UNIT 22

SQL 컴파일러

DB2는 시스템 카탈로그의 통계 자료를 이용하여 비용 기반의 옵티마이저를 제공합니다. 요청된 SQL문은 효율적인 액세스 플랜을 위하여 컴파일 과정에서 재작성됩니다. 액세스 플랜은 익스플레인 테이블, Visual Explain, db2expln, db2exfmt 등을 이용하여 확인합니다.

DB2 9.7 개발자 가이드

Developer Edition

- SQL 컴파일러
- 최적화 클래스
- 최적화 클래스 지정 방법
- 쿼리 재작성
- 뷰 병합
- 서브쿼리에서 조인으로의 변환
- 중복 조인 제거
- 공유 총계
- DISTINCT 제거
- 일반 술어 푸시다운
- 상관 해제
- 암시적 술어
- OR에서 IN으로의 변환
- 패키지
- 익스플레인 도구
- Visual Explain
- db2expln 유틸리티
- 동적 SQL문에 대한 db2expln 출력
- 정적 SQL문에 대한 db2expln 출력
- db2exfmt 유틸리티
- 동적 SQL문에 대한 db2exfmt 출력
- 정적 SQL문에 대한 db2exfmt 출력

TOPIC 22 01 SQL 컴파일러

Point

비용 기반 (Cost-based)의 최적화기(Optimizer)를 이용하여 요청된 SQL문에 대한 최소 비용의 액세스 플랜을 선택하는 과정을 SQL 컴파일이라고 합니다. 옵티마이저는 필요에 따라 요청된 SQL문을 보다 효율적인 SQL문으로 재작성하기도 합니다.

Tip

쿼리 그래프 모델(QGM, Query Graph Model)은 SQL문의 컴파일 과정 동안 사용되는 내부적인 데이터베이스로 메모리에 저장됩니다.

1 SQL 컴파일러는 여러 단계를 수행하여 입력된 SQL문을 컴파일하고, 실행할 수 있는 액세스 플랜을 생성합니다. 비용 기반의 옵티마이저를 이용하여 최적의 액세스 플랜을 결정하게 됩니다.

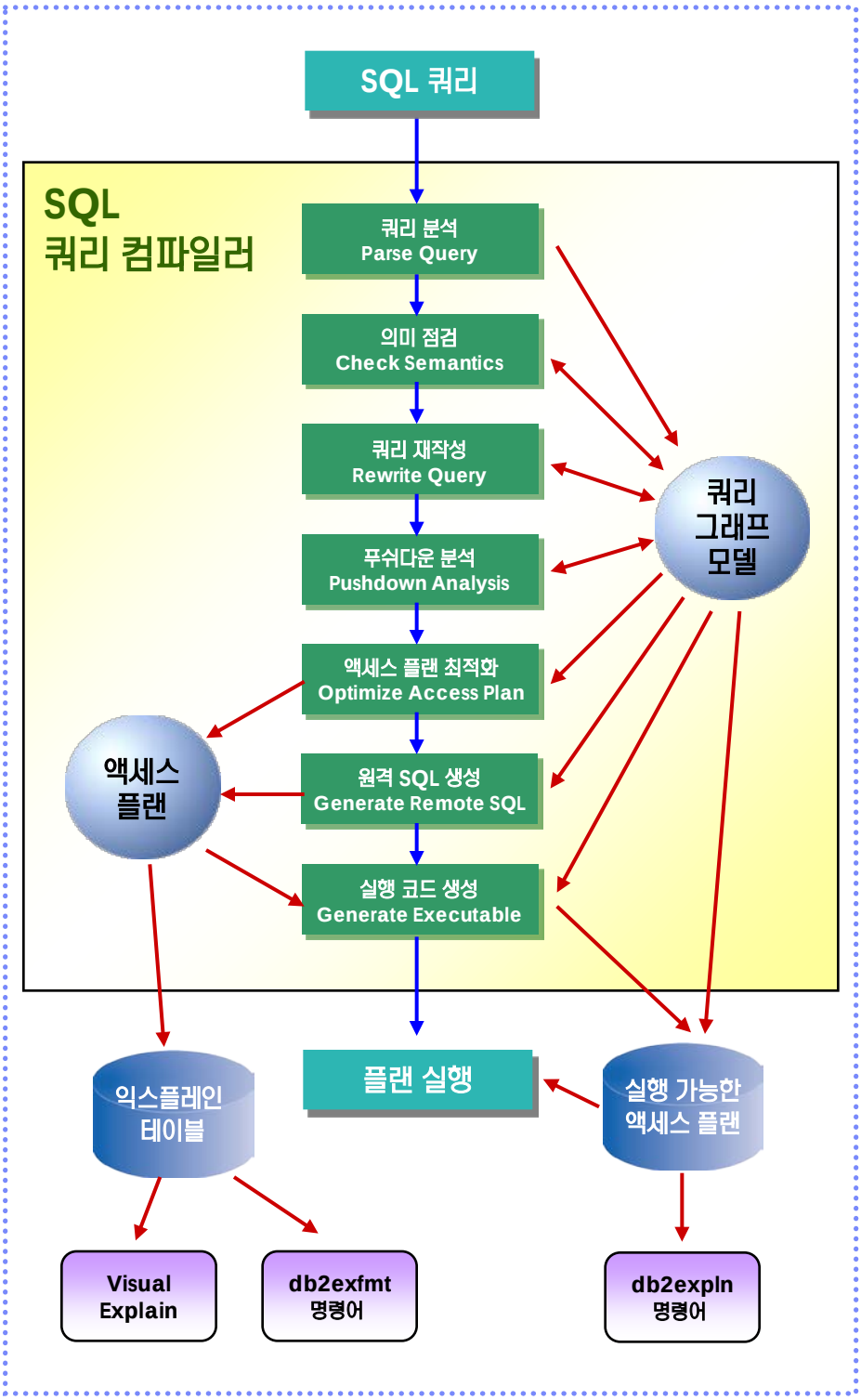

Figure 2201A • • • SQL 컴파일 과정

TOPIC 22 01 SQL 컴파일러

Point

비용 기반 (Cost-based)의 최적화기(Optimizer)를 이용하여 요청된 SQL문에 대한 최소 비용의 액세스 플랜을 선택하는 과정을 SQL 컴파일이라고 합니다. 옵티마이저는 필요에 따라 요청된 SQL문을 보다 효율적인 SQL문으로 재작성하기도 합니다.

Tip

옵티마이저가 가장 효율적인 액세스 플랜을 작성하려면 테이블 및 해당 데이터에 대한 최신 통계 정보가 필요하므로, 자주 변경하는 테이블에 대해서 적절한 간격으로 RUNSTATS를 실행하여 최신 통계 자료를 유지합니다.

RUNSTATS는 automatic으로 운영이 가능합니다. 수동으로 운영해야 할 이유가 없다면 automatic으로 설정하시기 바랍니다.

Tip

조인의 유형에는 Nested Loop 조인, Merge Scan 조인, Hash 조인이 있고, 액세스 유형에는 테이블 스캔과 인덱스 스캔이 있습니다.

Tip

'푸쉬다운 분석'과 '원격 SQL 생성' 단계는 Federated 된 데이터베이스에만 적용됩니다.

Tip

시스템의 통계 자료가 변경되거나, 새로운 인덱스가 추가된 경우에는 정적 SQL문에 대한 액세스 플랜을 명시적으로 재생성하도록 합니다.

2 SQL 컴파일러가 실행하는 각 과정에 대한 설명은 다음과 같습니다.

과 정	설 명
쿼리 구문 분석 Parse Query	SQL 쿼리를 분석하여 구문의 유효성을 확인합니다. 구문 오류가 검출되면, 처리를 중단하고 해당 SQL 오류를 리턴합니다. 쿼리 구문 분석이 완료되면, 쿼리에 대한 내부적인 표현이 QGM에 작성되고 저장됩니다.
의미 점검 Check Sementics	명령문의 각 부분에 의미상 불일치되는 부분이 있는지 확인합니다. 참조 제한 조건, 테이블 점검 제한 조건, 트리거와 뷰 등의 작동 의미를 QGM에 추가합니다. QGM은 쿼리 블록, 서브쿼리, 상관 관계, 파생된 테이블, 표현식, 데이터 유형, 데이터 유형 변환, 코드 페이지 변환, 파티션 키를 포함한 쿼리에 대한 모든 의미를 기록합니다. 의미 점검의 예로 YEAR 스칼라 함수에서 인수로 지정된 컬럼의 데이터 유형이 날짜 시간 데이터 유형인지 확인하는 것을 들 수 있습니다.
쿼리 재작성 Rewrite Query	QGM에 저장된 여러 가지 정보를 이용하여 쿼리를 최적화된 형식으로 변환하여 QGM에 저장합니다.
푸쉬다운 분석 Pushdown Anaysis	SQL문을 실행하는데 필요한 조작을 Federated 된 데이터 소스에서 원격으로 평가할 수 있는지 또는 푸시다운 하는지 여부를 분석합니다.
액세스 플랜 최적화 Optimize Access Plan	QGM에 저장된 정보를 기반으로 쿼리의 의미, 인덱스, 기본 테이블, 파생된 테이블, 서브쿼리, 상관 관계 등을 분석하고, 입력된 SQL문을 실행할 수 있는 다양한 액세스 플랜을 생성합니다. 옵티마이저는 시스템 카탈로그의 통계 자료를 이용하여 실행 비용을 측정하고, 생성된 액세스 플랜 중에서 최소 비용의 액세스 플랜을 선택합니다. 액세스 플랜에는 테이블의 액세스 순서, 사용할 인덱스 선택, 조인 방법, 액세스 유형 등이 포함됩니다.
원격 SQL 생성 Generate Remote SQL	선택한 액세스 플랜이 Federated 된 데이터 소스에서 작동할 수 있는 단계 세트로 구성되어야 하는 경우에는 데이터 소스 SQL dialect를 기본으로 하여 효율적인 SQL문을 작성합니다.
실행 코드 생성 Generate Executable	선택된 액세스 플랜과 QGM을 사용하여 쿼리에 대한 실행 액세스 플랜과 섹션을 작성합니다. 정적 SQL문에 대한 액세스 플랜은 데이터베이스의 시스템 카탈로그 테이블에 패키지로 저장되고, 동적 SQL문에 대한 액세스 플랜은 패키지 캐시에 저장됩니다.

3 패키지가 실행되면 데이터베이스 관리 프로그램은 시스템 카탈로그 테이블 또는 패키지 캐시에 저장된 정보를 사용하여 데이터에 액세스할 방법을 결정하고 쿼리를 실행하여 실행 결과를 반환합니다.

4 호스트 변수, 특수 레지스터, 매개 변수 표시 문자를 가지는 정적 SQL문과 동적 SQL문에 대한 액세스 플랜은 SQL 컴파일러가 선택한 기본 예상치 (Default Filtering Factor)를 이용합니다. 실행시 실제로 입력된 값을 이용한 액세스 플랜을 생성하려면, 바인드시 REOPT 옵션을 사용합니다.

TOPIC 22 02 최적화 클래스

Point

옵티마이저가 SQL문을 컴파일할 때 적용하는 최적화의 심화 정도를 결정하는 수치입니다. 숫자가 높을수록 다양한 알고리즘을 시도하여 액세스 플랜을 결정합니다. 기본 클래스는 5입니다.

Tip

Federated된 데이터베이스에 대한 쿼리에서, 최적화 클래스는 리모트 옵티마이저에 적용되지 않습니다.

Tip

클래스 2는 동적 프로그래밍 대신 Greedy 조인 열거를 사용하는 것을 제외하면 클래스 5와 매우 유사합니다.

1. 사용자는 SQL문이 컴파일될 때 적당한 최적화 클래스를 지정하여 옵티마이저가 가장 효율적인 액세스 플랜을 선택하도록 조절할 수 있습니다. 최적화 클래스를 개별적으로 지정하여 쿼리에 대한 런타임 성능을 향상시킬 수 있으나, 높은 최적화 클래스를 지정할수록 더 많은 컴파일 시간과 시스템 자원을 필요로 하게 됩니다.

2. 최적화 클래스는 다음과 같이 7가지로 구분됩니다.

