

Structured Query Language (SQL)

Part 1. DDL

Hyeokman Kim
School of Computer Science
Kookmin Univ.

ANSI/ISO SQL

SQL STANDARD

SQL (Structured Query Language) 표준의 역사

- 1970년: Dr. E.F.Codd의 관계형 DBMS (Relational DBMS) 논문 발표
 - 1974년: IBM SEQUEL (Structured English Query Language) 개발
 - 1979년: Oracle 상용 DBMS 발표
 - 1980년: Sybase SQL Server 발표 (이후 Sybase ASE로 개명)
 - 1983년: IBM DB2 발표 (연구용 제품 System R의 상용 버전)
- 1986년: ANSI/ISO SQL 표준 최초 제정 (SQL-86, [SQL1](#))
- 1992년: ANSI/ISO SQL 표준 개정 (SQL-92, [SQL2](#))
 - 1993년: MS SQL Server 발표 (Windows OS, Sybase Code 활용)
 - 관계형 DBMS의 폭발적 성장기, 벤더별 용어 및 기능 차이가 너무 커짐.
- 1999년: ANSI/ISO SQL 표준 개정 (SQL-99, [SQL3](#))
 - 추가 필수 기능의 대폭적 정의, 벤더별 호환 가능한 표준 재정
 - Oracle의 8i/9i 버전
- 2003년: ANSI/ISO SQL 표준 개정 ([SQL-2003](#))
 - 소폭 추가 개정, 현재 대부분의 상용 DBMS
- 2008년: ANSI/ISO SQL 표준 개정 ([SQL-2008](#))
 - 아직 상용 제품에 반영되지 않음.

□ 표준

- 강제적 사용 : 국가 표준
- 사용 권고(recommendation) : 국제 표준
- Note : 국제표준이 국가표준으로 채택되어야 강제적 적용이 가능함.

□ ANSI/ISO SQL과 벤더별 SQL

- SQL 표준은 기능을 명세함. 따라서 벤더별로 표준과 다른 용어를 쓰는 것은 허용됨.
- 일반적으로 벤더별로 SQL 문장의 차이는 적어지고 있으나, 데이터 유형과 내장함수에서는 여전히 차이가 많음.

SQL의 특징 및 기능

- SELECT-FROM-WHERE의 블록 사상 (block mapping)을 이용
- SQL-2003의 대표적 기능
 - Standard Join 기능 추가 (CROSS, OUTER JOIN 등 새로운 FROM 절 JOIN 기능들)
 - Scalar subquery, top-N query 등의 새로운 subquery 기능들
 - ROLLUP, CUBE, GROUPING SETS 등의 새로운 리포팅 기능
 - WINDOW FUNCTION 같은 새로운 개념의 분석 기능들

