위해 설정됩니다.
- 일괄처리 어플리케이션의 경우 확약 식별은 메세지 대기행렬로 송신될 수 있습니다. 어플리케이션이 시작될 때 실행되는 프로그램은 대기행렬에서 메세지를 검색할 수 있고 다시 프로그램을 시작할 수 있습니다.

어플리케이션 요구사항에 따라 어플리케이션을 시작하는 방법에는 몇 가지가 있습니다. 방법을 선택할 때 다음을 고려하십시오.

- 동시에 프로그램 사용자가 여럿 있을 때 시스템의 비정상 종료 후 각 사용자에 대한 확약 식별이 데이터 영역에서 서로 겹치게 되므로 하나의 데이터 영역을 통지 오브젝트로 사용할 수 없습니다.
- 통지 오브젝트에서 정보 삭제를 설계할 때 정보를 다음과 같이 사용한 후 즉시 실패가 발생하는 상황을 처리해야 합니다.
  - 정보가 즉시 삭제되는 경우 인터럽트된 트랜잭션 처리 전에 다른 장애가 발생하는 경우 그 정보는 존재하지 않습니다.
  - 통지 오브젝트의 정보는 인터럽트된 트랜잭션이 정상적으로 처리될 때까지 삭제되어서는 안됩니다. 이 경우 데이터베이스 파일이거나 메세지 대기행렬인 경우 둘 이상의 항목이 통지 오브젝트에 존재합니다.
  - 프로그램은 둘 이상의 항목이 있는 경우 마지막 레코드에 액세스해야 합니다.
- 통지 오브젝트는 시스템이나 작업 실패가 발생하거나 미확약 변경이 정상적인 작업 종료시 존재하는 경우에만 갱신되므로 워크스테이션 사용자에게 확약된 마지막 트랜잭션을 제공하기 위해서는 통지 오브젝트를 사용할 수 없습니다.
- 정보가 워크스테이션 사용자에게 표시되면 그것은 의미 있는 정보일 것입니다. 이를 위해서 프로그램은 통지 오브젝트의 코드를 사용자가 다시 시작하는 데 도움이 될 정보로 변환해야 합니다.
- 다시 시작하기 위한 정보는 워크스테이션 사용자가 필요로 하면 표시되어야 합니다. 프로그램 추가 논리는 더 이상 의미를 갖지 않을 때 정보가 표시되는 것을 방지합니다.
- 단일 통지 오브젝트 및 표준 처리 프로그램은 통지 오브젝트가 데이터베이스 파일인 경우 다시 시작 기능을 제공합니다. 이 표준 처리 프로그램은 각 개별 프로그램에 대한 변경을 최소화하기 위해 다시 시작할 수 있는 기능을 필요로 합니다.

통지 오브젝트에 대한 코드 예제는 다음을 참조하십시오.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

- 각 프로그램에 대한 고유한 통지 오브젝트
- 모든 프로그램에 대한 단일 통지 오브젝트

예: 각 프로그램에 대한 고유한 통지 오브젝트

이 주제에서는 고유한 통지 오브젝트를 사용하여 각 프로그램을 재시작하기 위한 지침 및 샘플 코드를 제공합니다.

각 작업에 대한 하나의 고유한 통지 오브젝트를 사용함으로써 동일한 프로그램을 사용하는 사용자가 여럿 있다고 하더라도 외부적으로 서술되는 확약 식별을 사용할 수 있습니다. 다음 예에서는 데이터베이스 파일이 통지 오브젝트로 사용되고 통지 오브젝트는 이 프로그램에 의해서만 사용됩니다.

프로그램에는 재고에 따라 영수증을 갱신해야 할 두 개의 데이터베이스 파일(PRDMSTP 및 PRDLOCP)이 있습니다. 프로그램이 사용하는 화면 파일의 이름은 PRDRCTD입니다. 데이터베이스 파일인 PRDRCTP는 통지 오브젝트로 사용됩니다. 이 통지 오브젝트는 프로그램에 파일로 정의되고 통지 기능에 대한 자료 구조 정의로도 사용됩니다.

실제 파일(PF) PRDMSTP의 DDS를 보려면 실제 파일(PF) PRDMSTP의 DDS(88See page)를 참조하십시오.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

실제 파일 **PRDLOCP**의 **DDS**

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDLOCR                   TEXT('Location record')
  2.00      A            PRODCT         3         COLHDG('Product' 'Number')
  3.00      A            LOCATN         6         COLHDG('Location')
  4.00      A            LOCAMT         5  0      COLHDG('Location' 'Amount')
  5.00      A                                     EDTCDE(Z)
  6.00      A          K PRODCT
  7.00      A          K LOCATN
```

화면 파일 **PRDRCTD**의 **DDS**

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                     REF(PRDMSTP)
  2.00      A          R PROMPT
  3.00      A                                     CA03(98 'End of program')
  4.00      A                                     SETOFF(71 'RESTART')
  5.00      A                                  1  20'PRODUCT RECEIPTS'
  6.00      A                                  3   2'Quantity'
  7.00      A            QTY           3  0I     +1
  8.00      A                                     +6'Product'
  9.00      A            PRODCT   R        I     +1
 10.00      A  61                                   ERRMSG('No record +
 11.00      A                                       found in the +
 12.00      A                                       master file' 62)
 13.00      A                                     +6'Location'
 14.00      A            LOCATN   R        I     +1REFFLD(LOCATN PRDLOCP)
 15.00      A  62                                   ERRMSG('No record +
 16.00      A                                       found in the +
 17.00      A                                       location file' 62)
 18.00      A                                  9   2'Last Transaction'
 19.00      A  71                                  +6'This is restart +
```

```
20.00      A                                        information'
21.00      A                                      DSPATR(HI BL)
22.00      A                                 12   2'Quantity'
23.00      A                                 12  12'Product'
24.00      A                                 12  23'Location'
25.00      A                                 12  35'Description'
26.00      A            LSTPRD    R          14  15REFFLD(PRODCT)
27.00      A            LSTLOC    R          14  26REFFLD(LOCATN *SRC)
28.00      A            LSTQTY    R          14   5REFFLD(QTY *SRC)
29.00      A                                      EDTCDE(Z)
30.00      A            LSTDSC    R          14  35REFFLD(DESCRP)
```