클래스	설명
0	최소의 최적화 알고리즘을 사용하므로, 쿼리 컴파일 오버헤드를 최소화하는 경우에만 지정하도록 합니다. 인덱스화가 잘 되어 있는 테이블에 액세스하는 매우 간단한 동적 SQL문으로 구성된 응용프로그램에 적합합니다. • 불균등 분포 분산 통계를 고려하지 않습니다. • 기본적인 쿼리 재작성 규칙만이 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • Nested Loop 조인, Index Scan 액세스 메소드만이 작동 가능합니다. • 리스트 프리페치, 스타 조인 전략은 고려되지 않습니다.
1	추가적으로 Merge Scan 조인, Table Scan이 사용되는 점을 제외하면 클래스 0과 유사합니다. • 불균등 분포 분산 통계를 고려하지 않습니다. • 쿼리 재작성 규칙의 일부 기능만 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • Merge Scna 조인, Table Scan이 사용됩니다. • 리스트 프리페치가 사용되지 않습니다.
2	모든 통계 자료와 쿼리 재작성 규칙, 리스트 프리페치, 요약 테이블 등이 적용됩니다. 복합 쿼리에 대한 대체 방법을 거의 고려하지 않으므로, 3 이상의 클래스보다 컴파일 시간이 적게 걸립니다. 특정 쿼리가 거의 반복되지 않는 DSS 또는 OLAP 환경의 복잡한 쿼리에 적합합니다. • 빈도 및 qantile 불균등 분포 분산 통계를 포함한 모든 통계가 사용됩니다. • MQT 라우팅을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • Greedy 조인 열거 기능이 사용됩니다. • 리스트 프리페치, MQT 라우팅을 포함한 액세스 메소드가 고려됩니다. • 스타 조인 전략은 고려되지 않습니다.
3	MVS/ESA용 DB2, OS/390 또는 z/OS의 쿼리 최적화 특성에 가장 유사하며, 중간 수준의 최적화를 위해 적당한 양의 리소스를 사용하므로 대부분의 응용프로그램에 적합합니다. • 불균등 분포 분산 통계는 발생하는 값을 추적하는 데 사용됩니다. • 조인의 서브쿼리 변환 등을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • 동적 프로그래밍 조인 열거 기능이 사용됩니다. • 리스트 프리페치, 인덱스 추가, 스타 조인 등의 액세스 메소드가 고려됩니다.
5	대부분의 다양한 쿼리 변환과 쿼리 최적화 기술을 효과적으로 적용하도록 설계되어 상당한 수준의 최적화를 실행하므로, 트랜잭션과 복합 쿼리를 모두 구성하는 혼합된 환경에 적합합니다. 복잡한 동적 SQL 쿼리에 대해서는 고려되는 액세스 플랜 조합 수를 줄입니다. • 빈도 및 quantile 불균등 분포 분산 통계를 포함한 모든 통계가 사용됩니다. • MQT 라우팅을 포함한 대부분의 쿼리 재작성 규칙이 적용됩니다. • 동적 프로그래밍 조인 열거 기능이 사용됩니다. • 리스트 프리페치, MQT 라우팅을 포함한 액세스 메소드가 고려됩니다.

TOPIC 22 02 최적화 클래스

Point

옵티마이저가 SQL문을 컴파일할 때 적용하는 최적화의 심화 정도를 결정하는 수치입니다. 숫자가 높을수록 다양한 알고리즘을 시도하여 액세스 플랜을 결정합니다. 기본 클래스는 5입니다.

클래스	설명
7	상당한 수준의 최적화를 실행합니다. 복잡한 동적 SQL 쿼리에 대해서도 모든 최적화 기법을 적용한다는 점을 제외하면 클래스 5와 유사합니다.
9	사용 가능한 모든 최적화 기법을 사용하도록 지시하므로 고려되는 가능한 액세스 플랜의 수를 크게 확장할 수 있습니다. 대형 테이블을 통해 매우 복잡하고 장시간에 걸쳐 실행되는 쿼리에 대해 향상된 액세스 플랜을 생성할 수 있는지 확인하는 용도로 사용합니다. • 사용 가능한 모든 통계를 이용합니다. • 모든 쿼리 재작성 규칙을 적용합니다. • 카테시안곱과 모든 가능한 조인 열거 기능을 사용합니다. • 모든 액세스 메소드를 적용합니다.

Tip

장시간 실행되는 쿼리를 분석하려면, db2batch로 쿼리를 실행하여 SQL문을 컴파일하는 데 걸린 시간과 실행에 걸린 시간을 확인합니다. 컴파일 시간을 줄이려면 낮은 최적화 클래스를 지정하고, 실행 시간을 줄이려면 높은 최적화 클래스를 지정합니다.

3 쿼리 최적화 클래스 1, 2, 3, 5, 7은 모두 일반적인 목적으로 사용하는 데 적합합니다. 쿼리 컴파일 시간을 더 줄여야 하거나, SQL문이 아주 단순한 경우에만 클래스 0을 고려합니다.

4 대부분의 SQL문은 기본 쿼리 최적화 클래스인 최적화 클래스 5를 사용하여 적합하게 최적화될 수 있습니다. 주어진 최적화 클래스에서 쿼리 컴파일 시간과 자원 소비는 쿼리의 복잡도, 특히 조인 및 서브쿼리 수에 의해 주로 영향을 받습니다. 그러나, 컴파일 시간과 자원의 사용은 수행된 최적화의 양에 의해서도 영향을 받습니다. 소규모의 데이터베이스, 간단한 동적 쿼리, 컴파일시 사용하는 메모리의 양을 제한, 컴파일 시간의 단축 등이 필요하다면, 최적화 클래스를 조절할 수 있습니다.

Tip

쿼리 생성 프로그램에 의해 생성된 SQL문에 대해서는 높은 쿼리 최적화 클래스가 유리합니다.

5 최적화 클래스를 선택하는 경우에는 일반적으로 다음과 같은 순서를 고려합니다.

- 기본값인 클래스 5로 시작하고, 다른 클래스를 사용하려면 클래스 1, 2, 3부터 시도합니다.
- 같은 컬럼에서 많은 Join 술어를 가진 테이블이 있거나 컴파일 시간이 중요하다면, 클래스 1 또는 2를 사용합니다.
- 하나 또는 적은 테이블만 액세스, 단일 또는 적은 행만 페치, 완전하고 고유한 인덱스를 사용하는 1 초 이하의 OLTP 쿼리에 대해서는 클래스 0 또는 1을 사용합니다.
- 30초를 초과하여 장시간 실행되는 쿼리에 대해서는 클래스 3, 5, 7 을 사용합니다.
- 클래스 3 이상은 동적 프로그래밍 조인 열거 알고리즘을 사용하여 많은 액세스 플랜을 고려하므로, 테이블 수가 증가하면 클래스 0, 1, 2보다 컴파일 시간이 많이 소요됩니다.

6 결정 지원 쿼리 또는 월말 보고 쿼리 등의 복합 쿼리는 기본값인 클래스 5 이상을 적용합니다.

- 대형 테이블에 대한 액세스
- 많은 수의 술어
- 많은 서브쿼리
- 많은 조인
- UNION 및 INTERSECT와 같은 많은 집합 연산자
- 많은 규정 행
- GROUP BY 및 HAVING 조작
- 중첩 테이블 표현식
- 많은 수의 뷰

TOPIC 22 03 최적화 클래스 지정 방법

Point

정적 SQL문은 PREP 또는 BIND 명령어의 QUERYOPT 옵션으로 지정하며, 동적 SQL문은 특수 레지스터리인 CURRENT QUERY OPTIMIZATION 으로 지정합니다. 데이터베이스 구성 변수인 dft_queryopt로 기본 최적화 클래스를 변경합니다.

1. 최적화 클래스는 dft_queryopt 데이터베이스 구성 변수로 조절합니다. get db cfg 명령어로 확인하고, update db cfg 명령어로 변경합니다. 기본값은 5 로 설정되어 있습니다.

```
$ db2 get db cfg for <DB명> | grep DFT_QUERYOPT
$ db2 update db cfg for <DB명> using DFT_QUERYOPT <최적화클래스>
```

2. 정적 SQL문에 대한 컴파일은 한 번만 실행되므로, 컴파일 시간과 자원은 한 번만 소요됩니다. 생성된 액세스 플랜은 시스템 카탈로그에 저장되어 여러 번 사용될 수 있습니다. 일반적으로, 정적 SQL은 기본 쿼리 최적화 클래스를 사용합니다.

3. 정적 SQL문에 대한 최적화 클래스는 프리컴파일이나 바인드 과정에서 QUERYOPT 라는 옵션을 이용하여 패키지 단위로 변경할 수 있습니다. 옵션을 지정하지 않으면, dft_queryopt 데이터베이스 구성 변수에 지정된 값을 사용합니다.

```
$ db2 prep <ESQL 소스파일명> QUERYOPT <최적화 클래스> bindfile
$ db2 bind <바인드 파일명> QUERYOPT <최적화 클래스>
```

4. 정적 SQL문에 대해 적용된 최적화 클래스는 SYSCAT.PACKAGES 카탈로그 테이블의 QUERYOPT 컬럼에서 확인할 수 있습니다.

```
$ db2 "select pkgname, QUERYOPT from syscat.packages"
```

5. 동적 SQL문은 실행시에 액세스 플랜이 생성되어 실행되므로, 동적 SQL문에 대한 컴파일 시간과 자원은 실행시마다 소요될 수도 있습니다. 동적 SQL문에 대한 액세스 플랜은 시스템 카탈로그에 저장되지 않고, 패키지 캐시에 저장됩니다. 액세스 플랜이 캐시된 이후에 환경이 변경되지 않았다면, 이후의 PREPARE문에 포함된 동일한 동적 SQL문은 패키지 캐시의 액세스 플랜을 재사용하기 때문에 동일한 동적 SQL문은 재컴파일할 필요가 없습니다. 만약, 환경적인 변경이 발생하면, 동적 SQL문은 재컴파일되어 패키지 캐시에 다시 저장됩니다.

6. 패키지 캐시의 크기는 PCKCACHESZ 데이터베이스 구성 변수로 조절합니다. get db cfg 명령어로 확인하고, update db cfg 명령어로 변경합니다. 기본값은 -1로 설정되어 있으므로 maxappls 데이터베이스 구성 변수에 지정된 값의 8배가 할당됩니다.

```
$ db2 get db cfg for <DB명> | grep PCKCACHESZ
$ db2 update db cfg for <DB명> using PCKCACHESZ <최적화클래스>
```

7. 동적 SQL문은 최적화 클래스는 SET CURRENT QUERY OPTIMIZATION 이라는 특수 레지스터리를 이용하여 변경합니다. 레지스터리의 값을 지정하지 않으면, dft_queryopt 데이터베이스 구성 변수에 지정된 값이 사용됩니다.

```
$ db2 set CURRENT QUERY OPTIMIZATION = <최적화 클래스>
```

Tip

PREP 명령어에서 최적화 클래스는 BIND 명령어에서 지정한 최적화 클래스의 값으로 대체됩니다.

Tip

REBIND 명령어로 패키지를 리바인드하면 정적 SQL문에 대해 기존의 최적화 클래스가 사용됩니다. 최적화 클래스를 변경하려면 BIND 명령어를 사용합니다.

Tip

FLUSH PACKAGE CACH문은 현재 패키지 캐시에 캐시된 모든 동적 SQL문을 제거합니다. 캐시된 동적 SQL문은 논리적으로 무효화되었으므로, 동일한 동적 SQL문이 요청되면 재컴파일됩니다.

Tip

동적 SQL문에 대해 항상 특정한 최적화 클래스가 적용되도록 하려면, 응용프로그램의 소스에 SET문을 포함시킬 수 있습니다.

TOPIC 22 04 쿼리 재작성

Point

SQL 컴파일러는 최적의 액세스 플랜을 선택하기 위하여 쿼리를 재작성합니다. 쿼리의 재작성의 유형으로는 조작 병합, 조작 이동, 술어 변환 등이 있습니다.

1 주요한 쿼리 재작성의 유형은 다음과 같습니다.

유 형	설 명
조작 병합	가장 적은 수의 조작, 특히 SELECT 조작을 갖도록 쿼리를 구성하기 위해, 쿼리를 재작성하여 쿼리 조작을 병합합니다. 뷰 병합, 서브쿼리에서 조인으로의 변환, 중복 조인 제거, 공유 집계 등을 예로 들 수 있습니다.
조작 이동	조작 및 술어의 개수를 최소로 하는 쿼리를 구성하기 위해 쿼리를 재작성하여 쿼리 조작을 이동시킵니다. DISTINCT 제거, 일반 술어 푸시다운, 상관 해제 등을 예로 들 수 있습니다.
술어 변환	기존의 술어를 특정 쿼리에 대해 보다 최적화된 술어로 변환하도록 쿼리를 재작성합니다. 암시적 술어의 추가, OR에서 IN으로의 변환을 예로 들 수 있습니다.

2 조작 병합의 예는 다음과 같습니다.

방 법	설 명
뷰 병합	뷰를 사용하는 SELECT문은 테이블의 조인 순서를 제한하며, 테이블의 중복 조인을 유발할 수도 있습니다. 뷰가 병합되면, 이러한 제한 사항을 없앨 수 있습니다.
서브쿼리에서 조인으로의 변환	SELECT문에 서브쿼리가 들어 있는 경우, 서브쿼리를 제거하여 조인으로 변환합니다.
중복 조인 제거	중복 조인을 제거하여 SELECT문을 단순화할 수 있습니다.
공유 집계	쿼리에서 사용된 특정 집계 함수를 쿼리에 포함된 기존의 다른 집계 함수들의 실행 결과를 이용하여 재작성함으로써 수행해야 할 계산의 개수를 줄일 수 있습니다.

3 조작 이동의 예는 다음과 같습니다.

방 법	설 명
DISTINCT 제거	DISTINCT 조작이 수행되는 지점을 이동시키거나, 조작을 완전히 제거하여 실행 비용을 줄일 수 있습니다.
일반 술어 푸시다운	조건에 맞는 자료의 범위 또는 개수를 더 많이 줄일 수 있는 술어가 먼저 쿼리에 적용될 수 있도록 하기 위하여 적용되는 술어의 순서를 변경할 수 있습니다.
상관 해제	파티션된 데이터베이스 환경에서 데이터베이스 파티션 사이에서의 결과 세트 이동은 비효율적입니다. 다른 데이터베이스 파티션으로 브로드캐스트해야 하는 크기를 또는 개수를 줄이는 것이 쿼리 재작성의 목적입니다.

4 술어 이동의 예는 다음과 같습니다.

방 법	설 명
암시적 술어의 추가	추가적인 테이블 조인을 고려할 수 있도록 숨어있는 술어가 해당 쿼리에 추가될 수 있습니다.
OR에서 IN으로의 변환	OR 술어를 IN 술어로 변환할 수 있습니다.

