최적의 SQL 작성 비법

엔코아 컨설팅
솔루션 사업본부 본부장
명재호 이사

CONTENTS

- 대부분의 개발자의 SQL 작성 모습
- 이런 SQL을 작성하자
- SQL 쉽다? vs 어렵다?
- 최적의 성능을 위한 SQL 작성시 고려사항
- SQL 작성 Process
- SQL 구조 Tree
- 사례 연구
- Summary

대부분의 개발자의 SQL 작성 모습

요구사항 → SQL → 결과

무슨 SQL을 작성했지?

이 SQL이 어떻게 수행될까?

이런 사항은 관심 밖!!!

오로지 관심은 결과값!!!

이런 SQL을 작성하자

요구사항 → 

→ 결과

내부가 훤히 들여다 보이는 SQL

무슨 SQL을 작성했는지 정확히 알 수 있는 SQL

실행계획을 정확히 예상할 수 있는 SQL

결과만이 아닌 성능이 담보된 SQL

SQL 쉽다? vs 어렵다?

- SQL 쉽다?
  - 'WHAT'만 필요할 뿐 'HOW'는 필요없다.
  - 배우기가 쉽다.
    - SELECT, FROM, WHERE
    - 프로그램을 모르는 END-USER도 손쉽게 SQL을 작성할 수 있다.
  - 동일한 결과를 얻을 수 있는 방법은 수도 없이 많다.
- SQL 어렵다?
  - 최적의 SQL을 작성하는 것은 쉽지 않다.
  - SQL은 SELECT, FROM, WHERE절만 알아서 되는 것은 아니다.
  - 수행되는 과정을 정의할 수 없다. à 성능을 담보할 수 없다.
  - **생각하지 않은 SQL을 성능을 보장 받을 수 없다!!!**
- **그렇다면 어떻게 해야 최적의 SQL을 작성할 수 있을까?**

최적의 성능을 위한 SQL 작성시 고려사항

- 집합적 사고
- 인덱스(Index(집합적 사고 및 인덱스 정책에 대한 부분은 추후 세미나에서 진행함))

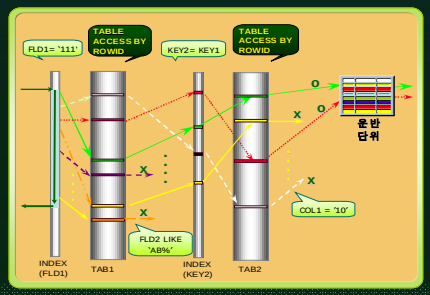
- 조인방식(Join Method)
- 조인순서(Join Order)
- 조인 연결고리(Join Link)
- 인라인 뷰(Inline View)
- 서브쿼리(Subquery)
- 실행계획(Execution Plan)

- 저장형 함수(Stored Procedure)
- 스칼라 서브쿼리(Scalar Subquery)
- 기타

고려사항 1 : Join Method

Nested Loops Join

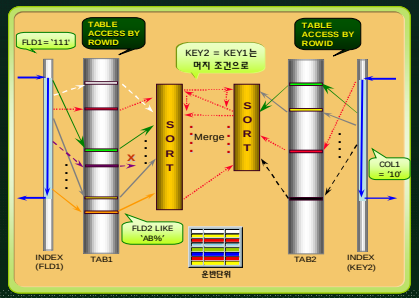
- 순차적
- 선행적
- 종속적
- 랜덤액세스
- 선택적
- 연결고리 상태, 방향성
- 부분범위 처리 가능
- 체크조건 영향력이 적음



OLTP 90%

Sort Merge Join

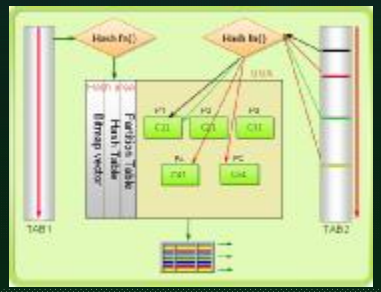
- 동시적
- 독립적
- 전체범위처리
- 스캔방식
- 연결고리 상태 무관
- 무방향성
- 과다한 정렬작업
- 체크조건 영향력이 높음