통지 오브젝트 및 외부 서술 자료 구조(PRDRCTP)의 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                      LIFO
  2.00      A                                      REF(PRDMSTP)
  3.00      A          R PRDRCTR
  4.00      A            USER          10
  5.00      A            PRODCT    R
  6.00      A            DESCRP    R
  7.00      A            QTY            3  0
  8.00      A            LOCATN    R               REFFLD(LOCATN PRDLOCP)
  9.00      A          K USER
```

프로그램은 통지 오브젝트를 다음과 같이 처리합니다.

- 처음에 프로그램은 통지 오브젝트를 임의로 처리하고 특정 키에 대해 존재하는 경우 레코드를 표시합니다.
  - 레코드가 여러 개 존재하는 경우 PRDRCTP 파일은 LIFO 순서이므로 이 키에 대한 마지막 레코드가 사용됩니다.
  - 레코드가 없는 경우 다시 시작할 필요가 없으므로 트랜잭션이 인터럽트되지 않습니다.
  - 첫 번째 성공적인 확약 조작 전에 프로그램이 실패하는 경우 재시작이 필요하다고 판단하지 않습니다.
- 통지 오브젝트를 지우는 루틴은 프로그램 끝에서 발생합니다.
  - 여러번 실패가 발생한 경우 루틴은 통지 오브젝트의 복수 레코드 삭제를 처리할 수 있습니다.
  - 시스템이 확약 식별을 데이터베이스 파일에 위치시킨다고 하더라도 확약 식별은 RPG 프로그램에 변수로 지정해야 합니다.
  - RPG에서는 자료 구조를 외부에서 서술하는 것이 가능하므로 자료 구조는 확약 식별을 지정할 수 있는 편리한 방법입니다. 이 예에서 자료 구조는 데이터베이스 파일이 통지 오브젝트로 사용한 동일한 외부 서술을 사용합니다.

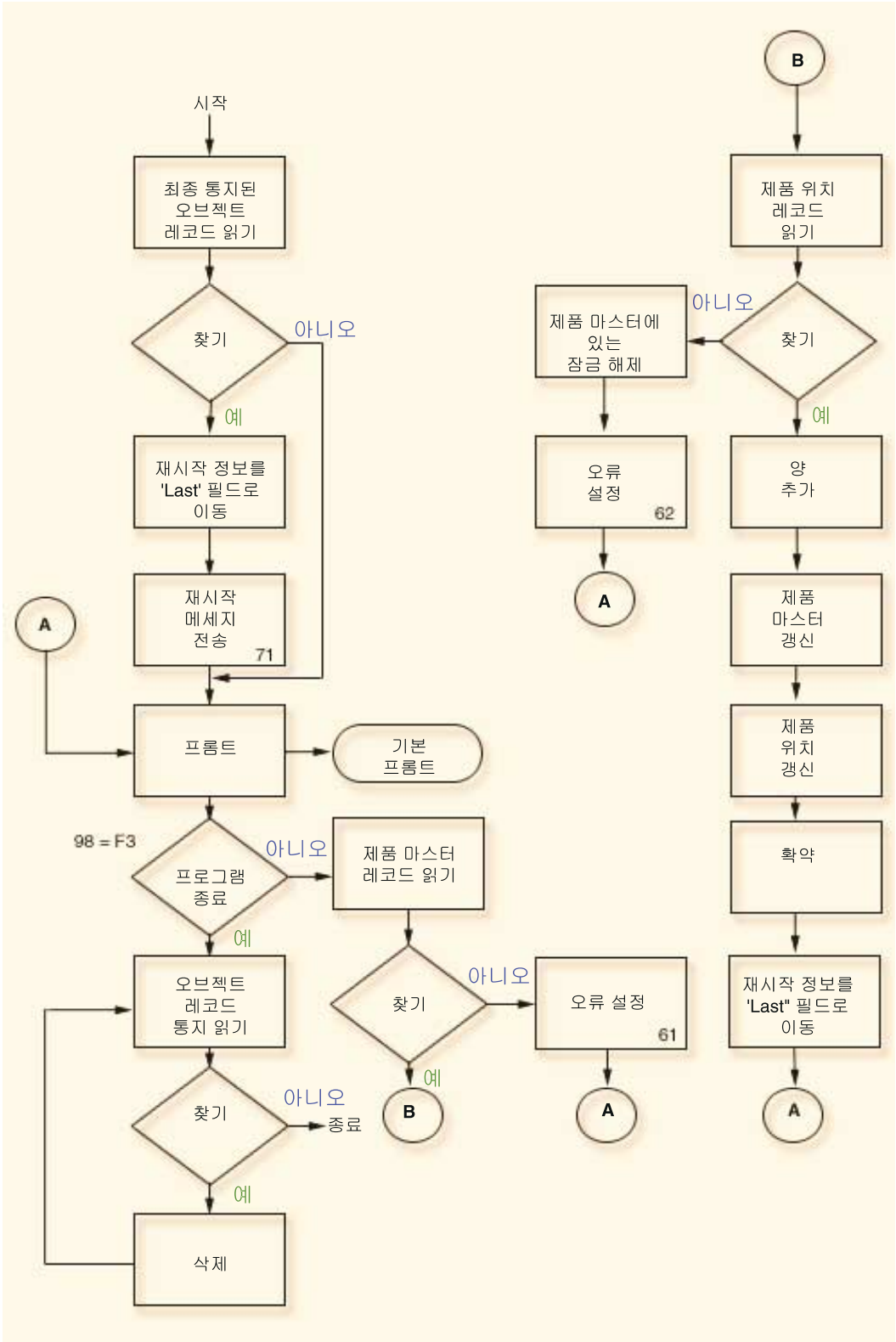
이 프로그램 처리 시 사용자에게 제품 번호, 위치 및 수량을 입력하도록 프롬트가 표시됩니다.

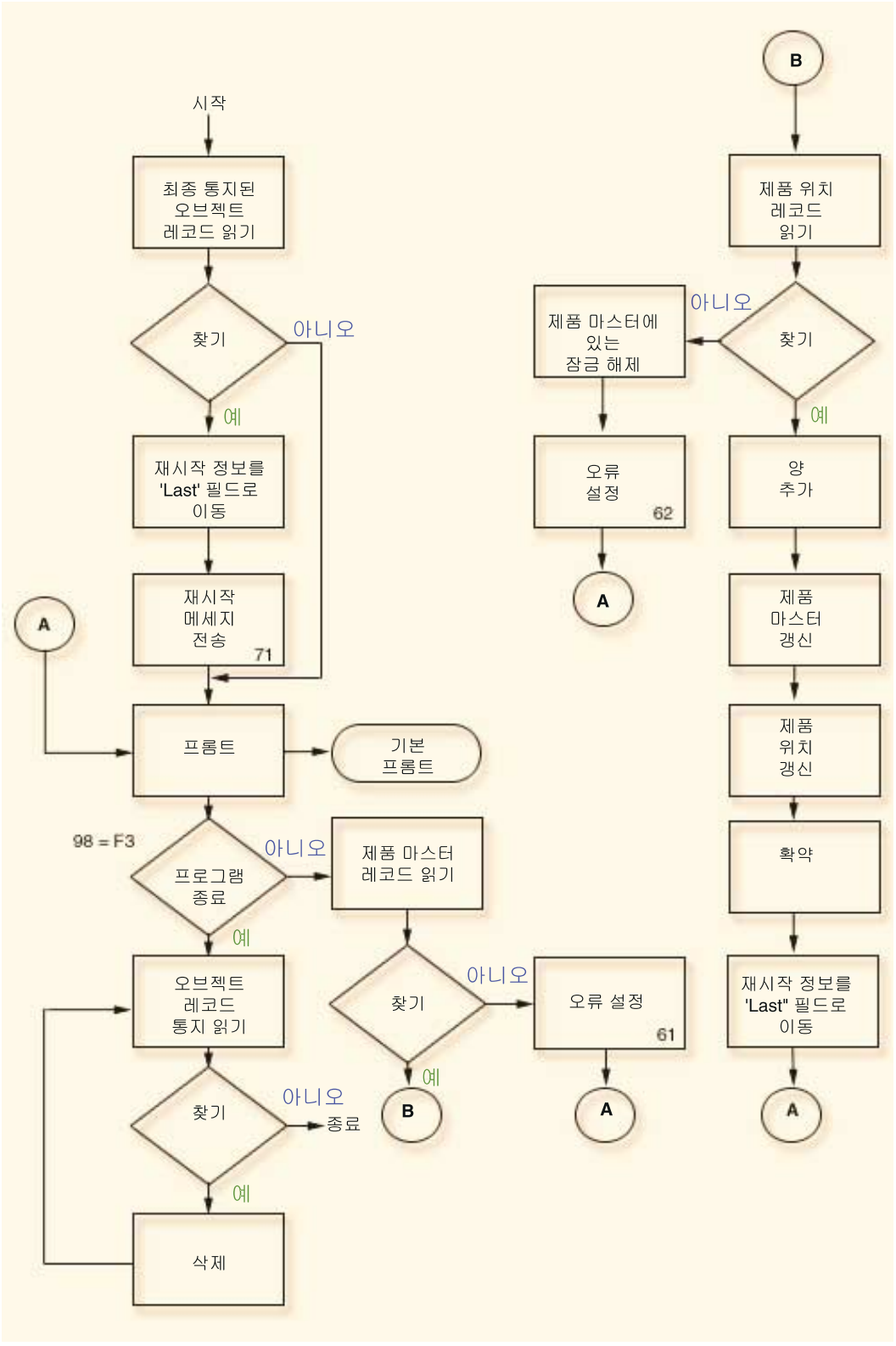
- 두 파일이 갱신되어야 합니다.
  - 제품 마스터 파일(PRDMSTP)
  - 제품 위치 파일(PRDLOCP)
- 각 파일의 레코드는 갱신되기 전에 먼저 존재해 있어야 합니다.

- 각 트랜잭션이 성공적으로 입력된 후 프로그램은 입력 필드를 대응하는 마지막 필드로 이동시킵니다. 이 마지막 필드는 마지막으로 입력된 내용에 대한 피드백으로서 각 프롬트에 의해 오퍼레이터에게 표시됩니다.
- 재시작을 위한 정보가 존재하는 경우 이 정보는 이러한 마지막 필드로 이동하고 특수 메세지가 화면에 표시됩니다.

이 프로세스가 다음 그림에 요약되어 있습니다. 사용자명이 프로그램으로 전달되어 통지 오브젝트에 고유한 레코드를 제공합니다.

프로그램 흐름


시작
최종 통지된 오브젝트 레코드 읽기
찾기
아니오
예
재시작 정보를 'Last' 필드로 이동
재시작 메세지 전송
71
A
프롬트
기본 프롬트
98 = F3
프로그램 종료
아니오
제품 마스터 레코드 읽기
예
오브젝트 레코드 통지 읽기
찾기
아니오
종료
예
삭제
찾기
아니오
오류 설정
61
예
B
A
B
제품 위치 레코드 읽기
찾기
아니오
제품 마스터에 있는 잠금 해제
예
오류 설정
62
A
양 추가
제품 마스터 갱신
제품 위치 갱신
확약
재시작 정보를 'Last" 필드로 이동
A


시작
최종 통지된 오브젝트 레코드 읽기
찾기
아니오
예
재시작 정보를 'Last' 필드로 이동
재시작 메세지 전송
71
A
프롬트
기본 프롬트
98 = F3
프로그램 종료
아니오
제품 마스터 레코드 읽기
예
오브젝트 레코드 통지 읽기
찾기
아니오
종료
예
삭제
찾기
아니오
오류 설정
61
예
B
A
B
제품 위치 레코드 읽기
찾기
아니오
제품 마스터에 있는 잠금 해제
예
오류 설정
62
A
양 추가
제품 마스터 갱신
제품 위치 갱신
확약
재시작 정보를 'Last" 필드로 이동
A

*

다음은 이 예에 대한 RPG 소스입니다. 통지 오브젝트(PRDRCTP 파일)가 프로그램의 시작과 끝에서 일반 파일로 사용되고 프로그램 호출 전에 CL(STRCMTCTL 명령)에서 통지 오브젝트로 지정되기도 합니다.

RPG 소스

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00      FPRDRCTD CF  E                    WORKSTN
  4.00      F*
  5.00      F* The following file is the specific notify object for this pgm.
  6.00      F*      It is accessed only in a restart situation and at the
  7.00      F*        end of the program to delete any records.  The records
  8.00      F*        are written to the notify object by Commitment Control.
  9.00      F*
 10.00      FPRDRCTP UF  E           K        DISK
 11.00      ICMTID     E DSPRDRCTP

 12.00      C           *ENTRY    PLIST
 13.00      C                     PARM           USER10 10
 14.00      C                     MOVE USER10    USER
 15.00      C*
 16.00      C*  Check for restart information - get last rcd per user
 17.00      C*    PRDRCTP file access path is in LIFO sequence
 18.00      C*
 19.00      C           USER      CHAINPRDRCTR             20     Not found
 20.00      C  N20                DO                               Restart
 21.00      C                     EXSR MOVLST                      Move to last
 22.00      C                     SETON                    71     Restart
 23.00      C                     END
 24.00      C*
 25.00      C*  Basic processing loop
 26.00      C*
 27.00      C           L00P      TAG
 28.00      C                     EXFMTPROMPT
 29.00      C   98                GOTO END                         End of pgm
 30.00      C           PRODCT    CHAINPRDMSTR             61     Not found
 31.00      C   61                GOTO L00P
 32.00      C           KEY       KLIST
 33.00      C                     KFLD           PRODCT
 34.00      C                     KFLD           LOCATN
 35.00      C           KEY       CHAINPRDLOCR             62     Not found
 36.00      C   62                DO
 37.00      C                     EXCPTRLSMST                      Release lck
 38.00      C                     GOTO L00P
 39.00      C                     END
 40.00      C                     ADD  QTY       ONHAND           Add
 41.00      C                     ADD  QTY       LOCAMT
 42.00      C                     UPDATPRDMSTR                     Update
 43.00      C                     UPDATPRDLOCR                     Update
 44.00      C*

 45.00      C*  Commit and move to previous fields
 46.00      C*
 47.00      C           CMTID     COMIT
 48.00      C                     EXSR MOVLST                      Move to last
 49.00      C                     GOTO L00P
```

```
50.00      C*
51.00      C*  End of program processing
52.00      C*
53.00      C           END       TAG
54.00      C                     SETON                     LR
55.00      C*56.00      C*  Delete any records in the notify object
57.00      C*
58.00      C           DLTLP     TAG
59.00      C           USER      CHAINPRDRCTR              20     Not found
60.00      C  N20                DO
61.00      C                     DELETPRDRCTR                     Delete
62.00      C                     GOTO DLTLP
63.00      C                     END
64.00      C*
65.00      C*  Move to -Last Used- fields for operator feedback
66.00      C*
67.00      C           MOVLST    BEGSR
68.00      C                     MOVE PRODCT    LSTPRD
69.00      C                     MOVE LOCATN    LSTLOC
70.00      C                     MOVE QTY       LSTQTY
71.00      C                     MOVE DESCRP    LSTDSC
72.00      C                     ENDSR
73.00      OPRDMSTR E                RLSMST
```

