

데이터베이스 JOIN의 모든 것

기본 개념부터 성능 최적화까지

박창희

1. DB JOIN이란 무엇인가?

정의

관계형 데이터베이스(RDBMS)에서 두 개 이상의 테이블을 **공통된 컬럼(키)**을 기준으로 연결하여, 하나의 테이블처럼 데이터를 조회하는 연산입니다.

목적

데이터 중복을 피하기 위해 정규화(Normalization) 과정을 거쳐 분리된 테이블들을 다시 결합하여 의미 있는 정보 집합을 만드는데 사용됩니다.

핵심 키워드



관계형 데이터베이스 (RDBMS)



테이블 연결 (데이터 결합)

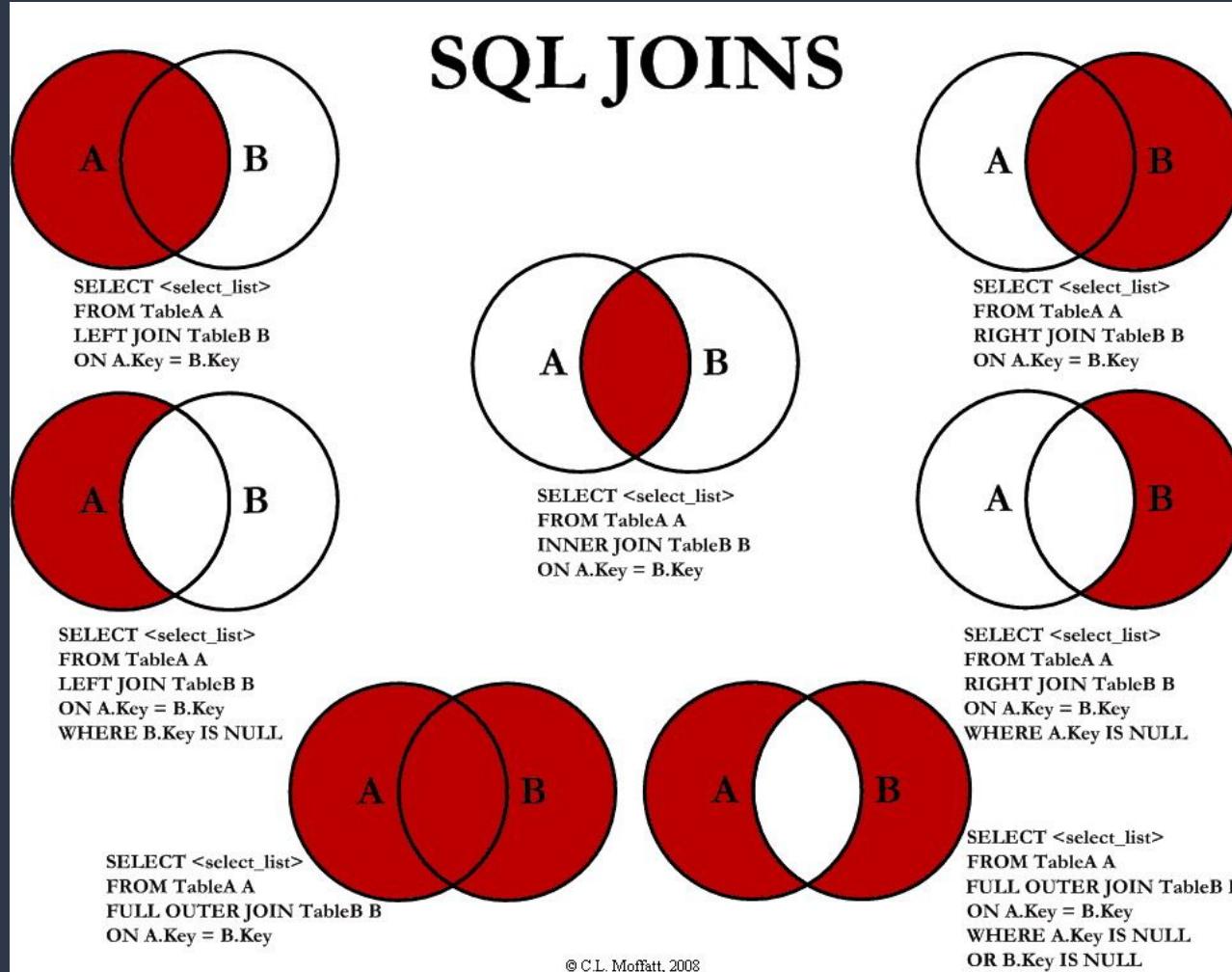


공통 컬럼 (Join Key)



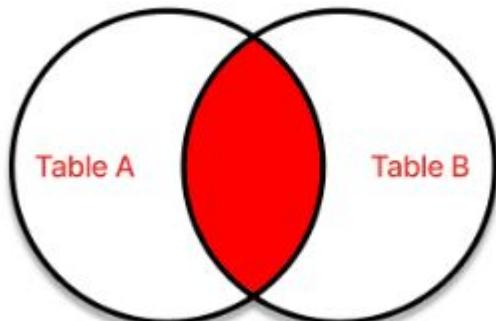