SQL 문장의 종류

□ DDL

- CREATE
- ALTER
- DROP
- RENAME

□ DML

- INSERT, DELETE, UPDATE
- SELECT

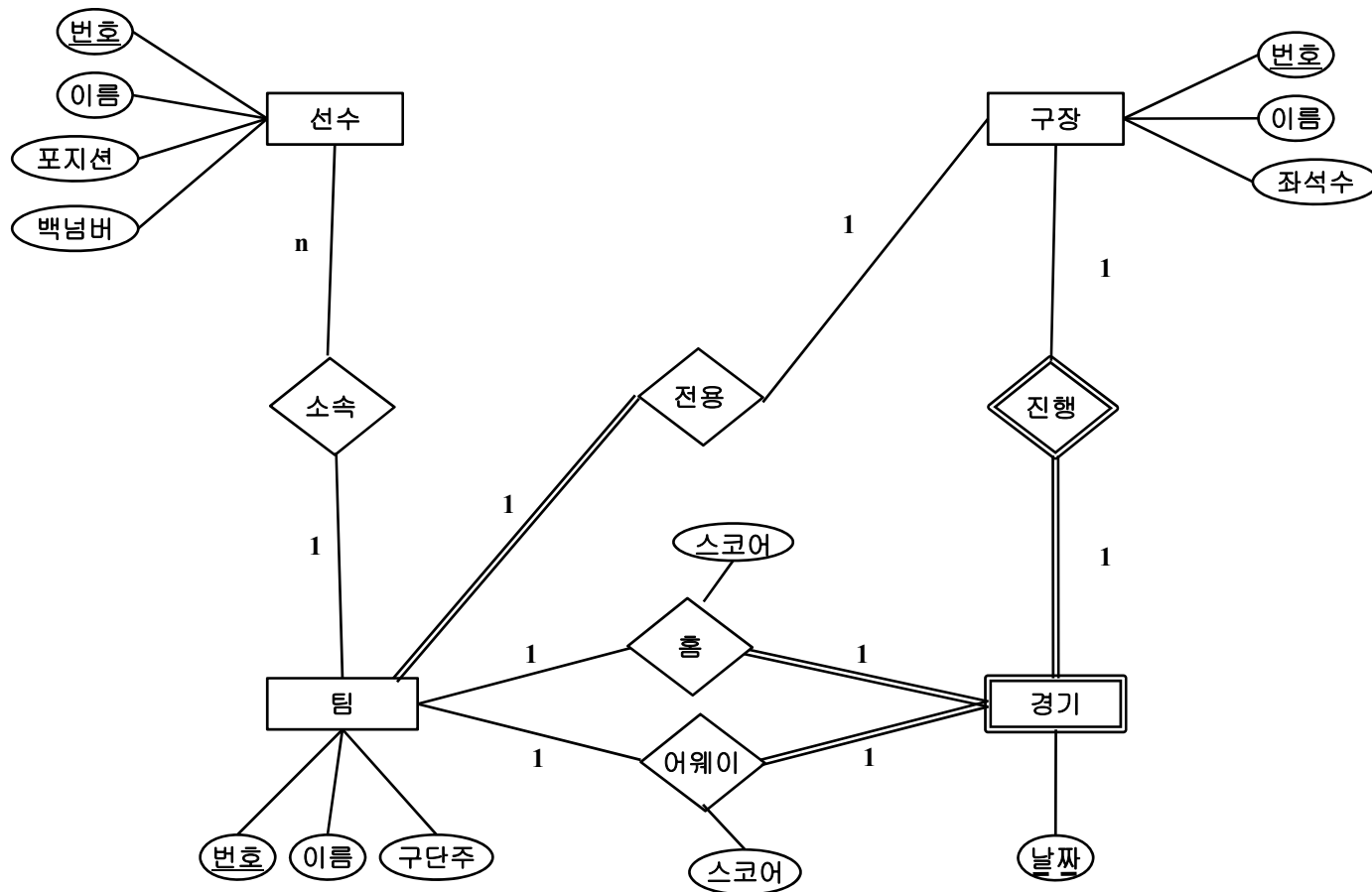
□ DCL

- GRANT
- REVOKE

□ TCL (Transaction Control Language)