예: 모든 프로그램에 대한 단일 통지 오브젝트

모든 프로그램에 대하여 하나의 통지 오브젝트를 사용하는 것은 다시 시작하기 위해 필요한 모든 정보가 같은 오브젝트에 들어 있고 통지 오브젝트에 대한 표준 접근 방식을 모든 프로그램에서 사용할 수 있으므로 장점이 많습니다. 여기에서는 사용자와 프로그램 ID를 고유하게 조합하여 프로그램이 다시 시작할 때 올바른 정보에 액세스하도록 하는 것입니다.

다시 시작하기 위해 필요한 정보가 프로그램마다 다를 수 있으므로 확약 식별에 대해 외부 서술 자료 구조를 사용하면 안됩니다. 하나의 통지 오브젝트를 사용하는 경우 앞의 프로그램에서 외부적으로 보다는 프로그램 내에서 자료 구조를 서술할 수 있습니다. 예를 들면 다음과 같습니다.

```
 1   10      USER
11   20      PGMNAM
21   23      PRODCT
24   29      LOCATN
30   49      DESC
50   51  0   QTY
52  220      DUMMY
```

이 통지 오브젝트를 사용하는 각 프로그램에서 확약 식별에 지정된 정보는 프로그램에 대하여 고유합니다(사용자 및 프로그램명은 고유하지 않음). 통지 오브젝트는 어떤 프로그램이든 확약 식별에 위치시킬 최대 정보를 포함할 수 있도록 충분히 커야 합니다.

예: 각 프로그램에 대한 고유한 통지 오브젝트에서 통지 오브젝트를 사용하는 더 많은 예를 제공합니다.

예: 표준 처리 프로그램을 시작하여 어플리케이션 시작

표준 처리 프로그램은 하나의 데이터베이스 파일을 모든 어플리케이션에 대한 통지 오브젝트로 사용하여 어플리케이션을 다시 시작하는 한 가지 방법입니다. 이 방법은 표준 프로그램을 사용하는 모든 어플리케이션에 대하여 사용자 프로파일명이 고유하다고 가정합니다.

이러한 방식의 경우 실제 파일 NFYOBJP는 통지 오브젝트로 사용되고 다음과 같이 정의됩니다.

```
고유한 사용자 프로파일명   10 문자
프로그램 식별              10 문자
다시 시작을 위한
     정보                  문자 필드
                           (프로그램을 다시 시작하는 데
                           필요한 최대 정보량을
                           수용할 수 있도록
                           충분히 커야
                           합니다.
                           이 필드는 어플리케이션 프로그램에서
                           필요합니다.
                           예에서는 길이를 200으로
                           가정합니다.)
```

파일은 SHARE(*YES)를 지정하여 작성됩니다. 파일의 처음 두 필드는 파일에 대한 키입니다. (이 파일 역시 RPG 프로그램의 자료 구조로 정의할 수 있습니다.)

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

다음은 표준 처리 프로그램에 대한 예제 코드입니다.

- 예: 표준 처리 프로그램 코드
- 예: 표준 확약 처리 프로그램
- 예: 표준 처리 프로그램을 사용하여 어플리케이션 재시작 결정

예: 표준 처리 프로그램 코드

다음은 표준 처리 프로그램을 사용한 예입니다. 다음 코드 예에 표시된 어플리케이션은 다음과 같이 수행됩니다.

1. 어플리케이션 프로그램은 매개변수로 사용자명을 받아 프로그램명과 함께 통지 오브젝트에서 고유 식별자로 사용합니다.
2. 어플리케이션 프로그램은 표준 확약 처리 프로그램으로 요구 코드 R을 전달하고 프로그램은 레코드가 통지 오브젝트에 있는지 판별합니다.
3. 표준 확약 처리 프로그램이 코드 1을 리턴하면 레코드를 찾은 것이고 어플리케이션 프로그램은 다시 시작하기 위해 필요한 정보를 사용자에게 표시합니다.
4. 어플리케이션 프로그램은 일반적인 처리를 수행합니다.
5. 트랜잭션이 완료되면 참조를 위해 값이 저장되고 워크스테이션 사용자는 이전 트랜잭션에서 무엇이 수행되었는지를 볼 수 있습니다.

   통지 오브젝트는 작업이나 시스템 장애가 발생한 경우에만 갱신되므로 통지 오브젝트는 저장된 정보를 제공하지 않습니다.

이 프로그램의 흐름은 처리 흐름을 참조하십시오.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

어플리케이션 프로그램 예

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00      FPRDRCTD CF  E                    WORKSTN
  4.00      F*
  5.00      F* The following is a compile time array which contains the
  6.00      F*    restart information used in the next example
  7.00      F*
  8.00      E                    RTXT   50  50  1               Restart text
  9.00      I*
 10.00      I* Data structure used for info passed to notify object
 11.00      I*
 12.00      ICMTID       DS
 13.00      I                                        1  10 USER
 14.00      I                                       11  20 PGMNAM
 15.00      I                                       21  23 PRODCT
 16.00      I                                       24  29 LOCATN
 17.00      I                                       30  49 DESCRP
 18.00      I                                     P 50  510QTY
 19.00      I                                       52 170 DUMMY
 20.00      I                                     171 220 RSTART
 21.00      C           *ENTRY    PLIST
 22.00      C                     PARM          USER10 10
 23.00      C*
 24.00      C*  Initialize fields used to communicate with std program
 25.00      C*
 26.00      C                     MOVE USER10    USER
 27.00      C                     MOVEL'PRDRC2'  PGMNAM
 28.00      C                     MOVE 'R'       RQSCOD           Read Rqs
 29.00      C                CALL 'STDCMT'
 30.00      C                     PARM           RQSCOD  1
 31.00      C                     PARM           RTNCOD  1
 32.00      C                     PARM           CMTID 220        Data struct
 33.00      C            RTNCOD   IFEQ '1'                       Restart
 34.00      C                     EXSR MOVLST                     Move to last
 35.00      C SETON                                       71    Restart
 36.00      C                     END
 37.00      C*
 38.00      C*  Initialize fields used in notify object
 39.00      C*
 40.00      C                     MOVEARTXT,1    RSTART           Move text
 41.00      C*
 42.00      C* Basic processing loop
 43.00      C*
 44.00      C            LOOP     TAG
 45.00      C                     EXFMTPROMPT
 46.00      C   98                GOTO END
 47.00      C            PRODCT   CHAINPRDMSTR              61    Not found
 48.00      C   61                GOTO LOOP
```

