## 2과목 1장 SQL기본

**● 데이터베이스의 종류**

(1) 관계형 DB : 릴레이션에 데이터를 저장하고 관리
릴레이션을 사용해서 집합 연산과 관계 연산을 할 수 있다.
(2) 계층형 DB : 트리 형태의 자료구조에 데이터를 저장하고 관리 (1:M 관계를 표현)
(3) 네트워크형 DB : 오너(Owner)와 멤버(Member) 형태로 데이터를 저장 (1:M / N:M 표현 가능)

**● 관계연산**

(1) 선택(SELECT) : 릴레이션 조건에 맞는 행(튜플)만을 조회
(2) 투영(PROJECT) : 릴레이션 조건에 맞는 속성만을 조회
(3) 결합(JOIN) : 여러 릴레이션의 공통된 속성을 사용해 새로운 릴레이션을 만든다
(4) 나누기(DIVIDE) : 기존 릴레이션에서 대상 릴레이션과 공통이 되는 행(튜플)을 제외하고 중복된 행도 제거

**● 테이블 구조**

(1) 기본키 : 하나의 테이블에서 유일성과 최소성을 만족하면서 해당 테이블을 대표하는 것
(2) 행 : 하나의 테이블에 저장되는 값 (튜플)
(3) 칼럼 : 어떤 데이터를 저장하기 위한 필드 (속성)
(4) 외래키 : 다른 테이블의 기본키를 참조하는 칼럼, 관계연산 중에서 결합연산(JOIN)을 하기 위해서 사용

**● SQL 종류**

(1) DDL : DB의 구조를 정의 (CREATE, ALTER, DROP, RENAME)
(2) DML : 테이블에서 데이터를 입력, 수정, 삭제, 조회
(INSERT, UPDATE, DELETE, SELECT)
(3) DCL : DB 사용자에게 권한을 부여하거나 회수
(GRANT, REVOKE)
(4) TCL : 트랜잭션을 제어하는 명령어 (COMMIT, ROLLBACK)
**√ 트랜잭션** : DB의 작업을 처리하는 단위

**● 트랜잭션의 특성**

(1) 원자성(Automicity) : 트랜잭션은 DB가 연산의 전부 또는 일부(All or Nothing) 실행만이 있다.
(2) 일관성(Consistency) : 트랜잭션 실행 결과로 DB의 상태가 모순되지 않아야 한다.
(3) 고립성(Isolation) : 트랜잭션 실행 중에 생성하는 연산의 중간결과는 다른 트랜잭션이 접근할 수 없다.
(4) 영속성(Durability) : 트랜잭션이 그 실행을 성공적으로 완료하면 그 결과는 영구적 보장이 되어야 한다.

**● ERD 구성요소**

엔터티(Entity), 관계(Relationship), 속성(Attribute)

**● SQL 실행순서**

(1) 파싱(Parsing) : SQL문의 문법을 확인하고 구문분석한다. 구문분석한 SQL문은 Library Cache에 저장
(2) 실행(Execution) : 옵티마이저가 수립한 실행계획에 따라 SQL을 실행
(3) 인출(Fetch) : 데이터를 읽어서 전송
**파싱 → 실행 → 인출**

---

**● 데이터 유형 ORACLE / SQL SERVER**

CHAR(s) : 고정 길이 문자열 정보 ( 'AA' = 'AA ')
VARCHAR(s) : 가변 길이 문자열 정보 ( 'AA' != 'AA ')
NUMBER / NUMERIC : 정수, 실수 등 숫자 정보
DATE / DATETIME : 날짜와 시각 정보

**● DDL**

(1) CREATE : 테이블 생성 / 테이블 생성 시 기본키, 외래키 기타 제약사항 등을 설정할 수 있다.
(2) ALTER : 테이블 변경 / 칼럼을 추가하거나 변경, 삭제 가능 / 기본키를 설정하거나 외래키 설정 가능
(3) DROP : 테이블 삭제 / 데이터의 구조뿐만 아니라 저장된 데이터도 모두 삭제

**● CREATE 예시**

```
CREATE TABLE EMP (
    empno number(10),   ----------①
    ename varchar2(20),   ----------②
    sal number(10, 2) default 0,   ---③
    deptno varchar2(4) not null,   ---④
    createdate date default sysdate,   -----⑤
    constraint emp_pk primary key (empno)   ---⑥
);
```