Tip

필요에 따라 IN 술어를 OR 술어로 변환할 수도 있습니다.

TOPIC 22 05 뷰 병합

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 뷰 병합을 들 수 있습니다. 뷰를 사용하는 SELECT문의 테이블의 조인 순서에 대한 제한과 테이블의 중복 조인을 제거할 수 있습니다.

1 뷰를 병합하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 KES.EMPL입니다.

```
$ db2 "SELECT id, name, salary FROM kes.empl"
ID          NAME                SALARY
-------------------------------------------------------------
    1       KES                 100
    2       KHY                 250
    3       JHS                 300
    4       JJY                 280
```

2 KES.EMPL 테이블에서 NAME 컬럼의 값이 'K' 로 시작되는 정보만 표시하는 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW kes.vname AS
    SELECT * FROM kes.empl WHERE name LIKE 'K%';
$ db2 –tf view.sql
```

3 KES.EMPL 테이블에서 SALARY 컬럼의 값이 200 이상인 정보만 표시하는 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW kes.vsalary AS
    SELECT * FROM kes.empl WHERE salary >= 200;
$ db2 –tf view.sql
```

4 KES.EMPL 테이블에서 NAME 컬럼의 값이 'K' 로 시작되고, SALARY 컬럼의 값이 200 이상인 사원의 정보만 표시하기 위하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   v1.id, v1.name, v2.salary
FROM     kes.vname v1, kes.vsalary v2
WHERE    v1.id = v2.id;
$ db2 –tf query.sql
```

5 쿼리 재작성 과정에서 두 개의 뷰는 병합되어 다음과 같이 재작성될 수 있습니다.

```
SELECT   v1.id, v1.name, v2.salary
FROM     kes.empl v1, kes.empl v2
WHERE    v1.id = v2.id
AND      v1.name LIKE 'K%'
AND      v2.salary >= 200
```

Tip

사용자가 작성한 SELECT문을 포함하는 두 개의 뷰로부터 SELECT문을 병합하면, 액세스 플랜 선택시에 보다 많은 선택 사항을 고려할 수 있습니다. 병합된 두 개의 뷰가 동일한 기본 테이블을 사용하는 경우에는 추가 재작성 작업을 수행할 수도 있습니다.

TOPIC 22 06 서브쿼리에서 조인으로의 변환

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 서브쿼리에서 조인으로의 변환을 들 수 있습니다. 서브쿼리가 제거되어 조인으로 변환될 수도 있습니다.

Tip

SAMPLE 데이터베이스를 생성하려면, 인스턴스 사용자로 로그인한 후에 db2sampl 명령어를 실행합니다.

1 서브쿼리에서 조인으로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE와 DEPARTMENT 입니다.

```
$ db2 "SELECT empno, lastname, workdept FROM employee"
EMPNO  LASTNAME       WORKDEPT
-------------------------------------------------------
000010 HAAS           A00
000020 THOMPSON       B01
000030 KWAN           C01
000050 GEYER          E01
000060 STERN          D11

$ db2 "SELECT deptno, deptname FROM department"
DEPTNO DEPTNAME
-------------------------------------------------------
A00    SPIFFY COMPUTER SERVICE DIV.
B01    PLANNING
C01    INFORMATION CENTER
D01    DEVELOPMENT CENTER
D11    MANUFACTURING SYSTEMS
```

2 서브쿼리를 포함한 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   empno, lastname
FROM     employee
WHERE    workdept IN
         (SELECT   deptno
          FROM     department
          WHERE    deptname = 'PLANNING');
$ db2 –tf query.sql
```

Tip

일반적으로 조인은 서브쿼리보다 더 효율적으로 실행됩니다.

3 서브쿼리는 다음과 같이 조인으로 변환될 수 있습니다.

```
SELECT   DISTINCT empno, lastname
FROM     employee emp,
         department dept
WHERE    emp.workdept = dept.deptno
AND      dept.deptname = 'PLANNING'
```

TOPIC 22 07 중복 조인 제거

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 중복 조인 제거를 들 수 있습니다. 중복 조인을 제거하여 SELECT문을 단순화하면 실행 비용을 줄일 수 있습니다.

1 서브쿼리에서 조인으로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE와 DEPARTMENT 입니다. 다음의 쿼리는 EMPLOYEE 테이블에 대한 불필요한 중복 조인을 포함하고 있습니다.

```
$ cat query.sql
SELECT   e1.empno, e1.firstnme, e1.lastname, e1.edlevel
FROM     employee e1, employee e2
WHERE    e1.empno = e2.empno
AND      e1.edlevel > 17
AND      e2.salary  > 35000 ;
$ db2 –tf query.sql
```

2 EMPLOYEE 테이블에 대한 중복 조인을 제거하여 다음과 같이 단순화될 수 있습니다.

```
SELECT   empno, firstnme, lastname, edlevel
FROM     employee
WHERE    edlevel > 17
AND      salary  > 35000
```

3 또 다른 예를 위해 다음과 같이 뷰를 작성합니다. EMPLOYEE와 DEPARTMENT 테이블에는 WORKDEPT와 DEPTNO 컬럼을 통해 참조 제한 조건이 설정되어 있다고 가정합니다.

```
$ cat view.sql
CREATE VIEW vpeople AS
      SELECT   lastname, salary, deptno, deptname, mgrno
      FROM     employee e, department d
      WHERE    e.workdept = d.deptno ;
$ db2 –tf view.sql
```

4 위에서 정의한 뷰를 이용하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT  lastname, salary FROM vpeople;
$ db2 –tf query.sql
```

5 VPEOPLE 뷰에서 사용되었던 EMPLOYEE와 DEPARTMENT 테이블에 대한 조인이 제거되고 아래와 같이 재작성될 수 있습니다.

```
SELECT   lastname, salary
FROM     employee
WHERE    workdept IS NOT NULL
```

Tip

쿼리를 실행하는 사용자에게 테이블이 아닌 뷰에 대한 액세스 권한만이 있다면, 이러한 상황에서는 쿼리를 재작성할 수 있다는 사실을 알더라도 테이블에 대한 액세스 권한이 없기 때문에 재작성하지 못할 수도 있습니다.

TOPIC 22 08 공유 총계

Point

쿼리의 재작성 유형 중에 조작 병합의 예로 공유 총계를 들 수 있습니다. 쿼리에서 사용된 특정 집계 함수를 쿼리에 포함된 기존의 다른 집계 함수들의 실행 결과를 이용하여 재작성함으로써 수행해야 할 계산의 개수를 줄여서 쿼리의 실행 비용을 줄일 수 있습니다.

1 쿼리에 사용되는 함수의 종류가 많으면 쿼리의 실행 속도는 당연히 떨어집니다. 쿼리에 이미 사용된 다른 함수를 이용하여 필요한 함수의 기능을 대체할 수 있는 있다면, 쿼리 내에서 수행될 계산의 개수를 줄이면서 더 개선된 액세스 플랜을 생성할 수 있습니다.

2 공유 총계를 이용하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE입니다. SUM, AVG, COUNT 함수를 사용하는 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   SUM(salary+bonus+comm) AS osum,
         AVG(salary+bonus+comm) AS oavg,
         COUNT(*) AS ocount
FROM     employee;
$ db2 –tf query.sql
```

Tip

AVG 함수를 대체하기 위해 사용된 osum / ocount 표현식에서 나눗셈 연산은 한 번만 실행됩니다.

3 SUM과 AVG 함수의 대상으로 사용되는 계산식인 SALARY+BONUS+COMM 이 동일하므로 AVG 함수는 SUM 함수와 COUNT 함수를 이용하여 SUM/COUNT 형식으로 재작성할 수 있습니다. AVG 함수의 대상으로 사용되었던 계산식이 제거되어 실행해야할 계산식의 개수를 줄였으므로 쿼리의 실행 비용을 줄일 수 있습니다.

```
SELECT   osum,
         osum / ocount
         ocount
FROM   (SELECT   sum(salary+bonus+comm) as osum,
                 count(*) as ocount
         FROM employee
        ) AS shared_agg;
```

TOPIC 22 09 DISTINCT 제거

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 DISTINCT 제거를 들 수 있습니다. DISTINCT 조작이 수행되는 지점을 이동시키거나, 조작을 완전히 제거하여 실행 비용을 줄일 수 있습니다.

1. DISTINCT를 제거하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE입니다.

2. EMPNO 컬럼은 EMPLOYEE 테이블의 기본키로 정의되어 있다고 가정합니다. 기본키를 생성하면, 자동적으로 고유한 인덱스가 생성됩니다.

```
$ db2 describe indexes for table employoee
인덱스          인덱스                          고유한   컬럼     컬럼
스키마          이름                            규칙     수       이름
--------------------------------------------------------------------------------
SYSIBM          SQL060511155131630              P        1        +EMPNO
```

3. 다음과 같이 불필요한 DISTINCT 키워드가 포함된 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   DISTINCT empno, firstnme, lastname
FROM     employee;
$ db2 -tf query.sql
```

4. EMPNO 컬럼이 EMPLOYEE 테이블의 기본 키로 정의되었고, 쿼리의 SELECT 절에 기본키인 EMPNO가 선택되기 때문에 리턴되는 각 행이 고유할 것이라는 것을 이미 알고 있습니다. 이러한 경우에는, DISTINCT 키워드를 사용한 경우와 사용하지 않은 경우의 실행 결과가 동일하다는 것이 보장되므로 위의 쿼리는 DISTINCT절을 제거하여 다음과 같이 재작성됩니다.

```
SELECT   empno, firstnme, lastname
FROM     employee
```

Tip

UNIQUE 옵션을 가진 고유 인덱스를 생성하고, SELECT 절에 고유 인덱스의 키 컬럼이 모두 포함되었다면, 기본키와 마찬가지로 DISTINCT 키워드는 제거될 수 있습니다.

TOPIC 22-10 일반 술어 푸시다운

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 일반 술어 푸시다운을 들 수 있습니다. 조건에 맞는 자료의 범위 또는 개수를 더 많이 줄일 수 있는 술어가 먼저 쿼리에 적용될 수 있도록 하기 위하여 적용되는 술어의 순서를 변경하여 비용을 줄일 수 있습니다.

1 일반 술어를 푸시다운하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 푸쉬다운이란 쿼리에서 표현된 여러 개의 술어 중에서 순서적으로 나중에 적용될 술어를 먼저 적용될 수 있도록 해당 술어의 적용 순서 레벨을 낮추는 방법입니다. 즉, WHERE 절의 AND 또는 OR로 연결된 여러 개의 술어 중에서 데이터 필터링의 정도가 높은 술어를 더 앞쪽으로 배치하여 술어의 적용 순서를 변경합니다.

2 EMPLOYEE 테이블에서 WORKDEPT 컬럼의 값이 'D11' 인 부서에 속한 모든 사원의 목록을 제공하는 다음과 같은 뷰를 생성합니다.

```
$ cat view.sql
CREATE VIEW d11_empl (empno, firstnme, lastname, phoneno) AS
SELECT   empno, firstnme, lastname, phoneno
FROM     employee
WHERE    workdept = 'D11';
$ db2 –tf view.sql
```

3 뷰를 이용하여 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   firstnme, phoneno
FROM     d11_empl
WHERE    lastname = 'BROWN';
$ db2 –tf query.sql
```

4 D11_EMPL 뷰에 대한 쿼리는 단순히 EMPLOYEE 테이블에 대한 쿼리로 재작성될 수도 있습니다. WORKDEPT = 'D11' 이라는 술어를 먼저 적용하는 것이 쿼리를 더 빠르게 실행할 수 있다고 판단되는 경우에는 일반 술어의 푸쉬다운이 발생하지 않습니다.

```
SELECT   firstnme, phoneno
FROM     employee
WHERE    workdept = 'D11'
AND      lastname = 'BROWN'
```

5 푸시다운이 유리하다고 판단되면, 뷰 D11_EMPLOYEE의 조건문에 원래의 쿼리에서 지정된 LASTNAME = 'BROWN' 이라는 술어가 먼저 적용될 수 있도록 푸시다운하여 다음과 같이 재작성될 수 있습니다.

```
SELECT   firstnme, phoneno
FROM     employee
WHERE    lastname = 'BROWN'
AND      workdept = 'D11'
```

Tip

UNION, GROUP BY, 중첩 테이블 표현식, 공통 테이블 표현식이 사용된 쿼리에서도 술어가 푸시다운될 수 있습니다.

TOPIC 22 11 상관 해제

Point

쿼리의 재작성 유형 중에 조작 이동의 예로 상관 해제를 들 수 있습니다. 파티션된 데이터베이스 환경에서 다른 데이터베이스 파티션으로 브로드캐스트해야 하는 크기를 또는 개수를 줄이면 쿼리의 실행 비용을 줄일 수 있습니다.