```
 49.00      C            KEY      KLIST
 50.00      C                     KFLD           PRODCT
 51.00      C                     KFLD           LOCATN

SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 52.00      C           KEY       CHAINPRDLOCR             62    Not found
 53.00      C   62                DO
 54.00      C                     EXCPTRLSMST                    Release lck
 55.00      C                     GOTO LOOP
 56.00      C                     END
 57.00      C                     ADD  QTY       ONHAND          Add
 58.00      C                     ADD  QTY       LOCAMT
 59.00      C                     UPDATPRDMSTR                   Update
 60.00      C                     UPDATPRDLOCR                   Update
 61.00      C*
 62.00      C*  Commit and move to previous fields
 63.00      C*
 64.00      C           CMTID     COMIT
 65.00      C                     EXSR MOVLST                    Move to last
 66.00      C                     GOTO LOOP
 67.00      C*  End of program processing
 68.00      C*
 69.00      C           END       TAG
 70.00      C                     MOVE 'D'       RQSCOD          Dlt Rqs
 71.00      C                     CALL 'STDCMT'
 72.00      C                     PARM           RQSCOD
 73.00      C                     PARM           RTNCOD
 74.00      C                     PARM           CMTID
 75.00      C                     SETON                     LR
 76.00      C*
 77.00      C*  Move to -Last Used- fields for operator feedback
 78.00      C*
 79.00      C           MOVLST    BEGSR
 80.00      C                     MOVE PRODCT    LSTPRD
 81.00      C                     MOVE LOCATN    LSTLOC
 82.00      C                     MOVE DESCRP    LSTDSC
 83.00      C                     MOVE QTY       LSTQTY
 84.00      C                     ENDSR
 85.00      OPRDMSTR E               RLSMST
 86.00 ** RTXT       Restart Text
 87.00 Inventory Menu - Receipts Option
```

예: 표준 확약 처리 프로그램 코드

표준 확약(STDCMT) 처리 프로그램은 모든 어플리케이션에서 사용하는 단일 통지 오브젝트와 통신하는 데 필요한 기능을 수행합니다. 확약 제어 기능이 자동으로 통지 오브젝트에 항목을 기록하지만 사용자 작성 프로그램은 통지 오브젝트를 처리해야 합니다. 표준 프로그램은 이러한 접근 방식을 단순화 및 표준화합니다.

프로그램은 전달된 매개변수를 검증하기 위해 작성되며 다음과 같이 적절한 조치를 취합니다.

O=열기

호출 프로그램은 리턴할 때 통지 오브젝트 파일이 열려 있도록 요구합니다. 통지 오브젝트가 RPG 프로

그램에 의해 내재적으로 열리므로 프로그램이 이를 닫으면 안됩니다. 인디케이터 98이 설정되어 프로그램 작업 영역을 보존하도록 LR을 OFF로 하여 리턴하고 통지 오브젝트를 열어 놓아 추가 오버헤드 없이 다시 호출될 수 있도록 합니다.

C=닫기
호출 프로그램이 더 이상 통지 오브젝트가 필요 없다고 판단하여 닫기를 요구합니다. 인디케이터 98을 OFF로 설정하여 통지 오브젝트를 완전히 닫습니다.

R=읽기
호출 프로그램이 키 필드가 일치하는 레코드를 읽어 다시 전달하도록 요구합니다. 프로그램은 전달된 키 필드를 사용하여 NFYOBJP에서 레코드를 검색하려고 시도합니다. 같은 키에 대하여 레코드가 중복 검색되면 마지막 레코드가 리턴됩니다. 그에 따라 리턴 코드가 설정되고 레코드가 존재하면 자료 구조 CMTID에 전달됩니다.

W=쓰기
호출 프로그램은 레코드를 통지 오브젝트에 기록하여 다음 번에 호출 프로그램이 호출될 때 다시 시작할 수 있도록 합니다. 프로그램은 전달된 자료의 내용을 NFYOBJP에 레코드로 기록합니다.

D=삭제
호출 프로그램은 이 키와 일치하는 레코드가 삭제되도록 요구합니다. 이 기능은 일반적으로 재시작 시 정보를 제거하기 위해 호출 프로그램의 정상적인 완료 시에 수행됩니다. 프로그램은 전달된 키 필드의 레코드를 삭제하려고 시도합니다. 레코드가 존재하지 않으면 다른 리턴 코드가 전달됩니다.

S=검색
호출 프로그램은 작성한 프로그램에 상관없이 특정 사용자에 대한 레코드를 검색합니다. 이 기능은 재시작이 필요함을 나타내기 위해 사인 온을 위해 프로그램에서 사용됩니다. 프로그램은 사용자명만 키로 사용하여 레코드가 존재하는지 찾아봅니다. 리턴 코드는 적절하게 설정되며 이 키에 대한 마지막 레코드(존재하는 경우)의 내용을 읽어 전달합니다.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

다음은 표준 확약 처리 프로그램 STDCMT를 보여줍니다.

표준 확약 처리 프로그램

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FNFYOBJP UF  E           K        DISK                    A
  2.00      ICMTID       DS
  3.00      I                                      1  10 UNQUSR
  4.00      I                                     11  20 UNQPGM
  5.00      I                                     21 220 BIGFLD
  6.00      C           *ENTRY    PLIST
  7.00      C                     PARM           RQSCOD  1
  8.00      C                     PARM           RTNCOD  1
  9.00      C                     PARM           CMTID 220
 10.00      C            UNQUSR   CABEQ*BLANKS   BADEND         H1 Invalid
 11.00      C            UNQPGM   CABEQ*BLANKS   BADEND         H2 Invalid
```

```
12.00      C*
13.00      C*  'O' for Open
14.00      C*
15.00      C            RQSCOD   IFEQ 'O'                         Open
16.00      C                     SETON                     98    End LR
17.00      C                     GOTO END
18.00      C                     END
19.00      C*
20.00      C*  'C' for Close
21.00      C*
22.00      C            RQSCOD   IFEQ 'C'                         Close
23.00      C                     SETOF                     98
24.00      C                     GOTO END
25.00      C                     END
26.00      C*
27.00      C*  'R' for Read - Get last record for the key
28.00      C*
29.00      C            RQSCOD   IFEQ 'R'                        Read
30.00      C            KEY      KLIST
31.00      C                     KFLD           UNQUSR
32.00      C                     KFLD           UNQPGM
33.00      C            KEY      CHAINNFYOBJR              51    Not found
34.00      C   51               MOVE '0'       RTNCOD
35.00      C   51               GOTO END
36.00      C                    MOVE '1'       RTNCOD            Found
37.00      C            LOOPl    TAG
38.00      C            KEY      READENFYOBJR                 20 EOF
39.00      C   20               GOTO END
40.00      C                    GOTO LOOP1
41.00      C                    END
42.00      C*
43.00      C*  'W' FOR Write
44.00      C*
45.00      C            RQSCOD   IFEQ 'W'                         Write
46.00      C                     WRITENFYOBJR
47.00      C                     GOTO END
48.00      C                     END
49.00      C*
50.00      C*  'D' for Delete - Delete all records for the key
51.00      C*
52.00      C           RQSCOD    IFEQ 'D'                         Delete
53.00      C           KEY       CHAINNFYOBJR              51    Not found
54.00      C   51               MOVE '0'       RTNCOD
55.00      C   51               GOTO END
56.00      C                    MOVE '1'       RTNCOD            Found
57.00      C           LOOP2     TAG
58.00      C                     DELETNFYOBJR
59.00      C           KEY       READENFYOBJR                 20 EOF
60.00      C  N20               GOTO LOOP2
61.00      C                    GOTO END
62.00      C                    END
63.00      C*
64.00      C*  'S' for Search for the last record for this user
65.00      C*              (Ignore the -Program- portion of the key)
66.00      C*
67.00      C           RQSCOD    IFEQ 'S'                        Search
68.00      C           UNQUSR    SETLLNFYOBJR                20 If equal
69.00      C  N20               MOVE '0'       RTNCOD
```

```
70.00      C  N20                GOTO END
71.00      C                     MOVE '1'       RTNCOD           Found
72.00      C           LOOP3     TAG
73.00      C           UNQUSR    READENFYOBJR                 20 EOF
74.00      C  N20                GOTO LOOP3
75.00      C                     GOTO END
76.00      C                     END
77.00      C*
78.00      C*  Invalid request code processing
79.00      C*
80.00      C                     SETON                     H2    Bad RQS code
81.00      C                     GOTO BADEND
82.00      C*
83.00      C*  End of program processing
84.00      C*
85.00      C           END       TAG
86.00      C  N98                SETON                     LR
87.00      C                     RETRN
88.00      C* BADEND tag is used then fall thru to RPG cycle error return
89.00      C           BADEND    TAG
```

예: 표준 처리 프로그램을 사용하여 어플리케이션 재시작 여부 결정

이 주제에서는 표준 처리 프로그램을 사용하여 비정상 IPL 후에 어플리케이션을 재시작할 것인지 여부를 결정하는 CL 코드 예를 설명합니다.