Rare

Hash Join

- 독립적
- 연결고리 상태 무관
- 조인 전 필터링
- 적절한 메모리 지정필요
- 동치조인(Equijoin)만 가능
- 대량 범위 처리 유리



OLAP 90%

고려사항 2 : Join Order

- Driving Table
  - 드라이빙 테이블이란 두개의 테이블이 조인을 할 경우 먼저 처리 되는 테이블을 의미한다.
  - WHERE절의 상수 조건에 가용할 수 있는 인덱스가 존재해야 함
- Driven Table
  - 두 개의 테이블이 조인을 할 경우 뒤에 처리되는 테이블을 의미한다.
  - 드라이빙 테이블로 부터 상수값을 공급받아 처리 됨
  - 연결고리를 통해 상수 값은 공급받게 됨
  - 따라서, 연결고리에 인덱스가 정상적으로 존재해야 함
  - 또는, 연결고리와 WHERE절의 상수조건을 포함한 인덱스가 존재해야 함
- 최적화된 Join Order란?
  - 모든 연결고리가 정상상태일 경우 처리범위가 적은 쪽에서 부터 드라이빙 되도록 처리 하는 것이 일반적으로 최적화된 조인 오더임

고려사항 3 : Join Link

- 연결고리(Join Link)란 무엇인가?
  - Where절에서 조인에 참여하는 각 테이블의 컬럼 간 다양한 연산자(=,〈〉, 〉=, in…)로 연결되어 있는 부분을 연결고리라고 함
- 연결고리 종류
  - 연결고리는 오로지 '='만 존재하는 것이 아님
- 연결고리 상태
  - 양쪽 정상
    - 조인 오더에 상관없이 항상 인덱스를 사용
    - 양쪽 연결고리가 정상일 경우 처리범위가 적은 테이블부터 드라이빙 되는 것이 유리
  - 한쪽 정상
    - 논리적으로 연결고리가 비정상인 테이블(연결고리에 인덱스가 사용할 수 없는 테이블)에서 연결고리가 정상적인 테이블로의 조인오더는 양쪽 정상과 같이 문제가 되지 않음
    - 단, 반대의 경우로 수행이 될경우 성능상의 문제를 야기할 수 있음
  - 양쪽 이상
    - 대부분 Nested Loops 조인 형태 보다는 Hash Join으로 풀리게 되는 경우가 대부분임

```
SELECT  a.FLD1, ... ,  b.FLD1, ...
FROM  TAB1  a , TAB2  b
WHERE  a.KEY1 = b.KEK1
    and  a.FLD1 =  '20'
    and  .....
```

고려사항 4 : Inline View

- Inline View의 정의
  - 인라인 뷰란 FROM 절 상에 오는 서브쿼리로서 VIEW처럼 동작을 함
  - 적절한 크기의 중간집합을 생성하기 위하여 사용됨
  - 데이터 처리의 순서를 의도적으로 지정하기 위해 사용됨.
  - 정상적이지 않은 데이터 구조를 연결하기 위해 사용됨.
  - 실행계획의 제어를 위한 목적으로 사용됨.
- Inline View vs View vs Subquery Factoring

	Inline View	Subquery Factoring	View
SQL 작성	Inline view가 사용되는 곳 마다 전체 view 쿼리를 작성	단일 SQL에서 한번만 view query를 작성 후 실제 SQL내에서는 Alias로 사용	View 명으로 사용
SQL간 공유	단일 SQL 한정	단일 SQL 한정	다수 SQL
VIEW의 실행 횟수	매번 실행	단일 SQL에서 한번	매번 실행