결과 집합 (Result Set)

2. JOIN 형태



3. JOIN의 기본 종류 (1/2)

INNER JOIN

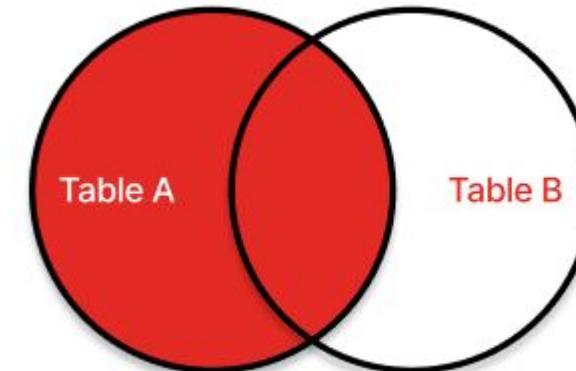


```
SELECT * FROM A INNER JOIN B ON A.key = B.key;
```

INNER JOIN (내부 조인)

두 테이블에 공통으로 존재하는 값들만 결합합니다. 즉, 조인 조건에 매칭되는 행들만 결과에 포함됩니다.

LEFT JOIN

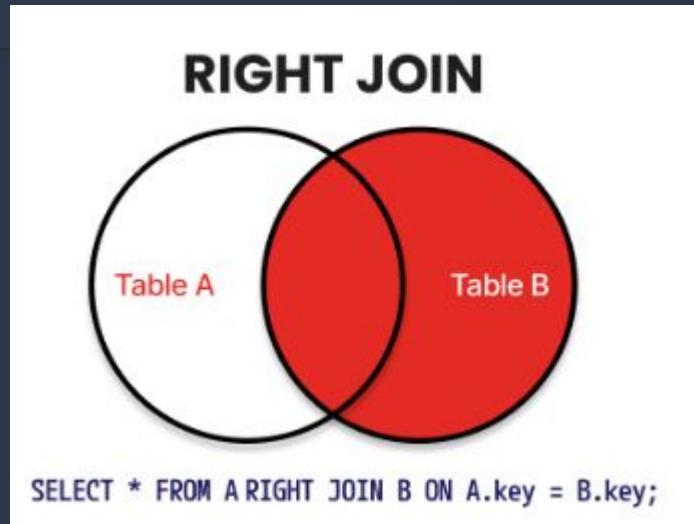


```
SELECT * FROM A LEFT JOIN B ON A.key = B.key;
```

LEFT (OUTER) JOIN (왼쪽 외부 조인)