1. 파티션된 데이터베이스 환경에서 PROJECT 테이블의 PROJNAME 컬럼과 EMPLOYEE 테이블의 SALARY, BONUS, COMM 컬럼을 이용하여 '%PROGRAMMING%' 프로젝트에서 작업 중이면서 총급여가 평균보다 적은 사원을 조회하는 쿼리를 실행합니다.

```
SELECT   p.projno, e.empno, e.lastname,
         e.salary+e.bonus+e.comm AS pay
FROM     employee e, project p
WHERE    p.respemp = e.empno
AND      p.projname like '%PROGRAMMING%'
AND      e.salary+e.bonus+e.comm <
             (SELECT   avg(e1.salary+e1.bonus+e1.comm)
              FROM     employee e1, project p1
              WHERE    p1.projname like '%PROGRAMMING%'
              AND      p1.projno = p.projno
              AND      e1.empno = p1.respemp)
```

2. 이 쿼리는 PROJECT 테이블에 대해 상관 관계가 있고, PROJECT와 EMPLOYEE 테이블의 데이터는 PROJNO 컬럼을 기준으로 동일한 파티션에 저장되어 있지 않으므로, PROJECT 테이블의 데이터를 다른 데이터베이스 파티션으로 브로드캐스트하게 될 수도 있습니다. 또한, 서브쿼리가 여러 번 평가되어야 하므로 실행 비용이 높습니다. EMPLOYEE와 PROJECT 테이블을 조인하여 PROJNO 컬럼별로 총급여를 계산하는 AVG_PROJ 라는 공통 테이블을 생성하여 AVG_COMP 컬럼에 PROJNO 컬럼의 값별로 평균 총급여를 저장하도록 합니다.

```
WITH     avg_proj(projno, avg_pay) AS
        (SELECT      p2.projno, avg(e1.salary+e1.bonus+e1.comm)
         FROM        employee e1, project p2
         WHERE       e1.empno = p2.respemp
         AND         p2.projname LIKE '%PROGRAMMING%'
         GROUP BY    p2.projno
```

3. 공통 테이블을 이용하여 재작성된 쿼리에서 AVG_PROJ 공통 테이블로 PROJNO 컬럼별 평균 총급여인 AVG_COMP를 먼저 계산하고, 그 결과를 EMPLOYEE 테이블이 포함된 모든 데이터베이스 파티션으로 브로드캐스트할 수 있습니다.

```
SELECT   p.projno, e.empno, e.lastname,
         e.salary+e.bonus+e.comm as compensation
FROM     project p, employee e, avg_proj a
WHERE    p.empno = e.empno
AND      p.projname LIKE '%programming%'
AND      p.projno = a.projno
AND      e.salary+e.bonus+e.comm < a.avg_comp
```

TOPIC 22-12 암시적 술어

Point

쿼리의 재작성 유형 중에 술어 이동의 예로 암시적 술어를 들 수 있습니다. 추가적인 테이블 조인을 고려할 수 있도록 숨어있는 술어를 해당 쿼리에 추가하여 쿼리의 실행 비용을 줄일 수 있습니다.

1. 암시적 술어를 추가하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 DEPARTMENT, EMPLOYEE, PROJECT 입니다. 다음의 쿼리는 DEPARTMENT 테이블의 ADMRDEPT 컬럼을 이용하여 'E01' 부서에게 보고하는 부서의 관리자들에 대한 정보와 해당 관리자가 담당하는 프로젝트의 목록을 표시합니다.

```
$ cat query.sql
SELECT   d.deptname d.mgrno, e.lastname, p.projname
FROM     department d,
         employee e,
         project p
WHERE    d.admrdept = 'E01'
AND      d.mgrno = e.empno
AND      e.empno = p.respemp;
$ db2 –tf query.sql
```

2. WHERE 절의 D.MGRNO = E.EMPNO 술어와 E.EMPNO = P.RESPEMP 술어로부터 DEPARTMENT 테이블과 PROJECT 테이블에 대해 MGRNO 컬럼과 RESPEMP 컬럼을 이용하여 DEPT.MGRNO = PROJ.RESPEMP 라는 암시적인 술어를 유추할 수 있습니다. 다음과 같은 암시적 술어를 추가하면, DEPARTMENT 테이블과 PROJECT 테이블에 대해 추가적으로 조인을 고려할 수 있습니다.

```
DEPT.MGRNO = PROJ.RESPEMP
```

3. 또 다른 예를 위해 EMPLOYEE 테이블의 WORKDEPT 컬럼과 DEPARTMENT 테이블의 DEPTNO 컬럼을 이용하여 부서 번호가 'E00' 보다 큰 부서와 이 부서에서 일하는 사원의 이름을 표시하는 다음과 같은 쿼리를 실행합니다.

```
$ cat query.sql
SELECT   empno, lastname, firstname, deptno, deptname
FROM     employee emp,
         department dept
WHERE    emp.workdept = dept.deptno
AND      dept.deptno > 'E00';
$ db2 –tf query.sql
```

4. 원래의 쿼리에 표현된 EMP.WORKDEPT = DEPT.DEPTNO 술어에 의하면 EMPLOYEE 테이블의 WORKDEPT 컬럼과 DEPARTMENT 테이블의 DEPTNO 컬럼은 동일한 것을 보장할 수 있으므로, DEPT.DEPTNO > 'E00' 술어로부터 유추된 EMP.WORKDEPT > 'E00' 이라는 암시적 술어를 추가하면, EMPLOYEE 테이블에서 조인되는 행의 수를 줄일 수 있으므로 쿼리의 WHERE 절에는 다음과 같은 술어가 추가됩니다.

```
EMP.WORKDEPT > 'E00'
```

TOPIC 22 13 OR에서 IN으로의 변환

Point

쿼리의 재작성 유형 중에 술어 이동의 예로 OR에서 IN으로의 변환을 들 수 있습니다. OR 절을 IN 술어로 변환하거나, IN 술어를 OR 절로 변환하여 쿼리의 실행 비용을 줄일 수 있습니다.

1. OR 술어를 IN 술어로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 동일한 컬럼으로 OR 절을 이용하여 표현한 조건식은 IN 술어로 변환되어 쿼리의 실행 비용을 줄일 수 있습니다.

2. OR 술어를 IN 술어로 변환하는 쿼리 재작성의 예는 다음과 같습니다. 사용되는 테이블은 SAMPLE 데이터베이스에서 제공되는 EMPLOYEE 입니다. 다음의 쿼리는 EMPLOYEE 테이블의 WORKDEPT 라는 동일한 컬럼으로 OR절을 이용하여 'D11', 'D21', 'E21' 부서에 속한 사원들의 정보를 표시합니다.

```
$ cat query.sql
SELECT *
FROM    employee
WHERE   workdept = 'D11'
OR      workdept = 'D21'
OR      workdept = 'E21';
$ db2 –tf query.sql
```

3. WORKDEPT 컬럼에 대한 인덱스가 없는 경우에 OR절을 다음과 같이 IN 술어로 재작성되어 쿼리를 보다 효율적으로 처리할 수 있습니다.

```
SELECT *
FROM    employee
WHERE   workdept IN ('D11', 'D21', 'E21')
```

Tip

조건식에 사용된 컬럼에 대한 인덱스가 존재한다면, 상황에 따라서 인덱스 ORING을 수행할 수 있도록 하기 위하여 IN 술어가 OR절로 재작성될 수도 있습니다.

TOPIC 22 14 패키지

Point

SQL 컴파일러가 정적 SQL문을 컴파일하여 생성한 액세스 플랜은 SYSIBM.SYSPACKAGES를 비롯한 여러 개의 데이터베이스 시스템 카탈로그 테이블에 패키지로 저장되어 관리되고, 패키지 캐시를 통해서 실행됩니다.

Tip

패키지명의 최대 길이는 8자입니다. BIND 명령어의 PACKAGE 옵션으로 변경할 수 있습니다. 기본값은 소스 파일명의 최초 8글자입니다.

Tip

바인드 파일에 포함된 SQL문은 db2bfd 명령어로 확인할 수 있습니다.

Tip

액세스 플랜은 SYSIBM.SYSPLAN 테이블에 바이너리 형식으로 저장되므로 익스플레인 유틸리티를 이용하여 확인합니다.

1 데이터베이스에 저장된 패키지 정보는 list packages 명령어를 이용하여 확인합니다.

```
>>-LIST--+-PACKAGES-+--+-----------------------------+---+------------+-->
         '-TABLES---'  |      .-USER----------------. |   '-SHOW DETAIL-'
                       '-FOR--+-ALL-----------------+-'
                              +-SCHEMA--schema-name-+
                              '-SYSTEM--------------'
```

Figure 2214A ••• LIST PACKAGES 명령어

2 list packages 명령어의 옵션은 다음과 같습니다.

옵 션	설 명
FOR USER	스키마명이 현재 세션의 사용자 ID와 동일한 패키지에 대한 정보를 표시합니다. 기본값이므로 생략할 수 있습니다.
FOR ALL	모든 패키지에 대한 정보를 표시합니다.
FOR SCHEMA <스키마명>	특정 스키마명을 가지는 패키지에 대한 정보를 표시합니다.
FOR SYSTEM	시스템 패키지에 대한 정보를 표시합니다.

3 list packages 명령어의 '유효' 컬럼에 표시되는 패키지의 상태는 다음 3가지로 구분됩니다.

상 태	설 명
Y	정상 상태의 패키지로 원하는 시점에 실행할 수 있습니다.
N	패키지가 종속되어 있는 테이블 등의 데이터베이스 오브젝트가 삭제된 패키지를 유효하지 않은(Invalid Package) 패키지라고 합니다. 이러한 패키지는 최초의 실행시에 자동적으로 리바인드됩니다.
X	패키지가 종속되어 있는 사용자 정의 함수나 저장 프로시저가 삭제되었기 때문에 사용할 수 없는 패키지를 작동 불능(Inoperative Package) 패키지라고 합니다. 이러한 패키지는 명시적으로 리바인드되어야 실행할 수 있습니다.

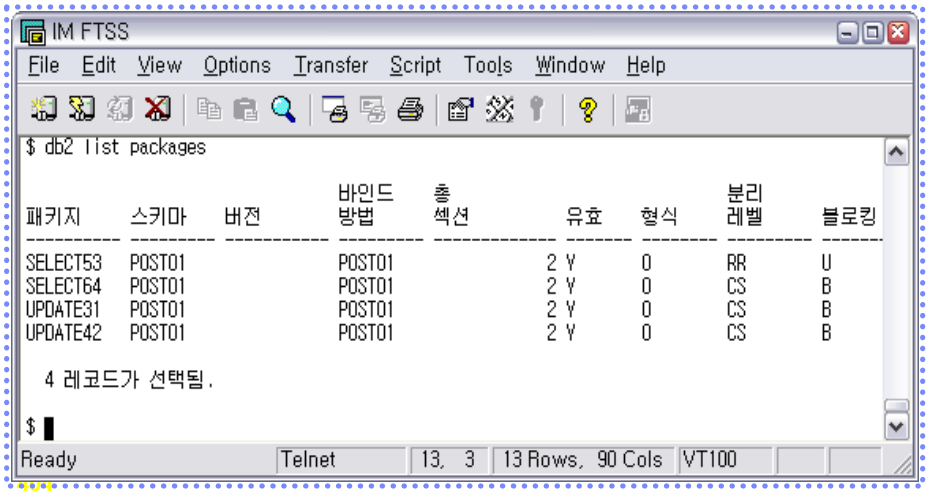

Figure 2214B ••• 패키지 정보

TOPIC 22 15 익스플레인 도구

Point

SQL 컴파일러가 정적 SQL문과 동적 SQL문에 대해 생성한 액세스 플랜을 확인하는 익스플레인 도구에는 GUI기반의 Visual Explain와 텍스트 기반의 db2exfmt, db2expln 등의 유틸리티가 있습니다.

Tip

이전 버전에서 사용하던 dynexpln 명령어 대신에 db2expln 명령어를 사용하도록 합니다.

Tip

db2expln 명령어는 매개 변수 표시 문자인 ? (물음표)를 포함한 동적 SQL문을 지원합니다.

Tip

Visual Explain 유틸리티를 최초로 실행할 때, 익스플레인 테이블이 자동적으로 생성됩니다.

Tip

실행하는 사용자의 ID를 스키마명으로 하는 익스플레인 테이블이 있어야 합니다.

1. SQL 컴파일러는 정적 또는 동적 SQL문의 액세스 플랜 및 환경에 대한 정보를 수집할 수 있습니다. 수집된 익스플레인 정보는 쿼리 처리를 위한 조작 시퀀스, 비용 정보, 각 술어에 대한 술어 및 선택성 추정치, SQL문에서 참조된 모든 오브젝트에 대한 통계 정보 등을 포함하므로, 개별 SQL 명령문의 실행 방법을 파악과 성능 향상을 위한 SQL문의 변경 또는 실행 환경 조절에 도움을 줄 수 있습니다.

2. 지원되는 익스플레인 도구와 특성은 다음과 같습니다.

특 성	Explain 테이블	Visual Explain	db2exfmt	db2expln
GUI 출력		O		
텍스트 출력			O	O
'Quick and Dirty' 정적 SQL문 분석				O
지원되는 정적 SQL	O	O	O	O
지원되는 동적 SQL	O	O	O	O
지원되는 CLI 응용프로그램	O	O	O	
DRDA Application Requester 지원	O			
상세 옵티마이저 정보	O	O	O	
다중 명령문 분석	O		O	O
응용프로그램 내에서 액세스 가능한 정보	O			

3. 익스플레인 테이블은 익스플레인 과정에서 생성되는 각종 정보를 보관하기 위한 테이블들입니다. 생성 스크립트는 인스턴스 사용자의 홈디렉토리에 있는 sqllib/misc/EXPLAIN.DDL 입니다. Visual Explain 또는 db2exfmt 유틸리티를 사용하려면, 익스플레인 테이블을 미리 생성하여야 합니다.

```
$ db2 connect to <DB명>
$ db2 -tvf $HOME/sqllib/misc/EXPLAIN.DDL
$ db2 list tables for all | grep EXPLAIN
```

4. 제어 센터에서 실행되는 Visual Explain은 GUI 방식으로 익스플레인 정보를 제공합니다.