초기 프로그램은 표준 확약 처리 프로그램을 호출하여 다시 시작하는 것이 필요한지 여부를 결정합니다. 그리고나서 워크스테이션 사용자가 재시작 여부를 결정할 수 있습니다.

초기 프로그램은 요구 코드 S(검색)를 표준 프로그램에 전달하고 표준 프로그램은 그 사용자에 대한 레코드를 검색합니다. 레코드가 존재하면 재시작 정보가 초기 프로그램으로 전달되고 워크스테이션 사용자에게 정보가 표시됩니다.

통지 오브젝트의 확약 식별에는 초기 프로그램이 표시할 수 있으며, 어떤 프로그램을 다시 시작해야 하는지를 식별하는 정보가 들어 있어야 합니다. 예를 들어 확약 식별의 마지막 50자는 이 정보를 포함시키기 위해 예약해 놓을 수 있습니다. 어플리케이션 프로그램에서 이 정보는 컴파일 시간 배열에 들어갈 수 있고 초기화 단계에 데이터 구조로 옮겨질 수 있습니다. 예: 표준 확약 처리 프로그램에 대한 코드에서는 이것을 어플리케이션 프로그램에 포함시키는 방식이 표시됩니다.

다음은 레코드가 통지 오브젝트에 존재하는지 여부를 판별하는 초기 프로그램의 예입니다.

주: 중요한 법적 고지사항에 관해 언급하는 코드 면책사항 관련 정보를 참조하십시오.

초기 프로그램 예

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00            PGM
  2.00            DCLF        CMTINLD
  3.00            DCL         &RQSCOD *CHAR LEN(1) VALUE(S)  /* Search */
  4.00            DCL         &RTNCOD *CHAR LEN(1)
  5.00            DCL         &CMTID *CHAR LEN(220)
  6.00            DCL         &USER *CHAR LEN(10)
```