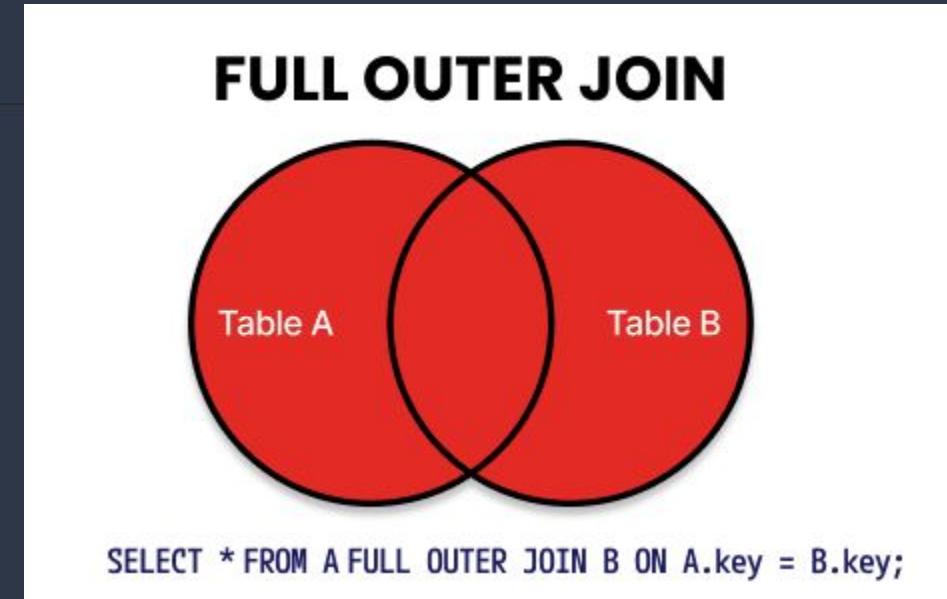
왼쪽 테이블의 모든 행을 우선 포함합니다. 오른쪽 테이블은 조인 조건에 매칭되는 행만 결합하고, 매칭되는 데이터가 없으면 NULL로 표시됩니다.

3. JOIN의 기본 종류 (2/2)



RIGHT (OUTER) JOIN (오른쪽 외부 조인)

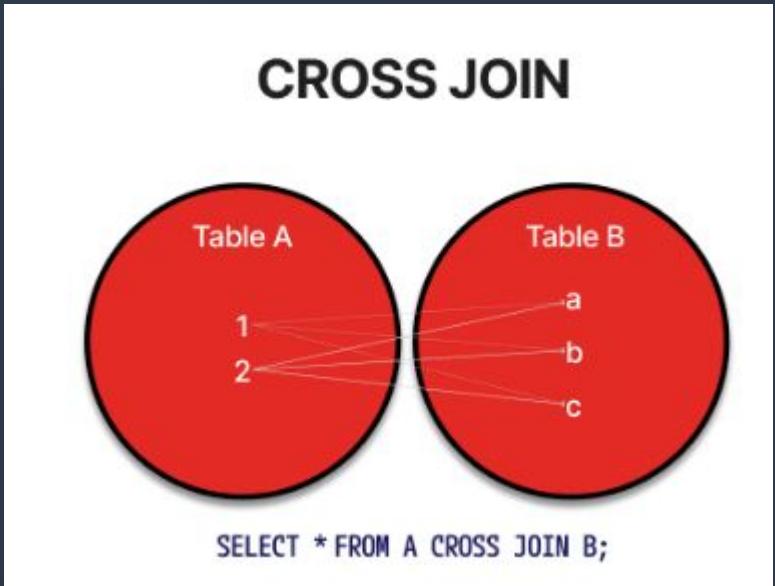
오른쪽 테이블의 모든 행을 우선 포함합니다. 왼쪽 테이블은 조인 조건에 매칭되는 행만 결합하고, 매칭되는 데이터가 없으면 NULL로 표시됩니다.



