

SQL - DDL

문혜영

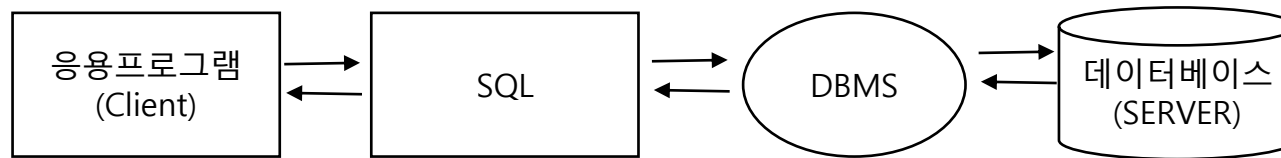
목차

- SQL 개요
- D D L
- 평가

SQL 개요

1. SQL 개념

- SQL(Structured Query Language)은 IBM이 1970년대 초에 SEQUEL(Structured English Query Language)로 개발-> SQL로 변경
- 1987년에 ISO(국제 표준화 기구)에서 SQL을 표준으로 제정



2. SQL 특징

- 문법이 사용하기 쉽습니다.
- 실행 순서와 상관없이 처리 내용을 명령할 수 있습니다.
- 데이터의 검색, 조작, 정의가 용이합니다.
- 표준 언어로 구성되어 있습니다.

3. 프로그래밍언어와 비교

프로그래밍 언어	SQL 언어
기계와의 의사소통 또는 명령을 위한 언어	데이터베이스에서 원하는 데이터를 조작하기 위한 질의 언어
각 언어마다 고유한 명령어를 습득	DBMS마다 조금은 다르지만 ANSI표준의 SQL 사용으로 기본을 익히면 다른 DBMS 사용 용이

4. SQL 명령어 종류

- DDL(Data Definition Language) : CREATE, ALTER, DROP, RENAME, TRUNCATE(테이블 정의와 구조 수정)
- DML(Data Manipulation Language) : SELECT, INSERT, UPDATE, DELETE(검색, 삽입, 수정, 삭제)
- DCL(Data Control Language) : GRANT, REVOKE(접근 권한 관리)
- Cf) TCL(Transaction Control Language) : COMMIT, ROLLBACK, SAVEPOINT(DML에서 실행한 사항을 관리)

- SQL 문 작성 규칙
 - SQL 문은 대·소문자 구분하지 않음(SELECT와 select는 동일)
 - 한 줄 또는 여러 줄로 작성할 수 있음
 - 문장 마지막은 세미콜론(;)으로 끝남
 - 명령어, 객체명, 변수명은 대/소문자 구분이 없음 -> 명령어는 대문자로, 나머지는 소문자로 작성하는 게 좋음
 - 데이터 값은 대/소문자를 구분함
 - 날짜와 문자열에는 작은 따옴표를 사용
 - 단어와 단어 사이는 공백 또는 줄바꿈으로 구분
 - 주석문
 - -- 이것은 주석입니다.
 - /* 여기부터
 - 여기까지 주석입니다. */

- 테이블 생성 규칙

- 테이블명
 - 객체를 의미할 수 있는 이름으로, **단수형**을 권고함
 - 다른 테이블의 이름과 **중복**되지 **않**아야 함
- 칼럼명
 - 한 테이블 내에서는 칼럼명이 중복되지 않아야 함
 - 테이블 생성시 각 칼럼들은 괄호 내에서 콤마로 구분됨
 - 칼럼 뒤에 데이터 유형이 반드시 지정되어야 함
- 테이블명 & 칼럼명
 - 사전에 정의된 예약어(Reserved word)는 사용 불가
 - 테이블명과 칼럼명에는 **문자, 숫자, 일부 기호**(_, \$, #)만 허용됨
 - 테이블명과 칼럼명은 **반드시 문자로 시작**해야 함 (숫자, 기호 불가)
- 제약조건명: 다른 제약조건의 이름과 중복되지 않아야 함

- Oracle의 주요 데이터 타입

Type	설명
CHAR	<ul style="list-style-type: none"> 고정 문자열 (기본 1Byte ~ 최대 2,000Byte) 변수에 할당된 값이 고정 길이보다 작은 경우, 나머지 공간에 공백이 채워짐 (참고: 'AA' = 'AA ')
VARCHAR2	<ul style="list-style-type: none"> 가변 길이 문자열 (기본 1Byte ~ 최대 4,000Byte) 변수에 할당된 값이 고정 길이보다 작은 경우, 나머지 공간은 사용하지 않음 (참고: 'AA' ≠ 'AA ')
NUMBER	<ul style="list-style-type: none"> 정수, 실수 등의 숫자 정보 MS SQL Server의 경우 10가지 이상의 숫자 타입 지원 전체 자리수와 소수 부분 자리수를 명시함 예: NUMBER(8,2) → 전체자리수 8, 소수 부분 2자리
DATE	<ul style="list-style-type: none"> 날짜와 시각 정보