① : 정수/실수형 값 저장할 속성
② : 가변길이 문자를 저장할 속성
③ : 소수점 둘 째 자리까지 저장하며 초기화 값은 0이다
④ : not null 제약조건 추가 (무조건 값이 있어야 한다)
⑤ : 날짜를 저장할 속성, 초기화 값은 현재 날짜(sysdate)
⑥ : 기본키 설정

**● 테이블 생성 시 주의사항**

- 테이블 이름은 중복될 수 없다.
- 한 테이블 내의 칼럼명은 중복될 수 없다.
- 각 칼럼들은 , 로 구분되고 ; 로 끝난다.
- 칼럼 뒤에 데이터 유형은 꼭 지정되어야 한다.
- 테이블명과 칼럼명은 반드시 문자로 시작해야 한다.
- A-Z, a-z, 0-9, _, $, # 만 사용 가능하다.
- DATETIME 타입에는 별도로 크기를 지정하지 않는다.
- 기본키가 2개라면
  constraint emp_pk primary key (empno, ename)

**● 테이블 생성 시 CASCADE 사용**

constraint d_fk foreign key(deptno) dept (deptno)
on delete cascade

casecade 는 참조무결성을 유지하기 위한 제약조건
참조하는 부모 테이블의 키가 삭제될 경우 참조하는 자식 테이블에서의 해당 데이터가 같이 삭제된다.

**● 제약조건**

PRIMARY KEY(기본키) : 기본키 정의
UNIQUE KEY(고유키) : 고유키 정의
NOT NULL : NULL 입력 금지
CHECK : 입력 값 범위 제한 / NULL 무시(NULL 가능)
FOREIGN KEY(외래키) : 외래키 정의

√ 테이블 구조 확인
oracle → DESC 테이블;
SQL server → sp_help 'dbo.테이블'

**● ALTER 예시**

(1) 칼럼추가
ALTER TABLE emp ADD (age number(2) default 1) ;
(2) 칼럼 변경
ALTER TABLE emp MODIFY (ename varchar(40) not null) ;
(3) 칼럼 삭제
ALTER TABLE emp DROP COLUMN age;
(4) 칼럼명 변경 (oracle / sql-server)
ALTER TABLE emp RENAME COLUMN ename TO newNM ;
sp_rename 변경전칼럼, 새로운칼럼, 'CLOUMN'
(5) 테이블 제약조건 삭제
ALTER TABLE emp DROP CONSTRAINT 조건명 ;
(6) 테이블 제약조건 추가
ALTER TABLE emp ADD CONSTRAINT 조건명 조건 (칼럼명) ;

**● DROP 예시**

(1) 테이블 삭제
DROP TABLE emp ;
(2) 참조된 제약사항까지 모두 삭제
DROP TABLE emp CASCADE CONSTRAINT ;

**● TRUNCATE 예시**

(1) 테이블 데이터 삭제
TRUNCATE TABLE emp ;

**● DROP / TRUNCATE / DELETE 차이점**

(1) DROP : DDL / 테이블 정의 자체를 완전 삭제
(2) TRUNCATE : DDL(일부 DML) / 최초 생성된 초기 상태로 저장공간을 재사용 가능하도록 한다.
(3) DELETE : DML / 데이터만 삭제

√ Dependent : 관계형 데이터베이스에서 자식 테이블의 FK 데이터 생성 시 부모 테이블에 PK가 없는 경우, 자식 테이블의 데이터 입력을 허용하지 않는 참조동작

---

**● 뷰(VIEW)**

- 테이블로부터 유도된 가상의 테이블
- 실제 데이터를 가지고 있지 않고 테이블을 참조해서 원하는 칼럼만 조회 가능
- 데이터 딕셔너리(Data Dictionary)에 SQL문 형태로 저장하고 실행 시에 참조된다.

**● 뷰의 특징**

- 참조한 테이블이 변경되면 뷰도 변경된다.
- 특정 칼럼만 조회시켜서 보안성을 향상시킬 수 있다.
- 뷰의 검색은 참조한 테이블과 동일하게 할 수 있지만 뷰에 대한 입력/수정/삭제 에는 제약이 발생한다.
- 한 번 생성된 뷰는 변경할 수 없고 변경을 원하면 삭제 후 재생성 해야 한다.
- ALTER문을 사용해서 뷰를 변경할 수 없다.