- COMMIT
- ROLLBACK
- SAVEPOINT

Sample Database : K-리그



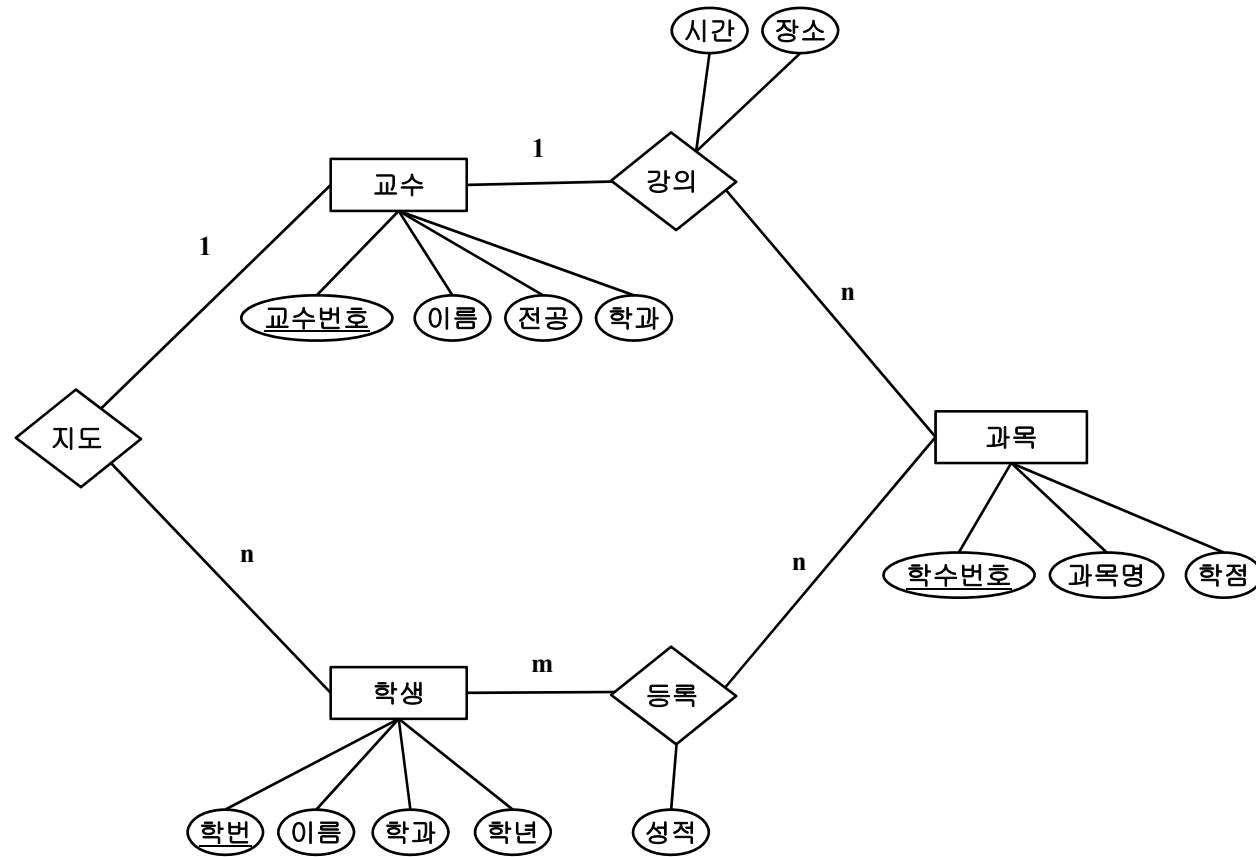
□ Schema

- 선수 (번호, 이름, 포지션, 백넘버, 소속팀번호*)
- 구장 (번호, 이름, 좌석수)
- 팀 (번호, 이름, 구단주, 전용구장번호*)
- 경기 (날짜, 구장번호*, 홈팀번호*, 어웨이팀번호*, 홈팀스코어, 어웨이팀스코어)

□ 변형된 스키마

- 선수 (번호, 이름, 포지션, 백넘버, 소속팀번호*)
- 구장 (번호, 이름, 좌석수, **홈팀번호***)
- 팀 (번호, 이름, 구단주, 전용구장번호*)
- 경기 (날짜, 구장번호*, 홈팀번호*, 어웨이팀번호*, 홈팀스코어, 어웨이팀스코어)

Sample Database : University



□ Schema

- 교수 (교수번호, 이름, 전공, 학과)
- 학생 (학번, 이름, 학과, 학년, 지도교수번호*)
- 과목 (학수번호, 과목명, 학점, 담당교수번호*, 시간, 장소)
- 등록 (학번*, 학수번호*, 성적)