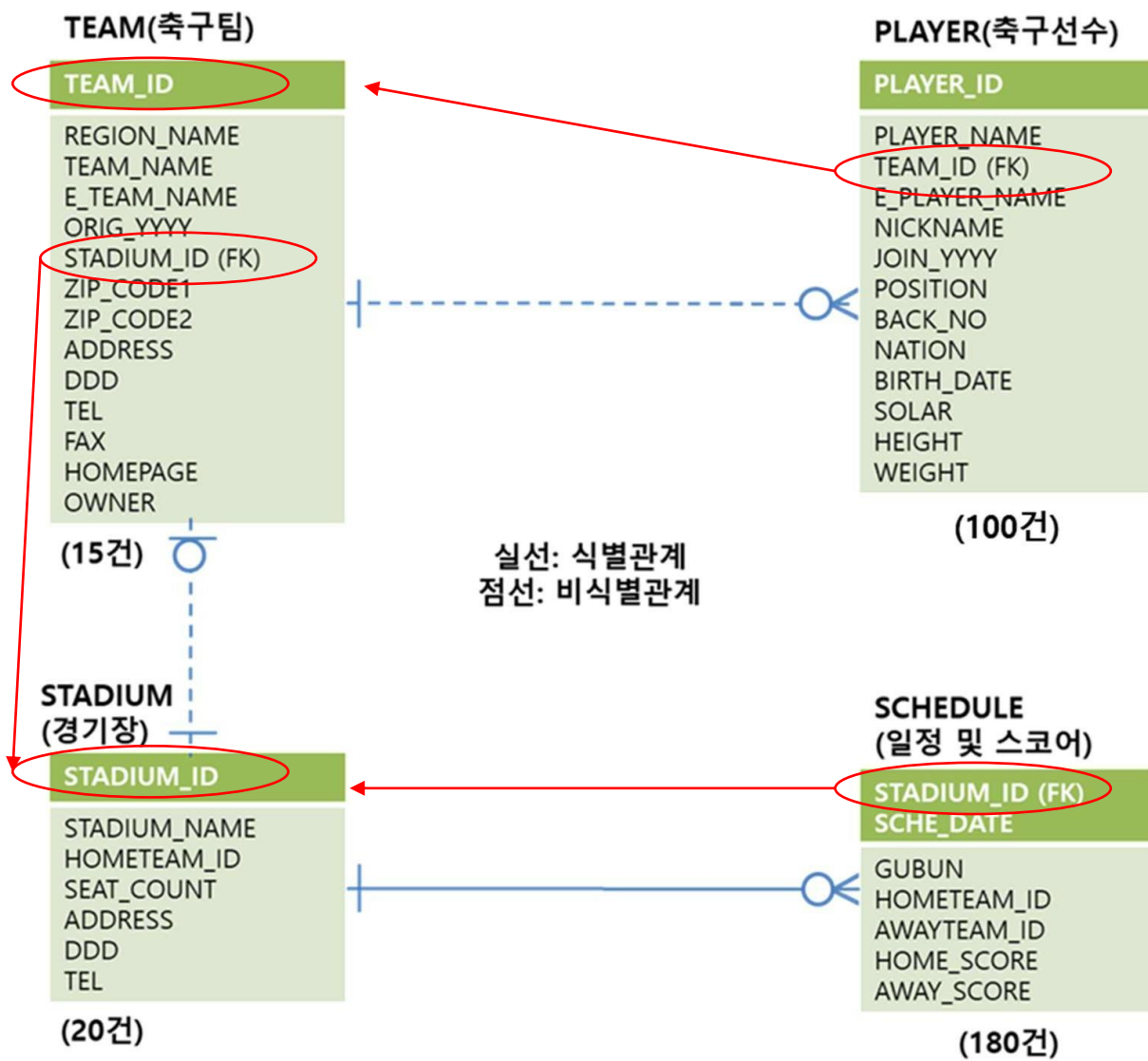
● 제약 조건 (Constraints)

구분	설명
PRIMARY KEY (기본키)	<ul style="list-style-type: none"> 기본키를 정의함 (하나의 테이블에 하나의 기본키 제약만 정의 가능) NOT NULL & UNIQUE
FOREIGN KEY (외래키)	<ul style="list-style-type: none"> 다른 테이블의 기본키를 참조 외래키 지정시 참조 무결성 제약 옵션 선택
NOT NULL	<ul style="list-style-type: none"> NULL 값의 입력을 허용하지 않는 제약 조건
UNIQUE	<ul style="list-style-type: none"> 해당 칼럼의 값이 테이블 내에서 유일해야 함을 제약하는 조건
CHECK	<ul style="list-style-type: none"> 입력할 수 있는 값을 제한하는 조건 (예: CONSTRAINT BACK_NO_CK CHECK (BACK_NO < 99))

- 제약 조건명을 명시적으로 부여할 수도 있고, 묵시적으로 제약 조건명 없이 제약 조건을 설정할 수도 있음

- 예제 데이터 (K-리그)