**● 뷰 생성/조회/삭제**

(1) 뷰 생성
CREATE VIEW v_emp AS SELECT * FROM emp ;
(2) 뷰 조회
SELECT * FROM v_emp ;
(3) 뷰 삭제
DROP VIEW v_emp ;

**● 뷰의 장단점**

(1) 장점
특정 칼럼만 조회할 수 있기 때문에 보안 기능 향상
데이터 관리가 간단, SELECT 문이 간단
하나의 테이블에 여러 개의 뷰를 생성 가능
(2) 단점
뷰는 독자적인 인덱스를 만들 수 없음
삽입/수정/삭제 연산에 제약이 있음
데이터 구조 변경 불가

---

**● INSERT**

(1) INSERT 문
INSERT INTO table (col1, col2, ...) VALUES (ex1, ex2, ...) ;
- 테이블의 모든 칼럼에 삽입하는 경우 칼럼명 생략 가능
- Auto Commit이 아니라면 Commit을 해야 한다.

(2) SELECT문으로 입력
INSERT INTO I_table SELECT * FROM table ;
- SELECT문으로 데이터를 조회해 테이블에 삽입 가능
- 단, 입력되는 테이블은 사전에 생성되어 있어야 한다.

(3) Nologging
ALTER TABLE table Nologging ;
- DB에 데이터를 입력하면 로그파일에 그 정보를 기록한다.
- Check point 라는 이벤트가 발생하면 로그파일의 데이터를 데이터 파일에 저장한다.
- Nologging 옵션은 로그파일의 기록을 최소화시켜서 입력시에 성능을 향상시키는 방법이다.
- Nologging 옵션은 Buffer Chace 라는 메모리 영역을 생략하고 기록한다.

**● UPDATE**
UPDATE emp SET ename = '조조' WHERE empno = 100 ;
- UPDATE 문은 데이터를 수정할 때 조건절이 나오는 행 수만큼 수정된다.

**● DELETE**
DELETE FROM emp WHERE empno = 100 ;
- DELETE 문으로 데이터를 삭제한다고 해서 테이블의 용량이 초기화되지는 않는다.

**● 테이블의 모든 데이터 삭제**
(1) DELETE FROM table : 데이터가 삭제되어도 테이블의 용량은 감소하지 않는다.
(2) TRUNCATE TABLE table : 데이터가 삭제되면 테이블의 용량을 초기화 한다. (정의는 삭제되지 않음)

**● SELECT**
SELECT * FROM emp WHERE empno = 100 ;

**● 와일드 카드**
* : 모든 % : 모든 _ : 한 글자

**● 합성 연산자 : 문자열 결합을 원할 경우 사용한다.**
|| (Oracle), + (SQL Server)
ex) SELECT ename || '님' FROM emp ; (Oracle)
ex) SELECT ename + '님' FROM emp (SQL Server)

**● ORDER BY 를 사용한 정렬 및 특징**
SELECT * FROM emp ORDER BY ename, sal DESC ;
- ORDER BY가 정렬을 하는 시점은 SELECT 다음
- ORDER BY는 정렬을 하기 때문에 메모리를 많이 사용
- 대량의 데이터를 정렬하면 정렬로 인한 성능저하가 발생
- 정렬을 위해 메모리 내부에 할당된 SORT_AREA_SIZE 사용
- 만약 SORT_AREA_SIZE가 너무 작으면 성능저하 발생
- 정렬을 회피하기 위해 인덱스를 생성할 때 사용자가 원하는 형태로 오름차순/내림차순 으로 생성해야 한다.
- 특별한 지정이 없으면 ORDER BY는 오름차순
- SELECT 절에서 정의하지 않는 칼럼 사용 가능
- SQL 문장의 제일 마지막에 위치
- ORDER BY 절에 칼럼명 대신 Alias 명이나 칼럼 순서를 나타내는 정수도 사용 가능

√ Oracle과 SQL Server 정렬 시 NULL 값 주의사항
Oracle 에서는 NULL을 가장 큰 값으로 취급하며
SQL Server 에서는 NULL을 가장 작은 값으로 취급한다.