```
 7.00           DCL        &INFO *CHAR LEN(50)
 8.00           RTVJOBA    USER(&USER)
 9.00           CHGVAR     &CMTID (&USER *CAT XX)
10.00                      /* The XX is required to prevent a blank Pgm nam */
11.00           CALL       STDCMT PARM(&RQSCOD &RTNCOD &CMTID)
12.00           IF         (&RTNCOD *EQ '1') DO /* RESTART REQD */
13.00           CHGVAR     &INFO %SST(&CMTID 171 50)
14.00           SNDRCVF    RCDFMT(RESTART)
15.00           ENDDO
16.00                      /*                                          */
17.00                      /* Enter normal initial program statements  */
18.00                      /*    or -TFRCTL- to first menu program      */
19.00                      /*                                          */
20.00           ENDPGM
```

트랜잭션 및 확약 제어의 문제점 해결

다음 페이지에서는 확약 제어의 문제점을 해결하는 데 필요한 타스크를 제공합니다.

확약 제어 오류
여기에서는 오류가 발생할 수 있는 조건을 설명하고 확약 제어 오류를 나열하며 오류를 처리할 수 있는 방법에 대해 설명합니다.

교착 상태 감지
이 타스크는 교착 상태 조건을 감지하기 위한 것입니다.

통신 장애 후 트랜잭션 회복
이 타스크는 시스템과의 통신에 장애가 발생한 후 리모트 시스템에서 작업 수행 트랜잭션을 처리하기 위한 것입니다.

확약 및 롤백 강제 실행 시점 및 재동기화 취소 시점
여기에서는 언제 그리고 어떻게 롤백 또는 확약을 강제 실행하는지 및 언제 재동기화를 취소하는지에 대해 설명합니다.

확약 제어 오류

확약 제어를 사용할 때 오류를 발생시키는 조건 및 그렇지 않은 조건을 이해하는 것이 중요합니다. 일반적으로 오류는 확약 정의를 사용하는 파일이 아직 열려 있을 때 확약 제어 종료(ENDCMTCTL) 명령을 실행하는 것과 같이 확약 제어 기능이 일관성 없이 사용될 때 발생합니다.

확약 처리 중의 오류

확약 조작 중에 통신 또는 시스템 장애가 발생한 경우 트랜잭션 관리자가 트랜잭션에 관여한 모든 시스템에서의 데이터 일관성을 유지할 수 있도록 재동기화를 수행해야 할 것입니다. 재동기화의 작동 기능 및 확약 조작에 영향을 미치는 방식은 다음과 같은 요소에 따라 달라집니다.

- 출력 대기 확약 옵션. 자세한 내용은 2단계 확약에 대한 확약 정의: 출력을 기다리지 않음을 참조하십시오.

- 트랜잭션 상태. 자세한 내용은 2단계 확약 제어에 대한 트랜잭션 상태를 참조하십시오.

장애가 복구할 수 없을 정도로 심각한 경우이거나 적시에 복구되기 어려운 경우 트랜잭션에 관계된 다른 시스템에 대한 시스템 오퍼레이터는 발견적(heuristic) 결정을 내려야 할 것입니다. 발견적(heuristic) 결정은 트랜잭션 중에 해당 시스템에서 변경된 내용을 확약하거나 롤백합니다. 그러한 결정 후 장애가 복구되고 이러한 결정으로 자료 무결성 문제가 발생했음이 재동기화에서 감지된 경우 QSYSOPR 메세지 대기행렬로 메세지 CPD83D9 또는 CPD83E9가 송신됩니다.

다음 정보는 확약 제어 오류에 대한 작업 세부사항을 제공합니다.

- 오류 조건
- 비 오류 조건
- 확약 제어 중에 모니터할 오류 메세지
- CALL 명령 후 모니터할 오류 메세지
- 정상적인 확약 또는 롤백 처리 실패

오류 조건

오류가 발생하면 프로그램에서 모니터할 수 있는 이탈 메세지가 송신됩니다. 다음은 확약 제어와 관련된 일반적인 오류입니다.

- 연속적인 STRCMTCTL 명령이 중간에 ENDCMTCTL 명령 개입 없이 실행됩니다.
- 파일이 확약 제어 하에 열리지만 STRCMTCTL 명령은 실행되지 않습니다.

  이것은 작업 레벨 확약 정의를 사용하는 활성 그룹 내에서 실행되는 프로그램에 대한 오류 조건이 아닙니다. 작업 레벨 확약 정의는 단일 프로그램에 의해서만 시작될 수 있지만 일단 프로그램에 의해 시작되면 작업 레벨 확약 정의는 활성 그룹 레벨 확약 정의를 사용하지 않는 활성 그룹에서 실행되는 프로그램에 의해 사용됩니다. 활성 그룹 레벨 확약 정의를 사용하는 활성 그룹 내에서 실행되는 프로그램은 먼저 STRCMTCTL 명령을 사용하여 활성 그룹 레벨 확약 정의를 시작해야 합니다.
- 확약 제어 하에서 출력을 위해 열린 파일은 저널되지 않습니다.
- 공유 파일의 첫 번째 열기 조작은 파일을 확약 제어 하에 위치시키지만 동일한 공유 파일의 후속 열기 조작은 그렇지 않습니다.
- 공유 파일의 첫 번째 열기 조작은 해당 파일을 확약 제어 하에 위치시키지 않지만 동일한 공유 파일의 후속 열기 조작은 그렇지 않습니다.
- 단일 트랜잭션에서 작업에 대한 레코드 잠금 한계에 도달했습니다.
- 프로그램에서 동일한 레코드에 대해 읽기 조작, 확약 조작 및 변경을 발행합니다. 확약 조작으로 레코드에 대한 잠금이 해제되므로 확약 조작 후에 읽기 조작을 다시 발행해야 합니다.
- 1단계 위치의 경우 확약 제어 하의 자원은 이미 확약 정의를 위해 확약 제어 하에 있는 자원과 같은 위치에 상주하지 않습니다.
- ENDCMTCTL 명령이 발행될 때 미확약 변경이 존재합니다.

  모든 파일이 닫히고 리모트 데이터베이스가 단절되고 종료될 확약 정의와 연관된 API 확약 자원이 없다면 이는 ENDCMTCTL 명령에 대한 오류 조건이 아닙니다.

- 확약, 롤백 또는 ENDCMTCTL 명령이 실행되고 STRCMTCTL 명령은 실행되지 않았습니다.

  활성 그룹 내에서 실행되는 프로그램의 경우 이것은 오류 조건이 아니며 작업 레벨 확약 정의가 활성 상태입니다. 작업 레벨 확약 정의는 단일 프로그램에 의해서만 시작될 수 있지만 일단 프로그램에 의해 시작되면 작업 레벨 확약 정의는 활성 그룹 레벨 확약 정의를 사용하지 않는 활성 그룹에서 실행되는 프로그램에 의해 사용됩니다. 활성 그룹 내에서 실행되고 활성 그룹 레벨 확약 정의를 사용하는 프로그램은 먼저 STRCMTCTL 명령을 사용하여 활성 그룹 레벨 확약 정의를 시작해야 합니다.
- ENDCMTCTL 명령이 실행되고 파일은 확약 정의에 대한 확약 제어 하에서 열려 있습니다.
- 저장 조작을 수행하는 작업에 확약 경계에 있지 않은 하나 이상의 확약 정의가 있습니다.
- 확약 가능한 자원을 갖는 다른 작업이 SAVACTWAIT 매개변수에 지정된 시간 내에 확약 경계에 도달하지 못하여 활동 중 보관 조작이 종료했습니다.
- API 확약 가능한 자원이 단일 작업에 대한 하나 이상의 확약 정의에 추가되었기 때문에 활동 중 보관 프로세스가 계속될 수 없습니다.
- 단일 작업에 대하여 1023개를 초과하는 확약 정의가 존재합니다.
- 자원 실패로 인해 리모트 위치로의 대화가 유실되었습니다. 이로 인해 트랜잭션이 롤백될 수 있습니다.
- 갱신을 위해 열린 1단계 자원이 확약 조작을 시작하지 않은 노드에 있습니다. 확약 요구를 시작한 자원 또는 노드 중 하나를 제거해야 합니다.
- 트랜잭션이 롤백 요청(RBR) 상태인 동안 확약 조작이 요구됩니다. 롤백 조작을 수행해야 합니다.
- API 종료 프로그램이 확약 요구 또는 롤백 요구를 발행합니다.
- 트리거 프로그램에서 그 트리거 프로그램이 호출된 확약 정의에 대한 확약 요구 또는 롤백 요구를 발행합니다.

트리거 프로그램은 별도의 확약 정의를 시작하고 그 정의에 대한 확약 또는 롤백 요구를 발행할 수 있습니다.

오류가 아닌 조건