고려사항 5 : Subquery

- Subquery의 정의
  - 메인쿼리에 종속되는 하위의 쿼리를 의미함
  - 종속의 의미는 반드시 메인쿼리의 집합 레벨을 변경할 수 없음
  - 메인쿼리는 서브쿼리의 모든 속성을 사용할 수 없음
  - 서브쿼리는 메인쿼리의 모든 속성을 사용할 수 있음
  - 대표적인 연산자로는 IN, EXISTS가 있음
- 제공자형 서브쿼리
  - 메인쿼리의 처리범위를 줄여줄 수 있을때 사용고려
  - 메인쿼리에 상수값을 공급하는 것과 동일한 효과를 얻을 수 있음
  - 서브쿼리로 부터 상수값을 공급받는 Where절의 조건에 인덱스가 필요함
- 확인자형 서브쿼리
  - 메인쿼리에서 산출된 결과과 다른 테이블의 조건을 만족하는지 여부를 확인하고자 할때 사용
  - 만족하는 결과만 존재하면 처리를 더 이상 만족하는 로우를 찾지 않음으로 처리가 빠름

고려사항 6 : Execution Plan의 확인

- 실행계획을 확인해야 하는 이유
  - SQL 작성시 자신이 예상하였던 실행계획과 옵티마이져가 생성한 실행계획과의 비교
  - 옵티마이져가 작성자가 생각했던 최적의 처리경로와 같이 실행계획을 수립했는지에 대한 판단이 필요
  - 옵티마이져의 한계로 인해 사용자가 생각하는 논리적인 판단과의 차이 확인

최적의 SQL로 보아도 무방함

사용자가 생각했던 최적의 처리 경로 = 옵티아미져가 생각한 최적의 처리 경로

사용자의 논리적 판단과 옵티마이져의 논리적 판단을 비교할 필요가 있음

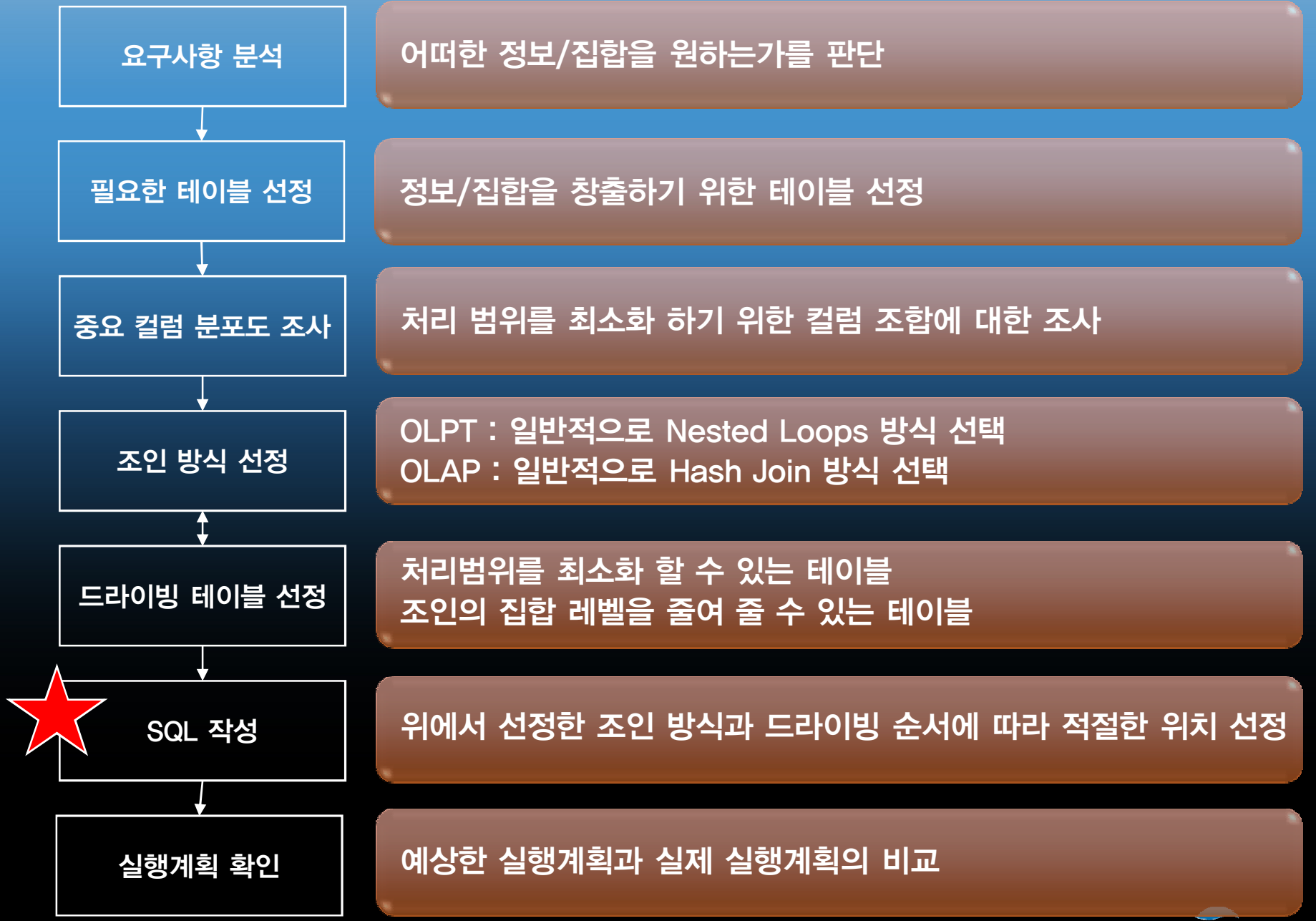
사용자가 생각했던 최적의 처리 경로 ≠ 옵티아미져가 생각한 최적의 처리 경로