5. db2exfmt 명령어는 익스플레인 테이블의 내용을 정해진 형식의 텍스트로 생성하는 도구입니다. 액세스 플랜 정보와 옵티마이저에 대한 정보를 확인하는데 사용됩니다.

6. db2expln 명령어는 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용됩니다. 하나 이상의 패키지에 사용 가능한 액세스 플랜 정보를 표시하며, 옵티마이저에 대한 정보는 표시하지 않습니다.

TOPIC 22-16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

기본적으로 원격 데이터베이스인 경우에는 데이터베이스 접속시에 사용자명과 암호를 제공해야 합니다.

1 제어 센터에서 원하는 데이터베이스를 선택한 후 SQL Explain 메뉴를 클릭합니다.

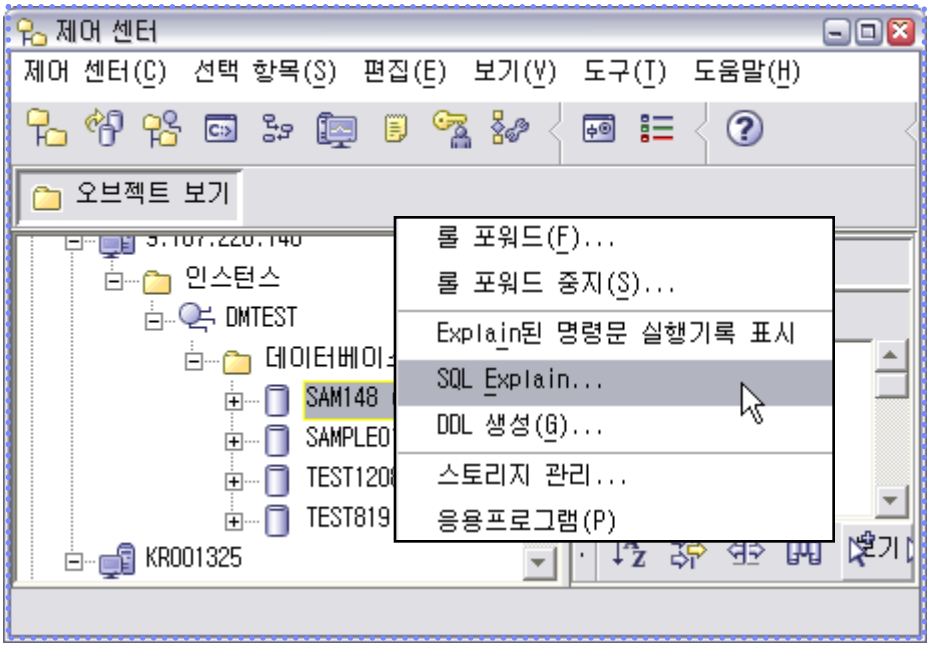

Figure 2216A ••• Visual Explain 메뉴 선택

2 동적 SQL문을 직접 입력하거나 가져오기를 눌러 동적 SQL문이 포함된 파일을 읽어옵니다. 확인 버튼을 클릭하면 익스플레인 정보가 생성됩니다. 접속한 사용자가 Visual Explain 을 최초로 실행할 때는 익스플레인 테이블이 자동적으로 생성됩니다.

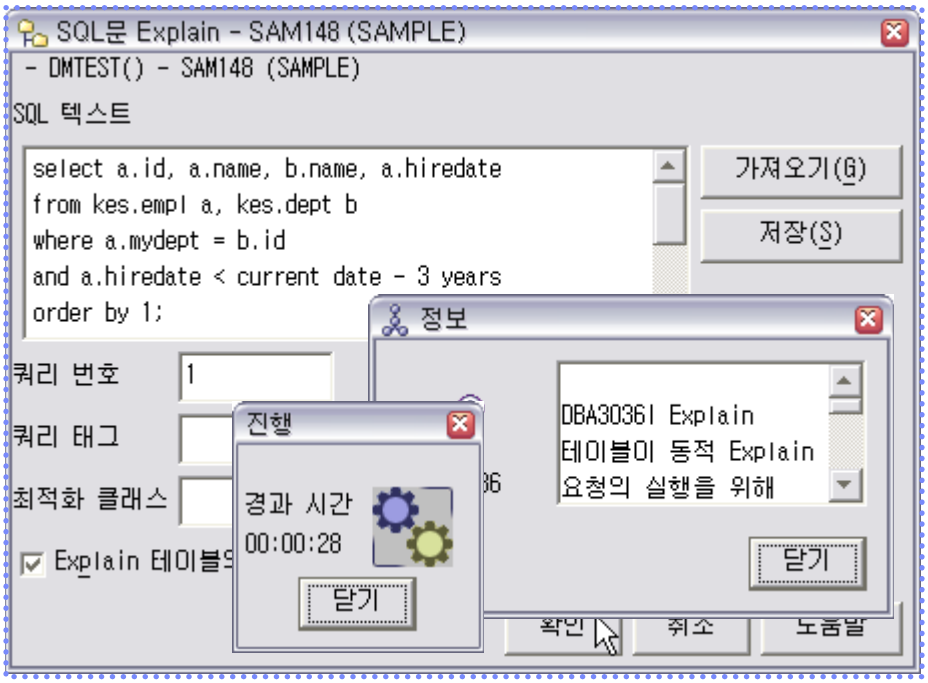

Figure 2216B ••• 동적 SQL문 입력

Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

괄호 안에 표시된 숫자는 각 조작별로 부여되는 일련 번호로, 큰 번호부터 화살표를 따라 역순으로 확인합니다.

Tip

각 다이어그램에는 조작명, 조작 일련 번호, 실행 비용 등이 표시됩니다.

3 생성된 액세스 플랜을 확인합니다.

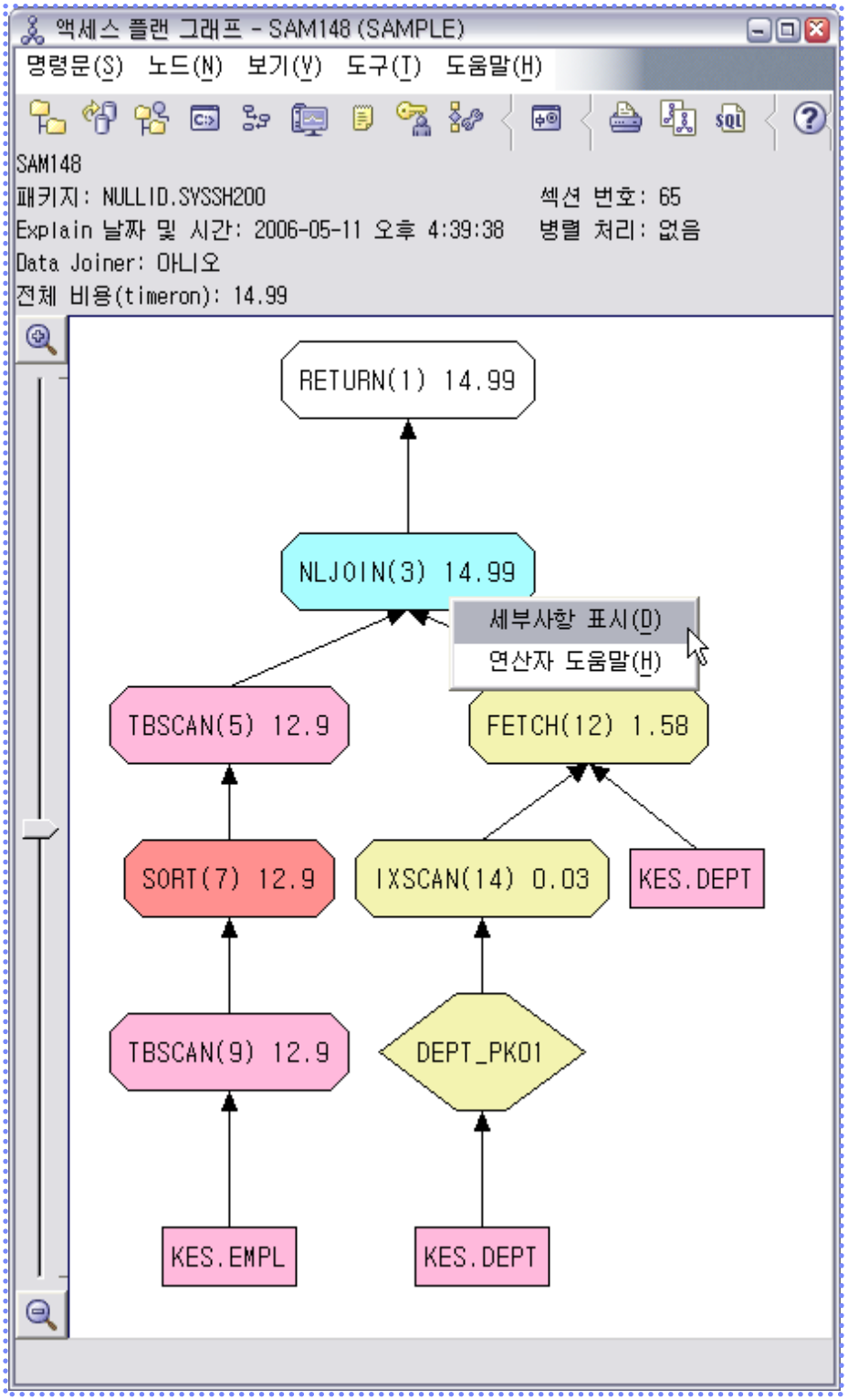

Figure 2216C ··· 액세스 플랜 그래프

TOPIC 22-16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

비용은 SQL문에 대한 액세스 플랜을 실행하는 데 필요한 자원의 추정 사용량으로 단위는 timeron 입니다.

Tip

timeron은 실제 경과 시간이 아니라, CPU 비용과 I/O 비용을 합하여 추산된 상대적인 비용입니다.

Tip

CPU 비용은 명령어의 수를 표시합니다. 현 시스템이 한 개의 명령어를 실행하는 CPU의 속도와 관련됩니다.

Tip

I/O 비용은 검색 및 전송된 페이지 수를 표시합니다. 현 시스템이 한 페이지를 처리하는 속도와 관련됩니다.

Tip

일반적으로 동일한 쿼리에 대한 여러 액세스 플랜을 비교할 때, timeron 단위의 비용이 적은 액세스 플랜이 빠른 것으로 간주하면 됩니다.

4 각 다이어그램에서 마우스의 오른쪽 버튼으로 조작별 세부 사항 정보를 확인할 수 있습니다.

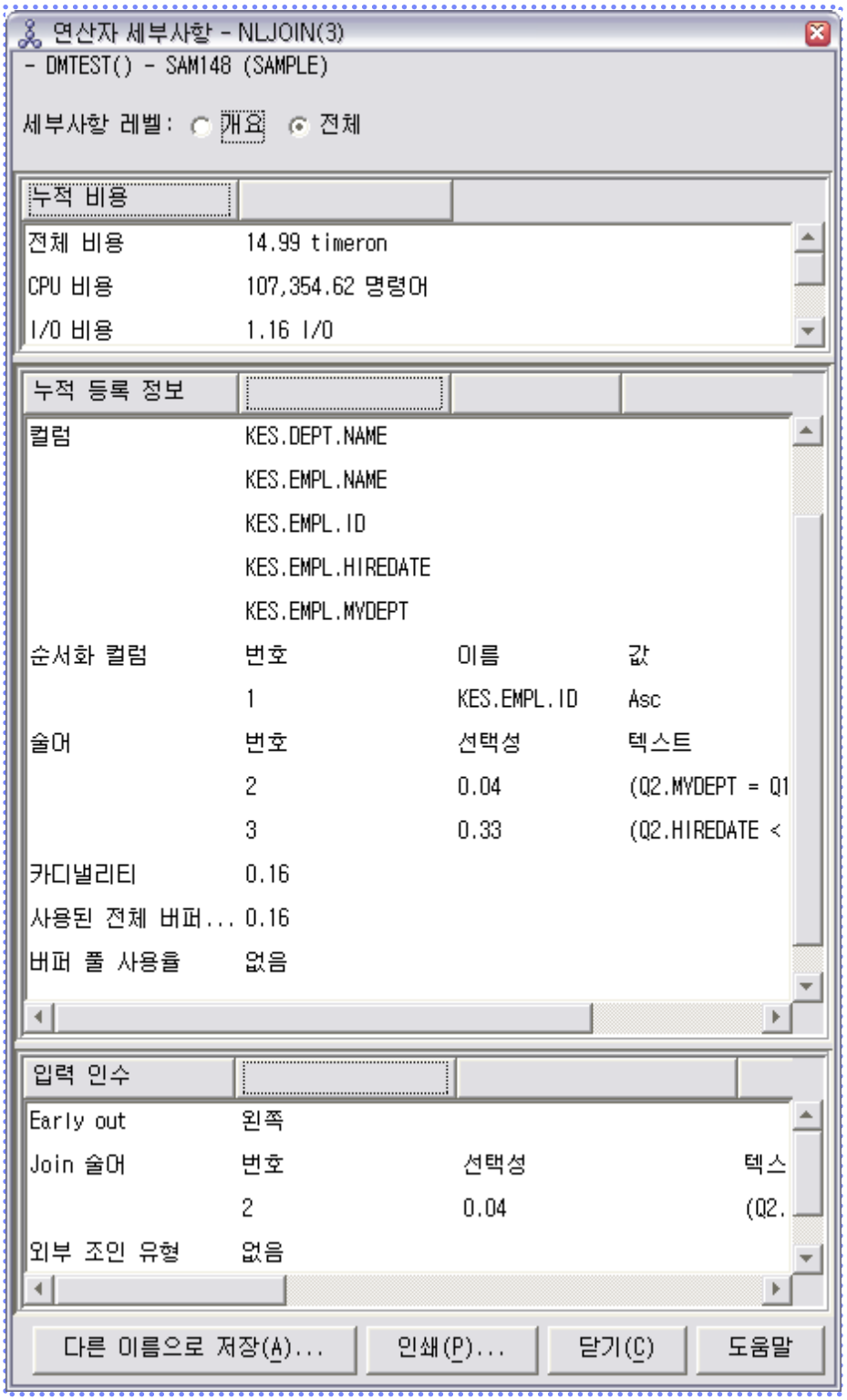

Figure 2216D ··· 연산자 세부 사항 표시

TOPIC 22 16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

연산자 유형 오른쪽 대괄호 안의 수는 각 노드에 대해 고유한 ID입니다. ID 옆의 숫자는 누적 비용을 표시합니다.