테이블	칼럼
PLAYER	선수ID , 선수명, 소속팀ID, 영문선수명, 선수별명, 입단년도, 포지션, 등번호, 국적, 생년월일, 양/음, 키, 몸무게
TEAM	팀ID , 연고지명, 팀명, 영문팀명, 창단년도, 운동장ID, 우편번호1, 우편번호2, 주소, 지역번호, 전화 번호, 팩스, 홈페이지, 구단주
STADIUM	운동장ID , 운동장명, 홈팀ID, 좌석수, 주소, 지역번호, 전화번호
SCHEDULE	운동장ID , 경기일자 , 경기진행여부, 홈팀ID, 원정 팀ID, 홈팀득점, 원정팀 득점



- 테이블 생성 SQL문 - STADIUM

```
CREATE TABLE STADIUM (  
    STADIUM_ID      CHAR(3) NOT NULL,  
    STADIUM_NAME    VARCHAR2(40) NOT NULL,  
    HOMETEAM_ID    CHAR(3),  
    SEAT_COUNT      NUMBER,  
    ADDRESS         VARCHAR2(60),  
    DDD            VARCHAR2(3),  
    TEL            VARCHAR2(10),  
    CONSTRAINT     STADIUM_PK PRIMARY KEY (STADIUM_ID)  
);
```

- 테이블 생성 SQL문 - TEAM

```
CREATE TABLE TEAM (  
    TEAM_ID          CHAR(3) NOT NULL,  
    REGION_NAME      VARCHAR2(8) NOT NULL,  
    TEAM_NAME        VARCHAR2(40) NOT NULL,  
    E_TEAM_NAME      VARCHAR2(50),  
    ORIG_YYYY        CHAR(4),  
    STADIUM_ID       CHAR(3) NOT NULL,  
    ZIP_CODE1        CHAR(3),  
    ZIP_CODE2        CHAR(3),  
    ADDRESS          VARCHAR2(80),  
    DDD              VARCHAR2(3),  
    TEL              VARCHAR2(10),  
    FAX              VARCHAR2(10),  
    HOMEPAGE         VARCHAR2(50),  
    OWNER            VARCHAR2(10),  
    CONSTRAINT TEAM_PK PRIMARY KEY (TEAM_ID),  
    CONSTRAINT TEAM_FK FOREIGN KEY (STADIUM_ID) REFERENCES STADIUM(STADIUM_ID)  
);
```

STADIUM 테이블이 먼저 생성되어 있어야함



- 테이블 생성 SQL문 - SCHEDULE

```
CREATE TABLE SCHEDULE (  
    STADIUM_ID      CHAR(3)    NOT NULL,  
    SCHE_DATE       CHAR(8)    NOT NULL,  
    GUBUN           CHAR(1)    NOT NULL,  
    HOMETEAM_ID     CHAR(3)    NOT NULL,  
    AWAYTEAM_ID     CHAR(3)    NOT NULL,  
    HOME_SCORE      NUMBER(2),  
    AWAY_SCORE      NUMBER(2),  
    CONSTRAINT      SCHEDULE_PK PRIMARY KEY (STADIUM_ID, SCHE_DATE),  
    CONSTRAINT      SCHEDULE_FK FOREIGN KEY (STADIUM_ID) REFERENCES STADIUM(STADIUM_ID)  
);
```

STADIUM 테이블이 먼저 생성되어 있어야 함

조건이름

외래키

STADIUM 테이블의 STADIUM_ID 를 참조함

- 테이블 생성 SQL문 - PLAYER

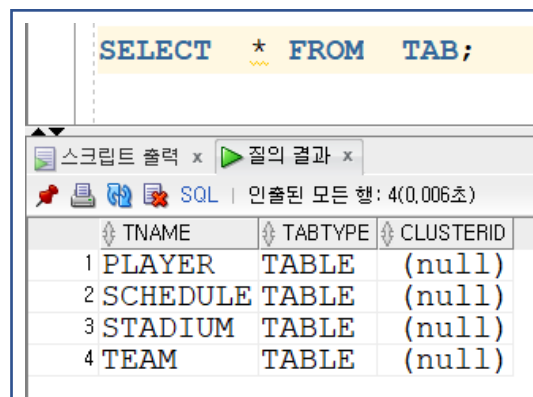
```
CREATE TABLE PLAYER (  
    PLAYER_ID    CHAR(7) NOT NULL,  
    PLAYER_NAME  VARCHAR2(20) NOT NULL,  
    TEAM_ID      CHAR(3) NOT NULL,  
    E_PLAYER_NAME VARCHAR2(40),  
    NICKNAME     VARCHAR2(30),  
    JOIN_YYYY    CHAR(4),  
    POSITION     VARCHAR2(10),  
    BACK_NO      NUMBER(2),  
    NATION       VARCHAR2(20),  
    BIRTH_DATE   DATE,  
    SOLAR        CHAR(1),  
    HEIGHT       NUMBER(3),  
    WEIGHT       NUMBER(3),  
    CONSTRAINT PLAYER_PK PRIMARY KEY (PLAYER_ID),  
    CONSTRAINT PLAYER_FK FOREIGN KEY (TEAM_ID) REFERENCES  
    TEAM(TEAM_ID)  
);
```

TEAM 테이블이 먼저 생성되어 있어야 함



DDL

- 테이블 목록 확인
 - `SELECT * FROM TAB;`



The screenshot shows a SQL query window with the command `SELECT * FROM TAB;` and its results. The results are displayed in a table with three columns: TNAME, TABTYPE, and CLUSTERID. There are four rows of data, all representing tables.