ANSI/ISO SQL

SQL DDL

1. Data Type

□ 숫자

– ANSI/ISO SQL

- ◆ SMALLINT, INTEGER, INT, BIGINT
- ◆ FLOAT(n), REAL, DOUBLE PRECISION
- ◆ NUMERIC(l,j), DECIMAL(l,j), DEC(l,j)

– Oracle

- ◆ NUMBER(n), NUMBER(n,m)

– SQL Server

- ◆ ANSI/ISO SQL과 비슷 (TINYINT 추가)
- ◆ MONEY, SMALLMONEY

□ 날짜

– ANSI/ISO SQL, Oracle, SQL Server 모두 동일

- ◆ DATE
- ◆ TIME : 단위는 Oracle 1초, SQL Server 3.33ms
- ◆ TIMESTAMP : DATE & TIME
- ◆ INTERVAL

□ 고정 문자열

– ANSI/ISO SQL, Oracle, SQL Server 모두 동일

◆ CHAR(n)

◆ 최소 1, 최대 Oracle 2,000, SQL Server 8,000 바이트


□ 가변 문자열 (varying character)

– ANSI/ISO SQL

◆ VARCHAR(n)

– Oracle

◆ VARCHAR2(n) : 최소 1, 최대 4,000 바이트

◆  Do not use the VARCHAR datatype. Use the VARCHAR2 datatype instead. Although the VARCHAR datatype is currently synonymous with VARCHAR2, the VARCHAR datatype is scheduled to be redefined as a separate datatype used for variable-length character strings compared with different comparison semantics. (from Oracle 10g manual)

– SQL Server

◆ VARCHAR(n) : 최소 1, 최대 8,000 바이트

고정 문자열과 가변 문자열의 차이

□ 저장 영역

- VARCHAR는 실제 데이터 크기만 저장
- VARCHAR가 CHAR에 비해 적은 공간을 사용하는 장점

□ 문자열의 비교 방법

- CHAR : 짧은 쪽의 끝에 공백(blank)을 채워서 같은 길이를 만든 후에 비교

'AA' = 'AA '

- VARCHAR : 맨 처음부터 한 문자씩 비교

'AA' ≠ 'AA '

2. CREATE SCHEMA

□ Syntax

```
CREATE SCHEMA 스키마명 AUTHORIZATION 소유자명
```

```
CREATE SCHEMA UNIVERSITY AUTHORIZATION SHLEE;
```

□ 스키마와 카탈로그

– 스키마

- ◆ 하나의 응용(사용자)에 속하는 테이블과 기타 구성요소 등의 그룹
- ◆ 스키마 이름, 스키마 소유자 포함

– 카탈로그

- ◆ 한 SQL 환경에서의 스키마들의 집합

3. DROP SCHEMA

□ Syntax

```
DROP SCHEMA 스키마명 [RESTRICT | CASCADE];
```

```
DROP SCHEMA UNIVERSITY RESTRICT;
```

– 옵션

- ◆ **RESTRICT** : 다른 스키마에 FK를 통한 참조 무결성 제약조건을 위반하는 튜플이 하나라도 존재하면, 스키마 제거 명령이 실행되지 않음.
- ◆ **CASCADE** : 다른 스키마에 FK를 통한 무결성 제약조건을 위반하는 튜플이 하나라도 존재하면, 그 튜플들도 같이 제거함.

4. CREATE TABLE

□ 제약조건(Constraint)

- NOT NULL

- UNIQUE : 고유한 값을 갖으며, 복수개의 널 값도 허용

- PRIMARY KEY : “UNIQUE” + “NOT NULL” 제약조건

- FOREIGN KEY : 참조 무결성 옵션을 선택할 수 있음.

- CHECK : 입력 가능한 (혹은 컬럼 값이 갱신될 때 유지되어야 하는) 컬럼 값의 범위 등을 제한하는 논리식을 저장.