Tip

쿼리가 병렬로 처리되면, 한 개의 쿼리를 처리하는데 여러 개의 데이터베이스 에이전트가 사용됩니다.

Tip

테이블 큐가 단일 노드 내의 데이터베이스 에이전트 사이에서 데이터를 전달하는데 사용되면 로컬 테이블 큐라고 합니다. 로컬 테이블 큐는 파티션 내 병렬 처리에 사용됩니다.

Tip

테이블 큐가 서로 다른 노드의 데이터베이스 에이전트 사이에서 데이터를 전달하는데 사용되면 로컬 테이블 큐라고 하지 않으며, 파티션간의 병렬 처리에 사용됩니다.

5 액세스 플랜 그래프는 액세스 플랜을 트리 구조로 표시합니다. 트리의 각 노드에 표시되는 다이어그램의 의미는 다음과 같습니다.

다이어그램	설명
KES.DEPT	직사각형. 연산자의 대상이 되는 테이블을 표시합니다.
DEPT_PK01	다이아몬드. 연산자의 대상이 되는 인덱스를 표시합니다.
SORT (7) 12.9	팔각형. 데이터에 대해 수행되어야 하는 조치 또는 인덱스와 테이블로부터의 출력을 저장하는 연산자를 표시합니다.
TQUEUE(27)	평행사변형. 쿼리를 처리하는 에이전트가 여러 개 있는 경우, 한 개의 데이터베이스 에이전트에서 다른 데이터베이스 에이전트로 테이블의 데이터를 전달하는데 사용되는 테이블 큐를 표시합니다.
SNAPSHOT_TBS	육각형. 테이블 함수를 표시합니다.

5 주요 연산자에 대한 설명은 다음과 같습니다.

연산자	설명
DELETE	테이블에서 행을 지웁니다.
FETCH	특정 레코드 ID를 사용하여 테이블에서 컬럼을 페치합니다.
FILTER	한 개 이상의 조건으로 데이터를 필터합니다.
GRPBY	지정된 컬럼과 함수의 공통값에 따라 행을 구분하고, 집합 함수를 평가합니다.
HSJOIN	둘 이상의 테이블이 조인 컬럼에서 해시된 해시 조인을 표시합니다.
INSERT	테이블에 행을 삽입합니다.
IXAND	두 개 이상의 인덱스를 스캔하여 나온 RID에 대해 AND 작업을 수행합니다.
IXSCAN	테이블의 인덱스를 스캔하여 일련의 순서화된 행을 산출합니다.
MSJOIN	병합 조인으로 외부 및 내부 테이블이 모두 조인-술어 순서로 되어야 합니다.
NLJOIN	중첩 루프 조인으로 외부 테이블 행마다 한 번씩 내부 테이블에 액세스합니다.
RETURN	쿼리로부터 사용자에게 데이터를 리턴함을 의미합니다.
RIDSCN	한 개 이상의 인덱스로부터 얻어진 일련의 행 ID(RID)를 스캔합니다.
SORT	지정된 컬럼의 순서로 행을 정렬하며, 중복된 항목을 제거할 수 있습니다.
TBSCAN	데이터 페이지에서 직접 필요한 모든 데이터를 읽어 행을 검색합니다.
TEMP	여러 번 다시 읽을 수 있도록 데이터를 임시 테이블에 저장합니다.
TQUEUE	데이터베이스 에이전트들 사이에 테이블 데이터를 전송합니다.
UNION	여러 테이블의 행을 이어 붙입니다.
UNIQUE	특정 컬럼에 대해 중복된 값을 가진 행을 제거합니다.
UPDATE	테이블에 있는 행을 갱신합니다.

TOPIC 22-16 Visual Explain

Point

제어 센터를 이용하면 GUI 형식의 익스플레인 도구인 Visual Explain 을 이용하여 SQL문의 액세스 플랜에 대한 익스플레인 정보를 확인할 수 있습니다.

Tip

과거에 실행했던 동적 SQL문에 대한 익스플레인 정보는 실행 기록 정보를 통해서 다시 확인할 수 있습니다.

Tip

정적 SQL문에 대한 익스플레인 정보는 실행 기록 정보에서 확인할 수 있습니다. PREP 또는 BIND 명령어로 패키지를 생성할 때, EXPLSNAP 옵션이 YES 또는 ALL로 설정되어야 합니다.

6 제어 센터에서 원하는 데이터베이스를 선택한 후, Explain된 명령문 실행 기록 표시 메뉴를 선택하면 지금까지 실행했던 익스플레인 정보를 확인할 수 있습니다.

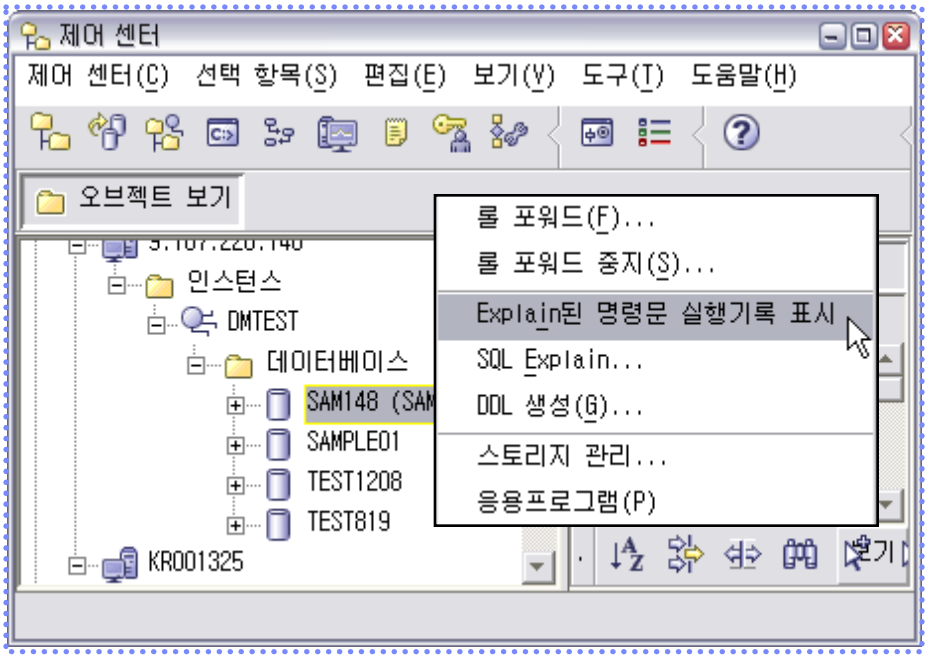

Figure 2216E ••• Explain된 명령 실행 기록 메뉴 선택

7 원하는 실행 기록 항목을 선택하여 해당하는 SQL 텍스트를 확인하거나, 기존의 액세스 플랜을 다시 표시할 수 있습니다. 쿼리 태그를 변경하거나, 실행 기록을 제거할 수 있습니다.

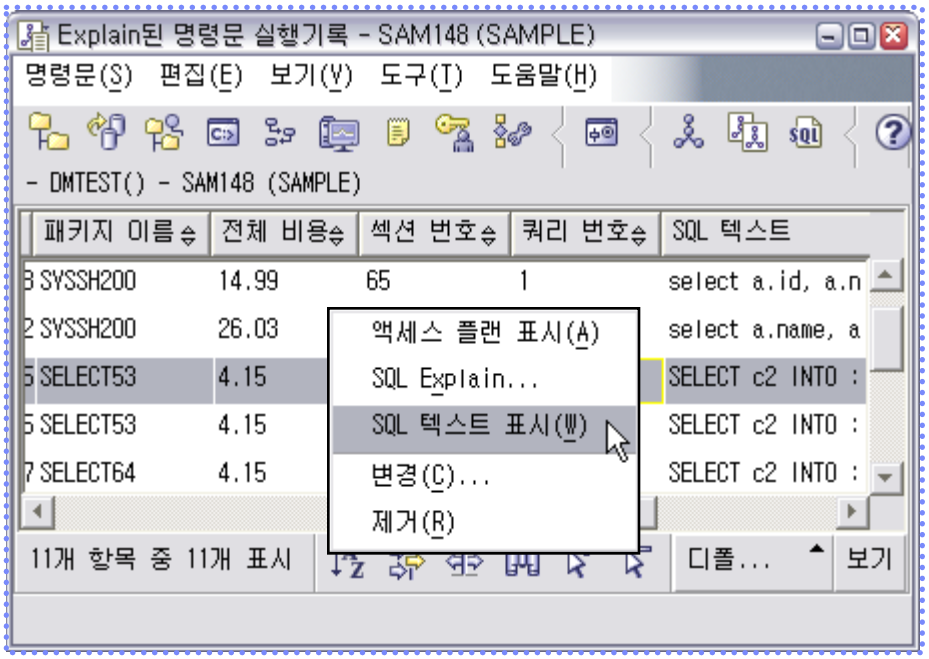

Figure 2216F ••• 동적 또는 정적 SQL문에 대한 익스플레인 실행 기록 선택

TOPIC 22 17 db2expln 유틸리티

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

dynexpln 명령어는 매개 변수 표시 문자로 사용되는 ? (물음표)를 포함한 동적 SQL문을 지원하지 않으므로, db2expln 명령어를 사용하도록 합니다.

1 db2expln 명령어를 이용하여 옵티마이저가 선택한 액세스 플랜을 표시합니다.

```
>>-db2expln--+-----------------------+--+--------------------+-------------->
             '-| connection-options |-'  '-| output-options |-'

>--+---------------------+--+---------------------+--+----------------------+->
   '-| package-options |-'  '-| dynamic-options |-'  '-| explain-options |-'
```

Connection Options:
-d <db> = Connect to the database named <db>.
-u <name> <pw> = Connect as user <name> with password <pw>.

Output Options:
-t = Send output to the terminal.
-o <file> = Write output to the file named <file>.

Package Options:
-c <pattern> = The package creator must match <pattern>.
-p <pattern> = The package name must match <pattern>.
-s <number> = The section number is <number>. Use 0 (zero) for all.

Dynamic Statement Options:
-q <sql> = The dynamic statement <statement> will be explained.
-f <file> = The dynamic statements contained in the <file> will be explained.

Explain Options:
-z <character> = Each SQL statement ends at <character>.
-g = Reconstruct the original optimizer plan graph.
-i = Show the operator ID numbers.

Figure 2217A ··· db2expln 명령어

Tip

섹션 번호에 0을 지정하면 모든 섹션에 대한 정보를 확인할 수 있습니다.

Tip

동적 SQL문을 포함하는 –q 옵션과 –f 옵션은 함께 사용할 수 없습니다.

Tip

-z 옵션을 지정할 때는 공백 문자를 입력한 후에, " (쌍따옴표) 사이에 SQL 구분 문자를 지정합니다.

Tip

출력 방향을 지정하는 -o 옵션과 –t 옵션은 함께 사용할 수 없습니다.

2 db2expln 명령어의 옵션에 대한 설명은 다음과 같습니다.

옵 션	설 명
-d <DB명>	DB명을 지정합니다.
-u <사용자명> <암호>	데이터베이스에 접속할 사용자명과 암호를 지정합니다.
-c <스키마명>	패키지의 소유자를 지정합니다.
-p <패키지명>	패키지명을 지정합니다.
-s <섹션 번호>	섹션 번호를 지정합니다.
-f <입력 파일명>	동적 SQL문이 포함된 입력 파일명을 지정합니다.
-q <SQL문>	동적 SQL문을 직접 입력합니다.
-z "<SQL문 구분자>"	SQL문의 구분자를 지정합니다. 기본은 ; (세미콜론) 입니다.
-o <출력 파일명>	출력 파일명을 지정합니다.
-t	화면으로 익스플레인 정보를 표시합니다.
-g -i	텍스트 형식의 트리 형태의 액세스 플랜을 포함시킵니다.

TOPIC 22 18 동적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

-q 옵션은 한 개의 동적 SQL문을 지정할 수 있습니다.

Tip

-f 옵션은 한 개 이상의 동적 SQL문을 지정할 수 있습니다. SQL문이 한 개 이상이거나, 한 줄 이상에 걸쳐서 표현될 때에는 반드시 –z 옵션을 이용하여 SQL 구분자를 지정해야 합니다. 일반적으로, SQL문의 구분자로 기본값인 ; (세미 콜론)을 사용합니다.

Tip

입력된 동적 SQL문에 대한 분리 수준, 블럭킹 옵션, 병렬 처리 여부 등의 정보가 표시됩니다.

Tip

입력된 동적 SQL문과 실행 비용, 액세스 플랜 등의 익스플레인 정보가 표시됩니다.