**● INDEX 를 사용한 정렬 회피 :** ORDER BY 없이 SELECT를 하면 기본키로 오름차순 정렬 된다. (기본키는 테이블 생성시 자동으로 인덱스 생성)

**● Distinct 와 Alias**
(1) Distinct : 중복된 데이터를 한 번만 조회
(2) Alias : 테이블명(칼럼명)을 간략하게 사용할 때 활용

**● WHERE 주의사항**
(1) LIKE : 와일드카드를 사용하지 않으면 '=' 와 같다.
ex) SELECT * FROM emp WHERE ename LIKE 'test1' ;
(2) Between : 지정된 범위에 있는 값을 조회한다.
ex) Between 100 and 200 (100과 200을 포함하고 그 사이에 있는 값을 조회)
(3) IN : OR의 의미를 가지고 있어 하나의 조건만 만족해도 조회가 된다.
ex) job IN ('clerk', 'manager')
(4) IN : 두 개의 칼럼을 동시에 조회할 수 있다.
ex) (job, ename) IN ( ('clerk', 'test1'), ('manager', 'test4') )

**● NULL**
- 모르는 값을 의미 / 값의 부재를 의미
- NULL과 모든 비교는 알 수 없음을 반환
- 숫자 혹은 날짜를 더하면 NULL이 된다.
- NULL 값은 비교연산자로 비교할 수 없다.
- 만약 비교연산자로 NULL을 비교하면 false 가 나온다.

**● NULL 조회 : IS NULL 또는 IS NOT NULL 사용**
ex) SELECT * FROM emp WHERE mgr IS NULL ;
ex) SELECT * FROM emp WHERE mgr IS NOT NULL ;

**● NULL 관련 함수**
(1) NVL : NULL 이면 다른 값으로 변경
- NVL(mgr, 0) : mgr 이 null 이면 0으로 변경
(2) NVL2 : NVL 함수와 DECODE를 하나로 만든 것
- NVL2(mgr, 1, 0) : mgr 이 not null 이면 1, null 이면 0
(3) NULLIF : 두 개의 값이 같으면 null, 다르면 첫 번째 값
- NULLIF(exp1, exp2) : exp1 == exp2 면 null, 다르면 exp1
(4) COALESCE : NULL이 아닌 최초의 표현식
- COALESCE(null, null, 'abc') : abc
- 표현식이 모두 NULL일 경우엔 결과도 NULL
- 인수에 NULL 값 상수만 지정한다면 오류 발생

---

● **GROUP BY, HAVING 특징**

- GROUP BY 절을 통해 소그룹별 기준을 정한 후, SELECT 절에 집계 함수를 사용한다.
- 집계 함수의 통계 정보는 NULL 값을 가진 행을 제외하고 수행한다.
- GROUP BY 절에서는 Alias 는 사용할 수 없다.
- 집계 함수는 WHERE 절에 올 수 없다.
- HAVING 절에는 집계함수를 이용해 조건 표시가 가능하다.
- HAVING 절은 일반적으로 GROUP BY 뒤에 위치한다.

● **집계함수 종류**

COUNT(*) : 행 수를 조회(NULL 포함)
COUNT(표현식) : 행 수를 조회(NULL 제외)
SUM() : 합계를 계산
AVG() : 평균을 계산
STDDEV() : 표준 편차
VARIAN() : 분산
MAX() / MIN() : 최댓값 / 최솟값

---

● **명시적 형변환 :** 형변환 함수를 사용해서 데이터 타입을 일치시키는 것

- 인덱스칼럼에 형변환을 수행 시 인덱스를 사용할 수 없다.
- 인덱스는 기본적으로 변형이라는 것이 발생하면 인덱스를 사용할 수 없다(물론 예외는 있음)

● **형변환 함수**

(1) TO_NUMBER(문자열) : 문자열을 숫자로 변환
(2) TO_CHAR(숫자 혹은 날짜, [FORMAT]) : 숫자 혹은 문자를 지정된 FORMAT의 문자로 변환
(3) TO_DATE(문자열, FORMAT) : 문자열을 지정된 FORMAT의 날짜형으로 변환

● **내장형 함수(Built-In Function)**

DUAL 테이블
- Oracle DB에 의해서 자동으로 생성되는 테이블
- DB의 모든 사용자가 사용할 수 있다.