FULL (OUTER) JOIN (완전 외부 조인)

양쪽 테이블의 모든 행을 포함합니다. 서로 매칭되지 않는 부분은 모두 NULL로 표시됩니다.

4. 기타 JOIN의 종류



CROSS JOIN (교차 조인)

ON 조건절 없이 두 테이블의 모든 행을 가능한 모든 조합으로 결합합니다.
Cartesian Product(데카르트 곱)라고도 하며, 심각한 성능 문제를 유발할 수 있습니다.



SELF JOIN (자체 조인)

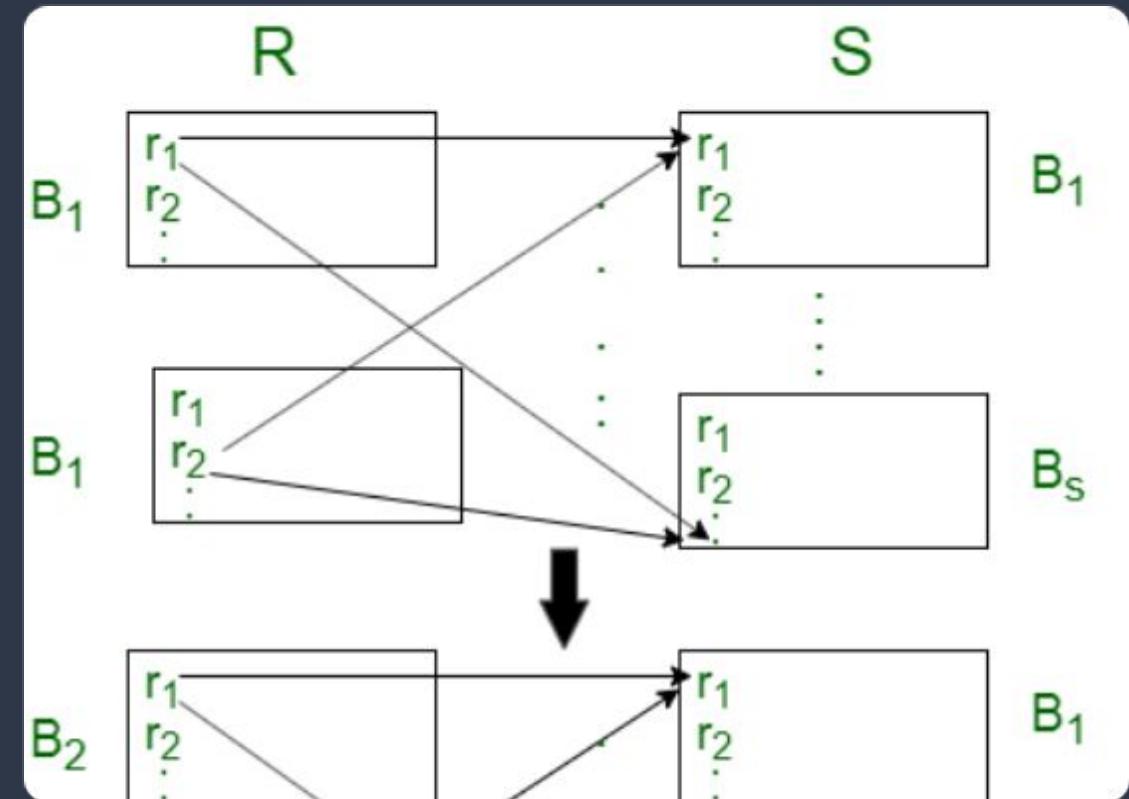
하나의 테이블이 자기 자신과 조인하는 방식입니다. 테이블에 별칭(Alias)을 붙여 서로 다른 테이블인 것처럼 사용합니다.(예:'직원'의 '관리자' 정보 찾기)

5. JOIN의 내부 구현 방식 (1/2)

Nested Loop Join (NL Join)

DBMS는 데이터를 효율적으로 결합하기 위해 여러 알고리즘을 사용합니다. NL Join은 이중 `for`문과 유사합니다.