다음과 같은 상황에서 오류 메세지가 발생할 것이라고 예상하기 쉽습니다. 그러나 확약 제어에는 이러한 상황이 허용됩니다. 다음은 확약 제어에서 오류가 발생하지 않는 상황입니다.

- 확약 또는 롤백 조작이 실행되고 확약 제어 하에 자원이 없습니다. 그러면 사용자는 확약 제어 하에 자원이 있는지 여부를 고려하지 않고 프로그램에 확약 또는 롤백 조작을 포함시킬 수 있습니다. 또한 확약 가능 변경을 하기 전에 확약 식별을 지정할 수 있습니다.
- 확약 또는 롤백 조작이 실행되고 미확약 자원 변경이 없습니다. 그러면 사용자는 미확약 자원 변경이 있는지 여부를 고려하지 않고 프로그램에 확약 또는 롤백 조작을 포함시킬 수 있습니다.
- 확약 제어 하의 파일이 닫히고 미확약 레코드가 존재합니다. 이 상황에서는 확약이나 롤백 조작을 수행하기 위해 다른 프로그램을 호출할 수 있습니다. 이것은 파일 공유 여부와 관계없이 발생합니다. 이 기능을 사용하여 서브프로그램은 복수의 프로그램이 포함되는 트랜잭션의 일부인 데이터베이스 변경을 수행할 수 있습니다.
- 작업이 정상적으로 또는 비정상적으로 종료했고, 하나 이상의 확약 정의에 대해 미확약 변경이 있습니다. 모든 확약 정의에 대한 변경이 롤백됩니다.

- 활성 그룹이 종료하고 활성 그룹 레벨의 확약 정의에 대한 지연 변경이 있습니다. 활성 그룹이 정상적으로 종료하고 동일한 활성 그룹 범위의 확약 제어 하에서 열려 있는 파일을 닫는 경우 시스템에 의해 내재적 확약이 수행됩니다. 그렇지 않으면 내재적 롤백이 수행됩니다.
- 프로그램이 확약되지 않은 변경된 레코드에 다시 액세스합니다. 이를 통해 프로그램은 다음을 수행할 수 있습니다.
  - 확약 조작을 지정하기 전에 레코드를 추가하고 갱신합니다.
  - 확약 조작을 지정하기 전에 동일한 레코드를 두 번 갱신합니다.
  - 확약 조작을 지정하기 전에 레코드를 추가하고 삭제합니다.
  - (확약 제어 하의) 다른 논리 파일로 미확약 레코드에 다시 액세스합니다.
- STRCMTCTL 명령에 LCKLVL(*CHG 또는 *CS)을 지정하고 읽기 전용에 대한 확약 조작을 사용하여 파일을 엽니다. 이 경우 요구에 대해 잠금이 발생하지 않습니다. 이것은 확약 제어가 유효하지 않은 것처럼 처리되지만 파일이 확약 제어 하에 있는 파일의 WRKJOB 메뉴 옵션에 나타나지 않습니다.
- STRCMTCTL 명령을 발행하고 확약 제어 하의 파일을 전혀 열지 않습니다. 이 상황에서 확약 제어 하에서 파일에 대한 레코드 레벨의 변경은 이루어지지 않습니다.

확약 제어 중 모니터하는 오류 메세지

메세지 종류 및 오류 발생 시점에 따라 확약 또는 롤백 조작에서 여러 다양한 오류 메세지가 리턴되거나 작업 기록부로 보내집니다.

이 메세지들은 다음과 같은 처리 중에 발생할 수 있습니다.

- 정상적인 확약 또는 롤백 처리
- 작업 처리 종료 시 확약 또는 롤백 처리
- 활성 그룹 종료 시 확약 또는 롤백 처리

활성 그룹 종료 또는 작업 프로세스 종료 시에는 다음과 같은 메세지를 모니터할 수 없습니다. CPFxxxx 메세지에 대해서만 모니터할 수 있습니다. CPDxxxx 메세지는 항상 진단 메세지로서 송신되며 모니터할 수 없습니다. 활성 그룹 종료 중의 활성 그룹 레벨 확약 정의 종료 시 또는 작업 종료 중의 확약 정의 시에 발생한 오류는 진단 메세지로 작업 기록부에 남게 됩니다.

확약 제어에 관련된 오류 메시지는 다음과 같습니다.

CPD8351
변경이 확약되지 않았을 수 있습니다.

CPD8352
리모트 위치 &3에서 변경이 확약되지 않았습니다.

CPD8353
관계형 데이터베이스 &1에 대한 변경이 확약되지 않았을 수 있습니다.

CPD8354
DDM 파일 &1에 대한 변경이 확약되지 않았을 수 있습니다.

CPD8355
DDL 오브젝트 &1에 대한 변경이 확약되지 않았을 수 있습니다.

CPD8356
롤백된 변경이 확약되었을 수 있습니다.

CPD8358
관계형 데이터베이스 &1에 대한 변경이 롤백되지 않았을 수 있습니다.

CPD8359
DDM 파일 &1에 대한 변경이 롤백되지 않았을 수 있습니다.

CPD835A
DDL 오브젝트 &3에 대한 변경이 롤백되지 않았을 수 있습니다.

CPD835C
&2의 &1 통지 오브젝트가 갱신되지 않았습니다.

CPD835D
DRDA 자원이 SQL 커서 보류를 허용하지 않습니다.

CPF835F
확약 또는 롤백 조작이 실패했습니다.

CPD8360
멤버 또는 파일 또는 이들 모두가 이미 할당 해제되었습니다.

CPD8361
&1 API 종료 프로그램이 확약 중에 실패했습니다.

CPD8362
&1 API 종료 프로그램이 롤백 중에 실패했습니다.

CPD8363
&1 API 종료 프로그램이 확약 중에 &4분 후에 종료되었습니다.

CPD8364
&1 API 종료 프로그램이 롤백 중에 &4분 후에 종료되었습니다.

CPD836F
확약 제어 조작 중에 프로토콜 오류가 발생했습니다.

CPD83D1
&4 API 자원은 최종 에이전트가 될 수 없습니다.

CPD83D2
확약 제어와 호환되지 않는 자원입니다.

CPD83D7
확약 조작이 롤백으로 변경되었습니다.

CPD83D9
발견적(heuristic) 혼합 조건이 발생했습니다.

CPF83DB
확약 조작 결과 롤백이 실행되었습니다.

CPD83DC
확약 또는 롤백 조작 결정에 문제 발생 시 조치 사용. 이유 &2.

CPD83DD
대화가 종료되었습니다. 이유 &1.

CPD83DE
리턴 정보가 유효하지 않습니다.

CPD83EC
&1 API 종료 프로그램이 롤백을 인가했습니다.

CPD83EF
롤백 조작이 다음 논리 작업 단위에 대해 시작되었습니다.

CPF8350
확약 정의를 찾을 수 없습니다.

CPF8355
ENDCMTCTL이 허용되지 않습니다. 지연 변경이 활동 상태입니다.

CPF8356
&1 로컬 변경이 확약되지 않고 확약 제어가 종료되었습니다.

CPF8358
&2의 &1 통지 오브젝트가 갱신되지 않았습니다.

CPF8359
롤백 조작이 실패했습니다.

CPF835A
확약 정의 &1의 종료가 취소되었습니다.

CPF835B
확약 제어 종료 중에 오류가 발생했습니다.

CPF835C
리모트 변경이 확약되지 않고 확약 제어가 종료되었습니다.

CPF8363
확약 조작이 실패했습니다.

CPF8364
확약 제어 매개변수 값이 유효하지 않습니다. 이유 코드 &3.

CPF8367
확약 제어 조작을 수행할 수 없습니다.

CPF8369
확약 제어 하에 API 확약 자원을 위치시킬 수 없습니다. 이유 코드 &1.

CPF83D0
확약 조작이 허용되지 않습니다.

CPF83D2
확약 완료 == 재동기화 진행 중이 리턴되었습니다.

CPF83D3
확약 완료 == 발견적(heuristic) 혼합이 리턴되었습니다.

CPF83D4
논리적 작업 단위 저널 항목이 송신되지 않았습니다.

CPF83E1
제한 위반으로 확약 조작이 실패했습니다.

CPF83E2
롤백 조작이 필요합니다.

CPF83E3
요구되는 네스팅 레벨이 활동 상태가 아닙니다.

CPF83E4
자원이 확약되지 않고 확약 제어가 종료되었습니다.

CPF83E6
확약 제어 조작이 완료되고 재동기화가 진행 중입니다.

CPF83E7
X/Open 글로벌 트랜잭션의 확약 또는 롤백이 허용되지 않습니다.

CALL 명령 후 오류 모니터

확약 제어를 사용하는 프로그램이 호출될 때 예기치 않은 오류가 발생하지 않았는지 모니터하고 오류가 발생한 경우 롤백 조작을 수행해야 합니다. 예를 들어 프로그램에서 RPG의 0으로 나누기 오류와 같은 예기치 않은 오류가 발생한 경우 미확약 레코드가 존재할 수 있습니다. 작업에 대한 INQMSGRPY(조회 메세지 응답) 매개변수의 상태에 따라 프로그램은 조회 메세지를 송신하거나 디폴트 조치를 수행합니다. 오퍼레이터가 응답하거나 디폴트 조치로 프로그램이 종료된 경우 미확약 레코드가 여전히 확약 또는 롤백 조작을 기다리며 존재합니다.

다른 프로그램이 호출되어 확약 조작을 발생시킨 경우 이전 프로그램에서 일부만 완료된 트랜잭션이 확약됩니다.

일부만 완료된 트랜잭션이 확약되는 것을 방지하려면 CALL 명령 후에 이탈 메세지를 모니터하십시오. 예를 들어 RPG 프로그램의 경우 다음 코딩을 사용하십시오.