- 실행계획에 대한 사례는 다음 세션 참고

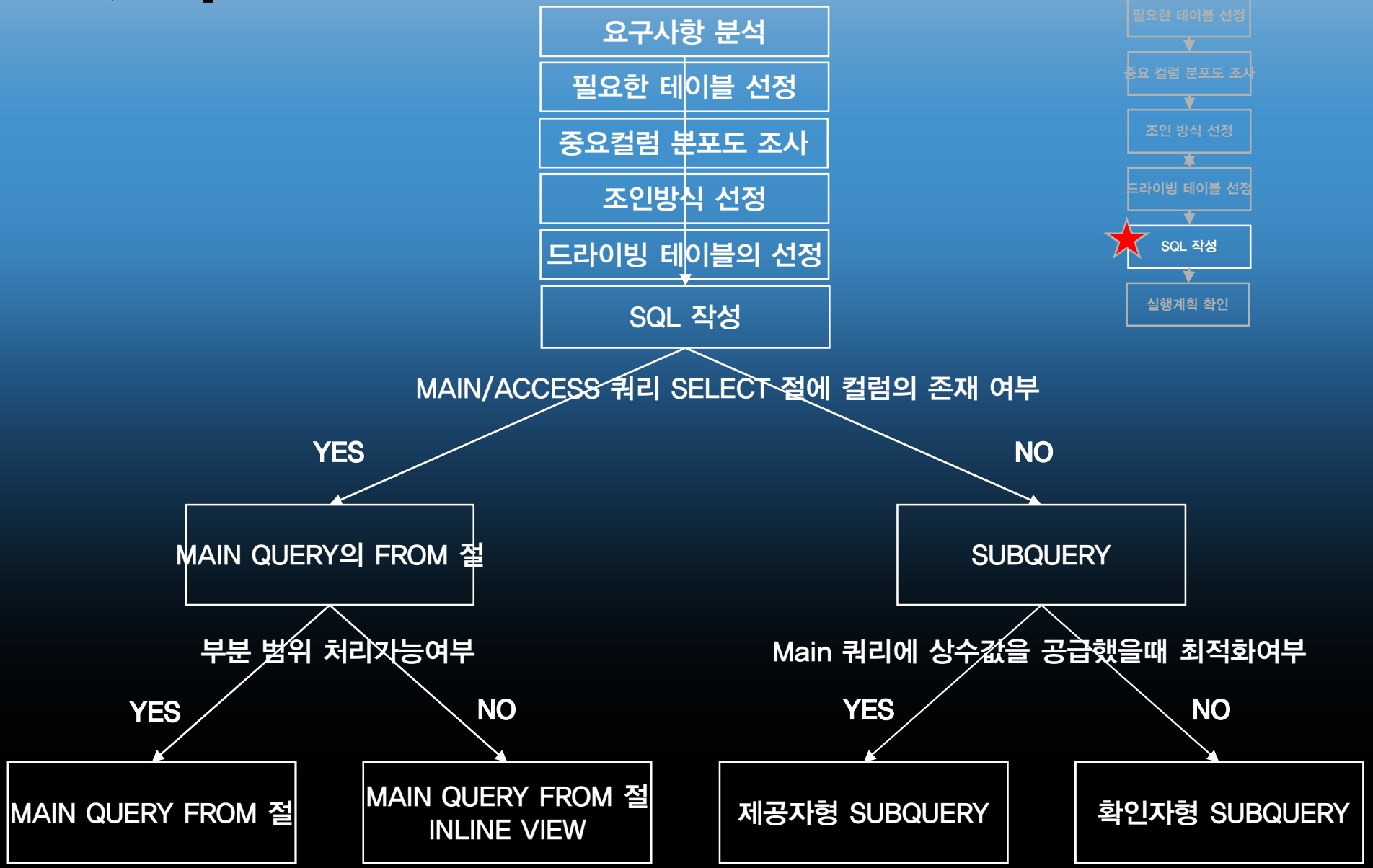
고려사항 정리

구분	인라인뷰 /조인	유니온 /그룹	저장형 함수	서브쿼리	스칼라 쿼리
M:M 관계의 데이터 연결	O	O	O	O	O
결과의 추출을 원할 때	O	O	O	△	O
다양한 추출컬럼이 필요할 때	O	O	△	△	△
양측 OUTER 조인	X	O	X	X	X
독자적으로 범위를 줄일 수 있을 때	O	O	O	O	O
다른 쪽에서 결과를 받는 것이 유리	X	X	O	O	O
배타적 관계의 연결	X	O	O	△	O
연결할 집합이 유사하지 않을 때	O	△	O	O	O
부분범위처리	△	X	O	△	O
기본키와 외부키가 아닌 경우 연결	O	O	O	O	O
단순히 조건 체크만 원할 때	△	X	△	O	△
단순히 조건의 상수값만 제공할 때	△	X	△	O	△