	TNAME	TABTYPE	CLUSTERID
1	PLAYER	TABLE	(null)
2	SCHEDULE	TABLE	(null)
3	STADIUM	TABLE	(null)
4	TEAM	TABLE	(null)

- 테이블 구조 확인
 - `DESCRIBE PLAYER;`



The screenshot shows a SQL query window with the command `DESCRIBE PLAYER;` and its results. The results are displayed in a table with three columns: 이름 (Name), 널? (Null?), and 유형 (Type). There are 14 rows of data, each representing a column in the PLAYER table.

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)

- 제약 조건의 지정

```
DROP TABLE PLAYER1 CASCADE CONSTRAINT;  
DROP TABLE PLAYER2 CASCADE CONSTRAINT;  
DROP TABLE PLAYER3 CASCADE CONSTRAINT;
```

```
CREATE TABLE PLAYER1(  
    PLAYER_ID          CHAR(7)          PRIMARY KEY,  
    PLAYER_NAME        VARCHAR2(20) NOT NULL,  
    NICKNAME           VARCHAR2(30) UNIQUE,  
    HEIGHT             NUMBER(3) CHECK (HEIGHT >= 150 AND HEIGHT <= 200),  
    TEAM_ID            CHAR(3)          REFERENCES TEAM(TEAM_ID)  
);  
  
CREATE TABLE PLAYER2(  
    PLAYER_ID          CHAR(7)          CONSTRAINT p2_pk_id PRIMARY KEY,  
    PLAYER_NAME        VARCHAR2(20) CONSTRAINT p2_nn_name NOT NULL,  
    NICKNAME           VARCHAR2(30) CONSTRAINT p2_un_nick UNIQUE,  
    HEIGHT             NUMBER(3) CONSTRAINT p2_ck_height CHECK (HEIGHT >= 150 AND HEIGHT <= 200),  
    TEAM_ID            CHAR(3)          CONSTRAINT p2_fk_tid REFERENCES TEAM(TEAM_ID)  
);
```


- 제약 조건의 지정 (cont'd)

```
CREATE TABLE PLAYER3(
    PLAYER_ID    CHAR(7),
    PLAYER_NAME  VARCHAR2(20) CONSTRAINT p3_nn_name NOT NULL,
    NICKNAME     VARCHAR2(30),
    HEIGHT       NUMBER(3),
    TEAM_ID      CHAR(3),

    CONSTRAINT p3_pk_id    PRIMARY KEY (PLAYER_ID),
    CONSTRAINT p3_un_nick  UNIQUE (NICKNAME),
    CONSTRAINT p3_ck_height CHECK (HEIGHT >= 150 AND HEIGHT <= 200),
    CONSTRAINT p3_fk_tid   FOREIGN KEY (TEAM_ID) REFERENCES TEAM(TEAM_ID)

);
```

제약조건의확인

```
SELECT * FROM ALL_CONSTRAINTS;
```

```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME IN ('PLAYER1', 'PLAYER2')
ORDER BY TABLE_NAME;
```

SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME IN ('PLAYER1', 'PLAYER2') ORDER BY TABLE_NAME;

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1 YYID	SYS C007105	R	PLAYER1	(null)
2 YYID	SYS C007103	P	PLAYER1	(null)
3 YYID	SYS C007104	U	PLAYER1	(null)
4 YYID	SYS C007102	C	PLAYER1	HEIGHT >= 150 AND HEIGHT <= 200
5 YYID	SYS C007101	C	PLAYER1	'PLAYER NAME' IS NOT NULL
6 YYID	P2 PK ID	P	PLAYER2	(null)
7 YYID	P2 UN NICK	U	PLAYER2	(null)
8 YYID	P2 CK HEIGHT	C	PLAYER2	HEIGHT >= 150 AND HEIGHT <= 200
9 YYID	P2 NN NAME	C	PLAYER2	'PLAYER NAME' IS NOT NULL
10 YYID	P2 FK TID	R	PLAYER2	(null)

- FK 제약 조건의 옵션

- 예) CONSTRAINT fk1 FOREIGN KEY (TEAM_ID) REFERENCES TEAM(Team_ID)

- ON DELETE CASCADE ON UPDATE RESTRICT;

- Referential Triggered Action

- ON UPDATE, ON DELETE

- Referential Action

- RESTRICT(Default): 기본값의 삭제 또는 갱신을 불허

- NO ACTION: RESTRICT와 동일하게 동작

- CASCADE:

- 기본키가 삭제되면 해당 값을 외래키로 갖는 레코드도 삭제

- 기본키가 갱신되면 이를 참조하는 외래키를 새로운 값으로 업데이트

- SET NULL

- 기본키가 삭제 또는 갱신되면 이를 참조하는 외래키를 NULL로 업데이트

- **FOREIGN KEY**

- 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 튜플 **삭제 시** 처리 방법을 지정하는 옵션
 - ON DELETE NO ACTION : 튜플을 삭제하지 못하게 함
 - ON DELETE CASCADE : 관련 튜플을 함께 삭제함
 - ON DELETE SET NULL : 관련 튜플의 외래키 값을 NULL로 변경함
 - ON DELETE SET DEFAULT : 관련 튜플의 외래키 값을 미리 지정한 기본 값으로 변경함
- 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 튜플 **변경 시** 처리 방법을 지정하는 옵션
 - ON UPDATE NO ACTION : 튜플을 변경하지 못하게 함
 - ON UPDATE CASCADE : 관련 튜플에서 외래키 값을 함께 변경함
 - ON UPDATE SET NULL : 관련 튜플의 외래키 값을 NULL로 변경함
 - ON UPDATE SET DEFAULT : 관련 튜플의 외래키 값을 미리 지정한 기본 값으로 변경함
- 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)
- 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)
ON DELETE CASCADE ON UPDATE CASCADE

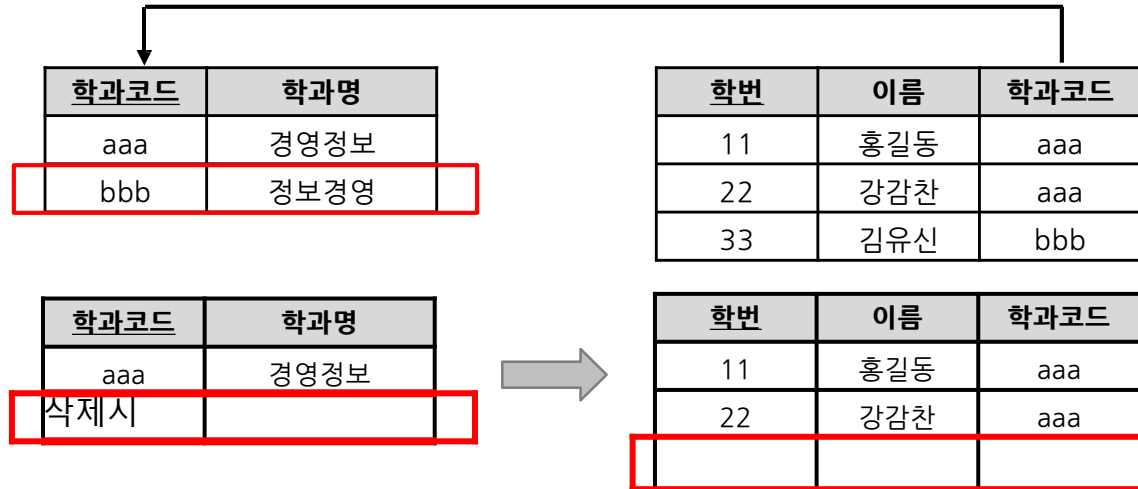
- 참조 무결성 제약조건 유지를 위한 튜플 삭제 예]



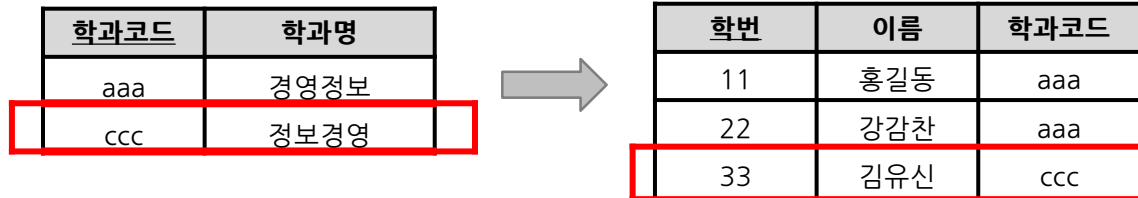
- ON DELETE NO ACTION : 부서 테이블의 튜플을 삭제하지 못하게 함
- ON DELETE CASCADE : 사원 테이블에서 홍보부에 근무하는 정소화 사원 튜플도 함께 삭제
- ON DELETE SET NULL : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 NULL로 변경
- ON DELETE SET DEFAULT : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 기본 값으로 변경

- FK 제약 조건의 옵션 (cont'd)