-  NULL 값의 표현

 - ◆ NULL : ASCII 코드 00

 - ◆ 공백(blank) : ASCII 코드 32

 - ◆ 숫자 0 (zero) : ASCII 코드 48

□ 제약조건의 서술 위치에 따라, 두 형태의 Syntax가 제공됨.

- 1. 컬럼 레벨 정의 방식

- 2. 테이블 레벨 정의 방식

Syntax 1 : 컬럼 레벨 정의 방식

```
CREATE TABLE 테이블명 (  
    {컬럼명 Datatype [NOT NULL] [DEFAULT 묵시값],}+  
    [UNIQUE (컬럼명_리스트),]  
    [PRIMARY KEY (컬럼명_리스트),]  
    {FOREIGN KEY (컬럼명_리스트) REFERENCES 기본테이블명[(컬럼명_리스트)]  
        [ON DELETE 옵션]  
        [ON UPDATE 옵션],}*  
    {CHECK (조건식),}*  
);
```

☞ 옵션 : NO ACTION, CASCADE, SET NULL, SET DEFAULT

☞ [...] : 생략 가능(0~1회),
{...} : 1회,
{...}+ : 1회 이상 반복,
{...}* : 0회 이상 반복

– CONSTRAINT 절을 사용하지 않음.

- ◆ UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약조건을 묵시적으로 명세

Syntax 2 : 테이블 레벨 정의 방식

```
CREATE TABLE 테이블명 (  
    {컬럼명 Datatype [NOT NULL] [DEFAULT 묵시값],}+  
    [[CONSTRAINT 조건명] UNIQUE (컬럼명_리스트),]  
    [[CONSTRAINT 조건명] PRIMARY KEY (컬럼명_리스트),]  
    {[CONSTRAINT 조건명]  
        FOREIGN KEY (컬럼명_리스트) REFERENCES 기본테이블명[(컬럼명_리스트)]  
        [ON DELETE 옵션]  
        [ON UPDATE 옵션],}*  
    {[CONSTRAINT 조건명] CHECK (조건식),}*  
);
```

☞ 옵션 : NO ACTION, CASCADE, SET NULL, SET DEFAULT

– CONSTRAINT 절을 사용

- ◆ UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK 제약조건을 명시적으로 명세.

– 제약조건 마다 조건명을 만들어야 하는 부담이 있으므로, 잘 사용하지 않음.

예제 : 선수 테이블의 생성 (컬럼 레벨 정의 방식)

□ Oracle

```
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),  
    PRIMARY KEY  (PLAYER_ID),  
    FOREIGN KEY  (TEAM_ID) REFERENCES TEAM(TEAM_ID),  
    CHECK (BACK_NO >= 0 AND BACK_NO <100)  
);
```

□ SQL Server

```
CREATE TABLE    PLAYER (  
    PLAYER_ID    CHAR(7)          NOT NULL,  
    PLAYER_NAME  VARCHAR(20)    NOT NULL,  
    TEAM_ID      CHAR(3)          NOT NULL,  
    E_PLAYER_NAME VARCHAR(40),  
    NICKNAME     VARCHAR(30),  
    JOIN_YYYY    CHAR(4),  
    POSITION      VARCHAR(10),  
    BACK_NO      TINYINT,  
    NATION        VARCHAR(20),  
    BIRTH_DATE    DATE,  
    SOLAR         CHAR(1),  
    HEIGHT        SMALLINT,  
    WEIGHT        SMALLINT,  
    PRIMARY KEY (PLAYER_ID),  
    FOREIGN KEY (TEAM_ID) REFERENCES TEAM(TEAM_ID),  
    CHECK (BACK_NO >= 0 AND BACK_NO <100)  
);
```

예제 : 선수 테이블의 생성 (테이블 레벨 정의 방식)

□ Oracle

```
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),  
    CONSTRAINT BACK_NO  
        CHECK (BACK_NO >= 0 AND BACK_NO <100)  
);
```