- **동작:** 바깥쪽(Driving) 테이블의 행을 하나씩 읽고, 해당 행과 매칭되는 데이터를 찾기 위해 안쪽(Driven) 테이블을 스캔합니다.
- **특징:**
 - 바깥쪽 테이블의 크기가 작을 때 유리합니다.
 - 안쪽 테이블의 조인 컬럼에 **인덱스(Index)**가 있으면 매우 빠릅니다.



5. JOIN의 내부 구현 방식 (2/2)

Sort Merge Join (SM Join)

- 동작:
 1. 두 테이블을 각각 조인 컬럼 기준으로 정렬(Sort)합니다.
 2. 정렬된 두 테이블을 병합(Merge)하면서 조인 조건을 만족하는 행을 결합합니다.
- 특징: 대용량 데이터를 조인할 때 효율적이며, 비등가(>, <) 조인에도 사용할 수 있습니다.

Hash Join (HJ)

- 동작:
 1. **Build Phase:** 작은 테이블(Build Table)을 읽어, 조인 컬럼을 키로 하는 해시 테이블을 메모리에 생성합니다.
 2. **Probe Phase:** 큰 테이블(Probe Table)을 읽으면서, 해시 테이블에서 매칭되는 데이터를 즉시 찾습니다.
- 특징: 등가(=) 조인에서 가장 효율적이며, 충분한 메모리 공간이 필요합니다.

6. 쿼리 옵티마이저: DBMS의 두뇌 (1/3)

1. 규칙 기반 옵티마이저 (RBO)

과거에 사용되던 방식. 사전에 정의된 '규칙'의 우선순위에 따라 실행 계획을 생성합니다. (예: "인덱스가 존재하면 항상 사용하라")

치명적 한계: 데이터의 실제 분포나 테이블 크기를 전혀 고려하지 않습니다. 90%를 조회할 때도 비효율적인 인덱스 스캔을 선택할 수 있습니다.

2. 비용 기반 옵티마이저 (CBO)

현대의 모든 DBMS 표준. 고정된 규칙이 아닌 '비용(Cost)' (I/O, CPU, Memory)을 기반으로 최적의 계획을 선택합니다.

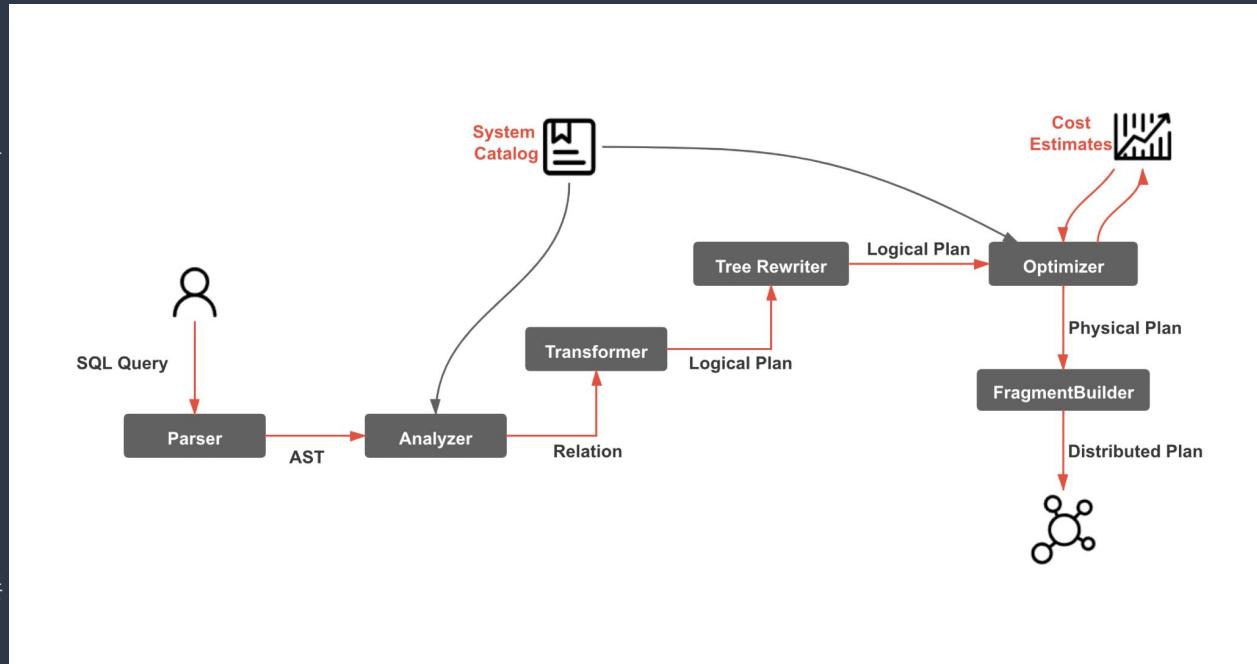
핵심: CBO는 '통계에 기반한 예측 모델'과 같습니다.

