



IBM Systems - iSeries  
Banco de Dados  
DB2 Multisystem

*Versão 5 Release 4*







IBM Systems - iSeries  
Banco de Dados  
DB2 Multisystem

*Versão 5 Release 4*

**Nota**

Antes de utilizar estas informações e o produto suportado por elas, leia as informações em “Avisos”, na página 63.

**Sétima Edição (Fevereiro de 2006)**

Esta edição se aplica à versão 5, release 4, modificação 0 do IBM i5/OS (número do produto 5722-SS1) e a todos os releases e modificações subsequentes, até que seja indicado o contrário em novas edições. Esta versão não é executada em todos os modelos RISC (Reduced Instruction Set Computer) nem é executada nos modelos CISC.

© Direitos Autorais International Business Machines Corporation 1998, 2006. Todos os direitos reservados.

# Índice

<b>DB2 Multisystem . . . . .</b>	<b>1</b>	NODENUMBER com o DB2 Multisystem . . . . .	37
O Que Há de Novo no V5R4 . . . . .	1	Registros Especiais com o DB2 Multisystem . . . . .	38
PDF Imprimível . . . . .	1	Desempenho e Escalabilidade com o DB2 Multisystem . . . . .	38
Introdução ao DB2 Multisystem . . . . .	2	Porque Você Deve Utilizar o DB2 Multisystem. . . . .	38
Benefícios da Utilização do DB2 Multisystem . . . . .	3	Como o DB2 Multisystem Ajuda a Expandir o Sistema de Banco de Dados . . . . .	40
DB2 Multisystem: Conceitos e Termos Básicos . . . . .	3	Design de Consulta para Desempenho com o DB2 Multisystem . . . . .	41
Introdução aos Grupos de Nós com o DB2 Multisystem . . . . .	5	Visão Geral da Otimização com o DB2 Multisystem . . . . .	42
Como os Grupos de Nós Funcionam com o DB2 Multisystem . . . . .	5	Implementação e Otimização de uma Consulta de Arquivo Único com o DB2 Multisystem . . . . .	42
Tarefas a Serem Concluídas Antes de Utilizar os Comandos de Grupo de Nós com o DB2 Multisystem . . . . .	6	Implementação e Otimização da Ordenação de Registros com o DB2 Multisystem . . . . .	44
Comando Criar Grupo de Nós (CRTNODGRP) . . . . .	6	Implementação e Otimização das Cláusulas UNION e DISTINCT com o DB2 Multisystem . . . . .	45
Comando Exibir Grupo de Nós (DSPNODGRP) . . . . .	8	Processamento dos Parâmetros DSTDTA e ALWCPYDTA com o DB2 Multisystem . . . . .	45
Comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) . . . . .	10	Implementação e Otimização de Operações de Junção com o DB2 Multisystem. . . . .	45
Comando Excluir Grupo de Nós (DLTNODGRP) . . . . .	11	Implementação e Otimização de Agrupamento com o DB2 Multisystem . . . . .	51
Arquivos Distribuídos com o DB2 Multisystem . . . . .	11	Suporte a Subconsulta com o DB2 Multisystem . . . . .	53
Comando Criar Arquivo Físico (CRTPF) e a Instrução SQL CREATE TABLE . . . . .	12	Planos de Acesso com o DB2 Multisystem . . . . .	53
Atividades do Sistema Depois que o Arquivo Distribuído É Criado . . . . .	14	Caminhos de Dados Abertos Reutilizáveis com o DB2 Multisystem . . . . .	53
Particionamento com o DB2 Multisystem . . . . .	21	Escritor de Resultado Temporário com o DB2 Multisystem . . . . .	55
Customização da Distribuição de Dados com o DB2 Multisystem . . . . .	23	Mensagens do Otimizador com o DB2 Multisystem . . . . .	57
Tabelas Particionadas . . . . .	24	Alterações no Comando Alterar Atributos da Consulta (CHGQRYA) com o DB2 Multisystem . . . . .	59
Criação de Tabelas Particionadas . . . . .	25	Resumo das Considerações sobre o Desempenho . . . . .	60
Modificação de Tabelas Existentes . . . . .	27	Informações Relacionadas ao DB2 Multisystem . . . . .	61
Índices com Tabelas Particionadas . . . . .	28	<b>Apêndice. Avisos . . . . .</b>	<b>63</b>
Desempenho e Otimização da Consulta . . . . .	29	Informações da Interface de Programação . . . . .	65
Considerações sobre Salvamento e Restauração . . . . .	33	Marcas Registradas . . . . .	65
Criação de Diário . . . . .	33	Termos e Condições . . . . .	65
Considerações sobre a Interface Nativa . . . . .	34		
Restrições . . . . .	34		
Funções Escalares Disponíveis com o DB2 Multisystem . . . . .	35		
PARTITION com o DB2 Multisystem . . . . .	35		
HASH com o DB2 Multisystem. . . . .	36		
NODENAME com o DB2 Multisystem . . . . .	36		



---

## DB2 Multisystem

Este tópico descreve os conceitos fundamentais do DB2 Multisystem, tal como arquivos de banco de dados relacional distribuído, grupos de nós e particionamento, e fornece as informações necessárias para criar e utilizar arquivos de banco de dados que são particionados por vários servidores iSeries. São fornecidas informações sobre como configurar os sistemas, como criar os arquivos e como os arquivos podem ser utilizados em aplicativos. Este tópico também descreve o particionamento de tabelas. O particionamento de tabelas é diferente do particionamento multissistema pelo fato de que ele é uma tabela particionada em um único servidor.

**Nota:** Utilizando os exemplos de código, você concorda com os termos do “Aviso de Isenção de Responsabilidade e Licença do Código” na página 62.



---

## O Que Há de Novo no V5R4

- | Este tópico realça algumas alterações no DB2 Multisystem para a V5R4.
- | O SQE (SQL Query Engine) fornece otimização direcionada a tabelas particionadas utilizando a otimização da expansão dinâmica da partição.

### Como saber o que é novo ou que foi alterado

Para ajudar a ver onde as alterações técnicas foram feitas, estas informações utilizam:

- A imagem  para marcar onde começam as informações novas ou alteradas.
- A imagem  para marcar onde terminam as informações novas ou alteradas.

Para localizar outras informações sobre as novidades ou alterações neste release, consulte Memorando para Usuários.

---

## PDF Imprimível

Utilize este documento para visualizar e imprimir um PDF destas informações.


Para visualizar ou fazer download da versão em PDF deste documento, selecione DB2 Multisystem (aproximadamente 869 KB).

### Salvando Arquivos PDF

Para salvar um PDF em sua estação de trabalho para exibição ou impressão:

1. Em seu navegador, clique com o botão direito do mouse no PDF (clique com o botão direito no link anterior).
- | 2. Clique na opção que salva o PDF localmente.
3. Navegue para o diretório no qual deseja salvar o PDF.
4. Clique em **Salvar**.

### Fazendo Download do Adobe Reader

- | É necessário ter o Adobe Reader instalado em seu sistema para visualizar ou imprimir esses PDFs. É possível fazer download de uma cópia gratuita no Web site da Adobe
- | ([www.adobe.com.br/products/acrobat/readstep.html](http://www.adobe.com.br/products/acrobat/readstep.html))  .

## Introdução ao DB2 Multisystem

O DB2 Multisystem é uma técnica de processamento paralelo que fornece maior escalabilidade para bancos de dados.

Utilizando o DB2 Multisystem, você tem a capacidade de conectar vários servidores iSeries (até 32 servidores) em conjunto em um cluster sem compartilhamento. (*Sem compartilhamento* significa que cada sistema na rede acoplada tem e gerencia sua própria memória principal e armazenamento em disco.) Assim que os sistemas são conectados, os arquivos do banco de dados podem ser propagados pelas unidades de armazenamento em cada sistema conectado. Os arquivos do banco de dados podem ter dados particionados (distribuídos) por um conjunto de sistemas e cada sistema tem acesso a todos os dados no arquivo. No entanto, para os usuários, o arquivo se comporta como um arquivo local em seu sistema. Da perspectiva do usuário, o banco de dados aparenta ser único: o usuário pode executar consultas em paralelo em todos os sistemas na rede e ter acesso em tempo real aos dados nos arquivos.

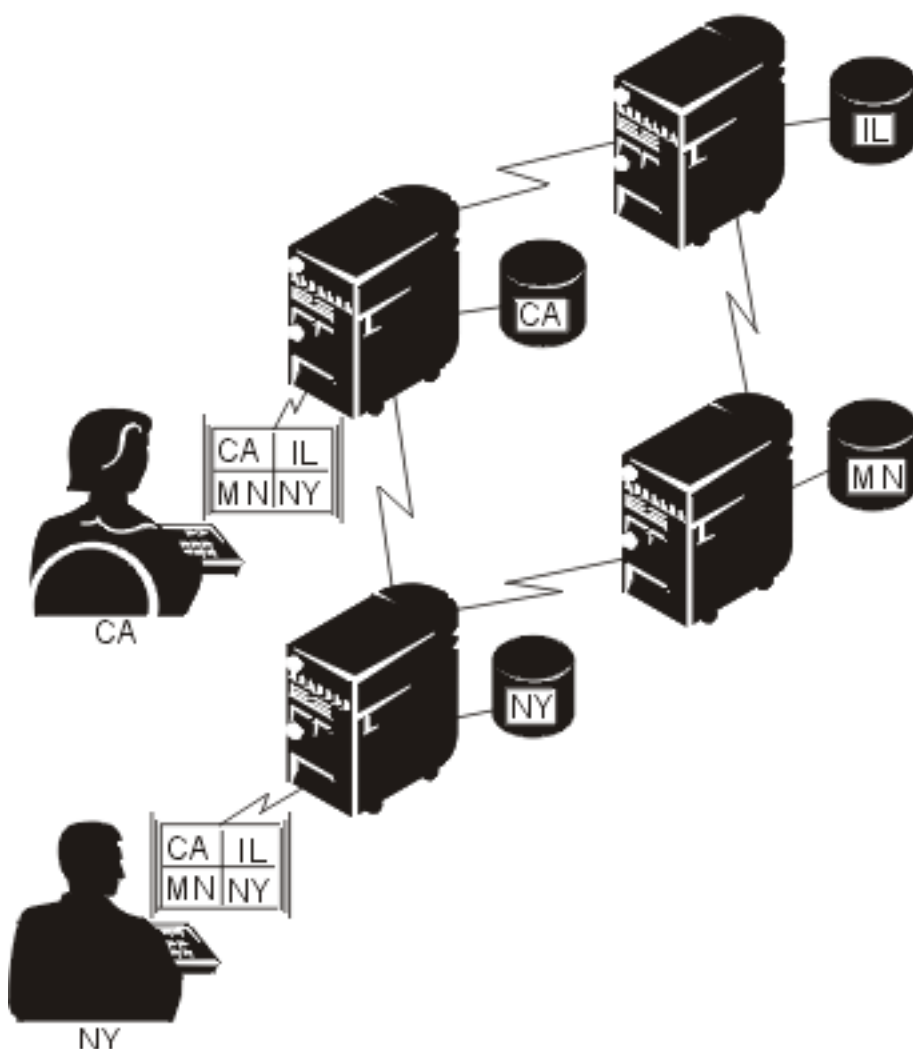


Figura 1. Distribuição de Arquivos do Banco de Dados pelos Sistemas

Essa técnica de processamento paralelo significa que um alto uso em um servidor não degrada o desempenho nos outros sistemas conectados à rede. Se tiver grandes volumes de dados e precisar executar consultas, o DB2 Multisystem fornece um método de execução dessas consultas que está entre os



mais eficientes disponíveis. Na maior parte dos casos, o desempenho da consulta é aprimorado porque as consultas não mais são executadas em relação a arquivos locais, mas em paralelo por vários servidores.

Se ainda não tiver instalado o DB2 Multisystem, consulte Instalar, Fazer Upgrade ou Excluir o i5/OS e o Software Relacionado para obter informações sobre a instalação de programas licenciados adicionais. Para instalar o DB2 Multisystem, utilize a opção 27 na lista de opções instaláveis para o sistema operacional.

## Benefícios da Utilização do DB2 Multisystem

Os benefícios da utilização do DB2 Multisystem incluem desempenho aprimorado de consultas, replicação de dados diminuída, maior capacidade do banco de dados e assim por diante.

É possível perceber os benefícios da utilização do DB2 Multisystem de diversas maneiras:

- O desempenho da consulta pode ser aprimorado pela execução em paralelo (partes da consulta são executadas simultaneamente em diferentes servidores)
- A necessidade por replicação de dados diminui porque todos os servidores podem acessar todos os dados
- É possível acomodar arquivos de banco de dados muito maiores
- Os aplicativos não mais se preocupam com o local dos dados remotos
- Quando for necessário crescimento, é possível redistribuir o arquivo por mais sistemas, e os aplicativos podem ser executados sem alteração nos novos sistemas

Com o DB2 Multisystem, é possível utilizar os mesmos métodos de E/S (Entrada/Saída), GETs, PUTs e UPDATES, ou métodos de acesso a arquivos utilizados no passado. Nenhum método de E/S adicional ou diferente nem métodos de acesso a arquivo são requeridos.

Os aplicativos não precisam ser alterados; quaisquer métodos de conectividade utilizados atualmente, exceto o OptiConnect, também funcionarão para quaisquer arquivos distribuídos que forem criados. Com o OptiConnect, é necessário utilizar as descrições de controlador do OptiConnect.

### Informações relacionadas

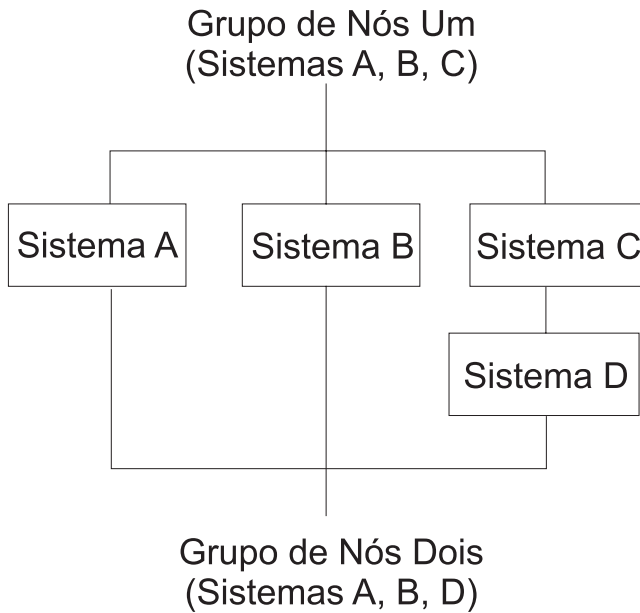
OptiConnect

## DB2 Multisystem: Conceitos e Termos Básicos

Um *arquivo distribuído* é um arquivo do banco de dados propagado por vários servidores iSeries. Esta seção descreve alguns dos conceitos principais utilizados na discussão da criação e utilização de arquivos distribuídos pelo DB2 Multisystem.

Cada servidor tem uma parte de um arquivo distribuído denominada *nó*. Cada servidor é identificado pelo nome que está definido para ele no diretório do banco de dados relacional.

Um grupo de sistemas que contém um ou mais arquivos distribuídos é denominado *grupo de nós*. Um *grupo de nós* é um objeto do sistema que contém a lista dos nós pelos quais os dados são distribuídos. Um sistema pode ser um nó em mais de um grupo de nós.



RBAL3509-1

Figura 2. Grupos de Nós

Um arquivo é distribuído por todos os sistemas em um grupo de nós utilizando-se *particionamento*. O *particionamento de tabelas*, descrito adicionalmente em Tabelas particionadas, aplica-se a tabelas particionadas em um único sistema.

Um *número de partição* é um número de 0 a 1.023. Cada número de partição é designado a um nó no grupo de nós. Cada nó pode receber vários números de partição. A correlação entre nós e números de partição é armazenada em um *mapa de partição*. O mapa de partição também é armazenado como parte do objeto do grupo de nós. É possível fornecer o mapa de partição ao criar um grupo de nós; caso contrário, o sistema gera um mapa padrão.

Um mapa de partição é definido utilizando-se um arquivo de particionamento. Um *arquivo de particionamento* é um arquivo físico que define um número de nó para cada número de partição.

Uma *chave de particionamento* consiste em um ou mais campos no arquivo que está sendo distribuído. A chave de particionamento é utilizada para determinar qual nó no grupo de nós deve conter fisicamente as linhas com determinados valores. Isso é feito utilizando-se *hash*, uma função do sistema operacional que obtém o valor da chave de particionamento de um registro e o mapeia para um número de partição. O nó correspondente a esse número de partição é utilizado para armazenar o registro.

O exemplo a seguir mostra quais números de partição e nós podem ser parecidos para uma tabela distribuída para dois sistemas. A tabela tem uma chave de particionamento igual a LASTNAME.

Tabela 1. Mapa de Partição

Número da Partição	Nó
0	SYSA
1	SYSB
2	SYSA
3	SYSB

No mapa de partição, o número de partição 0 contém SYSA, o número de partição 1 contém o nó SYSB, o número de partição 2 contém SYSA e o número de partição 3 contém SYSB. Esse padrão é repetido.

O hash da chave de particionamento determina um número que corresponde a um número de partição. Por exemplo, um registro que tenha o valor Andrews pode ter hash na partição número 1. Um registro que tenha o valor Anderson pode ter hash na partição número 2. Se você consultar o mapa de partição mostrado na Tabela 1 na página 4, os registros da partição número 1 são armazenados em SYSB, enquanto os registros da partição número 2 são armazenados em SYSA.

#### **Conceitos relacionados**

“Tabelas Particionadas” na página 24

O DB2 UDB para iSeries suporta tabelas particionadas utilizando SQL.

---

## **Introdução aos Grupos de Nós com o DB2 Multisystem**

Para permitir que os arquivos do banco de dados sejam visíveis em um conjunto de servidores iSeries, primeiro é necessário definir o grupo de sistemas (grupo de nós) no qual deseja colocar os arquivos.

Um grupo de nós pode ter de 2 a 32 nós definidos para ele. O número de nós definido para o grupo de nós determina o número de sistemas pelos quais o arquivo do banco de dados será criado. O sistema local deve ser um dos sistemas especificados no grupo de nós. Quando o sistema criar o grupo de nós, o sistema designa um número, começando pelo número 1, para cada nó.

## **Como os Grupos de Nós Funcionam com o DB2 Multisystem**

Um *grupo de nós* é um objeto do sistema (\*NODGRP) armazenado no sistema em que ele foi criado.

Um grupo de nós não é um objeto distribuído. O objeto do sistema \*NODGRP contém todas as informações sobre os sistemas no grupo, além de informações sobre como os dados nos arquivos de dados devem ser particionados (distribuídos). O particionamento padrão é que cada sistema (nó) receba um compartilhamento igual dos dados.

O particionamento é manipulado utilizando um algoritmo hash. Quando um grupo de nós for criado, os números de partição entre 0 e 1023 são associados ao grupo de nós. Com o particionamento padrão, um número igual de partições é designado a cada um dos nós no grupo de nós. Quando os dados são incluídos no arquivo, é feito hash dos dados na chave de particionamento, o que resulta em um número de partição. Não é feito hash do registro inteiro dos dados - apenas os dados na chave de particionamento passam pelo algoritmo hash. O nó associado ao número de partição resultante é onde o registro dos dados reside fisicamente. Portanto, com o particionamento padrão, cada nó armazena um compartilhamento igual dos dados, desde que haja registros suficientes de dados e um grande intervalo de valores.

Se não deseja que cada nó receba um compartilhamento igual dos dados ou se deseja controlar quais sistemas terão partes específicas dos dados, é possível alterar como os dados são particionados, especificando-se um esquema de particionamento personalizado no comando Criar Grupo de Nós (CRTNODGRP) com o parâmetro arquivo de partição (PTNFILE) ou alterando-se posteriormente o esquema de particionamento utilizando o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA). Utilizando o parâmetro PTNFILE, é possível configurar o número do nó para cada uma das partições no grupo de nós; em outras palavras, o parâmetro PTNFILE permite ajustar como deseja que os dados sejam particionados nos sistemas no grupo de nós. (O parâmetro PTNFILE é utilizado em um exemplo em Criando grupos de nós utilizando o comando CRTNODGRP com o DB2 Multisystem.) Para obter informações adicionais sobre particionamento, consulte Particionamento com o DB2 Multisystem.

Como um grupo de nós é um objeto do sistema, ele pode ser salvo e restaurado utilizando o comando Salvar Objeto (SAVOBJ) e o comando Restaurar Objeto (RSTOBJ). É possível restaurar um objeto de grupo de nós para o sistema no qual ele foi criado ou para qualquer um dos sistemas no grupo de nós. Se o objeto de grupo de nós for restaurado para um sistema que não esteja no grupo de nós, o objeto será inutilizável.

#### **Conceitos relacionados**

“Comando Criar Grupo de Nós (CRTNODGRP)”

Esta seção utiliza dois exemplos de comando da CL para mostrar como criar um grupo de nós utilizando o comando Criar Grupo de Nós (CRTNODGRP).

“Particionamento com o DB2 Multisystem” na página 21

*Particionamento* é o processo de distribuição de um arquivo pelos nós em um grupo de nós.

## Tarefas a Serem Concluídas Antes de Utilizar os Comandos de Grupo de Nós com o DB2 Multisystem

Antes de utilizar o comando Criar Grupo de Nós (CRTNODGRP) ou qualquer um dos comandos de grupo de nós, é necessário assegurar-se de que a rede do banco de dados relacional distribuído utilizado foi configurado apropriadamente.

Se essa for uma nova rede de banco de dados relacional distribuído, consulte Programação do Banco de Dados Distribuído para obter informações sobre como estabelecer a rede.

É necessário assegurar-se de que um sistema na rede está definido como o sistema local (\*LOCAL). Utilize o comando Trabalhar com Entradas do Diretório do RDB - Banco de Dados Relacional (WRKRDBDIRE) para exibir os detalhes sobre as entradas. Se um sistema local não estiver definido, isso poderá ser feito especificando-se \*LOCAL para o parâmetro nome do local remoto (RMTLOCNAME) do comando Incluir Entradas do Diretório do RDB (ADDRDBDIRE), por exemplo:

```
ADDRDBDIRE RDB(MP000) RMTLOCNAME(*LOCAL) TEXT ('New York')
```

O servidor iSeries em New York, denominado MP000, é definido como o sistema local no diretório do banco de dados relacional. É possível ter apenas um banco de dados relacional local definido em um servidor iSeries como o nome do sistema ou o nome do local especificado para esse sistema na configuração de rede. Isso pode ajudá-lo a identificar o nome de um banco de dados e correlacioná-lo a um sistema específico na rede de banco de dados relacional distribuído, especialmente se a rede for complexa.

Para que o DB2 Multisystem distribua arquivos corretamente para os servidores iSeries nos grupos de nós definidos, é necessário fazer com que os nomes dos bancos de dados remotos (RDB) sejam consistentes em todos os nós (sistemas) no grupo de nós.

Por exemplo, se planeja ter três sistemas no grupo de nós, cada sistema deve ter pelo menos três entradas no diretório do RDB. Em cada sistema, os três nomes devem ser o mesmo. Em cada um dos três sistemas, existe uma entrada para o sistema local, que é identificada como \*LOCAL. As outras duas entradas contêm as informações apropriadas do local remoto.

### Conceitos relacionados

Programação de Bancos de Dados Distribuídos

## Comando Criar Grupo de Nós (CRTNODGRP)

Esta seção utiliza dois exemplos de comando da CL para mostrar como criar um grupo de nós utilizando o comando Criar Grupo de Nós (CRTNODGRP).

No exemplo a seguir, um grupo de nós com particionamento padrão (particionamento igual entre os sistemas) é criado:

```
CRTNODGRP NODGRP(LIB1/GRP001) RDB(SYSTEMA SYSTEMB SYSTEMC SYSTEMD)  
TEXT('Grupo de nós para arquivos de teste')
```

Nesse exemplo, o comando cria um grupo de nós que contém quatro nós. Observe que cada um dos nós deve ter entradas RDB definidas (previamente incluídas no diretório do banco de dados relacional utilizando o comando ADDRDBDIRE) e que um nó deve estar definido como local (\*LOCAL).

O padrão dos atributos de particionamento é designar um quarto das partições para cada número de nó. Esse grupo de nós pode ser utilizado no parâmetro NODGRP do comando Criar Arquivo Físico (CRTPF) para criar um arquivo distribuído. Para obter informações adicionais sobre arquivos distribuídos, consulte Criando Arquivos Distribuídos com o DB2 Multisystem.