3 db2expln 명령어에서 –q 옵션을 이용하여 동적 SQL문을 직접 입력하면, 해당하는 동적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ db2expln –d <데이터베이스명> -q <동적 SQL문> -t -g -i
$ db2expln –d <데이터베이스명> -q <동적 SQL문> -o <출력 파일명> -g -i
```

4 db2expln 명령어에서 –f 옵션을 이용하여 동적 SQL문이 저장된 파일명을 입력하면, 해당하는 동적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ cat <입력 파일명>
<SQL문>;
$ db2expln –d <데이터베이스명> -f <입력 파일명> -z ";" -t -g -i
$ db2expln –d <데이터베이스명> -f <입력 파일명> -z ";" -o <출력 파일명> -g -i
```

5 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2expln -d sample -q "select * from kes.empl where id=1" -t -g

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

******************** DYNAMIC ****************************************

============== STATEMENT ============================

      Isolation Level                = Cursor Stability
      Blocking                       = Block Unambiguous Cursors
      Query Optimization Class       = 5
      Partition Parallel             = No
      Intra-Partition Parallel       = No
      SQL Path  = "SYSIBM", "SYSFUN", "SYSPROC", "POST01"

SQL Statement:

  select name
  from kes.empl
  where id =1

<다음 페이지에 계속됨>
```

TOPIC 22 18 동적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

DPF 환경에서는 데이터베이스 파티션간의 정보 이동이 포함되므로, 동일한 쿼리에 대한 익스플레인 정보가 더 복잡하게 표시될 수 있습니다. 검색되는 데이터에 대해 list prefetch 기법이 선택된다면, 다음과 같은 액세스 플랜이 표시될 수도 있습니다.

```
      RETURN
       ( 1)
        |
       BTQ
       ( 2)
        |
      FETCH
      (----)
     /     \
   RIDSCN  Table:
   ( 4)    KES
    |      EMPL
   SORT
   ( 5)
    |
   IXSCAN
   ( 6)
  /          \
Index:       Table:
KES          KES
EMPL_PK01    EMPL
```

6 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
<이전 페이지에서 계속됨>

Estimated Cost = 12.869981
Estimated Cardinality = 1.000000

(   2) Access Table Name = KES.EMPL  ID = 2,15
        |  Index Scan:  Name = KES.EMPL_PK01  ID = 1
        |  |  Regular Index (Not Clustered)
        |  |  Index Columns:
        |  |  |  1: ID (Ascending)
        |  #Columns = 7
        |  Single Record
        |  Fully Qualified Unique Key
        |  #Key Columns = 1
        |  |  Start Key: Inclusive Value
        |  |  |  |  1: 1
        |  |  Stop Key: Inclusive Value
        |  |  |  |  1: 1
        |  Data Prefetch: Eligible 0
        |  Index Prefetch: None
        |  Lock Intents
        |  |  Table: Intent Share
        |  |  Row  : Next Key Share
        |  Sargable Predicate(s)
(   2) |  |  Return Data to Application
        |  |  |  #Columns = 7
(   1) Return Data Completion

End of section


Optimizer Plan:

      RETURN
      ( 1)
        |
      FETCH
      ( 2)
     /     \
  IXSCAN   Table:
  ( 2)     KES
    |      EMPL
 Index:
 KES
 EMPL_PK01
```

TOPIC 22-19 정적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

일반적으로, 섹션 번호에 0을 입력하여 패키지에 포함된 모든 섹션에 대한 익스플레인 정보를 확인합니다.

7 정적 SQL문은 이미 컴파일되어 시스템 카탈로그에 패키지로 저장되어 있습니다. db2expln 명령어에서 –p와 –s옵션을 이용하여 정적 SQL문이 포함된 패키지명과 섹션 번호를 입력하면, 해당하는 정적 SQL문에 대한 익스플레인 정보를 확인할 수 있습니다.

```
$ db2expln –d <데이터베이스명> -c <스키마명> -p <패키지명> -s <섹션번호> -t
```

8 정적 SQL문에 대한 익스플레인을 확인하기 위해 ESQL 소스 파일을 작성합니다. 소스 파일명은 select64.sqc 로 하겠습니다.

```
#include <stdio.h>

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
        char c2[30+1];
EXEC SQL END DECLARE SECTION;

int main()
{
        EXEC SQL CONNECT TO sample;

        EXEC SQL    SELECT name
                    INTO :name
                    FROM kes.empl
                    WHERE id = 1;

        printf("name = %s\n",name);

        EXEC SQL CONNECT RESET;

    exit(0);
}
```

Tip

익스플레인 정보만 간단하게 확인하려면, BIND 파일을 생성하지 않고, PREP 명령어로 패키지를 바로 생성합니다.

9 PREP 명령어를 이용하여 패키지를 생성합니다.

```
$ db2 connect to sample
$ db2 prep select64.sqc
```

10 패키지를 생성하는 사용자명이 post01 이라면, 패키지명은 POST01.SELECT64 이 됩니다. list packages 명령어로 패키지에 관련된 정보를 확인합니다.

```
$ db2 list packages
                                 바인드    총                          분리
패키지      스키마    버전       방법      섹션    유효   형식   레벨   블로킹
-----------------------------------------------------------------------------------
SELECT64    POST01             POST01      1     Y      0      CS     U

1 레코드가 선택됨.
```

TOPIC 22 19 정적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

패키지명을 비롯하여 분리 수준, 블럭킹 옵션, 병렬 처리 여부 등의 패키지 정보가 표시됩니다.

Tip

패키지에 포함된 섹션별로 SQL문과 실행 비용, 액세스 플랜 등의 익스플레인 정보가 표시됩니다.

11 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2expln -d sample -c post01 -p select64 -s 0 -t -g

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

******************** PACKAGE ****************************************

Package Name = "POST01"."SELECT64"  Version = ""

      Prep Date = 2006/05/22
      Prep Time = 14:05:46

      Bind Timestamp = 2006-05-22-14.05.46.565000

      Isolation Level                = Cursor Stability
      Blocking                       = Block Unambiguous Cursors
      Query Optimization Class       = 5

      Partition Parallel             = No
      Intra-Partition Parallel       = No

      SQL Path  = "SYSIBM", "SYSFUN", "SYSPROC", "POST01"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT name INTO :H00002
  FROM kes.empl
  WHERE id =1


Section Code Page = 970

Estimated Cost = 12.869981
Estimated Cardinality = 1.000000

<다음 페이지에 계속됨>
```

TOPIC 22-19 정적 SQL문에 대한 db2expln 출력

Point

패키지에 포함된 정적 SQL문 또는 동적 SQL문에 대한 간단한 액세스 플랜과 쿼리의 실행 비용을 확인하는 데 주로 사용되는 명령어로 옵티마이저에 대한 정보는 표시하지 않습니다.

Tip

DPF 환경에서는 데이터베이스 파티션간의 정보 이동이 포함되므로, 동일한 쿼리에 대한 익스플레인 정보가 더 복잡하게 표시될 수 있습니다.

```
    RETURN
     ( 1)
      |
     BTQ
     ( 2)
     |
     FETCH
     ( 3)
    /    \
  IXSCAN  Table:
  ( 3)    KES
   |      EMPL
Index:
KES
EMPL_PK01
```

12 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Access Table Name = KES.EMPL  ID = 2,15
|  Index Scan:  Name = KES.EMPL_PK01  ID = 1
|  |  Regular Index (Not Clustered)
|  |  Index Columns:
|  |  |  1: ID (Ascending)
|  #Columns = 1
|  Single Record
|  Fully Qualified Unique Key
|  #Key Columns = 1
|  |  Start Key: Inclusive Value
|  |  |  |  1: 1
|  |  Stop Key: Inclusive Value
|  |  |  |  1: 1
|  Data Prefetch: Eligible 0
|  Index Prefetch: None
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Sargable Predicate(s)
|  |  Return Data to Application
|  |  |  #Columns = 1
Return Data Completion

End of section


Optimizer Plan:

    RETURN
    ( 1)
     |
    FETCH
    ( 2)
   /    \
 IXSCAN  Table:
 ( 2)    KES
  |      EMPL
Index:
KES
EMPL_PK01
```

TOPIC 22 20 db2exfmt 유틸리티

Point

익스플레인 테이블의 내용을 정해진 형식의 텍스트로 생성하는 도구입니다. 정적 SQL문과 동적 SQL문에 대한 액세스 플랜 정보와 옵티마이저에 대한 정보를 확인하는데 사용됩니다.

Tip

실행하는 사용자의 ID를 스키마명으로 하는 익스플레인 테이블이 있어야 합니다.

Tip

-e 옵션을 이용하여 익스플레인 테이블의 스키마명을 지정하지 않으면, DB에 접속한 사용자 ID가 스키마명으로 이용됩니다.

Tip

-g 옵션의 세부 옵션 설명입니다.
O – 그래프만 생성
T – 오퍼레이션별 총비용을 표시
I – I/O 비용을 표시
C – 예측되는 데이터 건수를 표시

Tip

-n 옵션과 –s 옵션에서는 %, * 등의 wild charcter를 이용하여 표시할 수 있습니다.

Tip

출력 방향을 지정하는 -o 옵션과 –t 옵션은 함께 사용할 수 없습니다.

Tip

-# 옵션에서 0을 지정하면, 모든 섹션에 대한 정보를 표시합니다.

1 필요시 <인스턴스 사용자의 홈디렉토리>/sqllib/misc/EXPLAIN.DDL 을 실행하여 익스플레인 테이블을 생성합니다.

```
$ db2 connect to <DB명>
$ db2 -tvf $HOME/sqllib/misc/EXPLAIN.DDL
```

2 db2exfmt 명령어를 이용하여 익스플레인 테이블에 저장된 정보를 파일로 저장합니다.

```
>>-db2exfmt--+-------------+--+-------------+--+--------+------------------>
             '---d--dbname-'  '---e--schema-'  '---f--O-'

>--+----------------+--+-----+--+-----------+--+-------------+------------->
   |      .-------. |  '---l-'  '---n--name-'  '---s--schema-'
   |      V       | |
   '---g----+---+-+-'
            +-O-+
            +-T-+
            +-I-+
            '-C-'

>--+-------------+--+---------------------+--+---------------+------------>
   +--o--outfile-+  '--u--userID--password-'  '--w--timestamp-'
   | .--t-.      |
   '-+----+------'

>--+--------------+--+----+--------------------------------------------><
   '--#--sectnbr--'  '--h-'
```

Figure 2220A • • • db2exfmt 명령어

3 db2exfmt 명령어의 옵션에 대한 설명은 다음과 같습니다.

옵션	설명
-d <DB명>	패키지가 저장된 데이터베이스명을 지정합니다.
-e <스키마명>	익스플레인 테이블의 스키마명을 지정합니다.
-g OTIC	액세스 플랜을 트리 형식의 텍스트 그래프로 표시합니다.
-1	영문자 L 의 소문자로 패키지명의 대소문자를 무시합니다.
-n %	익스플레인의 대상이 되는 오브젝트명을 지정합니다. %를 지정하면, 모든 오브젝트를 대상으로 합니다.
-s %	익스플레인의 대상이 되는 스키마명을 지정합니다. %를 지정하면, 모든 스키마를 대상으로 합니다.
-o <출력 파일명>	익스플레인 정보를 저장할 출력 파일명을 지정합니다.
-t	익스플레인 정보를 화면으로 표시합니다.
-u <사용자명> <암호>	데이터베이스에 접속할 사용자명과 암호를 지정합니다.
-w	익스플레인된 시간 소인을 지정합니다. -1(숫자 1)을 지정하면, 최근에 실행된 SQL문에 대한 익스플레인 정보를 표시합니다.
-# <섹션 번호>	섹션 번호를 지정합니다.

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

Tip

PREP 또는 BIND 명령어에서 지정한 EXPLAIN 옵션의 값이 ALL이고, CURRENT EXPLAIN MODE 레지스터 값이 NO이면, 실행시에 동적 SQL문의 익스플레인 정보를 캡쳐하여 저장합니다.

Tip

CURRENT EXPLAIN MODE 레지스터의 값이 NO가 아닌 경우에는 PREP 또는 BIND 명령어에서 지정한 EXPLAIN 옵션 값은 무시됩니다.