6. 쿼리 옵티마이저: DBMS의 두뇌 (2/3)

CBO(비용 기반 옵티마이저)의 작동 원리

- 통계 정보(Statistics) 수집: 총 행 수, 값의 분포도, 고유 값 수 등을 수집합니다.
- 후보 계획 생성: 동일 SQL을 실행할 다양한 '후보 실행 계획'을 생성합니다. (예: NLJ, HJ)
- 비용 추정(Cost Estimation): '통계 정보'를 바탕으로 각 후보 계획의 '총 비용'을 추정합니다.
- 최적안 선택: CBO는 추정된 비용이 '가장 낮은' 실행 계획을 최종 선택합니다.

경고: 통계 정보가 오래되어 실제와 다르면, CBO가 '부정확한' 비용을 맹신하고 최악의 실행 계획을 선택할 수 있습니다.



6. 쿼리 옵티마이저: DBMS의 두뇌 (3/3)



Nested Loop Join 선택

(필터링 후) Driving 테이블이 충분히 작고, Inner 테이블의 조인 컬럼에 효율적인 인덱스가 있을 때 선택됩니다.



Hash Join 선택

대용량 데이터를 조인하거나, 적절한 인덱스가 없어 NLJ의 비용이 더 크다고 판단될 때 선택됩니다.(메모리 필요)



Sort Merge Join 선택

데이터가 이미 정렬되어 있거나, Hash Join을 수행할 메모리가 부족할 때 차선책으로 선택될 수 있습니다.

7. 실행 계획(Execution Plan) 분석

"실행 계획"은 CBO가 선택한 최종 작업 절차 보고서입니다. EXPLAIN 명령어로 옵티마이저의 선택을 확인할 수 있습니다.

핵심 지표 읽기 (MySQL 기준)

- id: 실행 순서 (높을수록 먼저).
- type: 테이블 접근 방식. ALL(풀 스캔)은 최악, ref, eq_ref가 효율적입니다.
- key: CBO가 실제로 사용하기로 선택한 인덱스. NULL이면 인덱스를 못 썼다는 의미입니다.
- rows: CBO가 처리할 것으로 '추정'한 행 수. 실제와 너무 다르면 통계 정보가 오래된 것입니다.

The screenshot shows the dbForge Studio for Oracle interface. In the center, the 'EXPLAIN PLAN' window displays the execution plan for a query. The query itself is:

```
SET STATEMENT_ID = 'Price Change'
INTO plan_table
FOR UPDATE OF film
    SET replacement_cost = replacement_cost + 5
    WHERE film_id =
        (SELECT film_id FROM film_category
         WHERE category_id = 3);
```

The execution plan details the steps taken by the database to execute the query. On the left, the 'Database Explorer' pane shows the schema, including tables like ACTOR, ADDRESS, CATEGORY, CUSTOMERS, DEPARTMENTS, EMPLOYEES, and FILM, along with their columns.