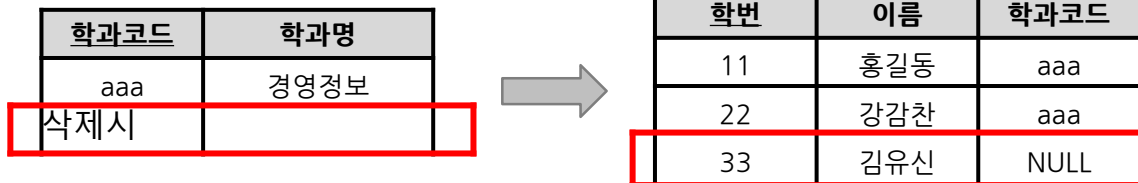
- ON DELETE CASCADE



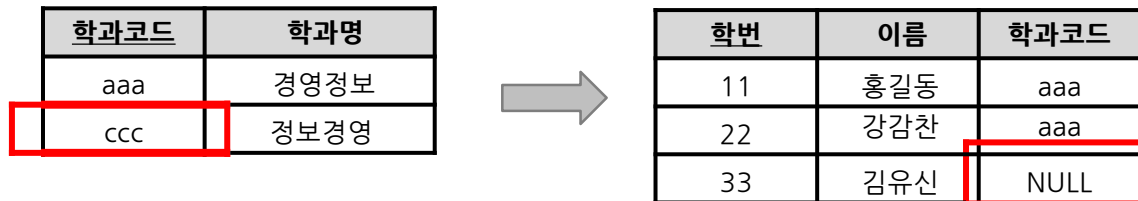
- ON UPDATE CASCADE



- ON DELETE SET NULL



- ON UPDATE SET NULL



- 기존 테이블을 활용한 테이블 생성 (SELECT 문 활용)

```
DROP TABLE PLAYER_TEMP CASCADE CONSTRAINT;
CREATE TABLE PLAYER_TEMP
AS SELECT * FROM PLAYER;
```

- 제약 조건은 NOT NULL만 복제됨
 - PK, FK, UNIQUE, CHECK 등은 수동으로 추가해야 함

```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME IN ('PLAYER', 'PLAYER_TEMP')
ORDER BY TABLE_NAME;
```

→

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1 MYID	PLAYER_FK	R	PLAYER	(null)
2 MYID	SYS_C007199	C	PLAYER	"PLAYER_ID" IS NOT NULL
3 MYID	SYS_C007200	C	PLAYER	"PLAYER_NAME" IS NOT NULL
4 MYID	SYS_C007201	C	PLAYER	"TEAM_ID" IS NOT NULL
5 MYID	PLAYER_PK	P	PLAYER	(null)
6 MYID	SYS_C007225	C	PLAYER_TEMP	"PLAYER_ID" IS NOT NULL
7 MYID	SYS_C007226	C	PLAYER_TEMP	"PLAYER_NAME" IS NOT NULL
8 MYID	SYS_C007227	C	PLAYER_TEMP	"TEAM_ID" IS NOT NULL

- ALTER 문을 통한 테이블 변경

- 칼럼의 추가/삭제/수정, 제약 조건의 추가/삭제
- (1) 칼럼의 추가 (ADD)
 - 새로 추가한 칼럼은 테이블의 맨 마지막에 **추가**됨

```
ALTER TABLE PLAYER_TEMP ADD (ADDRESS VARCHAR2(80));
DESCRIBE PLAYER_TEMP;
```

작업이 완료되었습니다.(0.024초)

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)



작업이 완료되었습니다.(0.03초)

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)
ADDRESS		VARCHAR2(80)

– (2) 칼럼의 삭제 (DROP COLUMN)

- 삭제 후 최소 하나 이상의 칼럼이 테이블에 존재해야 함

```
ALTER TABLE PLAYER_TEMP DROP COLUMN ADDRESS;  
DESCRIBE PLAYER_TEMP;
```

작업이 완료되었습니다.(0.03초)

이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)
ADDRESS		VARCHAR2(80)



작업이 완료되었습니다.(0.024초)

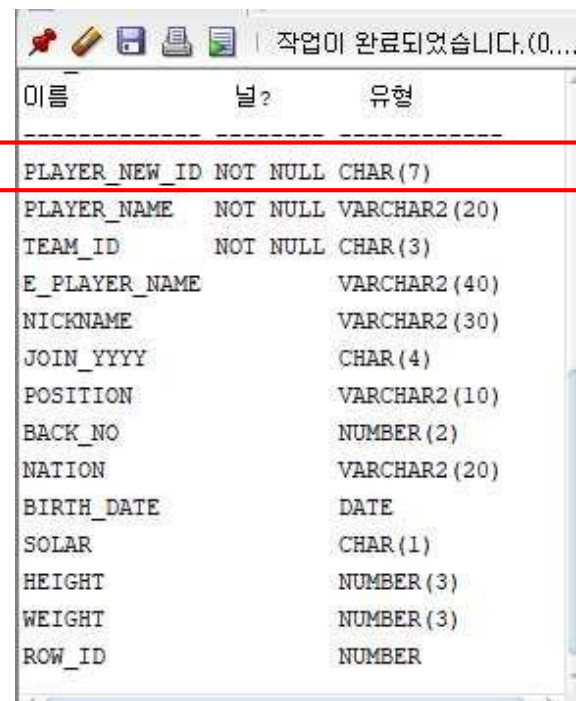
이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)