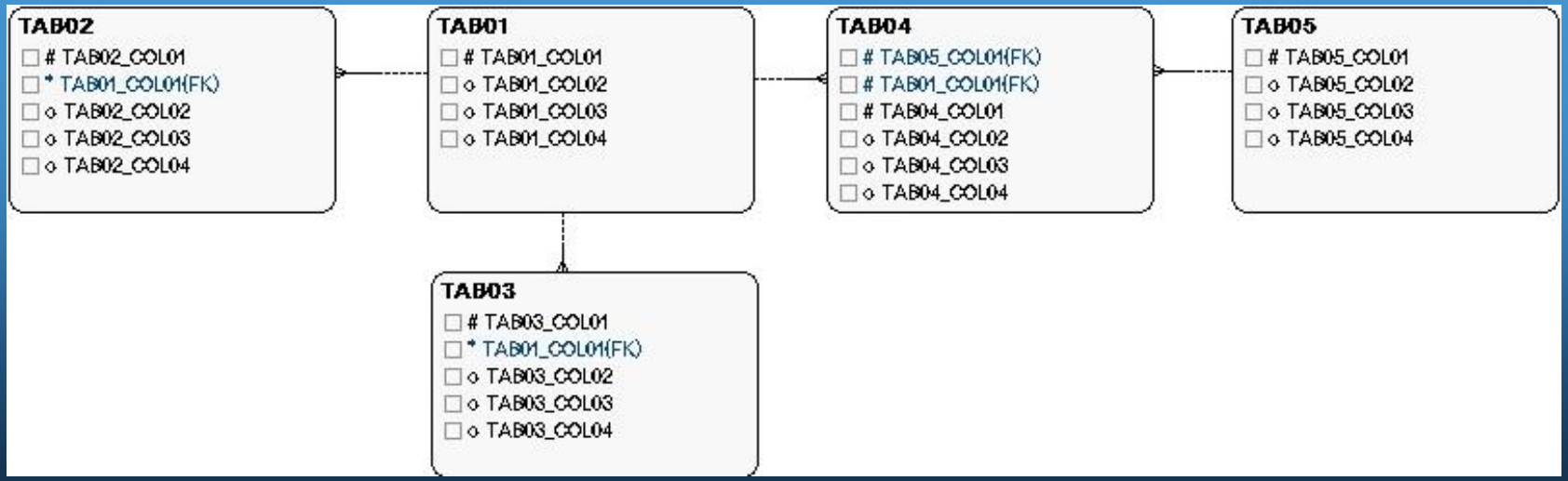
SQL 작성 Process



SQL 구조 Tree



사례 연구



TAB01	TAB02	TAB05
TAB01_COL01(=)	TAB02_COL02 〈〉	TAB05_COL02(between)
TAB01_COL03(=)	TAB02_COL04(=)	TAB05_COL03(like)

주어진 상수조건을 모두 만족하는게 되는 TAB01의 로우를 모두 출력하도록 하시오.

사례 연구 – 일반적인 작성 형태

①

```
SELECT *
  FROM TAB01
WHERE TAB01_COL1 ='……'
    AND TAB01_COL3='……'
```

②

```
SELECT *
  FROM TAB01, TAB02
WHERE TAB01_COL1 ='……'