ANSI/IS

□ SQL Server

```
CREATE TABLE    PLAYER (  
    PLAYER_ID    CHAR(7)                NOT NULL,  
    PLAYER_NAME  VARCHAR(20)            NOT NULL,  
    TEAM_ID      CHAR(3)                NOT NULL,  
    E_PLAYER_NAME VARCHAR(40),  
    NICKNAME     VARCHAR(30),  
    JOIN_YYYY    CHAR(4),  
    POSITION     VARCHAR(10),  
    BACK_NO      TINYINT,  
    NATION       VARCHAR(20),  
    BIRTH_DATE   DATE,  
    SOLAR        CHAR(1),  
    HEIGHT       SMALLINT,  
    WEIGHT       SMALLINT,  
    CONSTRAINT PLAYER_PK  
        PRIMARY KEY (PLAYER_ID),  
    CONSTRAINT PLAYER_FK  
        FOREIGN KEY (TEAM_ID) REFERENCES TEAM(TEAM_ID),  
    CONSTRAINT BACK_NO  
        CHECK (BACK_NO >= 0 AND BACK_NO <100)
```

ANSI/IS
);

예제 : 생성된 테이블의 구조 확인

□ Oracle

```
DESCRIBE PLAYER;
```

칼럼	NULL 가능	데이터 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)

□ SQL Server

– 저장 모듈(프로시저)인 sp_help 제공

```
Exec sp_help 'dbo.PLAYER'  
go
```

칼럼이름	데이터 유형	길이	NULL 가능
PLAYER_ID	CHAR(7)	7	NO
PLAYER_NAME	VARCHAR(20)	20	NO
TEAM_ID	CHAR(3)	3	NO
E_PLAYER_NAME	VARCHAR(40)	40	YES
NICKNAME	VARCHAR(30)	30	YES
JOIN_YYYY	CHAR(4)	4	YES
POSITION	VARCHAR(10)	10	YES
BACK_NO	TINYINT	1	YES
NATION	VARCHAR(20)	20	YES
BIRTH_DATE	DATE	3	YES
SOLAR	CHAR(1)	1	YES
HEIGHT	SMALLINT	2	YES
WEIGHT	SMALLINT	2	YES

SELECT문을 이용한 테이블 생성

□ Oracle

- CTAS : CREATE TABLE ~ AS SELECT ~
- 기존 제약조건 중 NOT NULL만 새로운 복제 테이블에 적용이 되고, 기본키, 고유키, 외래키, CHECK 등의 다른 제약 조건은 없어짐.

```
CREATE TABLE TEAM_TEMP  
AS  
SELECT *  
FROM TEAM;
```

칼럼	NULL 가능	데이터 유형
TEAM_ID	NOT NULL	CHAR(3)
REGION_NAME	NOT NULL	VARCHAR2(8)
TEAM_NAME	NOT NULL	VARCHAR2(40)
E TEAM_NAME		VARCHAR2(50)
ORIG_YYYY		CHAR(4)
STADIUM_ID	NOT NULL	CHAR(3)
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)

□ SQL Server

```
SELECT *  
INTO TEAM_TEMP  
FROM TEAM;
```

칼럼이름	데이터 유형	길이	NULL 가능
TEAM_ID	CHAR(3)	3	NO
REGION_NAME	VARCHAR(8)	4	NO
TEAM_NAME	VARCHAR(40)	40	NO
E_TEAM_NAME	VARCHAR(50)	50	YES
ORIG_YYYY	CHAR(4)	4	YES
STADIUM_ID	CHAR(3)	3	NO
ZIP_CODE1	CHAR(3)	3	YES
ZIP_CODE2	CHAR(3)	3	YES
ADDRESS	VARCHAR(80)	80	YES
DDD	VARCHAR(3)	3	YES
TEL	VARCHAR(10)	10	YES
FAX	VARCHAR(10)	10	YES
HOMEPAGE	VARCHAR(50)	50	YES
OWNER	VARCHAR(10)	10	YES