● **문자형 함수**

- ASCII(문자) : 문자 또는 숫자의 ASCII 값 반환
- CHR(ASCII) / CHAR(ASCII) : ASCII 값에 해당하는 문자 반환
- SUBSTR(문자열, m, n) : 문자열에서 m번째 위치부터 n개를 자른다. (SQL Server는 SUBSTRING)
- CONCAT(문자열1, 문자열2) : 문자열1, 2를 결합
- LOWER(문자열) : 영문자를 소문자로
- UPPER(문자열) : 영문자를 대문자로
- LENGTH(문자열) / LEN(문자열) : 문자열의 길이(공백포함)
- LTRIM(문자열, 지정문자) : 왼쪽에 지정된 문자 삭제
- RTRIM(문자열, 지정문자) : 오른쪽에 지정된 문자 삭제
- TRIM(문자열, 지정문자) : 양쪽에 지정된 문자를 삭제

(LTRIM, RTRIM, TRIM 에 지정문제 생략 시 공백을 삭제)

● **날짜형 함수**

- SYSDATE / GETDATE() : 현재날짜와 시각 출력
- EXTRACT / DATEPART : 날짜에서 데이터 출력

ex) EXTRACT('YEAR' | 'MONTH' | 'DAY' from SYSDATE)
: 날짜에서 년, 월, 일 조회

● **숫자형 함수**

ABS(n) : 절대값 반환
SIGN(n) : 양수는 1, 음수는 -1, 0은 0 반환
MOD(n1, n2) : n1 을 n2로 나눈 나머지 반환
CEIL/CEILING(n) : 올림 (크거나 같은 최소 정수 반환)
FLOOR(n) : 버림 (작거나 같은 최대 정수 반환)
ROUND(n, m) : 소수점 m 자리에서 반올림
TRUNC(n, m) : 소수점 m 자리에서 절삭

CEIL(38.433) → 39
FLOOR(38.433) → 38
ROUND(38.433) → 38
TRUNC(38.433) → 38

● **CASE**

(1) SIMPLE_CASE_EXPRESSION : CASE 다음에 바로 조건에 사용되는 칼럼이나 표현식을 표시

ex) CASE LOC WHEN 'a' THEN 'b'

(2) SEARCHED_CASE_EXPRESSION : WHEN 절에서 EQUI 조건을 포함한 여러 조건을 사용 가능

ex) CASE WHEN LOC = 'a' THEN LOC = 'b'

● **DECODE :** 특정 조건이 참이면 A, 거짓이면 B로 응답

ex) DECODE(empno, 1000, 'true', 'false')
→ empno = 1000 이면 true 아니면 false

---

● **ROWNUM**

- SELECT문의 결과에 대해서 논리적인 일련번호를 부여한다.
- 조회되는 행 수를 제한할 때 많이 사용된다.
- ROWNUM을 사용해서 한 개의 행을 가지고 올 수 있으나, 여러 개의 행을 가지고 올 때는 인라인 뷰를 사용해야 한다.

● **ROWID**

- Oracle DB 내에서 데이터를 구분할 수 있는 유일한 값
- 데이터가 어떤 데이터 파일, 어떤 블록에 저장되어 있는지를 알 수 있다.

**● ROWID 구조**

- 오브젝트 번호(1~6) : 오브젝트 별로 유일한 값을 가지고 있으며, 해당 오브젝트가 속해있는 값
- 상대 파일번호(7~9) : 테이블스페이스에 속해 있는 데이터 파일에 대한 상대 파일번호
- 블록 번호(10~15) : 데이터 파일 내부에서 어느 블록에 데이터가 있는지 알려준다.
- 데이터 번호(16~18) : 데이터 블록에 데이터가 저장되어 있는 순서를 의미

**● WITH**

- WITH 구문은 서브쿼리를 사용해서 임시 테이블이나 뷰처럼 사용할 수 있는 구문이다.
- 서브쿼리 블록에 별칭을 지정할 수 있다.
- 옵티마이저는 SQL을 인라인 뷰나 임시 테이블로 판단한다.

ex) 1> WITH viewData AS (SELECT * FROM EMP)
2> SELECT * FROM view Data

√ WITH TIES

ex) SELECT top(1) WITH TIES ename, sal FROM emp
ORDER BY sal DESC

가장 급여가 높은 1명을 출력한다.
단, 같은 급여를 받는 사원도 출력한다.