    AND TAB01_COL3='……'
    AND TAB02_02<>'……'
    AND TAB02_COL04='……'
    AND TAB01.TAB01_COL1 = TAB02.TAB01.COL01
```

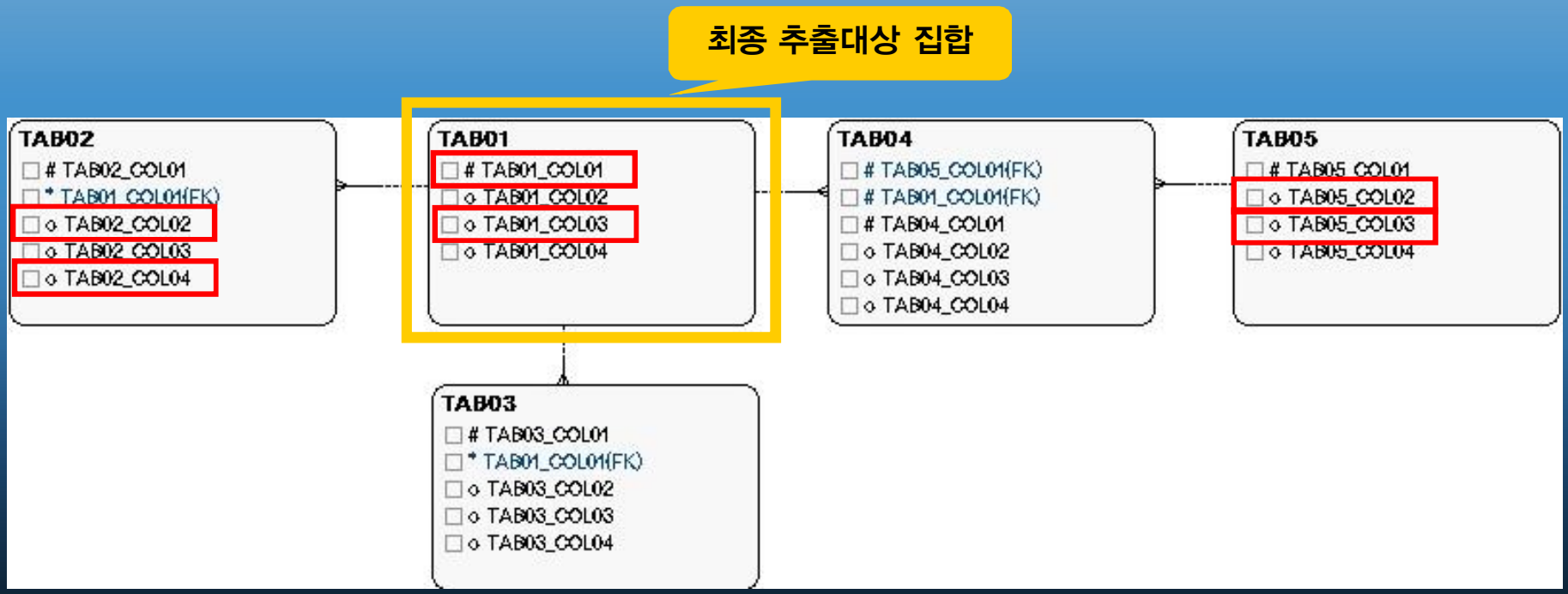
③

```
SELECT *
  FROM TAB01, TAB02, TAB05
WHERE TAB01_COL1 ='……'
    AND TAB01_COL3='……'
    AND TAB02_02<>'……'
    AND TAB02_COL04='……'
    AND TAB01.TAB01_COL1 = TAB02.TAB01.COL01
    AND TAB05_COL02 BETWEEN '……' AND '……'
    AND TAB05_COL3 LIKE '……%'
```