```
CALL RPGA
MONMSG MSGID(RPG9001)
EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

COBOL 프로그램인 경우는 다음과 같습니다.

```
CALL COBOLA
MONMSG MSGID(CBE9001)
EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

정상적인 확약 또는 롤백 처리 실패

확약 또는 롤백 처리 중에 오류가 발생할 수 있습니다. 다음 표에서는 이러한 처리를 네 가지 상황으로 나누어 보여줍니다. 가운데 열에서는 각 상황에서 오류가 발생했을 때 시스템이 취하는 조치를 설명합니다. 세 번째 열에서는 메세지에 대응하여 사용자 또는 어플리케이션에서 해야 하는 일을 제시해 놓았습니다. 이는 확약 제어가 시스템에 의해 처리되는 방식과 일치합니다.

상황	확약 또는 롤백 처리	제안 조치
레코드 레벨 I/O **확약** 실패	• 준비 웨이브 중에 오류가 발생하면 트랜잭션이 롤백되고 메세지 CPF83DB가 송신됩니다. • 확약된 웨이브 중에 오류가 발생하는 경우 확약 처리는 계속해서 가능한 많이 남아 있는 자원을 확약합니다. 확약 처리 끝에 메세지 CPF8363이 송신됩니다.	메세지에 대한 모니터. 처리가 필요함.

상황	확약 또는 롤백 처리	제안 조치
API 확약 자원에 대한 오브젝트 레벨 또는 확약 및 롤백 종료 프로그램이 확약 중에 실패했습니다.	• 준비 웨이브 중에 오류가 발생하면 트랜잭션이 롤백되고 메세지 CPF83DB가 송신됩니다. • 확약된 웨이브 중에 오류가 발생하는 경우 확약 처리는 계속해서 가능한 많이 남아 있는 자원을 확약합니다. 확약 자원 유형에 따라 다음 메세지 중 하나가 리턴됩니다. – CPD8353 – CPD8354 – CPD8355 – CPD8361 확약 처리 종료 시에 메세지 CPF8363이 송신됩니다.	메세지에 대한 모니터. 처리가 필요함.
레코드 레벨 I/O 롤백 실패	1. CPD8356이 리턴됩니다. 2. 오브젝트 레벨 또는 API 확약 자원에 대한 롤백 처리를 계속하려고 시도합니다. 3. 처리 종료 시 CPF8359를 리턴합니다.	메세지에 대한 모니터. 처리가 필요함.
롤백 중에 API 확약 자원에 대한 오브젝트 레벨 또는 확약 및 롤백 종료 프로그램이 실패했습니다.	1. 확약 자원 유형에 따라 다음 메세지 중 하나가 리턴됩니다. • CPD8358 • CPD8359 • CPD835A • CPD8362 2. 처리를 계속합니다. 3. 처리 종료 시 CPF8359를 리턴합니다.	메세지에 대한 모니터. 처리가 필요함.

작업 종료 중에 확약 또는 롤백 처리

앞의 표에서 설명된 상황은 모두 작업이 종료될 때에도 적용되는데 한 가지 예외는 다음 메세지 중 하나가 송신된다는 점입니다.

- 로컬 자원만 등록되어 있는 경우 CPF8356
- 리모트 자원만 등록되어 있는 경우 CPF835C
- 로컬 및 리모트 자원이 모두 등록되어 있는 경우 CPF83E4

또한 API 확약 가능 자원에 대한 확약 및 롤백 종료 프로그램이 호출된 경우 두 메세지 중 하나가 작업 완료와 관련되어 표시될 수도 있습니다. 확약 및 롤백 종료 프로그램이 5분 안에 완료되지 않으면 프로그램은 취소되고 진단 메세지 CPD8363(확약) 또는 CPD8364(롤백)가 보내지며 확약 또는 롤백의 나머지 처리가 계속됩니다.

IPL 시의 확약 또는 롤백 처리

앞의 표에서 설명된 상황은 모두 확약 정의에 대한 IPL 회복 시에도 적용되는데, 단 메세지 CPF835F가 메세지 CPF8359 또는 CPF8363 대신 송신됩니다. 특정 확약 정의에 대하여 송신된 메세지는 QDBSRVxx 작업

중 하나에 대한 작업 기록부나 QHST 기록부에 들어 갑니다. QHST 기록부에서 CPI8356 메세지는 특정 확약 정의에 대한 IPL 회복 시작을 나타냅니다. 메세지 CPC8351은 특정 확약 정의에 대한 IPL 회복 종료를 나타내며 확약 정의 회복에 대한 다른 메세지도 이 두 메세지 사이에서 찾아볼 수 있습니다.

API 확약 가능 자원에 대한 확약 및 롤백 종료 프로그램이 호출되면 두 메세지 중 하나는 확약 정의와 관련되어 나타날 수 있습니다. 확약 및 롤백 종료 프로그램이 5분 안에 완료되지 않으면 프로그램은 취소되고 진단 메세지 CPD8363(확약) 또는 CPD8364(롤백)가 보내지며 확약 또는 롤백의 나머지 처리가 계속됩니다.

교착 상태 감지

교착 상태 조건은 작업이 하나의 오브젝트인 오브젝트 A를 잠그고 다시 다른 오브젝트인 오브젝트 B에 대한 잠금을 확보하기 위해 기다리고 있을 때 발생할 수 있습니다. 동시에 오브젝트 B에 대한 잠금을 확보하고 있는 다른 작업이나 트랜잭션이 오브젝트 A에 대한 잠금을 확보하기 위해 기다리고 있는 경우입니다.

다음 단계를 수행하여 교착 상태 조건이 발생했는지 확인하고 다음과 같은 경우 이를 수정하십시오.

1. 활성 작업 리스트에서 일시 중단된 작업을 찾으십시오. 작업이 일시 중지되었는지 판별하려면 작업 상태 판별을 참조하십시오.
2. 작업이 잠금을 확보하기 위해 기다리고 있는 오브젝트를 찾아 보십시오. 트랜잭션 범위의 잠금을 수행하는 트랜잭션의 경우 트랜잭션에 대해 잠금이 수행된 오브젝트 표시에서 실행 단계를 참조하십시오. 작업 범위의 트랜잭션의 경우 세부사항: 활성 작업 등록 정보를 참조하십시오.
3. 작업이 잠금을 확보하려고 기다리고 있는 모든 오브젝트에 대하여 현재 잠금을 확보하고 있는 트랜잭션이나 작업의 리스트를 찾아보고 일시 중단된 작업이 요구한 레벨에 해당되는 충돌하고 있는 잠금을 찾아 보십시오.
4. 한 트랜잭션이 충돌하고 있는 잠금을 확보하고 있다면 이 트랜잭션과 연관된 작업을 표시하고 그 중 하나가 잠금을 기다리고 있는지 확인하십시오.
5. 대기 중인 이 작업이 초기의 일시 중단 작업에 의해 잠긴 오브젝트 중 하나를 잠그려하는지 여부를 판별하십시오. 초기의 일시 중단 작업에 의해 잠금된 오브젝트를 잠그려 하는 작업을 발견하는 경우 해당 오브젝트를 문제 지점으로 식별할 수 있습니다.
6. 트랜잭션을 조사하여 적절한 조치를 결정하십시오.
   a. 트랜잭션 등록 정보를 살펴 보고 어떤 어플리케이션이 그 트랜잭션을 시작했는지 찾아 본 후 어플리케이션 코드를 검토하십시오.
   b. 또는 트랜잭션 등록 정보에서 확약 주기 ID를 찾아 이 ID를 포함하는 항목에 대한 저널에서 탐색하여 이 지점까지의 트랜잭션 조치를 추적하십시오. 이를 위해서는 저널 항목 검색(RTVJRNE) 명령을 사용하여 CMTCYCID 매개변수를 지정할 수 있습니다.
   c. 적절한 정보를 얻은 후 사용자는 롤백 또는 확약 강제 실행 조치를 선택할 수 있습니다.

통신 장애 후 트랜잭션 회복

통신 장애가 발생한 경우 일반적으로 시스템은 자동으로 리모트 시스템과의 재동기화를 완료합니다. 그러나 리모트 시스템과의 통신이 다시 설정될 수 없는 경우와 같이 장애가 심각한 경우(예를 들면 통신 회선이 절단된 경우) 재동기화를 취소하고 트랜잭션을 직접 복원해야 합니다. 트랜잭션은 릴리스되어야 하는 잠금을 가지고 있는 경우도 있습니다.

1. iSeries Navigator 화면에서 작업 중인 트랜잭션에 대하여 확약 제어 정보를 표시하십시오.
2. 리모트 시스템과 재동기화시키려 하는 트랜잭션을 찾으십시오. 그 트랜잭션의 **진행 중인 재동기화** 필드를 **예**로 설정합니다.
3. 개별적인 트랜잭션의 자원 상태를 점검하여 리모트 시스템에 연결되어 있던 트랜잭션을 찾으십시오.
4. 트랜잭션 식별 후에 트랜잭션 상태에 따라 강제 확약 또는 강제 롤백을 해야 할 수도 있습니다.
5. 트랜잭션 등록 정보를 검토한 후 확약 또는 롤백을 결정할 수도 있습니다.
   - **작업 단위 ID**를 사용하여 다른 시스템에서 트랜잭션의 다른 부분을 찾을 수 있습니다.
   - 또한 트랜잭션 상태를 통해 확약 또는 롤백을 결정할 수도 있습니다. 예를 들어 데이터베이스 트랜잭션이 통신 장애 중에 2단계 확약을 수행하고 있었고 장애 후 상태가 ″준비″ 또는 ″최종 에이전트 지연″인 경우 그 트랜잭션에 대해 강제 확약을 실행하도록 선택할 수도 있습니다.
6. 문제의 트랜잭션에 대해 강제로 확약 또는 롤백을 수행한 후 장애가 발생한 그 트랜잭션에 대한 연결에서 재동기화를 중단하십시오.

회복에 대한 자세한 내용은 확약 및 롤백 강제 실행 시점 및 재동기화 취소 시점을 참조하십시오.

확약 및 롤백 강제 실행 시점 및 재동기화 취소 시점

확약, 롤백을 강제 실행하거나 재동기화를 취소하기 위한 결정을 **발견적(heuristic) 결정**이라고 합니다. 발견적(heuristic) 결정은 시스템이 강제로 트랜잭션을 확약하거나 롤백하도록 할 때 취하는 조치입니다. 발견적(heuristic) 결정을 내릴 때 결정이 트랜잭션의 다른 위치의 결과와 일치하지 않는 경우 트랜잭션은 발견적 혼합 상태가 됩니다. 사용자는 트랜잭션에 참여한 다른 모든 위치에 의해 취해진 조치에 대한 결정 및 데이터베이스 레코드 재동기화에 대한 책임을 져야 합니다.

발견적(heuristic) 결정을 내리기 전에 트랜잭션에 대해 가능한 많은 정보를 수집하십시오. 확약 정의에 연관된 작업을 표시하고 관련된 저널 및 파일이 무엇인지 기록하십시오. 이 정보는 나중에 저널 항목을 표시하고 저널된 변경사항을 수작업으로 적용 또는 제거해야 할 때 사용할 수 있습니다.

트랜잭션에 대한 정보를 찾을 수 있는 가장 좋은 위치는 해당 트랜잭션의 개시자인 위치를 찾아 보는 것입니다. 그러나 확약 또는 롤백 결정은 API 자원 또는 마지막 에이전트가 소유할 수 있습니다.

API 자원이 마지막 에이전트 자원으로 등록된 경우 확약 또는 롤백 여부에 대한 최종 결정은 API 자원이 소유합니다. 어플리케이션에 대한 정보 및 어플리케이션이 확약 또는 롤백에 대한 결정을 내리기 위해 API 자원을 사용하는 방식을 살펴 보아야 합니다.

트랜잭션이 마지막으로 선택된 에이전트를 가지고 있는 경우 마지막 에이전트가 확약 또는 롤백 결정을 소유합니다. 트랜잭션에 대한 정보는 마지막 에이전트의 상태를 살펴보아야 합니다.

복구할 수 없는 시스템 장애 또는 통신 장애로 인해 발견적(heuristic) 결정을 내리거나 재동기화를 취소해야 할 때 다음을 사용하여 확실하지 않은 트랜잭션을 모두 찾을 수 있습니다.

1. iSeries Navigator에서 작업하려는 시스템을 확장하십시오.
2. **데이터베이스** 및 해당 시스템의 로컬 데이터베이스를 확장하십시오.
3. **트랜잭션**을 확장하십시오.
4. **데이터베이스 트랜잭션** 또는 **글로벌 트랜잭션**을 확장하십시오.

이 화면에서 확약 정의, 재동기화 상태, 현재 작업 단위 ID 및 각 트랜잭션에 대한 현재 작업 단위 상태를 볼 수 있습니다. 다음과 같은 트랜잭션을 찾아 보십시오.

- **논리 작업 단위 상태**가 준비 또는 최종 에이전트 지연인 트랜잭션.
- **재동기화 진행 중** 상태가 예인 트랜잭션.

이 시스템에서 트랜잭션에 참여하고 있는 작업에 대해 작업하려면 해당 트랜잭션을 마우스 오른쪽 버튼으로 클릭하고 **작업**을 선택하십시오.

트랜잭션을 마우스 오른쪽 버튼으로 클릭하면 **확약 강제 실행**, **롤백 강제 실행** 또는 **재동기화 취소**도 선택할 수 있습니다.

발견적(heuristic) 결정을 내리거나 재동기화를 취소하기 전에 해당 트랜잭션과 연관된 다른 시스템에서의 작업 상태를 확인해보고자 할 수 있습니다. 리모트 시스템의 작업을 확인함으로써 시스템 간의 데이터베이스 불일치를 발생시킬 수 있는 결정을 방지하는 데 도움이 될 수 있습니다.

1. 작업하려는 트랜잭션을 마우스 오른쪽 버튼으로 클릭하십시오.
2. **자원 상태**를 선택하십시오.
3. 자원 상태 대화 상자에서 SNA 연결의 경우 **대화** 탭을 선택하고, TCP/IP 연결의 경우 **연결**을 선택하십시오.

각 대화 자원은 트랜잭션에 참여하고 있는 리모트 시스템을 나타냅니다. 리모트 시스템에서는 iSeries Navigator를 사용하여 해당 트랜잭션과 연관된 트랜잭션을 볼 수 있습니다.

작업 단위 ID의 기본 부분은 모든 시스템에서 동일합니다. 리모트 시스템에서 확약 제어 정보를 표시할 때 로컬 시스템에서 동일한 작업 단위 ID의 기본 부분을 찾아 보십시오.

예를 들어 로컬 시스템에서 작업 단위 ID가 `APPN.RCHASL97.X'112233445566`으로 시작하는 경우 역시 `APPN.RCHASL97.X'112233445566`으로 시작하는 작업 단위 ID를 리모트 시스템에서 찾아 보십시오.

확약 제어에 대한 관련 정보

아래에는 확약 제어 주제와 관련된 iSeries 매뉴얼 및 IBM 레드북(TM)(PDF 형식), 웹 사이트 및 Information Center 주제가 나열되어 있습니다. PDF를 보거나 인쇄할 수 있습니다.

매뉴얼

- COBOL/400 User's Guide (약 425 페이지)
- RPG/400 User's Guide (약 580 페이지)

레드북

- Connecting WebSphere to DB2 UDB Server (약 458 페이지)
- Advanced Functions and Administration for DB2 Universal Database for iSeries (약 372 페이지)
- Stored Procedures and Triggers on DB2 Universal Database for iSeries (약 424 페이지)
- Striving for Optimal Journal Performance on DB2 Universal Database for iSeries (약 184 페이지)

웹 사이트

The Open Group(www.opengroup.org)

기타 정보

- iSeries용 DB2 Universal Database 데이터베이스 프로그래밍
- iSeries용 DB2 Universal Database SQL 프로그래밍 개념
- XA API
- 저널 관리

보거나 인쇄하기 위해 워크스테이션에 PDF를 저장하려면 다음과 같이 하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽 버튼으로 클릭하십시오(위의 링크를 마우스 오른쪽 버튼으로 클릭).
2. **다른 이름으로 대상 저장...**을 클릭하십시오.
3. PDF를 저장하려는 디렉토리를 탐색하십시오.
4. **저장**을 클릭하십시오.

이 PDF를 보거나 인쇄하기 위해 Adobe Acrobat Reader가 필요한 경우 Adobe 웹 사이트 (www.adobe.com/products/acrobat/readstep.html)에서 사본을 다운로드할 수 있습니다.

Printed in U.S.A