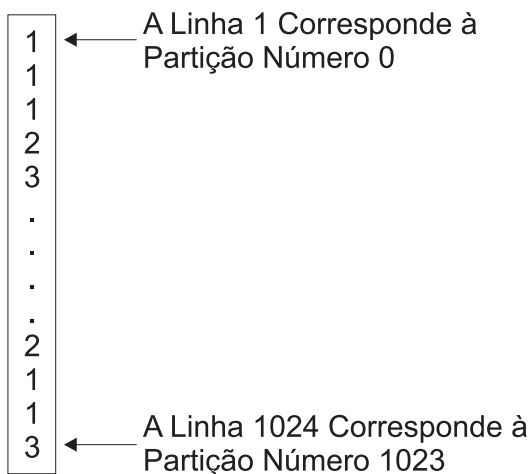
No exemplo a seguir, um grupo de nós com particionamento especificado é criado utilizando-se o parâmetro Arquivo de Particionamento (PTNFILE):

```
CRTNODGRP NODGRP(LIB1/GRP2) RDB(SYSTEMA SYSTEMB SYSTEMC)
PTNFILE(LIB1/PTN1)
TEXT('Particionar a maioria dos dados para SYSTEMA')
```

Nesse exemplo, o comando cria um grupo de nós que contém três nós (SYSTEMA, SYSTEMB e SYSTEMC). Os atributos de particionamento são obtidos do arquivo denominado PTN1. Esse arquivo pode ser configurado para forçar que uma maior porcentagem de registros esteja localizada em um sistema específico.

O arquivo PTN1 nesse exemplo é um arquivo de particionamento. Esse arquivo não é um arquivo distribuído, mas um arquivo físico local comum que pode ser utilizado para configurar um esquema de particionamento personalizado. O arquivo de particionamento deve ter um campo binário de 2 bytes. O arquivo de particionamento deve conter 1024 registros, onde cada registro contém um número de nó válido.

### Campo Binário de 2 Bytes



RBAL3508-0

Figura 3. Exemplo de Conteúdo do Arquivo de Particionamento PTNFILE

Se o grupo de nós contiver três nós, todos os registros no arquivo de particionamento devem ter os números 1, 2 ou 3. Os números de nós são designados na ordem em que os nomes do RDB foram especificados no comando Criar Grupo de Nós (CRTNODGRP). Uma maior porcentagem de dados pode ser forçada para um nó específico de forma que tenha mais registros que contenham esse número de nó no arquivo de particionamento. Esse é um método para customizar o particionamento com relação à quantidade de dados que reside fisicamente em cada sistema. Para customizar o particionamento com relação aos valores específicos que residem em nós específicos, utilize o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA). Consulte Alterando Grupos de Nós Utilizando o Comando CHGNODGRPA com o DB2 Multisystem para obter informações adicionais.

Você deve notar que, como as informações do grupo de nós estão armazenadas no arquivo distribuído, o arquivo não é imediatamente sensível a alterações no grupo de nós ou a alterações nas entradas de

diretório do RDB que estão incluídas no grupo de nós. É possível fazer modificações nos grupos de nós e nas entradas do diretório do RDB, mas até que utilize o comando CHGPF e especifique o grupo de nós alterado, os arquivos não alterarão seu comportamento.

Outro conceito é um *nó de visibilidade*. Um nó de visibilidade em um grupo de nós contém o objeto do arquivo (parte do mecanismo que permite a distribuição do arquivo entre vários nós), mas nenhum dado. Um nó de visibilidade retém um nível atual do objeto do arquivo a todo momento; o nó de visibilidade não tem dados armazenados nele. Em contraste, um nó (às vezes denominado *nó de dados*) contém dados. Como um exemplo de como é possível utilizar um nó de visibilidade em seu grupo de nós, assuma que o servidor iSeries utilizado pelos executivos de vendas faça parte de seu grupo de nós. Esses executivos provavelmente não desejam executar consultas regularmente, mas ocasionalmente podem desejar executar uma consulta específica. Em seu servidor, eles podem executar suas consultas, acessar dados em tempo real e receber os resultados das consultas. Portanto, embora nenhum dos dados esteja armazenado no servidor, como o sistema é um nó de visibilidade, os executivos podem executar a consulta sempre que for necessário.

Para especificar um nó como sendo um nó de visibilidade, é necessário utilizar o parâmetro PTNFILE no comando Criar Grupo de Nós (CRTNODGRP). Se o arquivo de particionamento não contiver registros de um número de nó específico, esse nó será um nó de visibilidade.

#### **Conceitos relacionados**

“Como os Grupos de Nós Funcionam com o DB2 Multisystem” na página 5

Um *grupo de nós* é um objeto do sistema (\*NODGRP) armazenado no sistema em que ele foi criado.

“Arquivos Distribuídos com o DB2 Multisystem” na página 11

Um *arquivo distribuído* é um arquivo do banco de dados propagado por vários servidores iSeries.

“Comando Alterar Atributos do Grupo de Nós (CHGNODGRPA)” na página 10

O comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) altera os atributos de particionamento de dados de um grupo de nós.

## **Comando Exibir Grupo de Nós (DSPNODGRP)**

O comando Exibir Grupo de Nós (DSPNODGRP) exibe os nós (sistemas) em um grupo de nós.

Também exibe o esquema de particionamento do grupo de nós (o particionamento será discutido posteriormente em Particionamento com o DB2 Multisystem).

O exemplo a seguir mostra como exibir um grupo de nós denominado GROUP1, além do esquema de particionamento associado ao grupo de nós. Essas informações são exibidas na estação de trabalho. Para obter detalhes completos sobre o comando DSPNODGRP, consulte o tópico Linguagem de Controle no Information Center.

```
DSPNODGRP NODGRP(LIB1/GROUP1)
```

Ao emitir o comando DSPNODGRP com o nome de um grupo de nós especificado, a tela Exibir Grupo de Nós é mostrada. Essa tela mostra os nomes dos sistemas (listados na coluna do sistema relacional) e o número do nó designado ao sistema. Esse é um método direto para determinar qual sistema tem determinado número de nó.

Exibir Grupo de Nós	
Grupo de Nós: GROUP1	Biblioteca: LIB1
<b>Relacional</b>	<b>Nó</b>
<b>Banco de Dados</b>	<b>Número</b>
SYSTEMA	1
SYSTEMB	2
SYSTEMC	3
<b>Final da Página</b> F3=Sair    F11=Dados de Particionamento    F12=Cancelar    F17=Início da Página    F18=Final da Página	

Figura 4. Exibir Grupo de Nós: correlação do banco de dados com o número do nó na tela grupo de nós

Para consultar o número do nó designado a cada número de partição, utilize F11 (Dados de Particionamento) na tela Exibir Grupo de Nós. A próxima tela mostra o número do nó designado a cada número de partição. O mapeamento entre o sistema e o número do nó (ou o número do nó do sistema) pode ser facilmente executado utilizando-se o comando DSPNODGRP.

Exibir Grupo de Nós	
Grupo de Nós: GROUP1	Biblioteca: LIB1
<b>Partição</b>	<b>Nó</b>
<b>Número</b>	<b>Número</b>
0	1
1	2
2	3
3	1
4	2
5	3
6	1
7	2
8	3
9	1
10	2
11	3
12	1
13	2
14	3
<b>Mais...</b>	
F3=Sair    F11=Dados do Nó    F12=Cancelar    F17=Início da página    F18=Final da Página	

Figura 5. Exibir Grupo de Nós: correlação do número da partição com o número do nó

O exemplo a seguir imprime uma lista dos sistemas no grupo de nós denominado GROUP2, além do esquema de particionamento associado:

```
DSPNODGRP NODGRP(LIB1/GROUP2) OUTPUT(*PRINT)
```

**Conceitos relacionados**

“Particionamento com o DB2 Multisystem” na página 21

Particionamento é o processo de distribuição de um arquivo pelos nós em um grupo de nós.

**Referências relacionadas**

Linguagem de Controle

## Comando Alterar Atributos do Grupo de Nós (CHGNODGRPA)

O comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) altera os atributos de particionamento de dados de um grupo de nós.

O grupo de nós contém uma tabela com 1.024 partições; cada partição contém um número de nó. Os números de nós foram designados quando o grupo de nós foi criado e correspondem aos bancos de dados relacionais especificados no parâmetro RDB do comando Criar Grupo de Nós (CRTNODGRP). Utilize o comando Exibir Grupo de Nós (DSPNODGRP) para consultar os valores de números de nós válidos e a correlação entre os números de nós e os nomes dos bancos de dados relacionais.

O comando CHGNODGRPA não afeta nenhum arquivo distribuído existente que tenha sido criado com a utilização do grupo de nós especificado. Para que o grupo de nós alterado seja utilizado, o grupo de nós alterado deve ser especificado ao criar um novo arquivo ou no comando Alterar Arquivo Físico (CHGPF). Para obter detalhes completos sobre o comando CHGNODGRPA, consulte o tópico Linguagem de Controle no Information Center.

Este primeiro exemplo mostra como alterar os atributos de particionamento do grupo de nós denominado GROUP1 na biblioteca LIB1:

```
CHGNODGRPA NODGRP(LIB1/GROUP1) PTNNBR(1019)
           NODNBR(2)
```

Nesse exemplo, o número de partição 1019 é especificado e quaisquer registros que tenham hash para 1019 são gravados no nó número 2. Isso fornece um método para especificar diretamente os números de partição para nós específicos em um grupo de nós.

O segundo exemplo altera os atributos de particionamento do grupo de nós denominado GROUP2. (GROUP2 é localizado utilizando-se a lista de procura de bibliotecas, \*LIBL.) É feito hash no valor especificado no parâmetro valor de dados de comparação (CMPDTA) e o número de partição resultante é alterado de seu número de nó existente para o número de nó 3. (Hash e particionamento são discutidos em Particionamento com o DB2 Multisystem.)

```
CHGNODGRPA NODGRP(GROUP2) CMPDTA('CHICAGO')
           NODNBR(3)
```

Quaisquer arquivos criados utilizando-se esse grupo de nós e que tenha uma chave de particionamento que consista em um campo de caractere armazena os registros que contêm 'CHICAGO' na chave de particionamento no nó número 3. Para permitir arquivos com vários campos na chave de particionamento, é possível especificar até 300 valores no parâmetro comparar dados (CMPDTA).

Ao digitar valores no parâmetro CMPDTA, você deve estar ciente de que os dados de caractere fazem distinção entre maiúsculas e minúsculas. Isso significa que 'Chicago' e 'CHICAGO' não resultam no mesmo número de partição. Os dados numéricos devem ser digitados apenas como dígitos numéricos; não utilize um ponto decimal, zeros iniciais ou zeros finais.

É feito hash em todos os valores para obter um número de partição, que é então associado ao número do nó especificado no parâmetro número do nó (NODNBR). O texto da mensagem de conclusão, CPC3207, mostra o número da partição que foi alterado. Tenha em mente que a emissão do comando CHGNODGRPA várias vezes e por muitos valores diferentes aumenta a chance de alterar o mesmo número de partição duas vezes. Se isso ocorrer, o número do nó especificado na alteração mais recente estará em efeito para o grupo de nós.

### Conceitos relacionados

“Comando Criar Grupo de Nós (CRTNODGRP)” na página 6

Esta seção utiliza dois exemplos de comando da CL para mostrar como criar um grupo de nós utilizando o comando Criar Grupo de Nós (CRTNODGRP).

“Particionamento com o DB2 Multisystem” na página 21

*Particionamento* é o processo de distribuição de um arquivo pelos nós em um grupo de nós.



“Customização da Distribuição de Dados com o DB2 Multisystem” na página 23

Como o sistema é responsável por fornecer os dados, você não precisa saber onde os registros residem na realidade. Entretanto, se deseja garantir que determinados registros sempre sejam armazenados em um sistema específico, poderá utilizar o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) para especificar onde esses registros residem.

#### Referências relacionadas

Linguagem de Controle

## Comando Excluir Grupo de Nós (DLTNODGRP)

O comando Excluir Grupo de Nós (DLTNODGRP) exclui um grupo de nós criado anteriormente.

Esse comando não afeta nenhum arquivo que tenha sido criado com a utilização do grupo de nós.

O exemplo a seguir mostra como excluir o grupo de nós denominado GROUP1. Quaisquer arquivos criados com esse grupo de nós não são afetados:

```
DLTNODGRP NODGRP(LIB1/GROUP1)
```

Embora a exclusão do grupo de nós não afete os arquivos criados utilizando esse grupo de nós, não é recomendado excluir grupos de nós depois de utilizá-los. Tão logo o grupo de nós seja excluído, não mais será possível utilizar o comando DSPNODGRP para exibir os nós e o esquema de particionamento. Entretanto, é possível utilizar o comando Exibir Descrição do Arquivo (DSPFD) e especificar TYPE(\*NODGRP) para consultar o grupo de nós associado a um arquivo específico.

#### Referências relacionadas

Linguagem de Controle

---

## Arquivos Distribuídos com o DB2 Multisystem

Um *arquivo distribuído* é um arquivo do banco de dados propagado por vários servidores iSeries.

Os arquivos distribuídos podem ser atualizados e acessados por métodos como SQL, ferramentas de consulta e o comando Exibir Membro do Arquivo Físico (DSPPFM). Para operações do banco de dados, os arquivos distribuídos são tratados exatamente como arquivos locais, em sua maioria. Para obter informações sobre as alterações do comando da CL para arquivos distribuídos, consulte o tópico Como os comandos da CL funcionam com arquivos distribuídos.

Para criar um arquivo de banco de dados como um arquivo distribuído, é possível utilizar o comando Criar Arquivo Físico (CRTPF) ou a instrução SQL CREATE TABLE. Os dois métodos são discutidos com mais detalhes neste tópico. O comando CRTPF tem dois parâmetros, grupo de nós (NODGRP) e chave de particionamento (PTNKEY), que criam o arquivo como um arquivo distribuído. É possível consultar o arquivo do banco de dados distribuído como um objeto (\*FILE com o atributo PF) que tem o nome e a biblioteca especificados ao executar o comando CRTPF.

Se deseja alterar um arquivo físico do banco de dados não-distribuído existente em um arquivo distribuído, é possível fazer isso utilizando o comando Alterar Arquivo Físico (CHGPF). No comando CHGPF, os parâmetros NODGRP e PTNKEY permitem fazer a alteração para um arquivo distribuído. Com o comando CHGPF, também é possível alterar os atributos de particionamento de dados para um arquivo de banco de dados distribuído existente especificando-se valores para os parâmetros NODGRP e PTNKEY. A especificação de valores para esses parâmetros faz com que os dados sejam redistribuídos de acordo com a tabela de particionamento no grupo de nós.

**Nota:** Todos os arquivos lógicos baseados no arquivo físico que está sendo alterado para um arquivo distribuído também se tornam arquivos distribuídos. Para arquivos grandes ou redes grandes, a redistribuição dos dados pode demorar e não deve ser feita com frequência.

#### Conceitos relacionados

“Comando Criar Grupo de Nós (CRTNODGRP)” na página 6

Esta seção utiliza dois exemplos de comando da CL para mostrar como criar um grupo de nós utilizando o comando Criar Grupo de Nós (CRTNODGRP).

“Como os Comandos da CL Funcionam com Arquivos Distribuídos” na página 15

Este tópico aborda a emissão de comandos da CL em arquivos distribuídos com o DB2 Multisystem.

## Comando Criar Arquivo Físico (CRTPF) e a Instrução SQL CREATE TABLE

Este tópico fornece informações sobre a criação de um arquivo físico distribuído utilizando o comando Criar Arquivo Físico (CRTPF) ou a instrução SQL CREATE TABLE.

Para criar um arquivo particionado, é necessário especificar os seguintes parâmetros no comando CRTPF:

- Um nome de grupo de nós para o parâmetro NODGRP
- O campo ou os campos a serem utilizados como a chave de particionamento (utilize o parâmetro PTNKEY)

A chave de particionamento determina onde (em qual nó) cada registro de dados reside fisicamente. A chave de particionamento é especificada quando o comando CRTPF é executado ou quando a instrução SQL CREATE TABLE é executada. Os valores dos campos que compõem a chave de particionamento de cada registro são processados pelo algoritmo HASH para determinar onde o registro está localizado.

Se tiver uma chave de particionamento com um campo único, todos os registros com o mesmo valor nesse campo residirão no mesmo sistema.

Se deseja criar um arquivo físico distribuído, o perfil do usuário deve existir em cada nó dentro do grupo de nós e o perfil do usuário deve ter a autoridade necessária para criar um arquivo distribuído em cada nó. Se for necessário criar um arquivo distribuído em uma biblioteca específica, essa biblioteca deverá existir em cada nó no grupo de nós e o perfil do usuário deverá ter a autoridade necessária para criar arquivos nessas bibliotecas. Se algum desses fatores não for verdadeiro, o arquivo não será criado.

A forma em que os sistemas estão configurados pode influenciar o perfil do usuário utilizado no sistema remoto. Para assegurar que o perfil do usuário seja utilizado no sistema remoto, esse sistema deve ser configurado como um local seguro. Para determinar se um sistema está configurado como um local seguro, utilize o comando Trabalhar com Listas de Configuração (WRKCFGL).

O exemplo a seguir mostra como criar um arquivo físico denominado PAYROLL que é particionado (especificado utilizando-se o parâmetro NODGRP) e tem uma única chave de particionamento no campo de número do funcionário (EMPNUM):

```
CRTPF FILE(PRODLIB/PAYROLL) SCRFILE(PRODLIB/DDS) SRCMBR(PAYROLL)
      NODGRP(PRODLIB/PRODGROUP) PTNKEY(EMPNUM)
```

Quando o comando CRTPF é executado, o sistema cria um arquivo físico distribuído para conter os dados locais associados ao arquivo distribuído. O comando CRTPF também cria arquivos físicos em todos os sistemas remotos especificados no grupo de nós.

O direito à propriedade do arquivo físico e a autoridade pública em todos os sistemas são consistentes. Essa consistência também inclui qualquer autoridade especificada no parâmetro AUT do comando CRTPF.

A instrução SQL CREATE TABLE também pode ser utilizada para especificar o grupo de nós e a chave de particionamento. No exemplo a seguir, uma tabela SQL denominada PAYROLL é criada. O exemplo utiliza a cláusula IN *nodgroup-name* e a cláusula PARTITIONING KEY.

```
CREATE TABLE PRODLIB/PAYROLL
      (EMPNUM INT, EMPLNAME CHAR(12), EMPFNAME CHAR (12))
      IN PRODLIB/PRODGROUP
      PARTITIONING KEY (EMPNUM)
```

Quando a cláusula PARTITIONING KEY não for especificada, a primeira coluna da chave primária, se estiver definida, será utilizada como a primeira chave de particionamento. Se nenhuma chave primária estiver definida, a primeira coluna definida para a tabela que não tenha um tipo de dados igual a data, hora, time stamp ou numérico de ponto flutuante será utilizada como a chave de particionamento.

Para verificar se um arquivo é particionado, utilize o comando Exibir Descrição do Arquivo (DSPFD). Se o arquivo for particionado, o comando DSPFD mostrará o nome do grupo de nós, os detalhes do grupo de nós armazenado no objeto do arquivo (incluindo o mapa de partição inteiro) e listará os campos da chave de particionamento.

Para obter uma lista das restrições que você precisa saber ao utilizar arquivos distribuídos com o DB2 Multisystem, consulte Restrições ao Criar ou Trabalhar com Arquivos Distribuídos com o DB2 Multisystem.

#### Conceitos relacionados

“Restrições ao Criar ou Trabalhar com Arquivos Distribuídos com o DB2 Multisystem”

Você deve estar ciente de algumas restrições ao criar ou trabalhar com arquivos distribuídos.

Programação de Bancos de Dados Distribuídos

## Restrições ao Criar ou Trabalhar com Arquivos Distribuídos com o DB2 Multisystem

Você deve estar ciente de algumas restrições ao criar ou trabalhar com arquivos distribuídos.

Essas restrições são:

- Os caminhos de acesso FCFO (Primeiro alterado primeiro que sai) não podem ser utilizados porque os caminhos de acesso são particionados por vários nós.
- Um arquivo distribuído pode ter no máximo um membro.
- Um arquivo distribuído não é permitido em uma biblioteca temporária (QTEMP).
- Os dados na chave de particionamento têm uma capacidade de atualização limitada. Geralmente, ao escolher uma chave de particionamento, é necessário escolher campos cujos valores não sejam atualizados. As atualizações na chave de particionamento são permitidas desde que a atualização não faça com que o registro seja particionado para um nó diferente.
- Campo de data, hora, time stamp ou numérico de ponto flutuante não podem ser utilizados na chave de particionamento.
- Os arquivos físicos de origem não são suportados.
- Os arquivos descritos externamente são suportados para arquivos distribuídos; os arquivos descritos no programa não são suportados.
- Se o caminho de acesso for exclusivo, a chave de particionamento deve ser um subconjunto do caminho de acesso da chave exclusiva.
- As restrições são suportadas; as restrições de referência são suportadas apenas se o grupo de nós dos arquivos da chave estrangeira e pai forem idênticos e *todos* os campos da chave de particionamento estiverem incluídos na restrição. A chave de particionamento deve ser um subconjunto dos campos da restrição. Além disso, para restrições exclusivas e primárias, se o caminho de acesso for exclusivo, a chave de particionamento deve ser um subconjunto do caminho de acesso da chave exclusiva.
- No comando CRTPF, o parâmetro do sistema deve ter o valor \*LCL especificado (CRTPF SYSTEM(\*LCL)). SYSTEM(\*RMT) não é permitido.
- Toda vez que um arquivo lógico é criado sobre um arquivo distribuído, o arquivo lógico também se torna distribuído, o que significa que não é possível construir um arquivo lógico local apenas sobre uma parte do arquivo físico em um nó específico. As visualizações SQL são uma exceção a isso se a

visualização for uma junção e se todos os arquivos físicos subjacentes não tiverem o mesmo grupo de nós. Nesse caso, a visualização é criada somente no sistema local. Embora essa visualização não seja distribuída, se você consultá-la, os dados serão recuperados de todos os nós, não apenas do nó onde a visualização foi criada.

Os arquivos de junção podem ser criados apenas utilizando-se SQL.

Para arquivos lógicos criados pelo DDS, apenas um arquivo básico é permitido.

- As tabelas de CCSIDs (Identificadores de conjunto de caracteres codificados) e SRTSEQ (seqüência de classificação) são resolvidas a partir do sistema de origem.
- O processamento de VLR (Registro de Comprimento Variável) não é suportado. Isso não significa que os campos de comprimento variável não são suportados para arquivos distribuídos. Essa restrição refere-se apenas às linguagens e aplicativos que requerem o processamento de VLR quando um arquivo é aberto.
- O processamento de EOFDLY (Retardo de fim de arquivo) não é suportado.
- O DFU (Utilitário de Arquivo de Dados) não funciona com arquivos distribuídos, porque o DFU utiliza o processamento de número de registro relativo para acessar registros.
- Um arquivo distribuído não pode ser criado em uma biblioteca localizada em um IASP (Conjunto de Armazenamento Auxiliar Independente).

#### **Conceitos relacionados**

“Comando Criar Arquivo Físico (CRTPF) e a Instrução SQL CREATE TABLE” na página 12

Este tópico fornece informações sobre a criação de um arquivo físico distribuído utilizando o comando Criar Arquivo Físico (CRTPF) ou a instrução SQL CREATE TABLE.

## **Atividades do Sistema Depois que o Arquivo Distribuído É Criado**

Assim que o arquivo for criado, o sistema assegura que os dados estejam particionados e que os arquivos permaneçam em níveis simultâneos.

Assim que o arquivo for criado, as seguintes atividades ocorrerão automaticamente:

- Todos os índices criados para o arquivo são criados em todos os nós.
- As informações sobre a utilização de arquivos distribuídos com o DB2 Multisystem são enviadas para todos os nós.
- O sistema impede que o arquivo seja movido e impede que sua biblioteca seja renomeada.
- Se o próprio arquivo for renomeado, seu novo nome será refletido em todos os nós.
- Vários comandos, como Alocar Objeto (ALCOBJ), Reorganizar Membro do Arquivo Físico (RGZPFM) e Iniciar Arquivo Físico do Diário (STRJRNP) agora afetam todas as partes do arquivo. Isso permite manter o conceito de um arquivo local ao trabalhar com arquivos particionados. Consulte Comandos da CL: Afetam todas as partes de um arquivo distribuído com o DB2 Multisystem para obter uma lista completa desses comandos da CL.

É possível emitir o comando Alocar Objeto (ALCOBJ) a partir de qualquer um dos nós no grupo de nós. Isso bloqueia todas as partes e assegura a mesma integridade concedida quando um arquivo local é alocado. Todas essas ações são manipuladas pelo sistema, o que evita a necessidade de digitar os comandos em cada nó.

No caso do comando Iniciar Arquivo Físico do Diário (STRJRNP), a criação do diário é iniciada em cada sistema. Dessa forma, cada sistema deve ter seu próprio diário e receptor de diário. Cada sistema tem suas próprias entradas de diário; a recuperação utilizando as entradas do diário deve ser feita individualmente em cada sistema. Os comandos para iniciar e finalizar a criação do diário afetam todos os sistemas no grupo de nós simultaneamente. Consulte Considerações sobre Criação de Diário com o DB2 Multisystem para obter informações adicionais.

- Vários comandos, como Fazer Dump de Objeto (DMPOBJ), Salvar Objeto (SAVOBJ) e Trabalhar com Bloqueios de Objetos (WRKOBJLCK) afetam apenas a parte do arquivo no sistema em que o comando foi emitido. Consulte Comandos da CL: Afetam apenas as partes locais de um arquivo distribuído com o DB2 Multisystem para obter uma lista completa desses comandos da CL.

Assim que um arquivo for criado como um arquivo distribuído, a abertura do arquivo resulta na abertura da parte local do arquivo, além das conexões feitas para todos os sistemas remotos. Quando o arquivo é criado, ele pode ser acessado a partir de qualquer sistema no grupo de nós. O sistema também determina quais nós e registros precisam ser utilizados para concluir a tarefa de E/S de arquivo (GETS, PUTs e UPDATES, por exemplo). Não é necessário influenciar fisicamente ou especificar nenhuma dessas atividades.