8. 인덱스와 JOIN 성능의 상관관계

1. 인덱스가 없는 경우

조인 조건 컬럼에 인덱스가 없다면, CBO는 가장 효율적일 수 있는 **Index Nested Loop Join (INLJ)**을 실행 계획 후보에서 원천적으로 배제합니다.

선택지는 BNLJ, Hash Join, Sort-Merge Join만 남게 됩니다. Extra 필드에 **Using join buffer (Block Nested Loop)** 경고가 나타날 확률이 높습니다.

2. 인덱스가 있는 경우

내부 테이블의 조인 컬럼에 인덱스를 생성하면, CBO는 INLJ라는 강력한 실행 계획을 '후보'에 올릴 수 있게 됩니다.

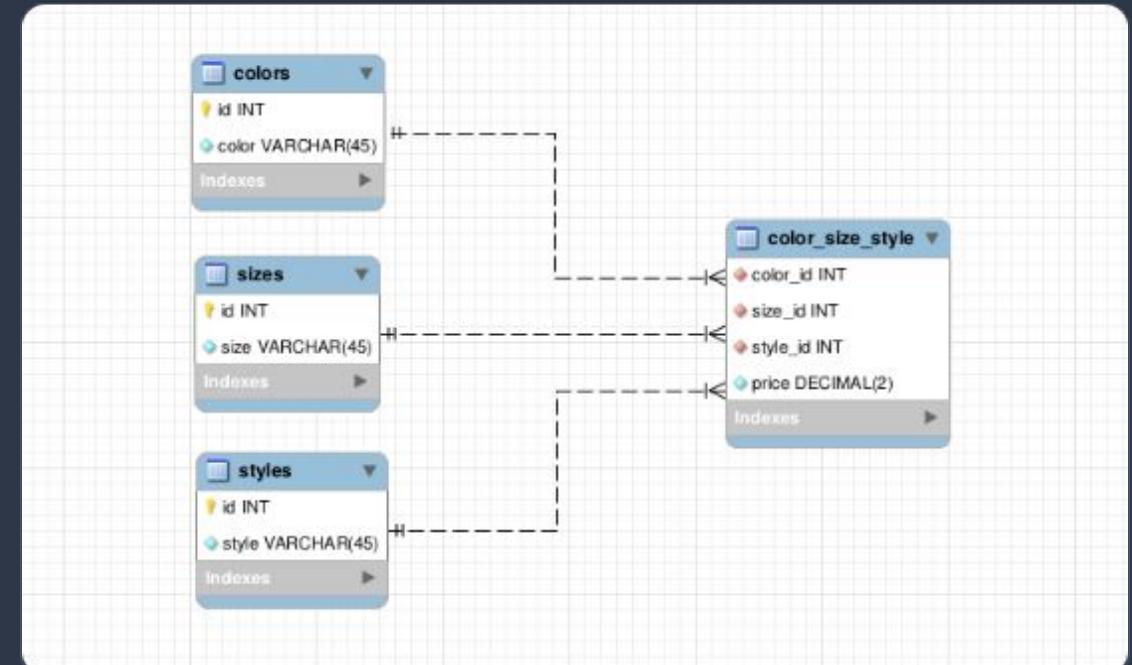
대부분의 경우 인덱스를 이용한 직접 탐색 비용은 풀 스캔이나 정렬/해싱 비용보다 훨씬 저렴하므로, CBO는 INLJ를 선택하게 되고 성능은 극적으로 향상됩니다.

9. 3중 조인 (Multi-Way JOIN)의 동작

Q: 3중 조인 (A, B, C)은 어떻게 동작하나요?