1 동적 SQL문에 대한 익스플레인 정보를 확인하려면, SQL문을 실행하기 전에 특수 레지스터인 CURRENT EXPLAIN MODE를 YES 또는 EXPLAIN 으로 설정합니다.

```
$ db2 set CURRENT EXPLAIN MODE <YES 또는 EXPLAIN>
$ db2 "SQL문"
```

2 CURRENT EXPLAIN MODE에 사용될 수 있는 특수 레지스터리의 값은 다음과 같습니다.

옵션	설명
NO	Explain 기능을 작동 불가능으로 설정합니다. 익스플레인 정보를 캡처하지 않으며, 기본값으로 사용됩니다.
YES	Explain 기능을 작동 가능으로 설정합니다. 동적 SQL문에 대한 익스플레인 정보를 캡처하여 익스플레인 테이블에 삽입하며, 동적 SQL문이 컴파일되고 실행됩니다.
EXPLAIN	Explain 기능을 작동 가능으로 설정합니다. 동적 SQL문에 대한 익스플레인 정보를 캡처하여 저장하며, 동적 SQL문 중에서 DML은 실행되지 않습니다.
REOPT	Explain 기능을 작동 가능으로 설정합니다. 실행 시간에 명령문 재최적화 되는 경우 (호스트 변수, 특수 레지스터, 매개 변수 표시 문자에 대한 실제 값이 사용 가능할 때)에 정적 또는 동적 SQL문에 대한 익스플레인 정보가 캡처되도록 합니다. SQL문이 실행 시간에 재최적화 되지 않을 경우 (명령문에 입력 변수가 없거나 REOPT 바인드 옵션이 NONE으로 설정된 경우)에 익스플레인 정보는 캡처되지 않습니다. REOPT 바인드 옵션을 ONCE로 설정하면, 익스플레인 정보는 명령문이 초기에 재최적화될 때 한 번만 캡처됩니다.
RECOMMEND INDEXES	SQL 컴파일러가 인덱스를 권장할 수 있게 합니다. 실행되는 쿼리는 ADVISE_INDEX 테이블에 권장되는 인덱스를 추가하고, 인덱스 사용에 대한 설명 정보를 캡처하여 설명 테이블에 저장합니다. 명령문은 컴파일되지도 않고, 실행되지도 않습니다.
EVALUATE INDEXES	SQL 컴파일러가 인덱스를 평가할 수 있게 합니다. 평가될 인덱스를 ADVISE_INDEX 테이블로부터 읽어 EVALUATE = Y로 표시해야 합니다. 옵티마이저는 카탈로그로부터의 값에 기초하여 가상 인덱스를 생성합니다. 실행되는 쿼리는 가상 인덱스에 기초하여 예측된 통계를 사용하여 컴파일되고 최적화됩니다. 명령문은 실행되지 않습니다.
RECOMMEND PARTITIONINGS	컴파일러가 특정 쿼리에 의해 액세스되는 각 테이블에 대해 가장 좋은 파티션을 권장하도록 지정합니다. 결과는 ADVISE_PARTITION 테이블에 저장되며, 쿼리는 실행되지 않습니다.
EVALUATE PARTITIONINGS	컴파일러가 ADVISE_PARTITION 테이블에 지정된 가상 파티션을 사용하여 쿼리의 예상 성능을 평가하도록 합니다.

3 CURRENT EXPLAIN MODE 의 값을 EXPLAIN 으로 설정하여 동적 SQL문을 실제로 실행하지 않고, 익스플레인 정보만 확인할 수 있습니다.

```
$ db2 set current explain mode EXPLAIN
```

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

Tip

- 익스플레인 정보를 확인한 후에는 반드시 CURRENT EXPLAIN MODE 레지스터의 값을 NO 또는 YES로 설정되어야 SQL문을 실행할 수 있습니다.

4 익스플레인을 원하는 동적 SQL문을 실행합니다.

```
$ db2 "select name from kes.empl where id = 1"
SQL0217W  Explain 정보 요청만 처리 중이므로 명령문이 실행되지 않았습니다.
SQLSTATE=01604
```

5 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2exfmt -d sample -g TIC -w -1 -n % -s % -# 0 –t

DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM
Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

******************** EXPLAIN INSTANCE ********************

DB2_VERSION:                    08.02.3
SOURCE_NAME:                    SQLC2E06
SOURCE_SCHEMA:                  NULLID
SOURCE_VERSION:
EXPLAIN_TIME:                   2006-05-22-16.29.01.594001
EXPLAIN_REQUESTER:              KR001325

Database Context:
----------------
        Parallelism:            None
        CPU Speed:              3.778754e-007
        Comm Speed:             0
        Buffer Pool size:       250
        Sort Heap size:         256
        Database Heap size:     600
        Lock List size:         50
        Maximum Lock List:      22
        Average Applications:   1
        Locks Available:        1122

Package Context:
---------------
        SQL Type:               Dynamic
        Optimization Level:     5
        Blocking:               Block All Cursors
        Isolation Level:        Cursor Stability

---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:        33

<다음 페이지에 계속됨>
```

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

6 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
        QUERYTAG:         CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     1

Original Statement:
------------------
select name
from kes.empl
where id=1

Optimized Statement:
-------------------
SELECT Q1."NAME" AS "NAME"
FROM KES.EMPL AS Q1
WHERE (Q1."ID" = 1)

Access Plan:
-----------
        Total Cost:       12.87
        Query Degree:     1

          Rows
         RETURN
         (   1)
          Cost
           I/O
            |
            1
         FETCH
         (   2)
          12.87
            1
        /----+---\
      1           4
   IXSCAN    TABLE: KES
   (   3)              EMPL
  0.0155141
      0
      |
      4
 INDEX: KES
          EMPL_PK01

<다음 페이지에 계속됨>
```

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

7 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Extended Diagnostic Information:
--------------------------------
Diagnostic Identifier:     1
Diagnostic Details: EXP0022W  Index has no statistics.
The index "KES "."EMPL_PK01" has not had runstats run on it. This can
lead to poor cardinality and predicate filtering estimates.
Diagnostic Identifier:     2
Diagnostic Details: EXP0020W  Table has no statistics.  The table "KES
"."EMPL" has not had runstats run on it. This can lead to poor
cardinality and predicate filtering estimates.

Plan Details:
-------------
        1) RETURN: (Return Result)
                Cumulative Total Cost:          12.87
                Cumulative CPU Cost:            52876
                Cumulative I/O Cost:            1
                Cumulative Re-Total Cost:       0.00280195
                Cumulative Re-CPU Cost:         7415
                Cumulative Re-I/O Cost:         0
                Cumulative First Row Cost:      12.8696
                Estimated Bufferpool Buffers:   2

                Arguments:
                ---------
                BLDLEVEL: (Build level)   DB2 v8.1.10.812 : s050811
                STMTHEAP: (Statement heap size)    2048

                Input Streams:
                -------------
                        4) From Operator #2
                                Estimated number of rows:     1
                                Number of columns:            1
                                Subquery predicate ID: Not Applicable
                                Column Names:
                                ------------
                                +Q2."NAME"

        2) FETCH : (Fetch)
                Cumulative Total Cost:          12.87
                Cumulative CPU Cost:            52876
                Cumulative I/O Cost:            1
                Cumulative Re-Total Cost:       0.00280195
                Cumulative Re-CPU Cost:         7415
```

<다음 페이지에 계속됨>

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

8 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
Cumulative Re-I/O Cost:        0
Cumulative First Row Cost:     12.8696
Estimated Bufferpool Buffers:  2

Arguments:
---------
MAXPAGES: (Maximum pages for prefetch)
        ALL
MAXPAGES: (Maximum pages for prefetch)
        ALL
PREFETCH: (Type of Prefetch)
        NONE
ROWLOCK : (Row Lock intent)
        NEXT KEY SHARE
TABLOCK : (Table Lock intent)
        INTENT SHARE
TBISOLVL: (Table access Isolation Level)
        CURSOR STABILITY

Input Streams:
-------------
        2) From Operator #3
                Estimated number of rows:     1
                Number of columns:            0
                Subquery predicate ID: Not Applicable

        3) From Object KES.EMPL
                Estimated number of rows:     4
                Number of columns:            1
                Subquery predicate ID: Not Applicable
                Column Names:
                ------------
                +Q1."NAME"

Output Streams:
--------------
        4) To Operator #1

                Estimated number of rows:     1
                Number of columns:            1
                Subquery predicate ID: Not Applicable
                Column Names:
                ------------
                +Q2."NAME"
```

<다음 페이지에 계속됨>

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

9 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
3) IXSCAN: (Index Scan)
        Cumulative Total Cost:          0.0155141
        Cumulative CPU Cost:            41056
        Cumulative I/O Cost:            0
        Cumulative Re-Total Cost:       0.00211421
        Cumulative Re-CPU Cost:         5595
        Cumulative Re-I/O Cost:         0
        Cumulative First Row Cost:      0.0155141
        Estimated Bufferpool Buffers:   1

        Arguments:
        ---------
        MAXPAGES: (Maximum pages for prefetch)
                ALL
        PREFETCH: (Type of Prefetch)
                NONE
        ROWLOCK : (Row Lock intent)
                NEXT KEY SHARE
        SCANDIR : (Scan Direction)
                FORWARD
        TABLOCK : (Table Lock intent)
                INTENT SHARE

        Predicates:
        ----------
        2) Start Key Predicate
                Relational Operator:            Equal (=)
                Subquery Input Required:        No
                Filter Factor:                  0.25

                Predicate Text:
                --------------
                (Q1."ID" = 1)

        2) Stop Key Predicate
                Relational Operator:            Equal (=)
                Subquery Input Required:        No
                Filter Factor:                  0.25

                Predicate Text:
                --------------
                (Q1."ID" = 1)
```

<다음 페이지에 계속됨>

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

10 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
                Input Streams:
                --------------
                        1) From Object KES.EMPL_PK01

                                Estimated number of rows:    4
                                Number of columns:           2
                                Subquery predicate ID: Not Applicable

                                Column Names:
                                ------------
                                +Q1.$RID$+Q1."ID"


                Output Streams:
                ---------------
                        2) To Operator #2

                                Estimated number of rows:    1
                                Number of columns:           0
                                Subquery predicate ID: Not Applicable


        Objects Used in Access Plan:
        ---------------------------

                Schema: KES
                Name:       EMPL_PK01
                Type:       Index
                            Time of creation:  2006-05-22-14.02.58.864000
                            Last statistics update:
                            Number of columns:                  1
                            Number of rows:                     4
                            Width of rows:                      -1
                            Number of buffer pool pages:        1
                            Distinct row values:                Yes
                            Tablespace name:                    USERSPACE1
                            Tablespace overhead:                12.670000
                            Tablespace transfer rate:           0.180000
                            Source for statistics:              Single Node
                            Prefetch page count:                32
                            Container extent page count:        32
                            Index clustering statistic:         20.000000
                            Index leaf pages:                   1
                            Index tree levels:                  1

<다음 페이지에 계속됨>
```

TOPIC 22 21 동적 SQL문에 대한 db2exfmt 출력

Point

특수 레지스터리인 CURRENT EXPLAIN MODE 값을 설정하여 동적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

11 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
                Index full key cardinality:          4
                Index first key cardinality:         4
                Index first 2 keys cardinality:      -1
                Index first 3 keys cardinality:      -1
                Index first 4 keys cardinality:      -1
                Index sequential pages:              1
                Index page density:                  100
                Index avg sequential pages:          -1
                Index avg gap between sequences:     -1
                Index avg random pages:              -1
                Fetch avg sequential pages:          -1
                Fetch avg gap between sequences:     -1
                Fetch avg random pages:              -1
                Index RID count:                     0
                Index deleted RID count:             0
                Index empty leaf pages:              0
                Base Table Schema:                   KES
                Base Table Name:                     EMPL
                Columns in index:                    ID

Schema: KES
Name:           EMPL
Type:           Table
                Time of creation:  2006-05-22-14.02.58.804001
                Last statistics update:
                Number of columns:                   7
                Number of rows:                      4
                Width of rows:                       37
                Number of buffer pool pages:         1
                Distinct row values:                 No
                Tablespace name:                     USERSPACE1
                Tablespace overhead:                 12.670000
                Tablespace transfer rate:            0.180000
                Source for statistics:               Single Node
                Prefetch page count:                 32
                Container extent page count:         32
                Table overflow record count:         0
                Table Active Blocks:                 -1
```

TOPIC 22 22 정적 SQL문에 대한 db2exfmt 출력

Point

PREP 또는 BIND 명령의 EXPLAIN 옵션을 이용하여 정적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하고, db2exfmt 명령어로 확인할 수 있습니다.

1 정적 SQL문에 대한 익스플레인 정보를 익스플레인 테이블에 저장하려면, PREP 또는 BIND 명령어로 패키지를 생성할 때, EXPLAIN 옵션을 YES 또는 ALL 으로 설정합니다.

```
$ db2 prep <ESQL 소스파일명> EXPLAIN <YES 또는 ALL>
$ db2 bind <바인드 파일명> EXPLAIN <YES 또는 ALL>
```

Tip

EXPLAIN 세부 옵션으로 사용되는 ALL과 REOPT 옵션은 Windows와 UNIX용 DB2에서만 지원됩니다.

2 PREP 또는 BIND 명령어의 EXPLAIN 옵션의 세부 옵션은 다음과 같습니다.

옵션	설명
NO	익스플레인 정보를 캡처하지 않습니다.
YES	정적 SQL문에 대한 익스플레인 정보를 저장합니다.
ALL	정적 SQL문과 동적 SQL문에 대한 익스플레인 정보를 저장합니다.
REOPT	실행 시간에 명령문 재최적화 되는 경우 (호스트 변수, 특수 레지스터, 매개 변수 표시 문자에 대한 실제 값이 사용 가능할 때)에 정적 또는 동적 SQL문에 대한 익스플레인 정보가 캡처되도록 합니다.

3 PREP 명령어에서 EXPLAIN 옵션을 YES 로 설정하여 패키지를 생성하면, 정적 SQL문에 대한 익스플레인 정보가 익스플레인 테이블에 저장됩니다.

```
$ db2 connect to sample
$ db2 prep select64.sqc EXPLAIN YES
```

Tip

헤더 부분을 제외하면, 정적 SQL문에 대한 익스플레인 예제와 결과가 동일합니다.

4 실제 익스플레인 정보의 출력 예는 다음과 같습니다.

```
$ db2exfmt  -d sample -g TIC -w -1 -n % -s % -# 0 –t
DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp.
    1991, 2002
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

******************** EXPLAIN INSTANCE ********************
DB2_VERSION:                    08.02.3
SOURCE_NAME:                    SELECT64
SOURCE_SCHEMA:                  KR001325
SOURCE_VERSION:
EXPLAIN_TIME:                   2006-05-22-17.13.15.731000
EXPLAIN_REQUESTER:              KR001325

Database Context:
----------------
<생략>
```