Observe que os pedidos da DRDA (Distributed Relational Database Architecture) e do DDM (Distributed Data Management) podem alvejar arquivos distribuídos. Os aplicativos distribuídos anteriormente que utilizam a DRDA ou o DDM para acessar um arquivo de banco de dados em um sistema remoto podem continuar a funcionar mesmo se esse arquivo de banco de dados foi alterado para ser um arquivo distribuído.

Você deve estar ciente de que a seqüência de chegada dos registros é diferente para arquivos de banco de dados distribuídos em relação aos de um arquivo de banco de dados local.

Como os arquivos distribuídos são fisicamente distribuídos pelos sistemas, não é possível contar com a seqüência de chegada ou o número de registro relativo dos registros. Com um arquivo de banco de dados local, os registros são manipulados em ordem. Se, por exemplo, você inserir dados em um arquivo de banco de dados local, os dados serão inseridos a partir do primeiro registro e continuando até o último registro. Todos os registros são inseridos em ordem seqüencial. Os registros em um nó individual são inseridos da mesma forma em que os registros são inseridos em um arquivo local.

Quando os dados forem lidos a partir de um arquivo de banco de dados local, os dados são lidos a partir do primeiro registro e continuando até o último registro. Isso não acontece com um arquivo de banco de dados distribuído. Com um arquivo de banco de dados distribuído, os dados são lidos dos registros (do primeiro ao último registro) no primeiro nó, em seguida do segundo nó e assim por diante. Por exemplo, ler o registro 27 não mais significa que um único registro foi lido. Com um arquivo distribuído, cada nó no grupo de nós contém seu próprio registro 27, nenhum dos quais é o mesmo.

#### **Conceitos relacionados**

“Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem” na página 17

Alguns comandos da CL, quando executados, afetam todas as partes do arquivo distribuído.

“Considerações sobre Criação de Diário com o DB2 Multisystem” na página 19

Embora os comandos Iniciar Arquivo Físico do Diário (STRJRNPF) e Finalizar Arquivo Físico do Diário (ENDJRNPF) sejam distribuídos para outros sistemas, a criação do diário real ocorre em cada sistema independentemente e para o receptor de diário próprio de cada sistema.

“Comandos da CL: Afetam Apenas Partes Locais de um Arquivo Distribuído com o DB2 Multisystem” na página 16

Alguns comandos da CL, quando executados, afetam apenas a parte do arquivo distribuído que está localizada no sistema local (o sistema a partir do qual o comando foi executado).

“Como os Comandos da CL Funcionam com Arquivos Distribuídos”

Este tópico aborda a emissão de comandos da CL em arquivos distribuídos com o DB2 Multisystem.

## **Como os Comandos da CL Funcionam com Arquivos Distribuídos**

Este tópico aborda a emissão de comandos da CL em arquivos distribuídos com o DB2 Multisystem.

Como um arquivo distribuído tem um tipo de objeto do sistema igual a \*FILE, muitos dos comandos da CL que acessam arquivos físicos podem ser executados em arquivos distribuídos. No entanto, o comportamento dos mesmos comandos da CL é alterado quando emitido em um arquivo distribuído versus um arquivo não-distribuído. Os tópicos a seguir abordam como os comandos da CL funcionam com arquivos distribuídos.

#### **Conceitos relacionados**

“Arquivos Distribuídos com o DB2 Multisystem” na página 11

Um *arquivo distribuído* é um arquivo do banco de dados propagado por vários servidores iSeries.

“Atividades do Sistema Depois que o Arquivo Distribuído É Criado” na página 14

Assim que o arquivo for criado, o sistema assegura que os dados estejam particionados e que os arquivos permaneçam em níveis simultâneos.

#### **Referências relacionadas**

Linguagem de Controle

### **Comandos da CL: Permitidos a Serem Executados em um Arquivo Distribuído com o DB2 Multisystem:**

Alguns comandos ou parâmetros específicos da CL não podem ser executados em arquivos distribuídos.

Esses parâmetros ou comandos da CL são:

- Parâmetro SHARE de Alterar Membro do Arquivo Lógico (CHGLFM)
- Parâmetro SHARE de Alterar Membro do Arquivo Físico(CHGPFM)
- Criar Objeto Duplicado (CRTDUPOBJ)
- Inicializar Membro do Arquivo Físico (INZPFM)
- Mover Objeto (MOVOBJ)
- Posicionar Arquivo do Banco de Dados (POSDBF)
- Remover Membro (RMVM)
- Renomear Biblioteca (RNMLIB) para bibliotecas que contenham arquivos distribuídos
- Renomear Membro (RNMM)
- Comando Sistema de Arquivo Integrado (IFS), COPY

### **Comandos da CL: Afetam Apenas Partes Locais de um Arquivo Distribuído com o DB2 Multisystem:**

Alguns comandos da CL, quando executados, afetam apenas a parte do arquivo distribuído que está localizada no sistema local (o sistema a partir do qual o comando foi executado).

Esses comandos da CL são:

- Aplicar Alterações Registradas (APYJRNCHG). Consulte Considerações sobre Criação de Diário com o DB2 Multisystem para obter informações adicionais sobre esse comando.
- Exibir Descrição do Objeto (DSPOBJD)
- Fazer Dump de Objeto (DMPOBJ)
- Finalizar Caminho de Acesso do Diário (ENDJRNAP)
- Remover Alterações Registradas (RMVJRNCHG). Consulte Considerações sobre Criação de Diário com o DB2 Multisystem para obter informações adicionais sobre esse comando.
- Restaurar Objeto (RSTOBJ)
- Salvar Objeto (SAVOBJ)
- Iniciar Caminho de Acesso do Diário (STRJRNAP)

É possível utilizar o comando Submeter Comando Remoto (SBMRMTCMD) para emitir qualquer comando da CL para todos os sistemas remotos associados a um arquivo distribuído. Ao emitir um comando da CL no sistema local e, em seguida, emitir o mesmo comando utilizando o comando SBMRMTCMD para um arquivo distribuído, é possível executar um comando da CL em todos os sistemas de um arquivo distribuído. Não é necessário fazer isso para comandos da CL que são executados automaticamente em todas as partes de um arquivo distribuído. Consulte Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem para obter informações adicionais.

O comando Exibir Descrição do Arquivo (DSPFD) pode ser utilizado para exibir informações sobre o grupo de nós para arquivos distribuídos. O comando DSPFD mostra o nome do grupo de nós, os campos na chave de particionamento e uma descrição completa do grupo de nós. Para exibir essas informações, é necessário especificar \*ALL ou \*NODGRP para o parâmetro TYPE no comando DSPFD.

O comando Exibir Membro do Arquivo Físico (DSPPFM) pode ser utilizado para exibir os registros locais de um arquivo distribuído; entretanto, se deseja exibir tanto dados remotos quanto dados locais, deverá especificar \*ALLDATA no parâmetro do registro (FROMRCD) no comando.

Ao utilizar o comando Salvar Objeto (SAVOBJ) ou o comando Restaurar Objeto (RSTOBJ) para arquivos distribuídos, cada parte do arquivo distribuído deve ser salva e restaurada individualmente. Uma parte do arquivo pode ser restaurada apenas para o sistema a partir do qual ela foi salva se ela deve ser mantida como parte do arquivo distribuído. Se necessário, o comando Alocar Objeto (ALLOBJ) pode ser utilizado para obter um bloqueio em todas as partes do arquivo para impedir que quaisquer alterações sejam feitas no arquivo durante o processo de salvamento.

O sistema distribuirá automaticamente qualquer arquivo lógico quando o arquivo for restaurado se as seguintes condições forem verdadeiras:

- O arquivo lógico foi salvo como um arquivo não-distribuído.
- O arquivo lógico será restaurado no sistema quando arquivos nos quais ele é baseado forem distribuídos.

As partes salvas do arquivo também podem ser utilizadas para criar um arquivo local. Para isso, é necessário restaurar a parte do arquivo para uma biblioteca diferente ou para um sistema que não estava no grupo de nós utilizado quando o arquivo distribuído foi criado. Para que todos os registros no arquivo distribuído possa ir para um arquivo local, é necessário restaurar cada parte do arquivo no mesmo sistema e, em seguida, copiar os registros para um arquivo agregado. Utilize o comando Copiar Arquivo (CPYF) para copiar os registros para o arquivo agregado.

#### **Conceitos relacionados**

“Atividades do Sistema Depois que o Arquivo Distribuído É Criado” na página 14  
Assim que o arquivo for criado, o sistema assegura que os dados estejam particionados e que os arquivos permaneçam em níveis simultâneos.

“Considerações sobre Criação de Diário com o DB2 Multisystem” na página 19  
Embora os comandos Iniciar Arquivo Físico do Diário (STRJRNPF) e Finalizar Arquivo Físico do Diário (ENDJRNPF) sejam distribuídos para outros sistemas, a criação do diário real ocorre em cada sistema independentemente e para o receptor de diário próprio de cada sistema.

“Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem”  
Alguns comandos da CL, quando executados, afetam todas as partes do arquivo distribuído.

#### **Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem:**

Alguns comandos da CL, quando executados, afetam todas as partes do arquivo distribuído.

Ao executar esses comandos no sistema, eles são executados automaticamente em todos os nós do grupo de nós.

Essa convenção permite manter a consistência no grupo de nós sem a necessidade de digitar o mesmo comando em cada sistema. Com alterações de autoridade, pode ocorrer alguma inconsistência no grupo de nós. Por exemplo, se o ID de um usuário for excluído de um sistema no grupo de nós, a habilidade em manter a consistência no grupo de nós será perdida.

Os erros de autoridade são manipulados individualmente.

Os seguintes comandos afetam todas as partes do arquivo distribuído:

- Incluir Membro do Arquivo Lógico (ADDLFM)
- Incluir Restrição do Arquivo Físico (ADDPFCST)
- Incluir Membro do Arquivo Físico (ADDPFM)
- Incluir Acionador do Arquivo Físico (ADDPFTRG)
- Alocar Objeto (ALCOBJ)
- Alterar Arquivo Lógico (CHGLF)
- Alterar Proprietário do Objeto (CHGOBJOWN)
- Alterar Arquivo Físico (CHGPF)
- Alterar Restrição do Arquivo Físico (CHGPFCSST)
- Limpar Membro do Arquivo Físico (CLRPFM)
- Copiar Arquivo (CPYF). Consulte Utilizando o Comando Copiar Arquivo (CPYF) com Arquivos Distribuídos com o DB2 Multisystem para obter informações adicionais sobre esse comando.
- Criar Arquivo Lógico (CRTLF)
- Desalocar Objeto (DLCOBJ)
- Excluir Arquivo (DLTF)
- Finalizar Arquivo Físico do Diário (ENDJRNPf). Consulte Considerações sobre Criação de Diário com o DB2 Multisystem para obter informações adicionais sobre esse comando.
- Conceder Autoridade ao Objeto (GRTOBJAUT)
- Remover Restrição do Arquivo Físico (RMVPPFCST)
- Remover Acionador do Arquivo Físico (RMVPPFTRG)
- Renomear Objeto (RNMOBJ)
- Reorganizar Membro do Arquivo Físico (RGZPFM)
- Revogar Autoridade do Objeto (RVKOBJAUT)
- Iniciar Arquivo Físico do Diário (STRJRNPf). Consulte Considerações sobre Criação de Diário com o DB2 Multisystem para obter informações adicionais sobre esse comando.

Para esses comandos, se algum objeto que não seja o arquivo distribuído for referido, é de sua responsabilidade criar esses objetos em cada sistema. Por exemplo, ao utilizar o comando Incluir Acionador do Arquivo Físico (ADDPFTRG), você deve assegurar-se de que o programa acionador exista em todos os sistemas. Caso contrário, ocorrerá um erro. O mesmo conceito é aplicado ao comando Iniciar Arquivo Físico do Diário (STRJRNPf), onde o diário deve existir em todos os sistemas.

Se o perfil do usuário não existir no nó remoto e você emitir o comando GRTOBJAUT ou o comando RVKOBJAUT, a autoridade será concedida ou revogada em todos os nós onde o perfil exista e será ignorada em todos os nós onde o perfil não exista.

#### **Conceitos relacionados**

“Atividades do Sistema Depois que o Arquivo Distribuído É Criado” na página 14

Assim que o arquivo for criado, o sistema assegura que os dados estejam particionados e que os arquivos permaneçam em níveis simultâneos.

“Comandos da CL: Afetam Apenas Partes Locais de um Arquivo Distribuído com o DB2 Multisystem” na página 16

Alguns comandos da CL, quando executados, afetam apenas a parte do arquivo distribuído que está localizada no sistema local (o sistema a partir do qual o comando foi executado).

“Comando Copiar Arquivo (CPYF) com Arquivos Distribuídos com o DB2 Multisystem” na página 19  
Quando o comando Copiar Arquivo (CPYF) é emitido, o sistema tenta executar o comando CPYF tão rapidamente quanto possível.

“Considerações sobre Criação de Diário com o DB2 Multisystem” na página 19

Embora os comandos Iniciar Arquivo Físico do Diário (STRJRNPf) e Finalizar Arquivo Físico do



Diário (ENDJRNPf) sejam distribuídos para outros sistemas, a criação do diário real ocorre em cada sistema independentemente e para o receptor de diário próprio de cada sistema.

#### *Considerações sobre Criação de Diário com o DB2 Multisystem:*

Embora os comandos Iniciar Arquivo Físico do Diário (STRJRNPf) e Finalizar Arquivo Físico do Diário (ENDJRNPf) sejam distribuídos para outros sistemas, a criação do diário real ocorre em cada sistema independentemente e para o receptor de diário próprio de cada sistema.

Como exemplo, você tem dois sistemas (A e B) nos quais tem um arquivo distribuído. Você precisa criar um diário e um receptor no sistema A e no sistema B, e o nome do diário e a biblioteca devem ser os mesmos nos dois sistemas. Ao emitir o comando STRJRNPf, ele é distribuído para os dois sistemas e a criação do diário é iniciada em ambos. Entretanto, o receptor de diário no sistema A contém apenas os dados das alterações ocorridas na parte do arquivo que reside no sistema A. O receptor de diário no sistema B contém apenas os dados das alterações ocorridas na parte do arquivo que reside no sistema B.

Isso afeta a estratégia de salvamento e restauração, bem como a estratégia de backup; por exemplo:

- Depois de emitir o comando STRJRNPf, é necessário salvar o arquivo do banco de dados de cada um dos sistemas no grupo de nós do arquivo.
- É necessário praticar o gerenciamento de diário padrão em cada um dos sistemas. É necessário alterar e salvar os receptores de diário apropriadamente, de forma que seja possível gerenciar o uso de espaço em disco para cada sistema. Ou ainda, é possível aproveitar o suporte ao gerenciamento de diário de alteração do sistema.

**Nota:** Apenas os nomes do diário devem ser os mesmos em cada um dos sistemas; os atributos não. Dessa forma, por exemplo, é possível especificar um valor de limite do receptor de diário diferente em sistemas diferentes, refletindo o espaço em disco disponível em cada um desses sistemas.

- Se for necessário recuperar uma parte de um arquivo de banco de dados distribuído, você precisa utilizar apenas o receptor de diário do sistema onde a parte do arquivo distribuído residia. Nesse receptor de diário, as alterações registradas são aplicadas utilizando o comando Aplicar Alterações Registradas (APYJRNCHG) ou as alterações registradas são removidas utilizando o comando Remover Alterações Registradas (RMVJRNCHG).
- **Não** é possível utilizar o receptor de diário de um sistema para aplicar ou remover as alterações registradas em uma parte do arquivo em outro sistema. Isso ocorre porque cada parte do arquivo em cada sistema tem seu próprio JID (Identificador de Diário Exclusivo).

#### **Conceitos relacionados**

“Atividades do Sistema Depois que o Arquivo Distribuído É Criado” na página 14

Assim que o arquivo for criado, o sistema assegura que os dados estejam particionados e que os arquivos permaneçam em níveis simultâneos.

“Comandos da CL: Afetam Apenas Partes Locais de um Arquivo Distribuído com o DB2 Multisystem” na página 16

Alguns comandos da CL, quando executados, afetam apenas a parte do arquivo distribuído que está localizada no sistema local (o sistema a partir do qual o comando foi executado).

“Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem” na página 17

Alguns comandos da CL, quando executados, afetam todas as partes do arquivo distribuído.

#### *Comando Copiar Arquivo (CPYF) com Arquivos Distribuídos com o DB2 Multisystem:*

Quando o comando Copiar Arquivo (CPYF) é emitido, o sistema tenta executar o comando CPYF tão rapidamente quanto possível.

Os parâmetros de comando especificados, os atributos de arquivo envolvidos na cópia e o tamanho e número de registros a serem copiados afetam a velocidade da execução do comando.

Ao copiar dados para arquivos distribuídos, o desempenho do comando de cópia pode ser aprimorado utilizando-se apenas os seguintes parâmetros no comando CPYF: FROMFILE, TOFILE, FROMMBR, TOMBR, MBROPT e FMTOPT(\*NONE) ou FMTOPT(\*NOCHK). Além disso, os parâmetros Do arquivo (FROMFILE) e Para arquivo (TOFILE) **não** devem especificar arquivos que contenham algum campo que aceite valores nulos. Em geral, quanto mais simples a sintaxe do comando de cópia, maior a chance de que a operação de cópia mais rápida seja obtida. Quando o método de cópia mais rápida estiver sendo utilizado ao copiar para um arquivo distribuído, a mensagem CPC9203 é emitida, indicando o número de registros copiados para cada nó. Normalmente, se essa mensagem não foi emitida, a cópia mais rápida não foi executada.

Ao copiar para um arquivo distribuído, você deve considerar as seguintes diferenças entre quando a cópia mais rápida é ou não utilizada:

- Para a cópia mais rápida, os registros são armazenados em buffer para cada nó. À medida que os buffers ficam cheios, eles são enviados para um nó específico. Se ocorrer um erro depois que quaisquer registros forem colocados em qualquer um dos buffers de nó, o sistema tentará enviar todos os registros que estão atualmente nos buffers de nó para o nó correto. Se ocorrer um erro enquanto o sistema está enviando registros para um nó específico, o processamento continuará para o próximo nó até que o sistema tente enviar todos os buffers de nó.

Em outras palavras, os registros que seguem um registro específico com erro podem ser gravados no arquivo distribuído. Essa ação ocorre devido ao bloqueio simultâneo feito em vários nós. Se você não deseja que os registros que seguem um registro com erro sejam gravados no arquivo distribuído, é possível forçar a não utilização da cópia mais rápida especificando-se ERRLVL(\*NOMAX) ou ERRLVL com um valor maior ou igual a 1 no comando CPYF.

Quando a cópia mais rápida não for utilizada, o bloqueio de registro será tentado, a menos que a abertura do arquivo de destino seja, ou se force a se tornar, uma abertura SEQONLY(\*NO).

- Quando a cópia mais rápida for utilizada, uma mensagem será emitida indicando que a abertura do membro foi alterada para SEQONLY(\*NO); entretanto, o arquivo de destino distribuído será aberto uma segunda vez para permitir o bloqueio dos registros. Você deve ignorar a mensagem sobre a alteração para SEQONLY(\*NO).
- Quando a cópia mais rápida for utilizada, várias mensagens serão emitidas indicando o número de registros copiados para cada nó. Em seguida, uma mensagem é enviada indicando o número total de registros copiados.
- Quando a cópia mais rápida não for utilizada, apenas a mensagem do número total de registros copiados será enviada. Nenhuma mensagem é enviada listando o número de registros copiados para cada nó.

Considere as seguintes restrições ao copiar de ou para arquivos distribuídos:

- O parâmetro FROMRCD pode ser especificado apenas com o valor \*START ou 1 ao copiar a partir de um arquivo distribuído. O parâmetro TORCD não pode ser especificado com um valor diferente do valor padrão \*END ao copiar a partir de um arquivo distribuído.
- O parâmetro MBROPT(\*UPDADD) não é permitido ao copiar para um arquivo distribuído.
- O parâmetro COMPRESS(\*NO) não é permitido quando o arquivo de destino for um arquivo distribuído e o arquivo de origem for um arquivo que pode ser excluído do banco de dados.
- Para copiar listagens de impressão, o valor de RCDNBR fornecido é a posição do registro no arquivo em um nó específico quando o registro for do tipo arquivo distribuído. O mesmo número de registro aparece várias vezes na listagem e cada um deles é um registro de um nó diferente.

#### **Conceitos relacionados**

“Comandos da CL: Afetam Todas as Partes de um Arquivo Distribuído com o DB2 Multisystem” na página 17

Alguns comandos da CL, quando executados, afetam todas as partes do arquivo distribuído.

## Particionamento com o DB2 Multisystem

*Particionamento* é o processo de distribuição de um arquivo pelos nós em um grupo de nós.

O particionamento é feito utilizando-se o algoritmo hash. Quando um novo registro for incluído, o algoritmo hash será aplicado aos dados na chave de particionamento. Em seguida, o resultado do algoritmo hash, um número entre 0 e 1023, será aplicado ao mapa de particionamento para determinar o nó no qual o registro reside.

O mapa de partição também é utilizado para otimização da consulta, atualizações, exclusões e junções. O mapa de partição pode ser customizado para forçar determinados valores de chave para nós específicos.

Por exemplo, durante a E/S, o sistema aplica o algoritmo hash aos valores nos campos da chave de particionamento. O resultado é aplicado ao mapa de partição armazenado no arquivo para determinar qual nó armazena o registro.

O seguinte exemplo mostra como esses conceitos estão relacionados:

O número do funcionário é a chave de particionamento e um registro é inserido no banco de dados com o número de funcionário igual a 56.000. O valor 56.000 é processado pelo algoritmo hash e o resultado é um número de partição igual a 733. O mapa de partição, que faz parte do objeto de grupo de nós e é armazenado no arquivo distribuído quando ele é criado, contém o número de nó 1 para a partição número 733. Dessa forma, esse registro é fisicamente armazenado no sistema no grupo de nós que recebeu o número de nó 1. A chave de particionamento (o parâmetro PTNKEY) foi especificada quando o arquivo particionado (distribuído) foi criado.

Os campos na chave de particionamento aceitam valores nulos. Entretanto, os registros que contêm um valor nulo na chave de particionamento têm hash para a partição número 0. Os arquivos com uma quantidade significativa de valores nulos na chave de particionamento podem resultar em desvio de dados na partição número 0, porque todos os registros com valores nulos têm hash para a partição número 0.

Depois de criar o objeto de grupo de nós e um arquivo de banco de dados relacional distribuído particionado, é possível utilizar o comando DSPNODGRP para visualizar o relacionamento entre os números de partição e os nomes de nós. Consulte Exibindo Grupos de Nós Utilizando o Comando DSPNODGRP com o DB2 Multisystem para obter informações adicionais sobre a exibição de números de partição, grupos de nós e nomes de sistemas.

Ao criar um arquivo distribuído, os campos da chave de particionamento são especificados no parâmetro PTNKEY do comando Criar Arquivo Físico (CRTPF) ou na cláusula PARTITIONING KEY da instrução SQL CREATE TABLE. Os campos com os tipos de dados DATE, TIME, TIMESTAMP e FLOAT não são permitidos em uma chave de particionamento.

### Conceitos relacionados

“Como os Grupos de Nós Funcionam com o DB2 Multisystem” na página 5

Um *grupo de nós* é um objeto do sistema (\*NODGRP) armazenado no sistema em que ele foi criado.

“Comando Exibir Grupo de Nós (DSPNODGRP)” na página 8

O comando Exibir Grupo de Nós (DSPNODGRP) exibe os nós (sistemas) em um grupo de nós.

“Comando Alterar Atributos do Grupo de Nós (CHGNODGRPA)” na página 10

O comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) altera os atributos de particionamento de dados de um grupo de nós.

## Plano para o Particionamento com o DB2 Multisystem

Na maior parte dos casos, é necessário planejar com antecedência para determinar como deseja utilizar o particionamento e as chaves de particionamento.

Como os dados devem ser divididos sistematicamente para disposição em outros sistemas? Quais dados você deseja unir em uma consulta com frequência? Qual é uma opção significativa ao fazer seleções? Qual é a forma mais eficaz de configurar a chave de particionamento para obter os dados necessários?

Ao planejar o particionamento, é necessário configurá-lo para que os sistemas mais rápidos recebam a maior parte dos dados. Você deve considerar quais sistemas aproveitam o paralelismo do SMP (Multiprocessamento Simétrico) para melhorar o desempenho do banco de dados. Observe que quando o otimizador de consulta constrói seu plano de acesso distribuído, o otimizador conta o número de registros no nó solicitante e multiplica esse número pelo número total de nós. Embora colocar a maior parte dos registros nos sistemas SMP tenha vantagens, o otimizador pode contrabalançar algumas dessas vantagens porque utiliza um número igual de registros em cada nó para seus cálculos. Para obter informações sobre SMP, consulte Conceitos de Programação SQL e Programação do Banco de Dados.

Se deseja influenciar o particionamento, será possível fazê-lo. Por exemplo, em seu negócio, os departamentos de vendas regionais utilizam determinados sistemas para realizar seu trabalho. Utilizando o particionamento, é possível forçar que os dados locais de cada região sejam armazenados no sistema apropriado para essa região. Dessa forma, o sistema que os funcionários na região noroeste dos Estados Unidos utilizam contém os dados da região noroeste.