④

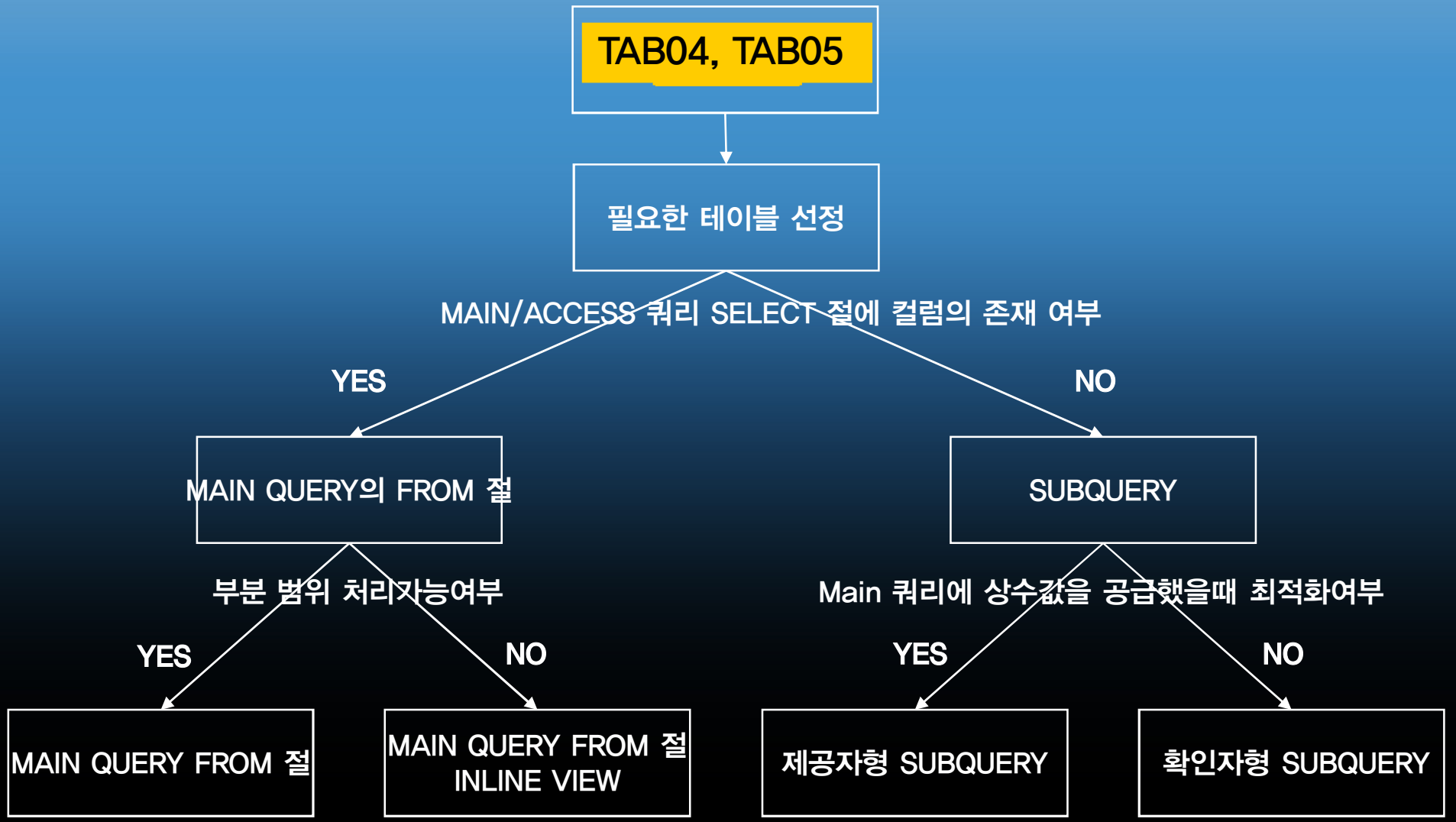
```
SELECT *
  FROM TAB01, TAB02, TAB05, TAB04
WHERE TAB01_COL1 ='……'
    AND TAB01_COL3='……'
    AND TAB02_02<>'……'
    AND TAB02_COL04='……'
    AND TAB01.TAB01_COL1 = TAB02.TAB01.COL01
    AND TAB05_COL02 BETWEEN '……' AND '……'
    AND TAB05_COL3 LIKE '……%'
    AND TAB04.TAB05_COL01 = TAB05.TAB05.COL01
    AND TAB04.TAB01_COL01 = TAB01.TAB01_COL01
```