A: 한 번에 3개를 조인하지 않습니다. 2개씩 순차적으로 조인합니다.

- 기본 동작 방식
 1. DBMS(옵티마이저)는 A, B, C 중 어떤 테이블을 먼저 조인할지 결정합니다.
 2. 예: (A JOIN B)를 먼저 수행하여 **중간 결과 집합 (Result_AB)**를 만듭니다.
 3. 이 중간 결과 집합과 나머지 테이블 C를 다시 조인합니다.
(Result_AB JOIN C)
- 옵티마이저의 핵심 역할
(A JOIN B) -> C, (B JOIN C) -> A 등 다양한 순서 중 총비용이 가장 낮은 최적의 조인 순서를 선택합니다.



10. 3중 조인과 성능의 관계

조인 순서가 성능을 결정합니다.

핵심은 '중간 결과 집합의 크기'입니다.

 **성능에 미치는 영향:** 첫 번째 조인(A JOIN B)의 결과 행 수가 적을수록, 두 번째 조인(Result_AB JOIN C)에서 처리할 데이터가 줄어들어 성능이 향상됩니다.

 **최적의 순서:** 옵티마이저는 일반적으로 중간 결과 행 수를 가장 적게 만드는 순서를 선호합니다.

 **잘못된 순서:** 만약 첫 번째 조인 결과가 매우 크면, 메모리 부담이 증가하고 두 번째 조인 시 처리할 데이터가 많아져 성능이 급격히 저하됩니다.

 **개발자의 역할:** 옵티마이저가 최적의 순서를 찾도록 WHERE절을 적절히 사용하고, 통계 정보를 최신으로 유지합니다.(필요시 '조인 힌트' 사용)

11. 요약 및 Q&A

요약 (Summary)

- JOIN은 관계형 DB에서 분리된 데이터를 결합하는 핵심 연산입니다.
- INNER, LEFT/RIGHT, FULL JOIN 등 목적에 맞는 JOIN을 사용해야 합니다.
- DBMS는 NL, Sort Merge, Hash Join 등 다양한 내부 알고리즘을 사용합니다.
- JOIN 성능은 '옵티마이저 (CBO)'와 '실행 계획', 그리고 '인덱스'에 의해 결정됩니다.
- 3중 이상의 다중 조인은 '조인 순서'와 '중간 결과 집합 크기'가 성능의 핵심입니다.

Q&A

Thank you.

Image Sources



<https://i.sstatic.net/UI25E.jpg>

Source: stackoverflow.com



<https://red9.com/wp-content/uploads/2025/05/sql-joins-visualized-venn-diagram-red9.png>

Source: red9.com



<https://www.acuitytraining.co.uk/wp-content/uploads/2022/01/sql-join-types.png>

Source: www.acuitytraining.co.uk



<https://www.acuitytraining.co.uk/wp-content/uploads/2022/01/sql-join-types.png.webp>

Source: www.acuitytraining.co.uk



<https://s33046.pcdn.co/wp-content/uploads/2020/02/sql-cross-join-working-principle.png>

Source: www.sqlshack.com



<https://dotnettrickscloud.blob.core.windows.net/article/sql%20server/3720241023165251.webp>

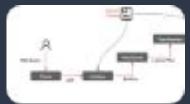
Source: www.scholarhat.com

Image Sources



https://media.geeksforgeeks.org/wp-content/uploads/20190924110021/two_block.png

Source: www.geeksforgeeks.org



<https://celerdata.com/hs-fs/hubfs/starrocks%20query%20planning.png?width=3156&height=1680&name=starrocks%20query%20planning.png>

Source: celerdata.com



<https://www.devart.com/dbforge/oracle/studio/images/oracle-explain-plan-example.png>

Source: www.devart.com



https://miro.medium.com/1*PTE93fxemX-qw0ewrsICTg.png

Source: medium.com