Para configurar o particionamento, é possível utilizar os parâmetros PTNFILE e PTNMBR do comando CRTPF. Utilize o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) para redistribuir um arquivo já particionado. Consulte Customizando a Distribuição de Dados com o DB2 Multisystem para obter informações adicionais.

As melhorias no desempenho são melhores para consultas feitas em arquivos grandes. Os arquivos com alta utilização para o processamento de transações mas raramente utilizados para consultas podem não ser os melhores candidatos para o particionamento e devem ser deixados como arquivos locais.

Para o processamento de junções, se for feita junção de dois arquivos em um campo específico, esse campo deve se tornar a chave de particionamento dos dois arquivos. Também é necessário assegurar-se de que os campos sejam do mesmo tipo de dados.

#### **Conceitos relacionados**

Conceitos de Programação SQL

Programação do Banco de Dados

“Customização da Distribuição de Dados com o DB2 Multisystem” na página 23

Como o sistema é responsável por fornecer os dados, você não precisa saber onde os registros residem na realidade. Entretanto, se deseja garantir que determinados registros sempre sejam armazenados em um sistema específico, poderá utilizar o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) para especificar onde esses registros residem.

## **Escolher Chaves de Particionamento com o DB2 Multisystem**

Para que o sistema processe o arquivo particionado da forma mais eficiente, há algumas dicas que devem ser consideradas ao configurar ou utilizar uma chave de particionamento.

Essas dicas são:

- A melhor chave de particionamento é aquela que tem vários valores diferentes e, portanto, a atividade de particionamento resulta em uma distribuição equilibrada dos registros de dados. Números de cliente, sobrenomes, números de indenização, CEPs (códigos de endereçamento postal regionais) e códigos de área de telefone são exemplos de boas categorias para utilizar como chaves de particionamento.

O sexo é um exemplo de uma má escolha de uma chave de particionamento porque tem apenas duas opções: masculino ou feminino. O sexo faz com que muitos dados sejam distribuídos para um único nó, em vez de propagarem pelos nós. Além disso, ao fazer uma consulta, o sexo como chave de particionamento faz com que o sistema processe muitos registros de dados. Ele é ineficiente; outro campo ou campos de dados podem reduzir o escopo da consulta e torná-la muito mais eficiente. Uma

chave de particionamento baseada em sexo é uma má opção nos casos em que a distribuição equilibrada dos dados é desejada no lugar de uma distribuição baseada em valores específicos.

Ao preparar a alteração de um arquivo local para um arquivo distribuído, é possível utilizar a função HASH para ter uma idéia de como os dados estão distribuídos. Como a função HASH pode ser utilizada em arquivos locais e com qualquer variedade de colunas, é possível tentar diferentes chaves de particionamento antes de realmente alterar o arquivo para que seja distribuído. Por exemplo, se planeja utilizar o campo de CEP de um arquivo, é possível executar a função HASH utilizando esse campo para ter uma idéia do número de registros que fazem HASH para cada número de partição. Isso ajuda na escolha dos campos da chave de particionamento ou na criação do mapa de partição nos grupos de nós.

- Não escolha um campo que precise ser atualizado com freqüência. Uma restrição aos campos da chave de particionamento é que eles podem ter os valores atualizados apenas se a atualização não forçar que o registro vá para um nó diferente.
- Não utilize muitos campos na chave de particionamento; a melhor opção é utilizar um campo. A utilização de muitos campos força o sistema a fazer mais trabalho em tempo de E/S.
- Escolha um tipo de dados simples, como um inteiro ou caractere de comprimento fixo, como a chave de particionamento. Essa consideração pode ajudar no desempenho porque o hash é feito para um único campo de tipo de dados simples.
- Ao escolher uma chave de particionamento, você deve considerar os critérios de junção e agrupamento das consultas normalmente executadas. Por exemplo, escolher um campo que nunca é utilizado como um campo de junção para um arquivo envolvido em junções, pode afetar negativamente o desempenho da junção. Consulte Design de Consulta para Desempenho com o DB2 Multisystem para obter informações sobre a execução de consultas que envolvam arquivos distribuídos.

#### **Conceitos relacionados**

“Design de Consulta para Desempenho com o DB2 Multisystem” na página 41

Este tópico fornece algumas orientações para o design de consultas para que elas utilizem recursos de consulta com mais eficácia ao executar consultas que utilizam arquivos distribuídos.

## **Customização da Distribuição de Dados com o DB2 Multisystem**

Como o sistema é responsável por fornecer os dados, você não precisa saber onde os registros residem na realidade. Entretanto, se deseja garantir que determinados registros sempre sejam armazenados em um sistema específico, poderá utilizar o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) para especificar onde esses registros residem.

Como exemplo, suponha que deseja que todos os registros com o CEP 55902 residam no sistema em Minneapolis, Minnesota. Ao emitir o comando CHGNODGRPA, é necessário especificar o CEP 55902 e o número de nó do sistema do nó local em Minneapolis.

Nesse ponto, o CEP 55902 alterou grupos de nós, mas os dados ainda estão distribuídos como estavam anteriormente. O comando CHGNODGRPA não afeta os arquivos existentes. Quando um arquivo particionado é criado, ele mantém uma cópia das informações do grupo de nós nesse momento. O grupo de nós pode ser alterado ou excluído sem afetar o arquivo particionado. Para que as alterações nos registros que serão redistribuídos entrem em efeito, é possível recriar o arquivo distribuído utilizando o novo grupo de nós ou utilizar o comando Alterar Arquivo Físico (CHGPF) e especificar o grupo de nós novo ou atualizado.

Utilizando o comando CHGPF é possível:

- Redistribuir um arquivo já particionado
- Alterar uma chave de particionamento (a partir do código de área do telefone para o ID da filial, por exemplo)
- Alterar um arquivo local para ser um arquivo distribuído
- Alterar um arquivo distribuído para ser um arquivo local.

**Nota:** Também é necessário utilizar o comando CHGNODGRPA para redistribuir um arquivo já particionado. O comando CHGNODGRPA pode ser utilizado opcionalmente com o comando CHGPF para fazer qualquer uma das demais tarefas.

Consulte Problemas de redistribuição ao incluir sistemas em uma rede para obter informações sobre a alteração de um arquivo local para um arquivo distribuído ou vice-versa.

#### **Conceitos relacionados**

“Comando Alterar Atributos do Grupo de Nós (CHGNODGRPA)” na página 10

O comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) altera os atributos de particionamento de dados de um grupo de nós.

“Plano para o Particionamento com o DB2 Multisystem” na página 21

Na maior parte dos casos, é necessário planejar com antecedência para determinar como deseja utilizar o particionamento e as chaves de particionamento.

“Problemas de Redistribuição ao Incluir Sistemas em uma Rede” na página 40

A redistribuição de um arquivo por um grupo de nós é um processo razoavelmente simples.

---

## **Tabelas Particionadas**

O DB2 UDB para iSeries suporta tabelas particionadas utilizando SQL.

O particionamento permite que os dados sejam armazenados em mais de um membro, mas a tabela aparece como um objeto para operações de manipulação de dados, como consultas, inserções, atualizações e exclusões. As partições herdam as características de design da tabela na qual elas foram baseadas, incluindo os nomes e tipos das colunas, restrições e acionadores.

O particionamento permite ter muito mais dados nas tabelas. Sem o particionamento, há um máximo de 4.294.967.288 linhas em uma tabela ou um tamanho máximo de 1,7 terabytes. Entretanto, uma tabela particionada pode ter várias partições, onde cada uma pode ter o tamanho máximo da tabela. Para obter informações sobre o tamanho máximo das tabelas particionadas, consulte os White Papers do DB2 UDB para iSeries.

O particionamento também pode aumentar o desempenho, a capacidade de recuperação e a capacidade de gerenciamento do banco de dados. Cada partição pode ser salva, restaurada, exportada, importada, eliminada ou reorganizada independentemente das demais partições. Além disso, o particionamento permite a rápida exclusão de conjuntos de registros agrupados em uma partição, em vez de processar linhas individuais de uma tabela não-particionada. A eliminação de uma partição fornece um desempenho significativamente melhor que a exclusão das mesmas linhas de uma tabela não-particionada.

Uma tabela particionada criada em um servidor iSeries é um arquivo de banco de dados com vários membros. Uma partição é o equivalente de um membro do arquivo do banco de dados. Portanto, a maioria dos comandos da CL utilizados para membros também é válida para cada partição de uma tabela particionada.

O DB2 Multisystem deve estar instalado no servidor iSeries para obter proveito do suporte a tabelas particionadas. Entretanto, existem algumas diferenças importantes entre o DB2 Multisystem e o particionamento. O DB2 Multisystem fornece duas formas de particionar os dados:

- É possível criar uma tabela distribuída para distribuir os dados entre diversos sistemas iSeries ou partições lógicas.
- É possível criar uma tabela particionada para particionar os dados em vários membros na mesma tabela de banco de dados em um sistema.

Em ambos os casos, a tabela é acessada como se não fosse particionada.

#### **Conceitos relacionados**

“DB2 Multisystem: Conceitos e Termos Básicos” na página 3

Um *arquivo distribuído* é um arquivo do banco de dados propagado por vários servidores iSeries. Esta seção descreve alguns dos conceitos principais utilizados na discussão da criação e utilização de arquivos distribuídos pelo DB2 Multisystem.

#### Informações relacionadas

White Papers do DB2 UDB para iSeries

## Criação de Tabelas Particionadas

Novas tabelas particionadas podem ser criadas utilizando-se a instrução CREATE TABLE.

A definição da tabela deve incluir o nome da tabela e os nomes e atributos das colunas na tabela. A definição também poderia incluir outros atributos da tabela, como a chave primária.

Há dois métodos disponíveis para particionamento: particionamento hash e particionamento de intervalo. O particionamento hash coloca linhas em intervalos aleatórios por um número de partições e colunas-chave especificadas pelo usuário. O particionamento de intervalo divide a tabela com base em intervalos de valores de coluna especificados pelo usuário. Especifique o tipo de particionamento que deseja utilizar com a cláusula PARTITION BY. Por exemplo, para particionar a tabela PAYROLL na biblioteca PRODLIB com a chave de particionamento EMPNUM em quatro partições, utilize o seguinte código:

```
CREATE TABLE PRODLIB.PAYROLL
(EMPNUM INT,
 FIRSTNAME CHAR(15),
  LASTNAME CHAR(15),
  SALARY INT)
PARTITION BY HASH(EMPNUM)
INTO 4 PARTITIONS
```

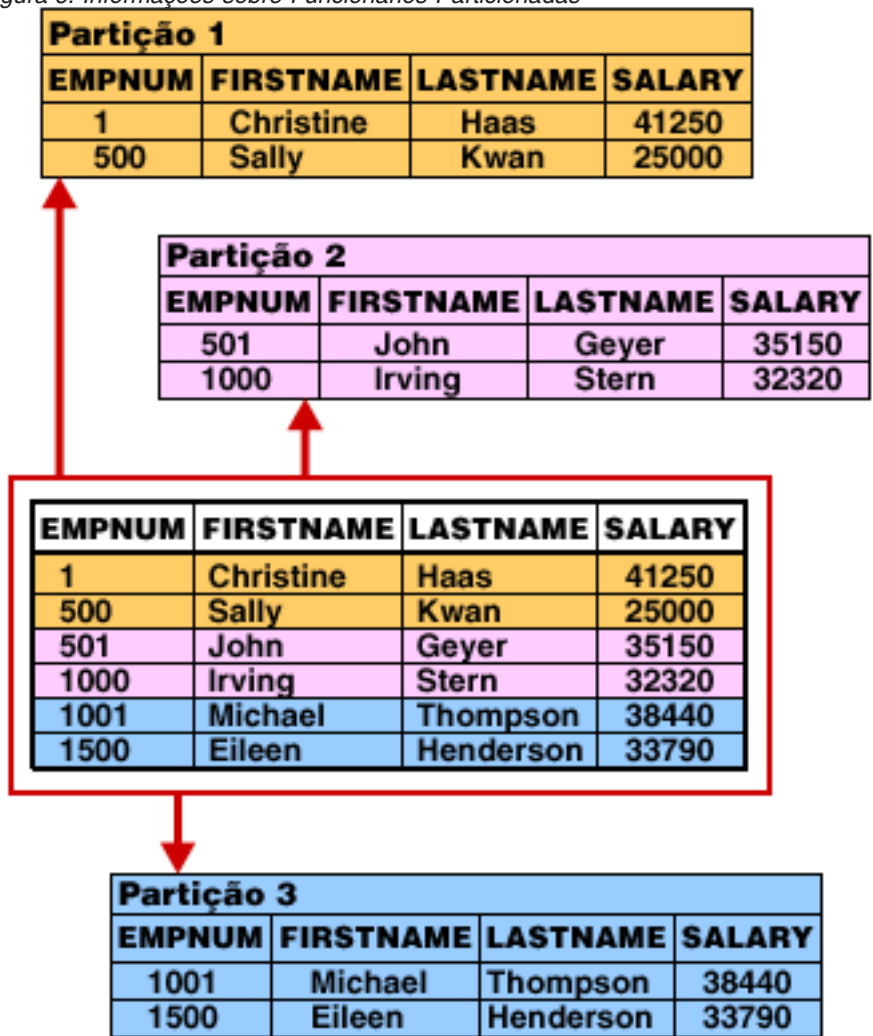
Ou, para particionar PAYROLL por intervalo, utilize o seguinte código:

```
CREATE TABLE PRODLIB.PAYROLL
(EMPNUM INT,
 FIRSTNAME CHAR(15),
  LASTNAME CHAR(15),
  SALARY INT)
PARTITION BY RANGE(EMPNUM)
(STARTING FROM (MINVALUE) ENDING AT (500) INCLUSIVE,
 STARTING FROM (501) ENDING AT (1000) INCLUSIVE,
 STARTING FROM (1001) ENDING AT (MAXVALUE))
```

Essa instrução resulta em uma tabela que contém três partições. A primeira partição contém todas as linhas em que EMPNUM é menor ou igual a 500. A segunda partição contém todas as linhas em que EMPNUM está entre 501 e 1000, inclusive. A terceira partição contém todas as linhas em que EMPNUM é

maior ou igual a 1001. A figura a seguir mostra uma tabela com dados particionados de acordo com esses valores.

Figura 6. Informações sobre Funcionários Particionadas



Quando uma tabela particionada é criada, uma restrição de verificação fornecida pelo sistema é incluída em cada partição. Essa restrição de verificação não pode ser exibida, alterada nem removida pelo usuário.

Para o particionamento de intervalo, essa restrição de verificação valida se os dados estão no intervalo apropriado. Ou, se a partição permitir valores nulos, a restrição de verificação valida se os dados são nulos.

Para o particionamento hash, essa restrição de verificação valida se os dados baseados na condição  $\text{Número da partição} = \text{MOD}(\text{Hash}(\text{campos}), \text{Número de partições}) + 1$  em que a função Hash retorna um valor entre 0 e 1023. Os valores nulos são sempre colocados na primeira partição.

Consulte a instrução CREATE TABLE no tópico Referência de SQL para cláusulas de particionamento e diagramas de sintaxe.

#### Conceitos relacionados



“De uma Tabela Não-particionada para uma Tabela Particionada”

Utilize *ADD partitioning-clause* da instrução ALTER TABLE para alterar uma tabela não-particionada em uma tabela particionada. A alteração de uma tabela existente para utilizar partições é semelhante à criação de uma nova tabela particionada.

#### Tarefas relacionadas

CREATE TABLE

#### Referências relacionadas

“Otimização de Restrição de Verificação” na página 31

O otimizador utiliza restrições de verificação (incluídas pelo usuário ou as restrições de verificação implícitas incluídas para a chave de particionamento) para reduzir as partições examinadas.

## Modificação de Tabelas Existentes

É possível alterar as tabelas não-particionadas para tabelas particionadas, alterar os atributos de tabelas particionadas existentes ou alterar tabelas particionadas para tabelas não-particionadas.

Utilize a instrução ALTER TABLE para fazer alterações em tabelas particionadas e não-particionadas.

Detalhes adicionais sobre a instrução ALTER TABLE e suas cláusulas estão disponíveis no tópico Referência de SQL.

#### Referências relacionadas

Referência SQL

## De uma Tabela Não-particionada para uma Tabela Particionada

Utilize *ADD partitioning-clause* da instrução ALTER TABLE para alterar uma tabela não-particionada em uma tabela particionada. A alteração de uma tabela existente para utilizar partições é semelhante à criação de uma nova tabela particionada.

Por exemplo, para incluir particionamento de intervalo na tabela não-particionada PAYROLL, utilize o seguinte código:

```
ALTER TABLE PRODLIB.PAYROLL
ADD PARTITION BY RANGE(EMPNUM)
(STARTING(MINVALUE) ENDING(500) INCLUSIVE,
STARTING(501) ENDING(1000) INCLUSIVE,
STARTING(1001) ENDING MAXVALUE))
```

#### Conceitos relacionados

“Criação de Tabelas Particionadas” na página 25

Novas tabelas particionadas podem ser criadas utilizando-se a instrução CREATE TABLE.

## Modificação de Tabelas Particionadas Existentes

É possível fazer várias alterações na tabela particionada utilizando as cláusulas da instrução ALTER TABLE.

Essas cláusulas são as seguintes:

- ADD PARTITION

Essa cláusula inclui uma ou mais partições hash ou uma partição de intervalo em uma tabela particionada existente. Certifique-se de que os parâmetros especificados não violam as regras a seguir; caso contrário, ocorrerão erros.

- Não utilize a cláusula ADD PARTITION para tabelas não-particionadas.
- Ao incluir partições hash, o número de partições incluídas deve ser especificado com o um inteiro positivo.
- Se fornecer um nome ou inteiro para identificar a partição, assegure-se de que ele ainda não esteja em uso por outra partição.
- Ao incluir partições de intervalo, os intervalos especificados não devem sobrepor os intervalos de nenhuma partição existente.

Por exemplo, para alterar a tabela PAYROLL na biblioteca PRODLIB com a chave de particionamento EMPNUM de forma que tenha quatro partições adicionais, utilize o seguinte código:

```
ALTER TABLE PRODLIB.PAYROLL  
ADD PARTITION 4 HASH PARTITIONS
```

- ALTER PARTITION

Essa cláusula altera o intervalo da partição de intervalo identificada. Assegure-se de que as seguintes condições sejam atendidas:

- A partição identificada deve existir na tabela.
- Os intervalos especificados não devem sobrepor os intervalos de nenhuma partição existente.
- Todas as linhas existentes da tabela particionada devem estar nos novos intervalos especificados na instrução ALTER TABLE.

- DROP PARTITION

Essa cláusula elimina uma partição de uma tabela particionada. Se a tabela especificada não for uma tabela particionada, será retornado um erro. Se a última partição restante de uma tabela particionada for especificada, será retornado um erro.

### Restrições ao Alterar o Tipo de Dados de uma Coluna:

Ao alterar o tipo de dados de uma coluna e esta fizer parte de uma chave de particionamento, há algumas restrições no tipo dos dados de destino.

Para particionamento de intervalo, o tipo dos dados de uma coluna utilizada para particionar uma tabela não pode ser alterado para BLOB, CLOB, DBCLOB, DATALINK, tipo de ponto flutuante ou um tipo distinto com base nesses tipos. Para particionamento hash, o tipo dos dados da coluna utilizada como parte de uma chave de particionamento não pode ser alterado para LOB, DATE, TIME, TIMESTAMP, tipo de ponto flutuante ou um tipo distinto com base em um desses.

#### Referências relacionadas

ALTER TABLE

### De uma Tabela Particionada para uma Tabela Não-particionada

Utilize a cláusula DROP PARTITIONING para alterar uma tabela particionada para uma tabela não-particionada.

Se a tabela especificada já for não-particionada, será retornado um erro. A alteração de uma tabela particionada que contém dados em uma tabela não-particionada requer a movimentação de dados entre as partições de dados. Não é possível alterar uma tabela particionada cujo tamanho seja maior que o tamanho máximo da tabela não-particionada para uma tabela não-particionada.

## Índices com Tabelas Particionadas

Os índices podem ser criados como particionados ou não-particionados. Uma índice particionado cria um índice individual para cada partição. Um índice não-particionado é um único índice que se estende por todas as partições da tabela.

Os índices particionados permitem aproveitar a otimização aprimorada das consultas. Se um índice exclusivo for particionado, as colunas especificadas no índice devem ser as mesmas ou um superconjunto da chave de particionamento dos dados.

Utilize a instrução CREATE INDEX para criar índices em tabelas particionadas. Para criar um índice para cada partição, utilize a cláusula PARTITIONED.

```
CREATE INDEX PRODLIB.SAMPLEINDEX  
ON PRODLIB.PAYROLL(EMPNUM) PARTITIONED
```

Para criar um índice único que se estenda por todas as partições, utilize a cláusula NOT PARTITIONED.

```
CREATE INDEX PRODLIB.SAMPLEINDEX  
ON PRODLIB.PAYROLL(EMPNUM) NOT PARTITIONED
```

Um EVI (Índice de Vetor Codificado) particionado pode ser criado apenas sobre uma tabela particionada. Não é possível criar um EVI não-particionado sobre uma tabela particionada.

Consulte a instrução CREATE INDEX no tópico Referência de SQL para obter informações adicionais sobre a criação de índices para tabelas particionadas.

Ao criar um índice SQL exclusivo, uma restrição exclusiva ou uma restrição de chave primária para uma tabela particionada, as seguintes restrições são aplicadas:

- Um índice pode ser particionado se as chaves do índice exclusivo forem as mesmas ou um superconjunto das chaves de particionamento.
- Se um índice exclusivo for criado com o valor padrão NOT PARTITIONED e as chaves do índice exclusivo forem um superconjunto das chaves de particionamento, o índice exclusivo será criado como particionado. Entretanto, se o usuário especificar explicitamente NOT PARTITIONED e as chaves do índice exclusivo forem um superconjunto das chaves de particionamento, o índice exclusivo será criado como **não** particionado.

#### Conceitos relacionados

“Desempenho e Otimização da Consulta”

As consultas que fazem referência a tabelas particionadas precisam ser consideradas com cuidado porque as tabelas particionadas normalmente são muito grandes. É importante entender os efeitos do acesso a várias partições no sistema e a aplicativos.

#### Tarefas relacionadas

CREATE INDEX

## Desempenho e Otimização da Consulta

As consultas que fazem referência a tabelas particionadas precisam ser consideradas com cuidado porque as tabelas particionadas normalmente são muito grandes. É importante entender os efeitos do acesso a várias partições no sistema e a aplicativos.

As tabelas particionadas podem aproveitar toda a otimização e o processamento paralelo disponíveis utilizando o SQL no DB2 Universal Database para iSeries. Para obter informações gerais sobre a otimização da consulta, consulte Desempenho e Otimização do Banco de Dados. Para tabelas particionadas, todos os métodos de acesso a dados descritos no tópico Desempenho e otimização do banco de dados podem ser utilizados para acessar os dados em cada partição. Além disso, se o recurso DB2 UDB Symmetric Multiprocessing estiver instalado, os métodos de acesso paralelo a dados estarão disponíveis para que o otimizador considere ao implementar a consulta.

Se as consultas precisarem acessar apenas uma partição individual em uma tabela particionada, a criação de um alias para essa partição individual e, em seguida, a utilização desse alias na consulta pode melhorar o desempenho. A consulta age como uma consulta de tabela não-particionada.

As tabelas particionadas são conceitualmente implementadas com o uma tabela aninhada em que cada partição é unida às outras partições.

Por exemplo, se executar a seguinte consulta:

```
SELECT LASTNAME, SALARY FROM PRODLIB.PAYROLL  
WHERE SALARY > 20000
```

A implementação da consulta poderá ser descrita como:

```
SELECT LASTNAME, SALARY FROM  
(SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00001)  
UNION ALL  
SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00002))
```

```

UNION ALL
SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00003))
X (LASTNAME, SALARY)
WHERE X.SALARY > 20000

```

A implementação das consultas da tabela particionada depende de qual mecanismo de consulta é utilizado: o CQE (Classic Query Engine) ou o SQE (SQL Query Engine). Para obter informações adicionais sobre os mecanismos de consulta, consulte o artigo Mecanismos SQE e CQE no tópico Desempenho e otimização do banco de dados. Existem considerações diferentes para cada mecanismo.

#### Conceitos relacionados

“Índices com Tabelas Particionadas” na página 28

Os índices podem ser criados como particionados ou não-particionados. Uma índice particionado cria um índice individual para cada partição. Um índice não-particionado é um único índice que se estende por todas as partições da tabela.