**● SELECT 문 실행 순서**

FROM → WHERE → GROUP BY →
HAVING → SELECT → ORDER BY

**● 연산자 우선순위**

() → NOT → 비교연산자 → AND → OR

---

**● DCL (Data Control Language) : GRANT / REVOKE**

**● GRANT :** DB 사용자에게 권한을 부여한다.

GRANT privileges ON object To user;

- privileges 는 권한, object는 테이블, user는 사용자

ex) GRANT select, update ON emp To kim
: kim 에게 emp 테이블의 조회,수정 권한 부여

**● Pribileges(권한)**

(1) SELECT/INSERT/UPDATE/DELETE : 조회/삽입/수정/삭제
(2) REFERENCES : 지정된 테이블을 참조하는 제약조건을 생성하는 권한을 부여
(3) ALTER : 지정된 테이블의 정의를 수정할 수 있는 권한
(4) INDEX : 지정된 테이블에 인덱스를 생성할 수 있는 권한
(5) ALL : 테이블에 대한 모든 권한

**● WITH GRANT OPTION**

(1) WITH GRANT OPTION : 특정 사용자에게 권한을 부여할 수 있는 권한을 부여

- A가 B에 권한 부여하고, B가 C에 권한 부여한 후 B의 권한을 회수하면 C의 권한까지 회수된다.

(2) WITH ADMIN OPTION : 테이블에 대한 모든 권한 부여

- A가 B에 권한 부여하고, B가 C에 권한 부여한 후 B의 권한을 회수하면 B의 권한만 회수된다.

**● REVOKE :** DB 사용자에게 부여된 권한을 회수한다.

REVOKE privileges ON object FROM user ;

---

**● TCL (Transaction Control Language)**

- 트랜잭션 : 밀접히 관련되어 분리될 수 없는 1개 이상의 DB조작
- COMMIT, ROLLBACK, SAVEPOINT

**● COMMIT**

- INSERT, UPDATE, DELETE 문으로 변경한 데이터를 DB에 반영한다.
- 변경 전 이전 데이터는 잃어버린다.
- DB 변경으로 인한 LOCK이 해제된다.
- COMMIT을 실행하면 하나의 트랜잭션 과정을 종료한다.

√ Auto Commit

- SQL*PLUS 프로그램을 정상적으로 종료하는 경우 자동 Commit
- DDL 및 DCL을 사용하는 경우 자동 Commit
- set autocommit on; 을 실행하면 자동 commit

**● ROLLBACK**

- 데이터에 대한 변경을 모두 취소하고 트랜잭션을 종료
- 트랜잭션 시작 이전의 상태로 되돌림
- ROLLBACK을 실행하면 LOCK이 해제된다.

**● SAVEPOINT**

- 트랜잭션을 작게 분할하여 관리
- 저장 지점, 현 시점에서 SAVEPOINT까지 트랜잭션의 일부만 ROLLBACK 가능

ex) SAVEPOINT <SAVEPOINT명> (Oracle)
ex) ROLLBACK TO <SAVEPOINT명> (Oracle)
ex) SAVE TRANSACTION <SAVEPOINT명> (SQL Server)
ex) ROLLBACK TRANSACTION <SAVEPOINT명> (SQL Server)

---

● **JOIN**

- 두 개 이상의 테이블을 결합하여 데이터를 출력하는 것
- 일반적으로 행들은 PK나 FK 값의 연관에 의해 JOIN이 성립된다.
- 어떤 경우에는 PK, FK 관계가 없어도 논리적인 값들의 연관만으로도 JOIN이 성립가능하다.

√ 테이블 개수에 따른 조인 횟수

- 5개의 테이블을 조인하기 위한 최소의 조인 횟수는 4이다.
- 최소의 조인 횟수 = (테이블수 – 1)

● **EQUI JOIN(동등 조인)** : 2개의 테이블 간에 칼럼 값들이 서로 정확하게 일치하는 경우 사용한다. 대부분 PK, FK의 관계를 기반으로 한다.

● **NON EQUI JOIN(비동등 조인)** : 동등 조인은 '=' 연산자를 사용한다. '=' 연산자가 아닌 다른 연산자를 사용한 경우는 모두 비동등 조인이다.

● **NATURAL JOIN** : WHERE절에서 조인 조건 추가 불가

- OWNER명 사용 불가 ex) PLAYER.NAME (X) / NAME (O)

● **CROSS JOIN** : WHERE절에서 JOIN 조건 추가 가능

**(조인에 대해선 다음 장에서 자세히 설명)**