사례 연구 – SQL Tree 구조 분석



TAB01	TAB02	TAB05
TAB01_COL01(=)	TAB02_COL02 〈〉	TAB05_COL02(between)
TAB01_COL03(=)	TAB02_COL04(=)	TAB05_COL03(like)

사례연구 – SQL 구조 Tree 분석



사례 연구 : 최종 SQL 작성

①

```
SELECT
  FROM TAB01
WHERE IN (SELECT
                  FROM TAB04, TAB05
    AND EXISTS (SELECT
                       FROM TAB02
                      WHERE                )
```

②

```
SELECT TAB01.*
  FROM TAB01
WHERE TAB01.TAB01_COL01 IN (SELECT TAB4.TAB01_COL01
                                  FROM TAB04, TAB05
                                 WHERE TAB04.TAB05_COL01 = TAB05.TAB05.COL01
                                   AND TAB05_COL02 BETWEEN '……' AND '……')
   AND EXISTS (SELECT 1
                 FROM TAB02
                WHERE TAB01.TAB01_COL1 = TAB02.TAB01.COL01
                  AND TAB02_COL02<>'……'
                  AND TAB02_COL04='……')
```

Summary

- SQL을 작성은 단순히 테이블을 나열하는 것이 아니다.
- 결과는 뿐만 아니라 성능 담보한 SQL을 작성해야 한다.
- SQL이 실행되는 과정을 먼저 생각하고 옵티마이져를 확인하라.
- 데이터 모델을 참조해서 SQL을 작성한다.
- 삼삼 정석들은 암기하자.
- 이들로 부터 응용이 만들어 진다.
- 쉬운 문제 100개 보다 어려운 문제 1개가 더 많이 얻는다.
- SQL 작성시 10분만 생각하고 시작하자.

- 이제는 ITA(EA)의 시대다. 준비하자.
- 개발자가 논리 데이터 모델을 활용하기 시작했다.

Q & A

감사합니다.

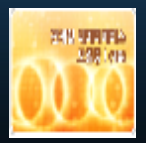
ESI 교육센타 7월 교육 안내(www.en-core.com)

Oracle 사용자를 위한 **SQL** 활용

기간 : **2007**년 **07**월 **02**일 ~ **2007**년 **07**월 **04**일

새로 쓴 대용량 데이터베이스 솔루션 **I**

기간 : **2007**년 **07**월 **09**일 ~ **2007**년 **07**월 **13**일



관계 형 데이터베이스 모델링 **I (**개론**)**

기간 : **2007**년 **07**월 **18**일 ~ **2007**년 **07**월 **20**일

새로 쓴 대용량 데이터베이스 솔루션 **for MS-SQL**

기간 : **2007**년 **07**월 **23**일 ~ **2007**년 **07**월 **26**일