Mecanismos SQE e CQE

#### Informações relacionadas

Desempenho e Otimização do Banco de Dados

### | Consultas Utilizando o SQL Query Engine

| O SQE (SQL Query Engine) fornece otimização direcionada a tabelas particionadas utilizando a otimização da expansão dinâmica da partição.

| Esse método de otimização direcionada primeiramente determina se uma consulta específica está estruturada de forma que certas partições na tabela particionada se beneficiem da otimização específica. Se a otimização direcionada for garantida, o otimizador determinará quais partições podem se beneficiar da otimização individual; em seguida, essas partições serão otimizadas separadamente. As partições restantes utilizam a técnica *de uma vez por todas*.

| O otimizador determina se o ambiente da tabela ou a consulta justifica a expansão dinâmica com base nas seguintes características:

- | • A tabela é particionada por intervalo e a consulta envolve a seleção de predicado no intervalo.
- | • A tabela tem um índice sobre uma ou algumas das partições, mas não todas (um *índice que se estende pela subpartição*).
- | • A tabela tem relativamente poucas partições.
- | • As definições de restrição na tabela ditam que apenas algumas partições participam.
- | • O tempo de execução estimado excede um limite específico.

| Se a expansão for justificada, o otimizador determinará as partições de destino utilizando técnicas estatísticas existentes, conforme apropriado. Por exemplo, para o particionamento de intervalo e a seleção de predicado, o otimizador examina as estatísticas ou o índice para determinar quais partições principais despertam interesse. Quando as partições de destino forem identificadas, o otimizador regravará a consulta. As partições de destino são redefinidas em uma operação UNION. As partições restantes permanecem como uma *instância única de tabela*. Essa instância única será então incluída na operação UNION juntamente com as partições de destino. Tão logo a regravação seja executada, o otimizador utiliza as técnicas de otimização existentes para determinar o plano para cada parte da UNION. No final da otimização, a instância única será convertida em outra operação UNION de suas partições contidas. O plano otimizado para a instância única é replicado pelas subárvores de UNION, dessa forma reduzindo drasticamente o tempo de otimização para tabelas particionadas.

| O SQL Query Engine também utiliza *eliminação de partição lógica* para otimizar tabelas particionadas. Esse método permite que o otimizador identifique oportunidades potenciais de eliminação da partição. O otimizador procura oportunidades em que um conjunto reduzido de respostas da tabela de origem pode ser aplicado à chave de definição da tabela da partição com um predicado de junção. Quando essas

| oportunidades forem identificadas, o otimizador construirá a lógica durante o tempo de execução da consulta para eliminar partições com base no conjunto de resultados da tabela de origem.

| Por exemplo, considere uma consulta na qual o otimizador identifica um predicado (nesse exemplo, uma cláusula WHERE) que envolva a chave de particionamento da tabela de partições. Se uma restrição em uma partição limitar a chave para um intervalo de 0 a 10 e um predicado na consulta identificar um valor de chave igual a 11, a partição poderá ser eliminada logicamente. Observe que a implementação da partição ainda será construída na consulta, mas o caminho será alterado pelo processamento direto da restrição combinada com o predicado. Essa lógica é construída com objetivos de capacidade de reutilização. Outra consulta com um predicado que identifique um valor de chave igual a 10 também pode ser implementada com esse plano. Se a partição foi removida fisicamente da implementação, a implementação não poderia ser reutilizada.

| Considere esse exemplo mais complicado:

```
| select *  
| from sales, timeDim  
| where sales.DateId = timeDim.DateId  
| and timeDim.Date > '09/01/2004'
```

| Nesse exemplo, sales é uma tabela particionada por intervalo em que sales.DateId é identificado como a chave de particionamento e sales.DateId = timeDim.DateId é identificado como um potencial predicado de redução. A consulta é modificada (aproximadamente) como:

```
| with sourceTable as (select * from timeDim where timeDim.Date > '09/01/2004')  
| select *  
| from sales, sourceTable  
| where sales.DateId = sourceTable.DateId  
| and sales.DateId IN (select DateId from sourceTable)
```

| Nesse exemplo, a junção original sobrevive, mas o predicado local adicional, sales.DateId IN (select DateId from sourceTable), é incluído. Agora esse novo predicado pode ser combinado com informações sobre a definição da partição para produzir uma resposta de quais partições participam. Esse resultado é produzido devido às seguintes condições:

- | • A existência de uma junção na chave de particionamento.
- | • A junção da tabela pode ser reduzida em um conjunto menor de valores de junção. Então, esses valores de junção poderão ser utilizados para alterar as partições na tabela particionada.

### | **Otimização de Restrição de Verificação:**

| O otimizador utiliza restrições de verificação (incluídas pelo usuário ou as restrições de verificação implícitas incluídas para a chave de particionamento) para reduzir as partições examinadas.

| Para obter informações adicionais sobre as restrições de chave de particionamento incluídas implicitamente, consulte Criando Tabelas Particionadas. No exemplo a seguir, assumo que PAYROLL seja particionada por intervalo:

```
| SELECT LASTNAME, SALARY  
| FROM PRODLB.PAYROLL  
| WHERE EMPNUM = :hv1
```

| O otimizador inclui novos predicados em cada partição na tabela:

```
| SELECT LASTNAME, SALARY FROM  
| (SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00001)  
| WHERE EMPNUM=:hv1 AND :hv1 <= 500  
| UNION ALL  
| SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00002)  
| WHERE EMPNUM=:hv1 AND :hv1 >= 501 AND :hv1 <=1000
```

```
| UNION ALL
| SELECT LASTNAME, SALARY FROM PRODLB.PAYROLL(PART00003)
| WHERE EMPNUM=:hv1 AND (:hv1 IS NULL OR :hv1 >= 1001))
| X (LASTNAME, SALARY)
```

| Se o valor de hv1 for 603, apenas as linhas da partição PART00002 serão examinadas.

#### | **Conceitos relacionados**

| “Criação de Tabelas Particionadas” na página 25

| Novas tabelas particionadas podem ser criadas utilizando-se a instrução CREATE TABLE.

#### | **Uso de Índice:**

| O SQE (SQL Query Engine) utiliza índices particionados e não-particionados para implementar consultas.

| Se um índice não-particionado for utilizado, o otimizador pode optar por implementar o índice não-particionado para cada partição. Isso implica que, por exemplo, mesmo se o índice não-particionado for exclusivo, cada partição será sondada pela linha.

### **Consultas Utilizando o Classic Query Engine**

Quando as consultas são implementadas utilizando o CQE (Classic Query Engine), você precisa estar ciente de como a consulta é otimizada, incluindo o uso de índice e materialização.

#### **Materialização:**

Ao executar o CQE (Classic Query Engine), uma tabela particionada é materializada sob algumas condições.

Essas condições são:

- Ela faz parte de uma consulta de junção.
- Ela tem uma cláusula GROUP BY.
- Ela tem uma função de coluna incorporada.

Para reduzir o tamanho da tabela temporária e reduzir o tempo de execução envolvido na consulta, qualquer seleção na consulta que possa ser utilizada durante a criação da tabela temporária para eliminar linhas será processada para diminuir o tamanho da tabela temporária e reduzir o tempo de processamento durante os casos de materialização.

**Nota:** Quando a tabela particionada é materializada, o tamanho da tabela temporária não pode exceder 4.294.967.288 linhas. O pushdown do processamento da seleção é executado para limitar o número de linhas na tabela temporária. Se o número de linhas exceder 4.294.967.288, um erro de limite de recursos será emitido e a consulta será finalizada.

#### **Considerações sobre Otimização de Consultas do CQE:**

O otimizador do CQE (Classic Query Engine) otimiza a consulta utilizando o primeiro membro da partição na tabela particionada. Esse método de acesso é utilizado para acessar linhas de todas as partições.

Os métodos de acesso a dados do recurso DB2 UDB Symmetric Multiprocessing não podem ser utilizados para acessar uma tabela particionada, mas podem ser utilizados ao processar tabelas temporárias e para criar índices temporários.

Se a tabela particionada for utilizada em uma consulta que contenha uma subconsulta, o otimizador não tentará implementar a consulta como uma consulta composta por junção. Isso impede que a tabela particionada seja materializada devido à junção.

## Uso de Índice:

O CQE (Classic Query Engine) utiliza índices não-particionados para implementar consultas.

Se existir um índice particionado, o otimizador não utiliza o índice para tomar decisões de otimização relativas ao método de acesso a ser utilizado durante o processamento da tabela particionada.

O otimizador cria índices temporários sobre tabelas particionadas permitindo a atualização dos dados ativos com ordenação.

Como os EVIs (Índices de Vetor Codificados) são criados apenas sobre uma partição única, o CQE não pode utilizar os EVIs para implementar consultas de tabelas particionadas.

## Considerações sobre Salvamento e Restauração

Uma tabela particionada pode ser salva e restaurada exatamente como qualquer outro arquivo de banco de dados.

As partições de uma tabela particionada são membros do banco de dados e, dessa forma, podem ser salvas e restauradas em conjunto ou individualmente. Considere os itens a seguir ao salvar e restaurar tabelas particionadas.

- Se restaurar apenas algumas das partições de uma tabela particionada para um sistema em que a tabela particionada não existia anteriormente, o sistema criará as partições que não foram restauradas. As partições criadas que não foram restauradas não podem ter nenhum dado restaurado.
- Se salvar uma tabela particionada por intervalo e, em seguida, eliminar uma ou mais partições, poderá restaurar as partições eliminadas para a tabela particionada.
- Se salvou uma tabela particionada por hash e, em seguida, alterar a tabela eliminando ou incluindo partições, não poderá restaurar a tabela que foi salva na tabela no sistema. Entretanto, se a tabela no sistema for excluída, será possível restaurar a tabela que foi salva.

Os aplicativos que utilizam os seguintes comandos da CL de salvamento e restauração devem ser alterados para utilizar Member \*ALL para processar todas as partições de uma tabela particionada:

- Restaurar Objeto (RSTOBJ)
- Salvar Objeto (SAVOBJ)
- Salvar Objeto de Restauração (SAVRSTOBJ)

### Tarefas relacionadas

Backup e Recuperação do Banco de Dados

## Criação de Diário

É possível criar o diário de uma tabela particionada da mesma forma que o faria com qualquer outro arquivo de banco de dados com vários membros. Ao criar o diário de uma tabela particionada, todas as partições da tabela são registradas no mesmo diário.

Os aplicativos que utilizam os seguintes comandos da CL de criação de diário devem ser alterados para utilizar Member \*ALL para processar todas as partições de uma tabela particionada:

- Aplicar Alterações Registradas (APYJRNCHG)
- Aplicar Extensão de Alterações Registradas (APYJRNCHGX)
- Exibir Diário (DSPJRN)
- Receber Entrada do Diário (RCVJRNE)
- Remover Alterações Registradas (RMVJRNCHG)
- Recuperar Entrada do Diário (RTVJRNE)

### Conceitos relacionados

## Considerações sobre a Interface Nativa

Uma tabela SQL é um arquivo físico do banco de dados com um membro (partição). Portanto, quando o arquivo é acessado por um aplicativo nativo, este lê e grava no membro abrindo o membro do arquivo.

Quando o arquivo (tabela SQL) é particionado, ele se torna um arquivo de vários membros e o aplicativo nativo precisa especificar o nome do membro (nome da partição). O aplicativo nativo pode evitar a necessidade de especificar o nome de um membro ao ler ou gravar dados alterando-se o aplicativo para utilizar um índice SQL baseado em todos os membros do arquivo físico.

Por exemplo, se o usuário criou um índice SQL com o seguinte código:

```
CREATE INDEX LIBNAME.INDEXNAME  
ON LIBNAME.TABLENAME(COLUMNNAME)  
NOT PARTITIONED
```

O aplicativo nativo poderá ler e gravar dados da tabela particionada sem precisar saber como os dados estão particionados.

Quando a tabela for particionada (se tornar um arquivo de vários membros), qualquer operação nativa anteriormente realizada na tabela deve ser feita para cada membro do arquivo de vários membros. Por exemplo, RGZPFM FILE(LIBNAME/TABLENAME) reorganiza apenas o membro \*FIRST. Para uma tabela particionada, é necessário utilizar o comando Reorganizar Membro do Arquivo Físico (RGZPFM) para cada membro. O comando Exibir Descrição do Arquivo (DSPFD), DSPFD FILE(LIBNAME/TABLENAME) TYPE(\*MBRLIST), lista todos os membros do arquivo.

### Referências relacionadas

- Reorganizar Membro do Arquivo Físico (RGZPFM)
- Exibir Descrição do Arquivo (DSPFD)

## Restrições

Existem algumas restrições para uma tabela particionada.

- As restrições de referência não são permitidas para uma tabela particionada.
- Se uma restrição de chave primária para uma tabela particionada for incluída e, em seguida eliminada, o índice da chave primária também será eliminada.
- Se uma restrição de chave primária for incluída em uma tabela particionada e, em seguida, removida pelo usuário, o usuário não terá permissão para manter a tabela com chave.
- Se uma tabela não-particionada existente não tiver uma restrição de chave primária, mas a tabela possuir chaves, as chaves serão removidas quando a tabela for alterada para uma tabela particionada.
- Os arquivos do DB2 Multisystem (arquivos distribuídos) já estão particionados por vários servidores e não podem ser particionados por vários membros em um único sistema.
- Não é permitido fazer uma atualização na chave de particionamento que tenta mover uma linha para uma partição diferente.
- O número de chaves de particionamento está restrito a 120.
- Todo o processamento de registros relativo a SQL é manipulado da mesma forma que o suporte do DB2 Multisystem. O número de registro relativo é determinado em cada partição individual, não a tabela como um todo. Por exemplo, ler o registro 27 significa ler o registro 27 em cada partição. Cada partição contém seu próprio registro 27, nenhum deles é igual.
- Há algumas restrições sobre o tipo de dados de uma coluna de chave de particionamento. Para particionamento de intervalo, o tipo dos dados de uma coluna utilizada para particionar uma tabela não pode ser BLOB, CLOB, DBCLOB, DATALINK, tipo de ponto flutuante ou um tipo distinto com



base nesses tipos. Para particionamento hash, o tipo dos dados da coluna utilizada como parte de uma chave de particionamento não pode ser LOB, DATE, TIME, TIMESTAMP, tipo de ponto flutuante ou um tipo distinto com base em um desses.

- Os aplicativos que utilizam os seguintes comandos da CL devem ser alterados para utilizar Member \*ALL para processar todas as partições de uma tabela particionada:
  - Limpar Membro do Arquivo Físico (CLRPFM)
  - Copiar do Arquivo de Importação (CPYFRMIMPF)
  - Copiar para Arquivo de Importação (CPYTOIMPF)
  - Excluir Arquivo de Rede (DLTNETF)
  - Abrir Arquivo de Consulta (OPNQRYF)
  - Executar Consulta (RUNQRY)
  - Trabalhar com Bloqueios de Objetos (WRKOBJLCK)
  - Aplicar Alterações Registradas (APYJRNCHG)
  - Aplicar Extensão de Alterações Registradas (APYJRNCHGX)
  - Exibir Diário (DSPJRN)
  - Receber Entrada do Diário (RCVJRNE)
  - Remover Alterações Registradas (RMVJRNCHG)
  - Recuperar Entrada do Diário (RTVJRNE)
  - Restaurar Objeto (RSTOBJ)
  - Salvar Objeto (SAVOBJ)
  - Salvar Objeto de Restauração (SAVRSTOBJ)

---

## Funções Escalares Disponíveis com o DB2 Multisystem

Para o DB2 Multisystem, as funções escalares estão disponíveis ao trabalhar com arquivos distribuídos.

Essas funções ajudam a determinar como distribuir os dados nos arquivos, além de determinar onde os dados estarão depois que o arquivo for distribuído. Ao trabalhar com arquivos distribuídos, os administradores de banco de dados podem considerar essas funções como sendo úteis ferramentas de depuração.

Essas funções escalares são PARTITION, HASH, NODENAME e NODENUMBER. É possível utilizar essas funções com o SQL ou o comando Abrir Arquivo de Consulta (OPNQRYF).

Para obter informações sobre a sintaxe das funções escalares do SQL, consulte a Referência de SQL. Para obter informações sobre a sintaxe das funções escalares do OPNQRYF, consulte o tópico Linguagem de Controle no Information Center.

### Referências relacionadas

Referência de SQL

Linguagem de Controle

## PARTITION com o DB2 Multisystem

Com a função PARTITION, é possível determinar o número da partição em que uma linha específica do banco de dados relacional distribuído está armazenada.

Saber o número da partição permite determinar qual nó no grupo de nós contém esse número de partição.

## Exemplos de PARTITION com o DB2 Multisystem

Aqui é fornecido um exemplo de como utilizar a função PARTITION.

- Localize o número de PARTITION de cada linha da tabela EMPLOYEE.

Instrução SQL:

```
SELECT PARTITION(CORPDATA.EMPLOYEE), LASTNAME
FROM CORPDATA.EMPLOYEE
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
FORMAT(FNAME)
MAPFLD((PART1 '%PARTITION(1)'))
```

- Selecione o número do funcionário (EMPNO) na tabela EMPLOYEE para todas as linhas em que o número da partição é igual a 100.

Instrução SQL:

```
SELECT EMPNO
FROM CORPDATA.EMPLOYEE
WHERE PARTITION(CORPDATA.EMPLOYEE) = 100
```

Comando OPNQRYF:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('%PARTITION(1) *EQ 100')
```

- Junte as tabelas EMPLOYEE e DEPARTMENT e selecione todas as linhas do resultado onde as linhas das duas tabelas têm o mesmo número de partição.

Instrução SQL:

```
SELECT *
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE PARTITION(X)=PARTITION(Y)
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
FORMAT(FNAME)
JFLD((1/PART1 2/PART2 *EQ))
MAPFLD((PART1 '%PARTITION(1)')
(PART2 '%PARTITION(2)'))
```

## HASH com o DB2 Multisystem

A função HASH retorna o número da partição aplicando a função hash nas expressões especificadas.

### Exemplo de HASH com o DB2 Multisystem

Utilize a função HASH para determinar o que devem ser as partições se a chave de particionamento for composta de EMPNO e LASTNAME.

- A consulta retorna o número da partição para cada linha em EMPLOYEE.

Instrução SQL:

```
SELECT HASH(EMPNO, LASTNAME)
FROM CORPDATA.EMPLOYEE
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
FORMAT(FNAME)
MAPFLD((HASH '%HASH(1/EMPNO, 1/LASTNAME)'))
```

## NODENAME com o DB2 Multisystem

Com a função NODENAME, é possível determinar o nome do RDB (banco de dados relacional) em que uma linha específica do banco de dados relacional distribuído está armazenada.

Saber o nome do nó permite determinar o nome do sistema que contém a linha. Isso pode ser útil para determinar se você deseja redistribuir certas linhas para um nó específico.

## Exemplos de NODENAME com o DB2 Multisystem

Aqui é fornecido um exemplo de como utilizar a função NODENAME.

- Localize o nome do nó e o número da partição para cada linha da tabela EMPLOYEE e o valor correspondente das colunas EMPNO para cada linha.

Instrução SQL:

```
SELECT NODENAME(CORPDATA.EMPLOYEE), PARTITION(CORPDATA.EMPLOYEE), EMPNO
FROM CORPDATA.EMPLOYEE
```

- Localize o nome do nó para cada registro da tabela EMPLOYEE.

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
FORMAT(FNAME)
MAPFLD((NODENAME '%NODENAME(1)'))
```

- Junte as tabelas EMPLOYEE e DEPARTMENT, selecione o número do funcionário (EMPNO) e determine o nó a partir do qual cada linha envolvida na junção é originado.

Instrução SQL:

```
SELECT EMPNO, NODENAME(X), NODENAME(Y)
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE X.DEPTNO=Y.DEPTNO
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
FORMAT(FNAME)
JFLD((EMPLOYEE/DEPTNO DEPARTMENT/DEPTNO *EQ))
MAPFLD((EMPNO 'EMPLOYEE/EMPNO')
(NODENAME1 '%NODENAME(1)')
(NODENAME2 '%NODENAME(2)'))
```

- Junte as tabelas EMPLOYEE e DEPARTMENT e selecione todas as linhas do resultado onde as linhas das duas tabelas estão no mesmo nó.

Instrução SQL:

```
SELECT *
FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
WHERE NODENAME(X)=NODENAME(Y)
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
FORMAT(FNAME)
JFLD((1/NODENAME1 2/NODENAME2 *EQ))
MAPFLD((NODENAME1 '%NODENAME(1)')
(NODENAME2 '%NODENAME(2)'))
```

## NODENUMBER com o DB2 Multisystem

Com a função NODENUMBER, é possível determinar o número do nó em que uma linha específica do banco de dados relacional distribuído está armazenada.

O número do nó é o número exclusivo designado a cada nó em um grupo de nós quando o grupo de nós foi criado. Saber o número do nó permite determinar o nome do sistema que contém a linha. Isso pode ser útil para determinar se você deseja redistribuir certas linhas para um nó específico.

## Exemplo de NODENUMBER com o DB2 Multisystem

Aqui é fornecido um exemplo de como utilizar a função NODENUMBER.

Se CORPDATA.EMPLOYEE for uma tabela distribuída, o número do nó de cada linha e o nome do funcionário serão retornados.

Instrução SQL:

```
SELECT NODENUMBER(CORPDATA.EMPLOYEE), LASTNAME
FROM CORPDATA.EMPLOYEE
```

Comando OPNQRYF:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))  
          FORMAT(FNAME)  
          MAPFLD((NODENAME '%NODENUMBER(1)')  
                (LNAME '1/LASTNAME'))
```

## Registros Especiais com o DB2 Multisystem

Para o DB2 Multisystem, todas as instâncias dos registros especiais são resolvidas no nó coordenador antes de enviar a consulta para os nós remotos. (Um *nó coordenador* é o sistema no qual a consulta foi iniciada.) Dessa forma, todos os nós executam a consulta com valores de registros especiais consistentes.

As seguintes regras se aplicam aos registros especiais:

- CURRENT SERVER sempre retorna o nome do banco de dados relacional do nó coordenador.
- O registro especial USER retorna o perfil do usuário que está executando o job no nó coordenador.
- CURRENT DATE, CURRENT TIME e CURRENT TIMESTAMP são obtidos do relógio de hora do dia no nó coordenador.
- CURRENT TIMEZONE é o valor de QUTCOFFSET do valor de sistema no nó coordenador.

## Função de Numeração Relativa de Registros com o DB2 Multisystem

A função RRN (numeração relativa de registros) retorna o número relativo do registro da linha armazenada em um nó de um arquivo distribuído.

A RRN não é exclusiva para um arquivo distribuído. Um registro exclusivo no arquivo é especificado se você combinar RRN com NODENAME ou NODENUMBER.

---

## Desempenho e Escalabilidade com o DB2 Multisystem

O DB2 Multisystem pode ajudá-lo a aumentar a capacidade do banco de dados, realizar melhorias no desempenho da consulta e fornecer acesso a banco de dados remoto utilizando um método mais fácil.

Utilizando o DB2 Multisystem, os usuários e aplicativos precisam acessar apenas o arquivo no sistema local. Não é necessário fazer alterações de código para poder acessar dados em um arquivo distribuído versus um arquivo local. Com funções como a DRDA (Distributed Relational Database Architecture) e o DDM (Distributed Data Management), o acesso deve ser dirigido explicitamente para um arquivo remoto ou um sistema remoto para que seja possível acessar esses dados remotos. O DB2 Multisystem manipula o acesso remoto de maneira que seja transparente aos usuários.

O DB2 Multisystem também fornece um caminho de crescimento simples para a expansão do banco de dados.

## Porque Você Deve Utilizar o DB2 Multisystem

As melhorias de desempenho podem ser bastante significativas para algumas consultas.

Testes mostraram que para consultas que têm uma grande quantidade de dados a ser processada, mas com um conjunto de resultados relativamente pequeno, o ganho de desempenho é quase proporcional ao número de sistemas em que o arquivo está distribuído. Por exemplo, suponha que você tenha um arquivo com 5 milhões (5.000.000) de registros que deseja consultar para os 10 principais produtores de receita. Com o DB2 Multisystem, o tempo de resposta para a consulta é reduzido aproximadamente pela metade pelo particionamento do arquivo de forma igual em dois sistemas. Em três sistemas, o tempo de resposta é aproximadamente um terço do tempo da execução da consulta em um único sistema. Esse cenário de melhor caso não se aplica a operações de junção complexas em que os dados precisam ser movidos entre os nós.

Se um arquivo for razoavelmente pequeno ou utilizado principalmente para o processamento de leitura/gravação de registros únicos, pouco ou nenhum ganho de desempenho poderá ser percebido com o particionamento do arquivo. Em vez disso, poderá ocorrer uma pequena degradação no desempenho. Nesses casos, o desempenho da consulta se torna mais dependente da velocidade da conexão física. No entanto, mesmo nessas situações, os usuários em todos os sistemas no grupo de nós ainda têm a vantagem de serem capazes de acessar os dados, muito embora eles sejam distribuídos, utilizando os métodos tradicionais de “arquivo local” do banco de dados com os quais eles estão familiarizados. Em todos os ambientes, os usuários têm os benefícios dessa transparência do sistema local e a possível eliminação da redundância de dados pelos sistemas no grupo de nós.

Outro recurso de paralelismo é o DB2 UDB Symmetric Multiprocessing. Com o multiprocessamento simétrico (SMP), quando um arquivo particionado é processado e se algum dos sistemas for um sistema multiprocessador, é possível alcançar um efeito multiplicador em termos de ganhos de desempenho. Se você particionou um arquivo por três sistemas e cada um deles é um sistema processador de 4 vias, as funções do DB2 Multisystem e SMP funcionam em conjunto. Utilizando o exemplo anterior de 5 milhões de registros, o tempo de resposta é aproximadamente um doze avos do que teria sido se a consulta fosse executada sem a utilização de nenhum dos recursos de paralelismo. Os tamanhos dos arquivos e os detalhes da consulta podem afetar o aprimoramento realmente visto.

Ao fazer consultas, grande parte do trabalho de execução da consulta é feita em paralelo, o que aprimora o desempenho geral do processamento de consultas. O sistema divide a consulta e processa as partes apropriadas da consulta no sistema apropriado. Isso contribui para o processamento mais eficaz e é feito automaticamente; não é necessário especificar nada para que esse processamento altamente eficiente ocorra.

**Nota:** O desempenho de algumas consultas podem não melhorar, especialmente se um grande volume de dados precisar ser movido.

Cada nó precisa processar apenas os registros que estão fisicamente armazenados nesse nó. Se a consulta especificar uma seleção na chave de particionamento, o otimizador de consulta pode determinar que apenas um nó precisa ser consultado. No exemplo a seguir, o campo ZIP é a chave de particionamento na instrução SQL do arquivo ORDERS:

```
SELECT NAME, ADDRESS, BALANCE FROM PRODLIB/ORDERS WHERE ZIP='48009'
```

Quando a instrução for executada, o otimizador determinará que apenas um nó precisa ser consultado. Lembre-se de que todos os registros que contêm o ZIP 48009 serão distribuídos para o mesmo nó.

No próximo exemplo de instrução SQL, a capacidade do processador de todos os servidores iSeries no grupo de nós pode ser utilizada para processar a instrução em paralelo:

```
SELECT ZIP, SUM(BALANCE) FROM PRODLIB/ORDERS GROUP BY ZIP
```

Outra vantagem de se fazer com que o otimizador direcione os pedidos de E/S apenas para sistemas que contenham dados pertinentes é que as consultas ainda poderão ser executadas se um ou mais dos sistemas não estiverem ativos. Um exemplo é um arquivo particionado de forma que cada filial de um negócio tenha seus dados armazenados em um sistema diferente. Se um sistema estiver indisponível, as operações de E/S de arquivos ainda poderão ser executadas para os dados associados às filiais restantes. O pedido de E/S falha para a filial que não está ativa.

O otimizador utiliza protocolos de confirmação em duas fases para assegurar a integridade dos dados. Como vários sistemas estão sendo acessados, se você solicitar o controle de confirmação, todos os sistemas utilizarão conversações protegidas. Uma *conversação protegida* significa que se ocorrer um defeito do sistema no meio de uma transação ou no meio de uma operação de banco de dados única, será feito rollback de todas as alterações feitas até esse ponto.

Quando forem utilizadas conversações protegidas, algumas opções de controle de confirmação são alteradas nos nós remotos para aprimorar o desempenho. A opção Aguardar por resultado é configurada

como Y e a opção Votação de leitura permitida é configurada como Y. Para aprimorar ainda mais o desempenho, é possível utilizar a API Alterar Opções de Confirmação (QTNCHGCO) para alterar a opção Aguardar por resultado para N no sistema em que as consultas são iniciadas. Consulte o tópico APIs no Information Center para entender os efeitos desses valores de opção de confirmação.

#### Referências relacionadas

APIs

### Dica de Aprimoramento de Desempenho com o DB2 Multisystem

Para o aprimoramento do desempenho, é possível especificar alguns valores para o parâmetro distribuir dados (DSTDTA).

Uma forma de garantir o melhor desempenho é especificar \*BUFFERED para o parâmetro DSTDTA no comando Substituir por Arquivo do Banco de Dados (OVRDBF). Isso instrui o sistema a recuperar dados de um arquivo distribuído o mais rápido possível, potencialmente mesmo às custas de atualizações imediatas que deverão ser feitas no arquivo. DSTDTA(\*BUFFERED) é o valor padrão para o parâmetro quando um arquivo é aberto com finalidades de leitura.

Os demais valores do parâmetro DSTDTA são \*CURRENT e \*PROTECTED. \*CURRENT permite que sejam feitas atualizações no arquivo por outros usuários, mas com algum custo de desempenho. Quando um arquivo for aberto para atualização, DSTDTA(\*CURRENT) é o valor padrão. \*PROTECTED fornece um desempenho semelhante a \*CURRENT, mas \*PROTECTED impede que as atualizações sejam feitas por outros usuários enquanto o arquivo está aberto.

### Como o DB2 Multisystem Ajuda a Expandir o Sistema de Banco de Dados

Utilizando arquivos de bancos de dados relacionais distribuídos, é possível expandir mais facilmente a configuração dos servidores iSeries.

Antes do DB2 Multisystem, se você quisesse migrar de um sistema para dois sistemas, havia vários problemas de banco de dados a resolver. Se você movesse metade dos usuários para um novo sistema, também poderia desejar mover metade dos dados para esse novo sistema. Isso o forçaria a regravar todos os aplicativos relacionados ao banco de dados, porque os aplicativos devem saber onde os dados residem. Depois de regravar os aplicativos, deveria utilizar um acesso remoto, como a DRDA (Distributed Relational Database Architecture) ou o DDM (Distributed Data Management), para acessar os arquivos pelos sistemas. Caso contrário, alguma função de replicação de dados poderia ser utilizada. Se fizesse isso, poderiam existir várias cópias dos dados, mais armazenamento seria utilizado e os sistemas também fariam o trabalho de manter as várias cópias dos arquivos nos níveis atuais.

Com o DB2 Multisystem, o processo de inclusão de novos sistemas na configuração é bastante simplificado. Os arquivos do banco de dados são particionados pelos sistemas. Em seguida, os aplicativos são motivos para o novo sistema. Os aplicativos não são alterados; não é necessário fazer alteração de programação alguma nos aplicativos. Agora os usuários podem trabalhar no novo sistema e imediatamente ter acesso aos mesmos dados. Se posteriormente mais crescimento for necessário, será possível redistribuir os arquivos pelo novo grupo de nós que inclui os sistemas adicionais.

### Problemas de Redistribuição ao Incluir Sistemas em uma Rede

A redistribuição de um arquivo por um grupo de nós é um processo razoavelmente simples.

É possível utilizar o comando Alterar Arquivo Físico (CHGPF) para especificar um novo grupo de nós para um arquivo ou uma nova chave de particionamento para um arquivo. O comando CHGPF pode ser utilizado para que um arquivo local se torne um arquivo distribuído, para que um arquivo distribuído se torne um arquivo local ou para redistribuir um arquivo distribuído por um conjunto de nós diferentes ou com uma chave de particionamento diferente.

Você deve estar ciente de que o processo de redistribuição pode envolver a movimentação de quase todos os registros no arquivo. Para arquivos muito grandes, isso pode ser um processo longo, durante o qual os dados no arquivo estão indisponíveis. Você não deve fazer a redistribuição de arquivos com frequência ou sem o planejamento apropriado.

Para alterar um arquivo físico local para um arquivo distribuído, é necessário especificar os parâmetros grupo de nós (NODGRP) e chave de particionamento (PTNKEY) no comando CHGPF. A emissão desse comando altera o arquivo para ser distribuído pelos nós no grupo de nós, e quaisquer dados existentes também serão distribuídos utilizando a chave de particionamento especificada no parâmetro PTNKEY.

Para alterar um arquivo distribuído para um arquivo local, é necessário especificar NODGRP(\*NONE) no comando CHGPF. Isso exclui todas as partes remotas do arquivo e força todos os dados de volta para o sistema local.

Para alterar a chave de particionamento de um arquivo distribuído, especifique os campos desejados no parâmetro PTNKEY do comando CHGPF. Isso não afeta os sistemas em que o arquivo foi distribuído. Isso faz com que todos os dados sejam redistribuídos, porque o algoritmo de hash precisa ser aplicado em uma nova chave de particionamento.

Para especificar um novo conjunto de sistemas pelos quais o arquivo deve ser distribuído, especifique o nome de um grupo de nós no parâmetro grupo de nós (NODGRP) do comando CHGPF. Isso resulta na distribuição do arquivo por esse novo conjunto de sistemas. É possível especificar uma nova chave de particionamento no parâmetro PTNKEY. Se o parâmetro PTNKEY não for especificado ou se \*SAME for especificado, a chave de particionamento existente será utilizada.

O comando CHGPF manipula a criação de novas partes do arquivo se o grupo de nós tiver novos sistemas incluídos nele. O comando CHGPF manipula a exclusão de partes do arquivo se um sistema não estiver no novo grupo de nós. Observe que se você deseja excluir e recriar um grupo de nós e, em seguida, utilizar o comando CRTPF para redistribuir o arquivo, deverá especificar o nome do grupo de nós no parâmetro NODGRP do comando CHGPF, mesmo se o nome do grupo de nós for o mesmo que aquele utilizado quando o arquivo foi criado pela primeira vez. Isso indica que você deseja que o sistema examine o grupo de nós e redistribua o arquivo. Entretanto, se você especificou um grupo de nós no parâmetro NODGRP e o sistema o reconheceu como idêntico ao grupo de nós atualmente armazenado no arquivo, nenhuma redistribuição ocorrerá, a menos que também tenha especificado PTNKEY.

Para arquivos com restrições de referência, se deseja utilizar o comando CHGPF para que o arquivo-pai e o arquivo dependente sejam arquivos distribuídos, você deverá fazer as seguintes tarefas:

- Remova a restrição de referência. Se não remover a restrição, o primeiro arquivo distribuído encontrará um erro de restrição, porque o outro arquivo no relacionamento de restrição de referência ainda não é um arquivo distribuído.
- Utilize o comando CHGPF para que os dois arquivos se tornem arquivos distribuídos.
- Inclua novamente a restrição de referência.

#### **Conceitos relacionados**

“Customização da Distribuição de Dados com o DB2 Multisystem” na página 23

Como o sistema é responsável por fornecer os dados, você não precisa saber onde os registros residem na realidade. Entretanto, se deseja garantir que determinados registros sempre sejam armazenados em um sistema específico, poderá utilizar o comando Alterar Atributos do Grupo de Nós (CHGNODGRPA) para especificar onde esses registros residem.

---

## **Design de Consulta para Desempenho com o DB2 Multisystem**

Este tópico fornece algumas orientações para o design de consultas para que elas utilizem recursos de consulta com mais eficácia ao executar consultas que utilizam arquivos distribuídos.

Este tópico também discute como as consultas que utilizam arquivos distribuídos são implementadas. Essas informações podem ser utilizadas para ajustar as consultas de forma que sejam executadas com mais eficácia em um ambiente distribuído.

Os arquivos distribuídos podem ser consultados utilizando-se SQL, o comando Abrir Arquivo de Consulta (OPNQRYF) ou qualquer interface de consulta no sistema. As consultas podem ser consultas de arquivo simples ou consultas de junção. É possível utilizar uma combinação de arquivos distribuídos e locais em uma junção.

Este capítulo assume que você esteja familiarizado com a execução e a otimização de consultas em um ambiente não-distribuído. Se deseja obter informações adicionais sobre esses tópicos:

- Os usuários do SQL devem consultar Referência de SQL e Conceitos de Programação SQL.
- Os usuários não-SQL devem consultar Programação de Banco de Dados e o tópico Referência de CL.

Este capítulo também mostra como aprimorar o desempenho de consultas distribuídos explorando o paralelismo e minimizando a movimentação de dados.

#### **Conceitos relacionados**

“Escolher Chaves de Particionamento com o DB2 Multisystem” na página 22

Para que o sistema processe o arquivo particionado da forma mais eficiente, há algumas dicas que devem ser consideradas ao configurar ou utilizar uma chave de particionamento.

Conceitos de Programação SQL

Programação do Banco de Dados

#### **Referências relacionadas**

Referência SQL

Referência da CL

## **Visão Geral da Otimização com o DB2 Multisystem**

As consultas distribuídas são otimizadas no nível distribuído e no nível local.

- A otimização no nível distribuído se concentra na divisão da consulta em etapas mais eficientes e na determinação de quais nós processam essas etapas. O otimizador distribuído é característico das consultas distribuídas. O otimizador distribuído é discutido neste tópico.
- A otimização no nível (etapa) local é a mesma otimização que ocorre em um ambiente não-distribuído; é o otimizador com o qual você provavelmente está familiarizado. A otimização no nível local é discutida brevemente neste tópico.

Uma suposição básica da otimização distribuída é que o número de registros armazenados em cada nó de dados é aproximadamente igual e todos os sistemas da consulta distribuída têm configurações semelhantes. As decisões tomadas pelo otimizador distribuído são baseadas no sistema e nas estatísticas de dados do sistema do nó coordenador.

Se uma consulta distribuída requerer mais de uma etapa, um arquivo de resultado temporário será utilizado. Um *arquivo de resultado temporário* é um arquivo temporário criado pelo sistema (armazenado na biblioteca QRECOVERY) utilizado para conter os resultados de uma etapa de consulta específica. O conteúdo do arquivo de resultado temporário é utilizado como entrada para a próxima etapa de consulta.

## **Implementação e Otimização de uma Consulta de Arquivo Único com o DB2 Multisystem**

Para fazer uma consulta de arquivo único, o sistema em que a consulta foi especificada, o nó coordenador, determina os nós do arquivo para o qual a consulta será enviada. Esses nós executam a consulta e retornam os registros consultados ao nó coordenador.



Todos os exemplos neste capítulo utilizam os seguintes arquivos distribuídos: DEPARTMENT e EMPLOYEE. O grupo de nós desses arquivos consiste em SYSA, SYSB e SYSC. Os dados são particionados pelo número do departamento.

A seguinte instrução SQL cria o arquivo distribuído DEPARTMENT.

```
CREATE TABLE DEPARTMENT
  (DEPTNO CHAR(3) NOT NULL,
   DEPTNAME VARCHAR(20) NOT NULL,
   MGRNO CHAR(6),
   ADMRDEPT CHAR(3) NOT NULL)
  IN NODGRP1 PARTITIONING KEY(DEPTNO)
```

Tabela 2. Tabela DEPARTMENT

Nó	Número do Registro	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
SYSA	1	A00	Support services	000010	A00
SYSB	2	A01	Planning	000010	A00
SYSC	3	B00	Accounting	000050	B00
SYSA	4	B01	Programming	000050	B00

A seguinte instrução SQL cria o arquivo distribuído EMPLOYEE.

```
CREATE TABLE EMPLOYEE
  (EMPNO CHAR(6) NOT NULL,
   FIRSTNME VARCHAR(12) NOT NULL,
   LASTNAME VARCHAR(15) NOT NULL,
   WORKDEPT CHAR(3) NOT NULL,
   JOB CHAR(8),
   SALARY DECIMAL(9,2))
  IN NODGRP1 PARTITIONING KEY(WORKDEPT)
```

Tabela 3. Tabela EMPLOYEE

Nó	Número do Registro	EMPNO	FIRSTNME	LASTNAME	WORK DEPT	JOB	SALARY
SYSA	1	000010	Christine	Haas	A00	Manager	41250
SYSA	2	000020	Sally	Kwan	A00	Clerk	25000
SYSB	3	000030	John	Geyer	A01	Planner	35150
SYSB	4	000040	Irving	Stern	A01	Clerk	32320
SYSC	5	000050	Michael	Thompson	B00	Manager	38440
SYSC	6	000060	Eileen	Henderson	B00	Accountant	33790
SYSA	7	000070	Jennifer	Lutz	B01	Programmer	42325
SYSA	8	000080	David	White	B01	Programmer	36450

A seguir é apresentada uma consulta que utiliza o arquivo distribuído definido acima, EMPLOYEE, com o índice EMPIDX criado sobre o campo SALARY. A consulta é inserida em SYSA.

Instrução SQL:

```
SELECT * FROM EMPLOYEE WHERE SALARY > 40000
```

Comando OPNQRYF:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('SALARY > 40000')
```

Nesse caso, SYSA envia a consulta acima para todos os nós de EMPLOYEE, incluindo SYSA. Cada nó executa a consulta e retorna os registros para SYSA. Como existe um índice distribuído no campo SALARY do arquivo EMPLOYEE, a otimização feita em cada nó decide se o índice deve ser utilizado.

No exemplo a seguir, a consulta é especificada em SYSA, mas a consulta é enviada para um subconjunto dos nós em que o arquivo EMPLOYEE existe. Nesse caso, a consulta é executada localmente apenas em SYSA.

Instrução SQL:

```
SELECT * FROM EMPLOYEE WHERE WORKDEPT = 'A00'
```

Comando OPNQRYF:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('WORKDEPT = 'A00')
```

O otimizador de consulta distribuída determina se há uma seleção de registro separável, WORKDEPT = 'A00', envolvendo a chave de particionamento, WORKDEPT, para essa consulta. O otimizador faz hash do valor 'A00' e, com base no valor do hash, localiza o nó em que todos os registros que satisfazem essa condição estão localizados. Nesse caso, todos os registros que satisfazem essa condição estão em SYSA, portanto a consulta é enviada apenas para esse nó. Como a consulta foi originada em SYSA, a consulta é executada localmente em SYSA.

As seguintes condições limitam o número de nós nos quais uma consulta é executada:

- Todos os campos da chave de particionamento devem ser uma seleção de registros que pode ser isolada
- Todos os predicados devem utilizar o operador igual (=)
- Todos os campos da chave de particionamento devem ser comparados a uma literal

**Nota:** Por motivos de desempenho, é necessário especificar os predicados de seleção de registros correspondentes à chave de particionamento para dirigir a consulta para um nó específico. A seleção de registro com as funções escalares NODENAME, PARTITION e NODENUMBER também podem dirigir a consulta para nós específicos.

## Implementação e Otimização da Ordenação de Registros com o DB2 Multisystem

Quando a ordenação for especificada em uma consulta, os critérios de ordenação são enviados juntamente com a consulta, de forma que cada nó possa executar a ordenação em paralelo. Dependendo do tipo de consulta especificado, será executada uma classificação ou mesclagem final no nó coordenador.

O melhor é uma mesclagem dos registros ordenados recebidos de cada nó. Uma mesclagem ocorre à medida que os registros são recebidos pelo nó coordenador. A principal vantagem de desempenho que uma mesclagem tem sobre uma classificação é que os registros podem ser retornados sem precisar classificar todos os registros de cada nó.

Uma classificação dos registros ordenados recebida de cada nó faz com que todos os registros de cada nó sejam lidos, classificados e gravados em um arquivo de resultado temporário antes que qualquer registro seja retornado.

Uma mesclagem pode ocorrer se a ordenação for especificada e nenhuma UNION e nenhum agrupamento final forem requeridos. Caso contrário, para ordenar consultas, uma classificação final é executada no nó coordenador.

O parâmetro Permitir Dados de Cópia (ALWCPYDTA) afeta como cada nó da consulta distribuída processa os critérios de ordenação. O parâmetro ALWCPYDTA é especificado nos comandos da CL Abrir Arquivo de Consulta (OPNQRYF) e Iniciar SQL (STRSQL) e também nos comandos do pré-compilador Criar SQLxxx (CRTSQLxxx):

- ALWCPYDTA(\*OPTIMIZE) permite que cada nó escolha a utilização de uma classificação ou um índice para implementar os critérios de ordenação. Essa opção é a melhor.

- Para o comando OPNQRYF e a API de consulta (QQQRY), ALWCPYDTA(\*YES) ou ALWCPYDTA(\*NO) reforça que cada nó utilize um índice que corresponda exatamente aos campos de ordenação especificados. Isso é mais restritivo do que o modo em que o otimizador processa a ordenação para arquivos locais.

## **Implementação e Otimização das Cláusulas UNION e DISTINCT com o DB2 Multisystem**

Se uma instrução SELECT unida se referir a um arquivo distribuído, a instrução será processada como em uma consulta distribuída.

O processamento da instrução pode ocorrer em paralelo. Entretanto, os registros de cada SELECT unido serão trazidos de volta ao nó coordenador para executar a operação de união. A esse respeito, os operadores de união são processados de forma serial.

Se uma cláusula ORDER BY for especificada com uma consulta de união, todos os registros de cada nó serão recebidos no nó coordenador e serão classificados antes que qualquer registro seja retornado.

Quando a cláusula DISTINCT for especificada para uma consulta distribuída, a inclusão de uma cláusula ORDER BY retorna registros mais rapidamente do que se nenhuma cláusula ORDER BY fosse especificada. DISTINCT com ORDER BY permite que cada nó ordene os registros em paralelo. Uma mesclagem final no nó coordenador lê os registros ordenados de cada nó, mescla os registros na ordem apropriada e elimina registros duplicados sem precisar fazer uma classificação final.

Quando a cláusula DISTINCT for especificada sem uma cláusula ORDER BY, todos os registros de cada nó serão enviados ao nó coordenador onde uma classificação é executada. Os registros duplicados são eliminados à medida que os registros classificados forem retornados.

## **Processamento dos Parâmetros DSTDTA e ALWCPYDTA com o DB2 Multisystem**

O parâmetro Permitir Dados de Cópia (ALWCPYDTA) pode alterar o valor especificado para o parâmetro Distribuir Dados (DSTDTA) do comando Substituir Arquivo do Banco de Dados (OVRDBF).

Se você especificou a utilização de dados ativos (DSTDTA(\*CURRENT)) no comando de substituição, um dos seguintes itens será verdadeiro:

- Uma cópia temporária foi requerida e ALWCPYDTA(\*YES) foi especificado
- Uma cópia temporária foi escolhida para um melhor desempenho e ALWCPYDTA(\*OPTIMIZE) foi especificado

e, em seguida, DSTDTA será alterado para DSTDTA(\*BUFFERED).

Se DSTDTA(\*BUFFERED) não for aceitável e a consulta não requerer uma cópia temporária, será necessário especificar ALWCPYDTA(\*YES) para manter DSTDTA(\*CURRENT) em efeito.

## **Implementação e Otimização de Operações de Junção com o DB2 Multisystem**

Além das considerações sobre desempenho para as consultas de junção não-distribuídas, existem outras considerações sobre desempenho para consultas que envolvam arquivos distribuídos.

As junções podem ser executadas apenas quando os dados forem compatíveis com a partição. O otimizador de consulta distribuída gera um plano que torna compatível a partição de dados, o que pode envolver a movimentação de dados entre nós.

Os dados são *compatíveis com a partição* quando os dados nas chaves de particionamento dos dois arquivos utilizaram o mesmo grupo de nós e tiverem hash para o mesmo nó. Por exemplo, o mesmo valor numérico armazenado em um campo inteiro grande ou um campo inteiro pequeno tem hash para o mesmo valor.

Os tipos de dados a seguir são compatíveis com a partição:

- Inteiro grande (4 bytes), inteiro pequeno (2 bytes), decimal compactado e numérico zonado.
- Caractere SBCS de comprimento fixo e variável, e DBCS-aberto, também ou apenas.
- Gráfico de comprimento fixo e variável.

Os tipos de dados data, hora, time stamp e numérico de ponto flutuante não são compatíveis com a partição porque não podem ser chaves de particionamento.

As junções que envolvem arquivos distribuídos são classificadas em quatro tipos: colocadas, direcionadas, reparticionadas e de difusão. As seções a seguir definem os tipos de junções e fornecem exemplos dos diferentes tipos de junção.

#### Conceitos relacionados

“Otimização de Junções com o DB2 Multisystem” na página 50

O otimizador de consulta distribuída gera um plano para unir arquivos distribuídos.

### Junção Colocada com o DB2 Multisystem

Em uma junção colocada, os registros correspondentes de arquivos que estão sendo unidos existem no mesmo nó.

Os valores da chave de particionamento dos arquivos que estão sendo unidos são compatíveis com a partição. Nenhum dado precisa ser movido para outro nó para executar a junção. Esse método é válido apenas para consultas em que todos os campos das chaves de particionamento são campos de junção e a operação de junção é o operador = (igual). Além disso, o *n*-ésimo campo (em que *n*=1 até o número de campos na chave de particionamento) da chave de particionamento do primeiro arquivo deve ser unido ao *n*-ésimo campo da chave de particionamento do segundo arquivo, e os tipos de dados dos *n*-ésimos campos devem ser compatíveis com a partição. Observe que todos os campos da chave de particionamento devem ser envolvidos na junção. Predicados de junção adicionais que não contenham campos da chave de particionamento não afetam sua capacidade em fazer uma junção colocada.

No exemplo a seguir, como o predicado da junção envolve os campos da chave de particionamento de ambos os arquivos e os campos são compatíveis com a partição, uma junção colocada pode ser executada. Isso implica que os valores correspondentes de DEPTNO e WORKDEPT estão localizados no mesmo nó.

Instrução SQL:

```
SELECT DEPTNAME, FIRSTNAME, LASTNAME
FROM DEPARTMENT, EMPLOYEE
WHERE DEPTNO=WORKDEPT
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
          FORMAT(JOINFMT)
          JFLD((DEPTNO WORKDEPT *EQ))
```

Registros retornados por essa consulta:

Tabela 4. Exibição dos Resultados da Consulta

DEPTNAME	FIRSTNAME	LASTNAME
Support services	Christine	Haas
Support services	Sally	Kwan

Tabela 4. Exibição dos Resultados da Consulta (continuação)

DEPTNAME	FIRSTNME	LASTNAME
Planning	John	Geyer
Planning	Irving	Stern
Accounting	Michael	Thompson
Accounting	Eileen	Henderson
Programming	Jennifer	Lutz
Programming	David	White

No exemplo a seguir, o predicado de junção adicional MGRNO=EMPNO não afeta a capacidade de executar uma junção colocada, porque as chaves de particionamento ainda estão envolvidas em um predicado de junção.