- (3) 칼럼명 변경 (RENAME COLUMN)
 - 해당 칼럼의 모든 정의가 그대로 유지됨

```
ALTER TABLE PLAYER_TEMP RENAME COLUMN PLAYER_ID TO PLAYER_NEW_ID;  
DESCRIBE PLAYER_TEMP;
```



이름	널?	유형
PLAYER_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)
ROW_ID		NUMBER



이름	널?	유형
PLAYER_NEW_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)
NATION		VARCHAR2(20)
BIRTH_DATE		DATE
SOLAR		CHAR(1)
HEIGHT		NUMBER(3)
WEIGHT		NUMBER(3)
ROW_ID		NUMBER

- (4) 칼럼의 정의 수정 (MODIFY)
 - 이미 입력되어 있는 값에 **영향**을 미치는 변경은 허용하지 않음
 - 데이터 타입 변경
 - 테이블에 아무 행도 없거나, 해당 칼럼이 NULL만 갖고 있을 때 가능
 - 칼럼의 크기 변경
 - 칼럼의 크기 **확대** → 항상 가능
 - 칼럼의 크기 축소 → 테이블에 아무 행도 없거나, 해당 칼럼이 NULL만 갖고 있거나, 현재 저장된 값을 수용할 수 있는 크기로의 축소만 가능
 - DEFAULT 값 추가 및 수정
 - 추가 및 수정 이후 삽입되는 행에만 영향을 미침

- (4) 칼럼의 정의 수정 (MODIFY) - cont'd
 - NOT NULL 제약조건 **추가**
 - 테이블에 아무 행도 없거나, 해당 칼럼에 NULL이 없을 때 가능
 - NOT NULL 제약조건 **삭제** → 항상 가능
 - NOT NULL 제약조건 추가 / 삭제
 - ALTER TABLE 테이블명 MODIFY (속성명 NOT NULL / NULL)
- Q) **PLAYER_TEMP** 테이블에서 **PLAYER_NAME** 속성이 NULL값을 허용하도록 정의를 변경하시오.

```
ALTER TABLE PLAYER_TEMP MODIFY (PLAYER_NAME NULL) ;
DESC          PLAYER_TEMP ;
```

작업이 완료되었습니다.(0..

이름	널?	유형
PLAYER_NEW_ID	NOT NULL	CHAR(7)
PLAYER_NAME	NOT NULL	VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)



작업이 완료되었습니다.(0..

이름	널?	유형
PLAYER_NEW_ID	NOT NULL	CHAR(7)
PLAYER_NAME		VARCHAR2(20)
TEAM_ID	NOT NULL	CHAR(3)
E_PLAYER_NAME		VARCHAR2(40)
NICKNAME		VARCHAR2(30)
JOIN_YYYY		CHAR(4)
POSITION		VARCHAR2(10)
BACK_NO		NUMBER(2)

– (5) 제약 조건의 추가/삭제 (ADD/DROP CONSTRAINT)

- 테이블 생성 이후에도 제약 조건을 추가/삭제할 수 있음
- 예) PLAYER_TEMP → TEAM에 대한 FK

```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME = 'PLAYER_TEMP';
```

SQL | 인출된 모든 행: 2(0초)

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1	MYID	SYS_C007225	C	PLAYER_TEMP	"PLAYER_NEW_ID" IS NOT NULL
2	MYID	SYS_C007227	C	PLAYER_TEMP	"TEAM_ID" IS NOT NULL

ALTER TABLE PLAYER_TEMP
ADD CONSTRAINT PLAYER_TEMP_FK
FOREIGN KEY (TEAM_ID) REFERENCES TEAM(Team_ID);

ALTER TABLE PLAYER_TEMP
DROP CONSTRAINT PLAYER_TEMP_FK;

SQL | 인출된 모든 행: 3(0초)

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1	MYID	PLAYER_TEMP_FK	R	PLAYER_TEMP	(null)
2	MYID	SYS_C007225	C	PLAYER_TEMP	"PLAYER_NEW_ID" IS NOT NULL
3	MYID	SYS_C007227	C	PLAYER_TEMP	"TEAM_ID" IS NOT NULL

- Q) PLAYER_TEMP 테이블에서 PLAYER_NEW_ID를 PK로 지정하시오.
(제약조건 명 : PLAYER_TEMP_PK)

```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME='PLAYER_TEMP';
```

SQL | 인출된 모든 행: 2(0초)

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1 MYID	SYS_C007225	C	PLAYER_TEMP	"PLAYER_NEW_ID" IS NOT NULL
2 MYID	SYS_C007227	C	PLAYER_TEMP	"TEAM_ID" IS NOT NULL



```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME='PLAYER_TEMP';
```

SQL | 인출된 모든 행: 3(0초)

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION
1 MYID	SYS_C007225	C	PLAYER_TEMP	"PLAYER_NEW_ID" IS NOT NULL
2 MYID	PLAYER_TEMP_PK	P	PLAYER_TEMP	(null)
3 MYID	SYS_C007227	C	PLAYER_TEMP	"TEAM_ID" IS NOT NULL



- RENAME 문을 통한 테이블 명칭 변경

- RENAME 기존 테이블명 TO 새 테이블명
- Q) PLAYER_TEMP 테이블의 명칭을 OLD_PLAYER로 변경하시오.