```
SQL:      SELECT DEPTNAME, FIRSTNME, LASTNAME
           FROM DEPARTMENT, EMPLOYEE
           WHERE DEPTNO=WORKDEPT AND MGRNO=EMPNO

OPNQRYF:  OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
           FORMAT(JOINFMT)
           JFLD((DEPTNO WORKDEPT *EQ) (MGRNO EMPNO *EQ))
```

Registros retornados por essa consulta:

Tabela 5. Exibição dos Resultados da Consulta

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Accounting	Michael	Thompson

## Junção Direcionada com o DB2 Multisystem

Na junção direcionada, as chaves de particionamento de pelo menos um dos arquivos são utilizadas como os campos de junção.

Os campos de junção não correspondem às chaves de particionamento dos demais arquivos. Os registros de um arquivo são direcionados ou enviados aos nós do segundo arquivo com base no hash dos valores dos campos de junção utilizando o mapa de partição e o grupo de nós do segundo arquivo. Tão logo os registros tenham sido movidos para os nós do segundo arquivo por meio de um arquivo distribuído temporário, uma junção colocada será utilizada para unir os dados. Esse método é válido apenas para consultas de junção idêntica em que todos os campos da chave de particionamento são campos de junção para pelo menos um dos arquivos.

Na consulta a seguir, o campo de junção (WORKDEPT) é a chave de particionamento do arquivo EMPLOYEE; entretanto, o campo de junção (ADMRDEPT) não é a chave de particionamento para DEPARTMENT. Se a junção foi tentada sem mover os dados, os registros resultantes estariam ausentes porque o registro 2 de DEPARTMENT seria unido aos registros 1 e 2 de EMPLOYEE e esses três registros são armazenados em nós diferentes.

Instrução SQL:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
       FROM DEPARTMENT, EMPLOYEE
       WHERE ADMRDEPT = WORKDEPT AND JOB = 'Manager'
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
         FORMAT(JOINFMT)
         QRYSLT('JOB *EQ 'Manager')
         JFLD((ADMRDEPT WORKDEPT *EQ))
```

Os registros de DEPARTMENT necessários para executar a consulta estão prontos e é feito hash dos dados em ADMRDEPT utilizando o mapa de particionamento e o grupo de nós de EMPLOYEE. É criado um arquivo temporário semelhante ao seguinte:

Tabela 6. Uma Tabela Temporária

Nó Antigo	Novo Nó	DEPTNAME	ADMRDEPT (Nova chave de particionamento)
SYSA	SYSA	Support services	A00
SYSB	SYSA	Planning	A00
SYSC	SYSC	Accounting	B00
SYSA	SYSC	Programming	B00

Essa tabela temporária é unida a EMPLOYEE. A junção funciona porque ADMRDEPT é compatível com a partição de WORKDEPT.

Tabela 7. Tabela Unida EMPLOYEE

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

## Junção Reparticionada com o DB2 Multisystem

Em uma junção reparticionada, as chaves de particionamento dos arquivos não são utilizadas como os campos de junção.

Os registros dos dois arquivos devem ser movidos fazendo-se hash dos valores do campo de junção de cada um dos arquivos. Como nenhum dos campos da chave de particionamento dos arquivos está incluído nos critérios de junção, os arquivos devem ser reparticionados fazendo-se hash em uma nova chave de particionamento que inclua um ou mais dos campos de junção. Esse método é válido apenas para consultas de junção idêntica.

Instrução SQL:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
FROM DEPARTMENT, EMPLOYEE
WHERE MGRNO = EMPNO
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
FORMAT(JOINFMT)
JFLD((MGRNO EMPNO *EQ))
```

Nesse exemplo, os dados devem ser redistribuídos porque nem MGRNO nem EMPNO é uma chave de particionamento.

Os dados de DEPARTMENT são redistribuídos:

Tabela 8. Dados de DEPARTMENT Redistribuídos

Nó Antigo	Novo Nó	DEPTNAME	MGRNO (Nova chave de particionamento)
SYSA	SYSB	Support services	000010
SYSB	SYSB	Planning	000010
SYSC	SYSC	Accounting	000050
SYSA	SYSC	Programming	000050

Os dados de EMPLOYEE são redistribuídos:

Tabela 9. Dados de EMPLOYEE Redistribuídos

Nó Antigo	Novo Nó	FIRSTNME	LASTNAME	EMPNO (Nova chave de particionamento)
SYSA	SYSB	Christine	Haas	000010
SYSA	SYSC	Sally	Kwan	000020
SYSB	SYSA	John	Geyer	000030
SYSB	SYSB	Irving	Stern	000040
SYSC	SYSC	Michael	Thompson	000050
SYSC	SYSA	Eileen	Henderson	000060
SYSA	SYSB	Jennifer	Lutz	000070
SYSA	SYSC	David	White	000080

Registros retornados por essa consulta:

Tabela 10. Exibição dos Resultados da Consulta

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

## Junção de Difusão com o DB2 Multisystem

Em uma junção de difusão, todos os registros selecionados de um arquivo são enviados ou difundidos para todos os nós do outro arquivo antes que a junção seja executada.

Esse é o método de junção utilizado para todas as consultas de junção não-idênticas. Esse método também é utilizado quando os critérios de junção utilizam campos que têm um tipo de dados igual a data, hora, time stamp ou numérico de ponto flutuante.

No exemplo a seguir, o otimizador de consultas distribuídas decide difundir EMPLOYEE, porque a aplicação da seleção JOB = 'Manager' resulta na difusão de um conjunto menor de registros. O arquivo temporário em cada nó no grupo de nós contém todos os registros selecionados. (Os registros são duplicados em cada nó.)

Instrução SQL:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
FROM DEPARTMENT, EMPLOYEE
WHERE DEPTNO <> WORKDEPT AND JOB = 'Manager'
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
FORMAT(JOINFMT)
QRYSLT('JOB *EQ 'Manager')
JFLD((DEPTNO WORKDEPT *NE))
```

O otimizador de consultas distribuídas envia os dois registros selecionados a seguir para cada nó:

Tabela 11. Registros que o otimizador de consultas distribuídas envia para cada nó

Nó Antigo	Novo Nó	FIRSTNME	LASTNAME	WORKDEPT
SYSA	SYSA, SYSB, SYSC	Christine	Haas	A00

Tabela 11. Registros que o otimizador de consultas distribuídas envia para cada nó (continuação)

Nó Antigo	Novo Nó	FIRSTNME	LASTNAME	WORKDEPT
SYSC	SYSA, SYSB, SYSC	Michael	Thompson	B00

Registros retornados por essa consulta:

Tabela 12. Exibição dos Resultados da Consulta

DEPTNAME	FIRSTNME	LASTNAME
Support services	Michael	Thompson
Planning	Christine	Haas
Planning	Michael	Thompson
Accounting	Christine	Haas
Programming	Christine	Haas
Programming	Michael	Thompson

## Otimização de Junções com o DB2 Multisystem

O otimizador de consulta distribuída gera um plano para unir arquivos distribuídos.

O otimizador de consulta distribuída examina os tamanhos de arquivos, o número esperado de registros selecionados para cada arquivo e o tipo de junções distribuídas possíveis; em seguida, o otimizador divide a consulta em várias etapas. Cada etapa cria um arquivo de resultado intermediário utilizado como entrada para a próxima etapa.

Durante a otimização, um custo é calculado para cada etapa de junção com base no tipo de junção distribuída. O custo reflete, em parte, a quantidade de movimentação de dados requerida para essa etapa de junção. O custo é utilizado para determinar o plano distribuído final.

O máximo de processamento possível é concluído durante cada etapa; por exemplo, seleção de registros isolada a uma determinada etapa é executada durante essa etapa, e tantos arquivos quanto for possível serão unidos para cada etapa. Cada etapa de junção pode envolver mais de um tipo de junção distribuída. Uma junção colocada e uma junção direcionada podem ser combinadas em uma junção colocada, direcionando-se primeiramente o arquivo necessário. Uma junção direcionada e uma junção reparticionada podem ser combinadas direcionando-se primeiramente todos os arquivos e, em seguida, executando a junção. Observe que as junções direcionadas e reparticionadas são na realidade apenas uma junção colocada, com um ou mais arquivos direcionados antes da ocorrência da junção.

Ao unir arquivos distribuídos com arquivos locais, o otimizador de consulta distribuída calcula um custo, semelhante ao custo calculado ao unir arquivos distribuídos. Com base nesse custo, o otimizador de consulta distribuída pode optar por executar uma das seguintes ações:

- Difundir todos os arquivos locais para os nós de dados do arquivo distribuído e executar uma junção colocada.
- Difundir todos os arquivos locais e distribuídos para os nós de dados do maior arquivo distribuído e executar uma junção colocada.
- Direcionar os arquivos distribuídos de volta para o nó coordenador e executar a junção nele.

### Conceitos relacionados

“Implementação e Otimização de Operações de Junção com o DB2 Multisystem” na página 45

Além das considerações sobre desempenho para as consultas de junção não-distribuídas, existem outras considerações sobre desempenho para consultas que envolvam arquivos distribuídos.

## Chaves de Particionamento sobre Campos de Junção com o DB2 Multisystem:



Nas seções anteriores sobre os tipos de junções, é possível notar que a movimentação de dados é requerida para todos os tipos de junção distribuída, exceto uma junção colocada.

Para eliminar a necessidade de movimentação de dados e maximizar o desempenho, todas as consultas devem ser gravadas de forma que seja possível uma junção colocada. Em outras palavras, as chaves de particionamento dos arquivos distribuídos devem corresponder aos campos utilizados para unir os arquivos. Para consultas que são executadas com frequência, é mais importante fazer com que as chaves de particionamento correspondam aos campos de junção do que corresponder aos critérios de ordenação ou agrupamento.

## Implementação e Otimização de Agrupamento com o DB2 Multisystem

O método de implementação para agrupamento em consultas que utilizam arquivos distribuídos depende do fato da chave de particionamento estar incluída nos critérios de agrupamento.

O agrupamento é implementado utilizando o agrupamento de uma etapa ou o agrupamento de duas etapas.

### Referências relacionadas

“Agrupamento e Junções com o DB2 Multisystem” na página 52

Se a consulta contiver uma junção, a chave de particionamento utilizada para determinar o tipo de agrupamento que pode ser implementado é baseada em qualquer reparticionamento dos dados requeridos para implementar a junção.

### Agrupamento em Uma Etapa com o DB2 Multisystem

Se todos os campos da chave de particionamento forem campos de GROUP BY, o agrupamento poderá ser executado utilizando-se o agrupamento em uma etapa, porque todos os dados do grupo estão no mesmo nó.

O código a seguir é um exemplo do agrupamento em uma etapa.

Instrução SQL:

```
SELECT WORKDEPT, AVG(SALARY)
FROM EMPLOYEE
GROUP BY WORKDEPT
```

Comando OPNQRYF:

```
OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT)
          GRPFLD(WORKDEPT)
          MAPFLD((AVGSAL '%AVG(SALARY)'))
```

Como WORKDEPT é a chave de particionamento e o campo de agrupamento, todos os valores semelhantes de WORKDEPT estão nos mesmos nós; por exemplo, todos os valores A00 estão em SYSA, todos os valores A01 estão em SYSB, todos os valores B00 estão em SYSC e todos os valores B01 estão em SYSA. O agrupamento é executado em paralelo nos três nós.

Para implementar o agrupamento em uma etapa, todos os campos da chave de particionamento deve ser campos de agrupamento. Os campos adicionais que não são chave de particionamento também podem ser campos de agrupamento.

### Agrupamento em Duas Etapas com o DB2 Multisystem

Se a chave de particionamento não estiver incluída nos campos de agrupamento, o agrupamento deverá ser feito utilizando o agrupamento em duas etapas, porque os valores semelhantes de um campo não estão localizados no mesmo nó.

O código a seguir é um exemplo do agrupamento em duas etapas.

Instrução SQL:

```
SELECT JOB, AVG(SALARY)
FROM EMPLOYEE
GROUP BY JOB
```

Comando OPNQRYF:

```
OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT2)
          GRPFLD(JOB)
          MAPFLD((AVGSAL '%AVG(SALARY)'))
```

Nesse exemplo, observe que para o grupo onde JOB é Clerk, o valor Clerk está em dois nós diferentes no arquivo distribuído EMPLOYEE. O agrupamento é implementado primeiramente pela execução do agrupamento em paralelo nos três nós. Isso resulta em um agrupamento inicial que é colocado em um arquivo temporário no nó coordenador. A consulta é modificada e o agrupamento é executado novamente no nó coordenador para obter o conjunto final de resultados de agrupamento.

O agrupamento de arquivo inteiro (sem agrupamento por campos) é sempre implementado utilizando duas etapas.

Se a consulta contiver uma cláusula HAVING ou o parâmetro Agrupar Expressão de Seleção (GRPSLT) no comando OPNQRYF, todos os grupos da primeira etapa de agrupamento serão retornados ao nó coordenador. Em seguida, a cláusula HAVING ou o parâmetro GRPSLT é processado como parte da segunda etapa de agrupamento.

Se a consulta contiver uma função de coluna DISTINCT (agregada) e o agrupamento em duas etapas for requerido, nenhum agrupamento será feito na primeira etapa. Em vez disso, todos os registros são retornados ao nó coordenador e todo o agrupamento é executado no nó coordenador como parte da segunda etapa.

## Agrupamento e Junções com o DB2 Multisystem

Se a consulta contiver uma junção, a chave de particionamento utilizada para determinar o tipo de agrupamento que pode ser implementado é baseada em qualquer reparticionamento dos dados requeridos para implementar a junção.

No exemplo a seguir, uma junção reparticionada é executada antes do agrupamento, o que resulta na nova chave de particionamento MGRNO. Como MGRNO agora é a nova chave de particionamento, o agrupamento pode ser executado utilizando o agrupamento de uma etapa.

Instrução SQL:

```
SELECT MGRNO, COUNT(*)
FROM DEPARTMENT, EMPLOYEE
WHERE MGRNO = EMPNO
GROUP BY MGRNO
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT2)
          JFLD((MGRNO EMPNO *EQ))
          GRPFLD(MGRNO)
          MAPFLD((CNTMGR '%COUNT'))
```

No exemplo a seguir, uma junção reparticionada é executada antes do agrupamento, o que resulta na nova chave de particionamento EMPNO. Como EMPNO agora é a chave de particionamento no lugar de WORKDEPT, o agrupamento não pode ser executado utilizando o agrupamento de uma etapa.

Instrução SQL:

```
SELECT WORKDEPT, COUNT(*)
FROM DEPARTMENT, EMPLOYEE
WHERE MGRNO = EMPNO
GROUP BY WORKDEPT
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT3)
          JFLD((MGRNO EMPNO *EQ))
          GRPFLD(WORKDEPT)
          MAPFLD((CNTDEPT '%COUNT'))
```

#### **Conceitos relacionados**

“Implementação e Otimização de Agrupamento com o DB2 Multisystem” na página 51  
O método de implementação para agrupamento em consultas que utilizam arquivos distribuídos depende do fato da chave de particionamento estar incluída nos critérios de agrupamento.

## **Suporte a Subconsulta com o DB2 Multisystem**

Os arquivos distribuídos podem ser especificados em subconsultas.

Uma subconsulta pode incluir suas próprias condições de pesquisa que podem, por sua vez, incluir subconsultas. Portanto, uma instrução SQL pode conter uma hierarquia de subconsultas. Os elementos da hierarquia que contêm subconsultas estão em um nível mais alto do que as subconsultas que eles contêm. Consulte Conceitos de Programação SQL para obter informações adicionais sobre a utilização de subconsultas.

#### **Conceitos relacionados**

Conceitos de Programação SQL

## **Planos de Acesso com o DB2 Multisystem**

Os planos de acesso armazenados para consultas que se referem a arquivos distribuídos são diferentes dos planos de acesso armazenados para arquivos locais.

O plano de acesso armazenado para uma consulta distribuída é o plano de acesso distribuído que contém informações sobre como a consulta é dividida em várias etapas e nos nós em que cada etapa da consulta é executada. As informações sobre como a etapa é implementada localmente em cada nó não são salvas no plano de acesso; essas informações são construídas em tempo de execução.

## **Caminhos de Dados Abertos Reutilizáveis com o DB2 Multisystem**

Os ODPs (Caminhos de Dados Abertos) reutilizáveis têm considerações especiais para consultas distribuídas. Como a maioria dos demais aspectos das consultas distribuídas, os ODPs têm dois níveis: distribuído e local.

O ODP distribuído é o ODP de coordenação. Um ODP distribuído associa a consulta ao usuário e controla os ODPs locais. Os ODPs locais estão posicionados em cada sistema envolvido na consulta e obtêm pedidos pelo ODP distribuído.

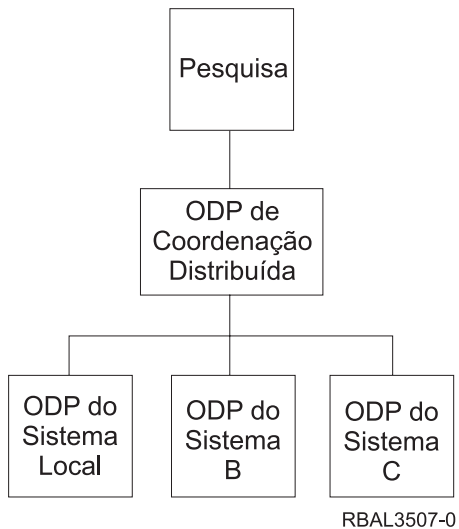


Figura 7. Exemplo de um ODP

Por exemplo, se for feito um pedido para executar SQL FETCH, o pedido será feito para o ODP distribuído. Em seguida, o sistema obtém esse pedido e executa a recuperação de registro apropriada nos ODPs locais.

Com as consultas distribuídas, é possível reutilizar os ODP distribuídos, ainda que um ou mais dos ODPs locais não sejam reutilizáveis; no entanto, se o ODP de consulta distribuída não for reutilizável, os ODPs locais sempre serão não reutilizáveis. Isso é permitido de forma que:

- Cada sistema local possa decidir a melhor forma de abrir seu ODP local (reutilizável versus não reutilizável).
- Independentemente dos métodos ODP locais, o ODP distribuído pode ser aberto como reutilizável o quanto for possível para manter os recursos ativos, como as comunicações.

O sistema tenta tornar o ODP distribuído reutilizável sempre que possível, mesmo quando um ODP local subjacente não for reutilizável. Se isso ocorrer, o sistema manipula a atualização do ODP da seguinte forma:

- Efetua ciclo por cada ODP local
- Executa uma atualização de um ODP local reutilizável
- Executa um fechamento “brusco” e a reabertura de um ODP não reutilizável
- Conclui todas as atualizações restantes do próprio ODP distribuído que é necessário

O ODP distribuído é reutilizável com mais frequência do que os ODPs locais, porque o ODP distribuído não é afetado por algumas das coisas que tornam os ODPs locais não reutilizáveis, como uma variável de host em uma cláusula LIKE ou o otimizador escolher não reutilizável, de forma que uma operação de criação de índice a partir de índice possa ser executada. Os casos que tornariam não reutilizáveis os ODPs distribuídos são um subconjunto daqueles que afetam os ODPs locais. Esse subconjunto inclui os seguintes itens:

- A utilização de arquivos temporários com finalidade diferente de uma classificação. Elas são denominadas consultas distribuídas de várias etapas, e a mensagem de depuração do otimizador CPI4343 é sinalizada para esses casos.
- Alterações na lista de bibliotecas, que podem afetar os arquivos consultados.
- Alterações em OVRDBF, que afetam os arquivos consultados.
- Alterações de valores para os registros especiais USER ou CURRENT TIMEZONE.
- Alterações de CCSID do job.

- A emissão do comando Recuperar Recursos (RCLRSC).

A capacidade de reutilização do ODP local é afetada pelas mesmas condições já existentes para os ODPs de consulta não-distribuída. Portanto, as mesmas considerações são aplicadas a eles e aos ODPs de consulta local.

## Escritor de Resultado Temporário com o DB2 Multisystem

Os escritores de resultado temporário são jobs iniciados pelo sistema que estão sempre ativos.

No sistema, os escritores de resultado temporário são jobs pareados denominados QQQTEMP1 e QQQTEMP2. Os escritores de resultado temporário manipulam pedidos dos jobs que estão executando consultas. Esses pedidos consistem em uma consulta (da etapa de consulta) a ser executada e no nome de um arquivo temporário do sistema a ser preenchido a partir dos resultados da consulta. O escritor de resultado temporário processa o pedido e preenche o arquivo temporário. Em seguida, o arquivo temporário intermediário é utilizado pelo job solicitante para concluir a consulta original.

O exemplo a seguir mostra uma consulta que requer um escritor de resultado temporário e as etapas necessárias para processar a consulta.

Instrução SQL:

```
SELECT COUNT(*)
      FROM DEPARTMENT a, EMPLOYEE b
      WHERE a.ADMRDEPT = b.WORKDEPT
            AND b.JOB = 'Manager'
```

Comando OPNQRYF:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
          FORMAT(FMTFILE)
          MAPFLD((CNTFLD '%COUNT'))
          JFLD((1/ADMRDEPT 2/WORKDEPT))
          QRYSLT('2/JOB = 'Manager')
```

WORKDEPT é a chave de particionamento para EMPLOYEE, mas ADMRDEPT não é a chave de particionamento para DEPARTMENT. Como a consulta deve ser processada em duas etapas, o otimizador divide a consulta nas seguintes etapas:

```
INSERT INTO SYS_TEMP_FILE
SELECT a.DEPTNAME, a.ADMRDEPT
FROM DEPARTMENT a
```

e

```
SELECT COUNT(*) FROM SYS_TEMP_FILE x, EMPLOYEE b
WHERE x.ADMRDEPT = b.WORKDEPT AND b.JOB = 'Manager'
```

Se um escritor de resultado temporário tiver permissão para o job (controlado pelas opções de Alterar Atributos da Consulta, CHGQRYA), o otimizador:

1. Cria o arquivo temporário (SYS\_TEMP\_FILE) na biblioteca QRECOVERY.
2. Envia o pedido que preenche SYS\_TEMP\_FILE no escritor de resultado temporário.
3. Continua a concluir a abertura da consulta final (enquanto o escritor de resultado temporário está preenchendo o arquivo temporário).
4. Depois que a consulta final for aberta, aguarda até que o escritor de resultado temporário tenha concluído o preenchimento do arquivo temporário antes de retornar o controle para o responsável pela chamada.

**Conceitos relacionados**

“Alterações no Comando Alterar Atributos da Consulta (CHGQRYA) com o DB2 Multisystem” na página 59

O comando CHGQRYA tem dois parâmetros aplicáveis a consultas distribuídas.

### **Job Escritor de Resultado Temporário: Vantagens com o DB2 Multisystem**

A vantagem da utilização de um job escritor de resultado temporário ao processar um pedido é que o escritor de resultado temporário pode processar seu pedido ao mesmo tempo (em paralelo) em que o job principal está processando outra etapa da consulta.

As vantagens de desempenho da utilização de um escritor de resultado temporário são as seguintes:

- O escritor de resultado temporário pode utilizar o suporte paralelo a SMP (Multiprocessamento Simétrico) na conclusão de sua etapa de consulta, enquanto o job principal pode continuar com etapas mais complexas que não contribuem elas próprias tão facilmente para o processamento paralelo (como a otimização de consulta, análise de índice e assim por diante).
- Como o processamento do arquivo distribuído requer interação de comunicações, uma quantidade considerável de tempo normalmente gasta no aguardo do envio e recebimento pode ser descarregada para o escritor de resultado temporário, o que deixa o job principal fazer outros trabalhos.

### **Job Escritor de Resultado Temporário: Desvantagens com o DB2 Multisystem**

O escritor de resultado temporário também tem desvantagens que devem ser consideradas ao determinar sua utilidade para consultas.

As desvantagens são:

- O escritor de resultado temporário é um job separado. Conseqüentemente, ele pode encontrar conflitos com o job principal, como os seguintes:
  - O job principal pode ter um arquivo bloqueado para ele mesmo. Nesse caso, o escritor de resultado temporário não pode acessar os arquivos e não pode concluir sua etapa de consulta.
  - O job principal pode ter criado o arquivo distribuído sob controle de confirmação e ainda não confirmou a criação. Nesse caso, o escritor de resultado temporário não pode acessar o arquivo.
- O escritor de resultado temporário pode encontrar uma situação que ele não pode manipular da mesma forma que o job principal. Por exemplo, se uma mensagem de consulta for sinalizada, o escritor de resultado temporário poderá cancelá-la, enquanto o job principal poderá optar por ignorar a mensagem e continuar.
- Os escritores de resultado temporário são compartilhados por todos os jobs no sistema. Se diversos jobs tiverem pedidos para os escritores de resultado temporário, os pedidos serão enfileirados enquanto os escritores tentam processá-los.

**Nota:** O sistema é enviado com três pares de jobs escritor de resultado temporário.

- A tentativa de analisar uma consulta (por meio de mensagens de depuração, por exemplo) pode ser complicada se um escritor de resultado temporário estiver envolvido (porque uma etapa da consulta é executada em um job separado).

**Nota:** O sistema não permite que o escritor de resultado temporário seja utilizado para consultas em execução sob o controle de confirmação \*CS ou \*ALL. Isso ocorre porque o job principal pode ter registros bloqueados no arquivo, o que pode fazer com que o escritor de resultado temporário deixe de fora esses registros e não possa ser concluído.

### **Controle do Escritor de Resultado Temporário com o DB2 Multisystem**

Por padrão, as consultas não utilizam o escritor de resultado temporário. Seu uso, no entanto, pode ser ativado utilizando-se o comando Alterar Atributos da Consulta (CHGQRYA).

O parâmetro Uso do Job Assíncrono (ASYNCJ) no comando CHGQRYA é utilizado para controlar o uso do escritor de resultado temporário. O parâmetro ASYNCJ tem as seguintes opções aplicáveis:

- \*DIST ou \*ANY permitem que os job do escritor de resultado temporário sejam utilizadas para consultas que envolvam arquivos distribuídos.
- \*LOCAL ou \*NONE impedem que o escritor de resultado temporário seja utilizado para consultas de arquivos distribuídos.

## Mensagens do Otimizador com o DB2 Multisystem

O otimizador de consulta distribuída do i5/OS fornece mensagens informativas sobre o processamento da consulta atual quando o job estiver no modo de depuração.

Essas mensagens, que mostram como a consulta distribuída foi processada, são adicionais às mensagens existentes do otimizador. Essas mensagens aparecem para o comando Abrir Arquivo de Consulta (OPNQRYF), DB2 UDB Query Manager e SQL Development Kit, SQL interativo, SQL incorporado e em qualquer HLL (Linguagem de Alto Nível) do servidor iSeries. Cada mensagem aparece no log do job; você precisa apenas colocar o job no modo de depuração.

É possível avaliar o desempenho da consulta distribuída utilizando as mensagens informativas colocadas no log do job pelo gerenciador de banco de dados. O gerenciador de banco de dados pode enviar qualquer uma das seguintes mensagens distribuídas ou mensagens existentes do otimizador quando apropriado. As variáveis com e comercial (&1, &X) são variáveis de substituição que contêm o nome de um objeto ou outro valor de substituição quando a mensagem aparece no log do job:

- CPI4341 Executando consulta distribuída.
- CPI4342 Executando junção distribuída para consulta.
- CPI4343 Mensagens de depuração do otimizador para etapa &1 de &2 da consulta distribuída.
- CPI4345 O arquivo de resultado distribuído temporário &4 foi construído para a consulta.

Essas mensagens fornecem feedback sobre como uma consulta distribuída foi executada e, em alguns casos, indicam os aprimoramentos que podem ser feitos para ajudar a consulta a executar mais rapidamente. As causas e respostas do usuário para as mensagens são parafraseadas a seguir. A ajuda real da mensagem é mais completa e deve ser utilizada ao tentar determinar o significado e as respostas de cada mensagem.

Uma explicação detalhada de cada mensagem é apresentada a seguir:

### **CPI4341**

Executando consulta distribuída.

Essa mensagem indica que um único arquivo distribuído foi consultado e não foi processado em várias etapas. Essa mensagem lista os nós do arquivo em que a consulta foi executada.

### **CPI4342**

Executando junção distribuída para consulta.

Essa mensagem indica que ocorreu uma junção distribuída. Essa mensagem também lista os nós em que a junção foi executada, além dos arquivos que foram unidos.

### **CPI4343**

Mensagens de depuração do otimizador para etapa &1 de &2 da consulta distribuída.

Essa mensagem indica que uma consulta distribuída foi processada em várias etapas e lista o número da etapa atual. Após essa mensagem são exibidas todas as mensagens do otimizador para essa etapa.

### **CPI4345**

O arquivo de resultado distribuído temporário &4 foi construído para a consulta.

Essa mensagem indica que um arquivo de resultado distribuído temporário foi criado e lista um código de razão indicando porque o arquivo temporário foi requerido. Essa mensagem também mostra a chave de particionamento utilizada para criar o arquivo e os nós nos quais o arquivo temporário foi criado.

O exemplo a seguir mostra como examinar as mensagens do otimizador distribuído geradas para determinar como a consulta distribuída foi processada. O exemplo utiliza os arquivos distribuídos EMPLOYEE e DEPARTMENT.

```
SQL:      SELECT A.EMPNO, B.MGRNO, C.MGRNO, D.EMPNO
          FROM  EMPLOYEE A, DEPARTMENT B, DEPARTMENT C, EMPLOYEE D
          WHERE A.EMPNO=B.MGRNO
                AND B.ADMRDEPT=C.DEPTNO
                AND C.DEPTNO=D.WORKDEPT

OPNQRYF:  OPNQRYF FILE((EMPLOYEE) (DEPARTMENT) (DEPARTMENT) (EMPLOYEE))
          FORMAT(JFMT)
          JFLD((1/EMNO 2/MGRNO *EQ)
              (2/ADMRDEPT 3/DEPTNO)
              (3/DEPTNO 4/WORKDEPT))
```

A seguinte lista de mensagens do otimizador distribuído é gerada:

- CPI4343 As mensagens de depuração do otimizador para a etapa &1 de &4 da consulta distribuída são apresentadas a seguir:
  - CPI4345 O arquivo de resultado distribuído temporário \*QQTDF0001 foi construído para a consulta.  
O arquivo B foi direcionado para o arquivo temporário \*QQTDF0001.
- CPI4343 As mensagens de depuração do otimizador para a etapa &2 de &4 da consulta distribuída são apresentadas a seguir:
  - CPI4342 Executando junção distribuída para a consulta.  
Os arquivos B, C e \*QQTDF0001 foram unidos. Essa foi uma combinação de uma junção colocada (entre os arquivos B e C) e uma junção direcionada (com o arquivo \*QQTDF0001).
  - CPI4345 O arquivo de resultado distribuído temporário \*QQTDF0002 foi construído para a consulta.  
O arquivo distribuído temporário \*QQTDF0002 foi criado para conter o resultado da junção dos arquivos B, C e \*QQTDF0001. Esse arquivo foi direcionado.
- CPI4343 As mensagens de depuração do otimizador para a etapa &3 de &4 da consulta distribuída são apresentadas a seguir:
  - CPI4345 O arquivo de resultado distribuído temporário \*QQTDF0003 foi construído para a consulta.  
O arquivo A foi direcionado para o arquivo temporário \*QQTDF0003.
- CPI4343 As mensagens de depuração do otimizador para a etapa &4 de &4 da consulta distribuída são apresentadas a seguir:
  - CPI4342 Executando junção distribuída para a consulta.  
Os arquivos \*QQTDF0002 e \*QQTDF0003 foram unidos. Essa foi uma junção reparticionada, porque os dois arquivos foram direcionados antes da ocorrência da junção.

As ferramentas adicionais que você pode desejar utilizar ao ajustar consultas para desempenho incluem os comandos da CL Imprimir Informações de SQL (PRTSQLINF), que se aplicam aos pacotes e programas SQL e Alterar Atributos da Consulta (CHGQRYA).



## Alterações no Comando Alterar Atributos da Consulta (CHGQRYA) com o DB2 Multisystem

O comando CHGQRYA tem dois parâmetros aplicáveis a consultas distribuídas.

Os dois parâmetros são: ASYNCJ (uso do job assíncrono) e APYRMT (aplicar remoto).

**Nota:** Diferentemente dos outros parâmetros, ASYNCJ e APYRMT não têm valores do sistema. Se um valor diferente do padrão for necessário, o valor deverá ser alterado para cada job.

### Referências relacionadas

“Escritor de Resultado Temporário com o DB2 Multisystem” na página 55

Os escritores de resultado temporário são jobs iniciados pelo sistema que estão sempre ativos.

## Parâmetro Uso de Job Assíncrono (ASYNCJ) com o DB2 Multisystem

É possível utilizar o parâmetro ASYNCJ para controlar o uso do escritor de resultados temporário.

O parâmetro ASYNCJ tem as seguintes opções:

- \*ANY — permite que os jobs do escritor de resultados temporário sejam utilizados para consultas de banco de dados que envolvam arquivos distribuídos.
- \*DIST — permite que os jobs do escritor de resultados temporário sejam utilizados para consultas de banco de dados que envolvam arquivos distribuídos.
- \*LOCAL — permite que os jobs do escritor de resultados temporário sejam utilizados apenas para consultas de arquivos locais. Embora essa opção seja permitida, atualmente não existe suporte do sistema para a utilização de escritores de resultados temporários para o processamento de consultas locais. \*LOCAL foi incluído para desativar o escritor de resultados temporário para consultas distribuídas, embora permita o estabelecimento de comunicações assíncronas.
- \*NONE — nunca utilizar o escritor de resultados temporário. Além disso, quando o processamento distribuído for executado, as comunicações serão executadas de forma síncrona. Isso pode ser muito útil ao analisar consultas porque permite que as mensagens de depuração de consultas de sistemas remotos sejam retornadas ao sistema local.

O exemplo a seguir mostra como desativar o uso de job assíncrono para o processamento de arquivo distribuído:

```
CHGQRYA ASYNCJ(*LOCAL)
```

Esse comando impede que os jobs assíncronos sejam utilizados para consultas que envolvam arquivos distribuídos.

O exemplo a seguir mostra como desativar completamente o uso de job assíncrono:

```
CHGQRYA ASYNCJ(*NONE)
```

Esse comando impede que os jobs assíncronos sejam utilizados para quaisquer consultas. Além disso, para consultas que envolvam arquivo distribuídos, as comunicações com os sistemas remotos são feitas de maneira síncrona.

O exemplo a seguir mostra como utilizar o comando CHGQRYA em combinação com o comando Iniciar Depuração (STRDBG) para analisar uma consulta distribuída:

```
STRDBG UPDPROD(*YES)
CHGQRYA ASYNCJ(*NONE)
STRSQL
      SELECT COUNT(*) FROM EMPLOYEE A
```

As seguintes mensagens de depuração são colocadas no log do job:

A conexão atual é com o banco de dados relacional SYSA.  
Job DDM iniciado.  
As mensagens de depuração do otimizador para a etapa 1 de 2 da consulta distribuída são apresentadas a seguir:  
O arquivo de resultado distribuído temporário \*QQTDF0001 foi construído para a consulta.  
As mensagens a seguir foram criadas no sistema de destino SYSB.  
O acesso de seqüência de chegada foi utilizado para o arquivo EMPLOYEE.  
O acesso de seqüência de chegada foi utilizado para o arquivo EMPLOYEE.  
As mensagens de depuração do otimizador para a etapa 2 de 2 da consulta distribuída são apresentadas a seguir:  
O acesso de seqüência de chegada foi utilizado para o arquivo EMPLOYEE.  
ODP criado.  
Bloco utilizado para consulta.

## Parâmetro Aplicar Remoto (APYRMT)

É possível utilizar o parâmetro APYRMT para especificar se as outras opções de CHGQRYA devem ser aplicadas aos jobs de sistema remoto associados que são utilizados no processamento dos pedidos de consulta distribuídos.

O parâmetro APYRMT tem as seguintes opções:

- \*YES — aplicar as opções de CHGQRYA aos jobs remotos. Isso requer que o job remoto tenha autoridade para utilizar o comando CHGQRYA. Caso contrário, um erro será sinalizado para o job remoto.
- \*NO — aplicar as opções de CHGQRYA apenas localmente.

O exemplo a seguir impede que as opções de CHGQRYA sejam aplicadas remotamente:

```
CHGQRYA DEGREE(*NONE) APYRMT(*NO)
```

Nesse caso, o suporte paralelo a SMP (Symmetric Multiprocessing) é impedido no nó do coordenador, mas os sistemas remotos podem escolher seu próprio grau de paralelismo.

Além desses parâmetros, você deve estar ciente de como o parâmetro Consultar Limite de Tempo (QRYTIMLMT) funciona. O limite de tempo especificado no parâmetro é aplicado a cada etapa (local e remota) da consulta distribuída; ele não é aplicado à consulta inteira. Portanto, é possível que uma etapa da consulta encontre o limite de tempo, enquanto outra pode continuar sem problemas. Embora a opção de limite de tempo possa ser bastante útil, ele deve ser utilizada com cuidado em consultas distribuídas.

## Resumo das Considerações sobre o Desempenho

Existem fatores de desempenho que devem ser considerados ao desenvolver consultas que utilizem arquivos distribuídos.

1. Para o comando OPNQRYF e a API de consulta (QQQQRY), a especificação de ALWCPYDTA(\*OPTIMIZE) permite que cada nó escolha um índice ou uma classificação para satisfazer a ordenação especificada.
2. Para o comando OPNQRYF e a API de consulta (QQQQRY), especificar ALWCPYDTA(\*YES) ou ALWCPYDTA(\*NO) reforça que cada nó utilize um índice que corresponda exatamente aos campos de ordenação especificados. Isso é mais restritivo do que o modo em que o otimizador processa a ordenação para arquivos não-distribuídos.
3. A inclusão de uma cláusula ORDER BY em uma seleção DISTINCT pode retornar registros mais rapidamente pois não necessita de uma classificação final no sistema solicitante.
4. A inclusão de todos os campos da chave de particionamento nos campos de agrupamento geralmente resulta no agrupamento de uma etapa, que é executado melhor do que o agrupamento de duas etapas.
5. A inclusão de todos os campos da chave de particionamento nos critérios de junção geralmente resulta em uma junção distribuída colocada.

6. A inclusão de todos os campos da chave de particionamento em uma seleção de registros igual e separável, geralmente resulta no processamento da consulta em apenas um nó.
7. A inclusão de alguma das funções escalares a seguir em uma seleção de registros igual e separável, geralmente resulta no processamento da consulta em apenas um nó:
  - NODENAME
  - NODENUMBER
  - PARTITION

---

## Informações Relacionadas ao DB2 Multisystem

Aqui estão listados os tópicos do Information Center relacionados ao tópico DB2 Multisystem.

- APIs fornece informações para o programador experiente sobre como utilizar as APIs (Interfaces de Programação de Aplicativos) para algumas funções do sistema operacional.
- Backup, Recuperação e Serviços de Mídia fornece informações sobre configuração e gerenciamento de:
  - Criação de diário, proteção do caminho de acesso e controle de confirmação
  - ASPs (Conjuntos de Armazenamento Auxiliares) do usuário
  - Proteção de disco (paridade do dispositivo, espelhamento e checksum)Esse tópico também fornece informações sobre o desempenho da mídia de backup media e as operações de salvamento/restauração. Inclui ainda tópicos avançados de backup e recuperação, como a utilização do suporte salvar enquanto ativo, salvamento e restauração para um release diferente e dicas e técnicas de programação.
- Linguagem de Controle fornece uma descrição da CL (Linguagem de Controle) do servidor iSeries e seus comandos (outros comandos são descritos nas respectivas publicações do programa licenciado). O tópico também fornece uma visão geral de *todos* os comandos da CL do servidor iSeries e descreve as regras de sintaxe necessárias para codificá-los.
- Programação do Banco de Dados fornece uma discussão detalhada da organização do banco de dados do servidor iSeries, incluindo informações sobre como criar, descrever e atualizar arquivos do banco de dados no sistema.
- Programação de Banco de Dados Distribuído fornece informações sobre a preparação e o gerenciamento de um servidor iSeries em um banco de dados relacional distribuído utilizando a DRDA (Distributed Relational Database Architecture). Ele descreve o planejamento, a configuração, a programação, a administração e a operação de um banco de dados relacional distribuído em mais de um servidor em um ambiente semelhante a um sistema.
- Instalar, Fazer Upgrade ou Excluir o i5/OS e o software relacionado inclui informações de planejamento e instruções passo a passo de procedimentos para instalar o sistema operacional e os programas licenciados.
- OptiConnect descreve o suporte a OptiConnect, que pode conectar vários servidores utilizando um cabo de fibra óptica. O OptiConnect permite acessar bancos de dados entre sistemas mais rapidamente e permite descarregar o trabalho em outro servidor. Tópicos adicionais incluem informações sobre configuração, instalação e operação.
- Conceitos de Programação SQL fornece informações sobre como utilizar o programa licenciado DB2 para DB2 UDB Query Manager e SQL Development Kit. O tópico mostra como acessar dados na biblioteca de bancos de dados e como preparar, executar e testar um programa aplicativo que contém instruções SQL incorporadas. O tópico também contém exemplos de instruções SQL/400 e uma descrição da função SQL interativo, descrevendo também conceitos e regras comuns para a utilização de instruções SQL/400 em COBOL/400, ILE COBOL/400, PL/I, ILE C/400, FORTRAN/400, RPG/400, ILE RPG/400 e REXX. Conceitos de Programação SQL e Programação de Banco de Dados também fornecem informações sobre o DB2 UDB Symmetric Multiprocessing.
- Referência de SQL fornece informações sobre como utilizar instruções DB2 SQL e fornece detalhes sobre a utilização correta das instruções. Exemplos de instruções incluem diagramas de sintaxe, parâmetros e definições. Uma lista das limitações do SQL e uma descrição da SQLCA (Área de Comunicação SQL) e da SQLDA (Área do Descritor SQL) também são fornecidas.

---

## Aviso de Isenção de Responsabilidade e Licença do Código

A IBM concede-lhe uma licença de direitos autorais não exclusivos para usar os exemplos de código de programação, a partir dos quais você pode gerar funções idênticas adaptadas a uma necessidade específica.

| SUJEITA ÀS GARANTIAS ESTABELECIDAS POR LEI, QUE NÃO PODEM SER EXCLUÍDAS, A IBM,  
| SEUS DESENVOLVEDORES E FORNECEDORES DO PROGRAMA NÃO OFERECEM GARANTIA OU  
| CONDIÇÕES, SEJAM EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO SE LIMITANDO ÀS  
| GARANTIAS IMPLÍCITAS OU ÀS CONDIÇÕES DE MERCADO, ADEQUAÇÃO A UM DETERMINADO  
| PROPÓSITO E NÃO-INFRAÇÃO EM RELAÇÃO AO PROGRAMA OU SUPORTE TÉCNICO, SE  
| HOVER.

| SOB NENHUMA CIRCUNSTÂNCIA, A IBM, OS DESENVOLVEDORES OU FORNECEDORES DO  
| PROGRAMA SÃO RESPONSÁVEIS PELOS ITENS A SEGUIR, MESMO SE INFORMADOS DE SUA  
| POSSIBILIDADE:

- | 1. DANOS OU PERDA DE DADOS;
- | 2. DANOS DIRETOS, ESPECIAIS, ACIDENTAIS OU INDIRETOS, OU QUALQUER ESPÉCIE DE DANO  
| DE CONSEQÜÊNCIA ECONÔMICA; OU
- | 3. PERDA DE LUCROS, NEGÓCIOS, RECEITAS, BENS OU ECONOMIAS.

| ALGUMAS JURISDIÇÕES NÃO PERMITEM A EXCLUSÃO OU LIMITAÇÃO DE DANOS DIRETOS,  
| ACIDENTAIS OU CONSEQÜENCIAIS, PORTANTO, ALGUMAS OU TODAS AS LIMITAÇÕES OU  
| EXCLUSÕES ACIMA PODEM NÃO SE APLICAR AO CLIENTE.

---

## Apêndice. Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos.

É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte um representante IBM local para obter informações sobre produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM ou outros direitos legalmente protegidos poderá ser utilizado em substituição a este produto, programa ou serviço. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não-IBM são de inteira responsabilidade do usuário.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum direito sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil  
Av. Pasteur, 138-146  
Botafogo  
Rio de Janeiro, RJ  
CEP 22290-240

Para pedidos de licenças com relação a informações sobre DBCS (Conjunto de Caracteres de Byte Duplo), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie suas consultas por escrito para:

IBM World Trade Asia Corporation Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**O parágrafo a seguir não se aplica ao Reino Unido e a nenhum país em que tais disposições não estejam de acordo com a legislação local:** A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA" SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS DE MERCADO, NÃO-INFRAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações, portanto, esta disposição pode não se aplicar ao Cliente.

Estas informações podem conter imprecisões técnicas ou erros tipográficos. Periodicamente, são feitas alterações nas informações aqui contidas; tais alterações serão incorporadas em novas edições da publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação.

Quaisquer referências neste documento para Web sites que não são IBM são fornecidas apenas para conveniência e não servem de maneira alguma como endosso para estes Web sites. Os materiais destes Web sites não são partes dos materiais para este produto IBM e a utilização de tais Web sites é de seu próprio risco.

A IBM pode utilizar ou distribuir todas os comentários fornecidos pelo Cliente da maneira que achar conveniente, sem que isso implique em qualquer compromisso ou obrigação para com o Cliente.

Os licenciados deste programa que desejam obter informações adicionais sobre o mesmo com o objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil  
Av. Pasteur, 138-146  
Botafogo  
Rio de Janeiro, RJ  
CEP 22290-240

Estas informações podem estar disponíveis, observadas as condições e os termos apropriados, incluindo, em alguns casos, o pagamento de uma taxa.

- | O programa licenciado descrito nestas informações e todo o material licenciado disponível são fornecidos
- | pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato Internacional de Licença do
- | Programa IBM, do Acordo de Licença IBM para Código de Máquina ou de qualquer outro acordo
- | equivalente.

Todos os dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido feitas em sistemas em nível de desenvolvimento e não há garantia de que tais medidas serão as mesmas em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

As informações relativas a produtos não-IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não-IBM. Dúvidas sobre os recursos de produtos não-IBM devem ser encaminhadas diretamente a seus fornecedores.

Todas as declarações relacionadas aos objetivos e intenções futuras da IBM estão sujeitas à alteração ou cancelamento sem aviso prévio, e representam apenas metas e objetivos.

Este documento contém exemplos de dados e relatórios utilizados em operações comerciais de rotina. Para ilustrá-las da forma mais completa possível, os exemplos podem incluir nomes de pessoas, empresas, marcas e produtos. Todos esses nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa real é mera coincidência.

#### LICENÇA DE DIREITOS AUTORAIS:

Estas informações contêm exemplos de programas aplicativos na linguagem fonte, ilustrando as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir tais programas de amostra em qualquer formato sem pagamento à IBM, para fins de desenvolver, usar, comercializar ou distribuir programas aplicativos que estejam de acordo com a interface de programação de aplicativos da plataforma operacional para a qual os programas de amostra são escritos. Estes exemplos não foram testados completamente em todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade em manutenção ou função destes programas.

Cada cópia ou parte deste exemplo de programas ou qualquer trabalho derivado deve incluir um aviso de copyright com os dizeres:

© (o nome de sua empresa) (ano). Partes deste código são derivadas dos Programas de Exemplo da IBM IBM. ©Copyright IBM Corp. \_digite o(s) ano(s)\_. Todos os direitos reservados.

Se estiver visualizando estas informações em cópia eletrônica, as fotos e ilustrações podem não aparecer.

---

## Informações da Interface de Programação

Esta publicação do DB2 Multisystem documenta as Interfaces de Programação planejadas que permitem ao cliente gravar programas para obter os serviços do IBM i5/OS.

---

## Marcas Registradas

Os termos a seguir são marcas registradas da International Business Machines Corporation nos Estados Unidos e/ou em outros países:

- | C/400
- | COBOL/400
- | DB2
- | DB2 Universal Database
- | Distributed Relational Database Architecture
- | DRDA
- | eServer
- | e(logotipo)server
- | i5/OS
- | IBM
- | IBM (logo)
- | Integrated Language Environment
- | iSeries
- | RPG/400
- | SQL/400

Outros nomes de empresas, produtos e serviços podem ser marcas registradas ou marcas de serviço de terceiros.

---

## Termos e Condições

As permissões para o uso dessas publicações estão sujeitas aos seguintes termos e condições.

**Uso Pessoal:** essas publicações podem ser reproduzidas para uso pessoal, não comercial, desde que todos os avisos de propriedade sejam preservados. Não é possível distribuir, exibir ou fazer trabalhos derivados dessas publicações ou de nenhuma parte desse documento, sem consentimento expresso da IBM.

**Uso Comercial:** é permitido reproduzir, distribuir e expor essas publicações exclusivamente dentro de sua empresa, desde que todos os avisos de propriedade sejam preservados. Não é possível fazer trabalhos derivados dessas publicações, ou reproduzir, distribuir ou exibir essas publicações ou qualquer parte deste documento fora da sua empresa, sem o consentimento expresso da IBM.

Exceto conforme concedido expressamente nessa permissão, nenhuma outra permissão, licença ou direito é concedido, seja expressa ou implícita, às publicações ou a qualquer informação, dados, software ou outra propriedade intelectual contida neste documento.

A IBM reserva-se o direito de revogar as permissões aqui concedidas, sempre que, a seu critério, o uso das publicações prejudicar seus interesses ou, conforme determinação da IBM, as instruções anteriormente citadas não estiverem sendo seguidas da forma apropriada.

Não é permitido fazer download, exportar ou reexportar estas informações, exceto em total conformidade com todas as leis e regulamentos aplicáveis, incluindo todas as leis e regulamentos de exportação dos Estados Unidos.

A IBM NÃO FORNECE NENHUMA GARANTIA SOBRE O CONTEÚDO DESSAS PUBLICAÇÕES. AS PUBLICAÇÕES SÃO FORNECIDAS "NO ESTADO EM QUE SE ENCONTRAM" E SEM GARANTIA DE

NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS DE MERCADO, NÃO-INFRAÇÃO E DE ADEQUAÇÃO A UM DETERMINADO PROPÓSITO.







Impresso em Brazil