SQL | 인출된 모든 행: 5(0,01,

	OBJECT_NAME	OBJECT_TYPE
1	PLAYER_TEMP	TABLE
2	STADIUM	TABLE
3	TEAM	TABLE
4	PLAYER	TABLE
5	SCHEDULE	TABLE



SQL | 인출된 모든 행: 5(0,00,

	OBJECT_NAME	OBJECT_TYPE
1	SCHEDULE	TABLE
2	PLAYER	TABLE
3	TEAM	TABLE
4	STADIUM	TABLE
5	OLD_PLAYER	TABLE

- DROP 문을 통한 테이블 삭제

```
DROP TABLE TEAM;
```



```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME= ' TEAM';
```

	OWNER	CONSTRAINT NAME	CONSTRAINT TYPE	TABLE NAME
1	YYID	TEAM FK	R	TEAM
2	YYID	SYS C007165	C	TEAM
3	YYID	SYS C007166	C	TEAM
4	YYID	SYS C007167	C	TEAM
5	YYID	SYS C007168	C	TEAM
6	YYID	TEAM PK	P	TEAM

```
DROP TABLE TEAM
```

```
CASCADE CONSTRAINT;
```

관련된 제약조건을 함께 삭제

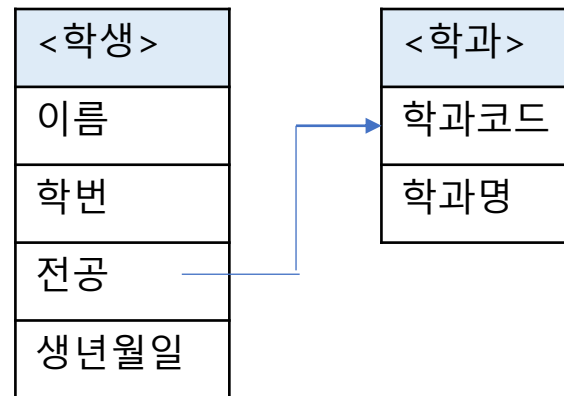
```
SELECT * FROM ALL_CONSTRAINTS
WHERE TABLE_NAME= ' TEAM';
```



- 예제) '이름', '학번', '전공', '생년월일'로 구성된 <학생> 테이블을 정의하는 SQL문을 작성하시오.

- 단, 제약 조건은 다음과 같다.

- '이름'은 NULL이 올 수 없다.
- '학번'은 기본키이다.
- '전공'은 <학과> 테이블의 '학과코드'를 참조하는 외래키로 사용된다.
- 생년월일 속성은 DATE 자료형을 갖는다.
- <학과> 테이블에서 삭제가 일어나면
관련된 튜플들의 전공 값을 NULL로 만든다.
- <학과> 테이블에서 '학과코드'가 변경되면
전공 값도 같은 값으로 변경한다.
- '생년월일'은 1980-01-01 이후의 데이터만 저장할 수 있으며
제약 조건의 이름은 '생년월일제약'으로 한다.
- 각 속성의 데이터 타입은 적당하게 지정한다.



평가

<고객>
<u>고객아이디</u>
고객이름
나이
등급
직업
적립금

<고객> 테이블 조건

- 고객아이디 속성은 기본키이다.
- 고객이름과 등급 속성은 값을 반드시 입력해야한다.
- 적립금 속성은 값을 입력하지 않으면 0이 기본으로 입력되게 하시오.

<주문>
<u>주문번호</u>
주문고객
주문제품
수량
배송지
주문일자

- <주문> 테이블 조건
- 주문번호 속성이 기본키이다.
- 주문고객 속성이 고객테이블의 고객아이디 속성을 참조하는 외래키이다.
- 주문제품 속성이 제품 테이블의 제품 번호 속성을 참조하는 외래키이다.

<제품>
<u>제품번호</u>
제품명
재고량
단가
제조업체

- <제품> 테이블 조건
- 제품번호 속성이 기본키이다.
- 재고량이 항상 0개 이상 10,000개이하를 유지하도록 제품 테이블을 생성하시오.

- 생성한 <고객> 테이블에 [가입날짜] 속성을 추가하시오.
- 추가한 <고객> 테이블의 [가입날짜] 속성을 삭제하시오.
- <고객> 테이블에 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 추가하시오. (제약조건 명 CHK_AGE)
- <고객>테이블의 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 삭제하시오(제약조건 명 CHK_AGE)

수고하셨습니다 🙌