ON DELETE 옵션

- 데이터베이스의 참조 무결성(referential integrity)을 유지하는 장치
 - 옵션의 종류: “어떤 튜플을 삭제할 때”
 - ◆ NO ACTION
 - ◆ CASCADE: FK를 통해 해당 튜플을 참조하는 모든 튜플들도 같이 삭제함.
(cascade deletion)
 - ◆ SET NULL : FK를 통해 해당 튜플을 참조하는 모든 튜플들의 FK 값을 널로 함.
 - ◆ SET DEFAULT

예제 : University Database

```
CREATE TABLE 교수 (  
    교수번호 NUMBER(5) NOT NULL,  
    이름 CHAR(10) NOT NULL,  
    전공 VARCHAR2(40),  
    학과 VARCHAR2(20),  
    PRIMARY KEY (교수번호),  
);
```

```
CREATE TABLE 학생 (  
    학번 NUMBER(7) NOT NULL,  
    이름 CHAR(10) NOT NULL,  
    학과 VARCHAR2(20),  
    학년 NUMBER(1),  
    지도교수번호 NUMBER(5)  
    PRIMARY KEY (학번),  
    FOREIGN KEY (지도교수번호) REFERENCES 교수(교수번호)  
    ON DELETE SET NULL  
);
```

```

CREATE TABLE 과목 (
    학수번호 NUMBER(5) NOT NULL,
    과목명 CHAR(10) NOT NULL,
    학점 NUMBER(1) NOT NULL,
    담당교수번호 NUMBER(5),
    시간 CHAR(10),
    장소 CHAR(10),
    PRIMARY KEY (학수번호),
    FOREIGN KEY (담당교수번호) REFERENCES 교수(교수번호)
        ON DELETE SET NULL
);

```

```

CREATE TABLE 등록 (
    학번 NUMBER(7) NOT NULL,
    학수번호 NUMBER(5) NOT NULL,
    성적 CHAR(2),
    PRIMARY KEY (학번, 학수번호),
    FOREIGN KEY (학번) REFERENCES 학생(학번)
        ON DELETE CASCADE,
    FOREIGN KEY (학수번호) REFERENCES 과목(학수번호)
        ON DELETE CASCADE
);

```

□ ON DELETE SET NULL

- 교수에서 튜플이 삭제되면,
 - ◆ 삭제된 교수 튜플을 참조하는 모든 학생 튜플들의 지도교수번호를 NULL로 변경함.
 - ◆ 삭제된 교수 튜플을 참조하는 모든 과목 튜플들의 담당교수번호를 NULL로 변경함.

□ ON DELETE CASCADE

- 등록에서 튜플이 삭제되면, 학생과 과목의 튜플은 영향을 받지 않음.
- 학생(혹은 과목)에서 튜플이 삭제되면,
 - ◆ 삭제된 학생(혹은 과목) 튜플을 참조하는 모든 등록 튜플들을 삭제함.

예제 : K-League Database

Set null
혹은 cascade

2. 삭제
혹은 null

선수(PLAYER) 테이블		
PLAYER_ID	PLAYER_NAME	TEAM_ID
2012137	이교트	K06
2012136	오비나	K10
2012135	윤환일	K02
2012134	페르난도	K04
2012133	레오	K03
2012132	실바	K07
2012131	무스타파	K04
2012130	에니	K01
2012129	알리 송	K01
2012128	자스민	K08
2012127	디디	K06

운동장(STADIUM) 테이블		
STADIUM_ID	STADIUM_NAME	SEAT_COUNT
D03	전주월드컵경기장	28000
B02	성남종합운동장	27000
C06	포항스틸야드	25000
D01	광양전용경기장	20009
B05	서울월드컵경기장	66806
B01	인천월드컵경기장	35000
C05	창원종합운동장	27065
C04	울산문수경기장	46102
D02	대전월드컵경기장	41000
B04	수원월드컵경기장	50000
A02	광주월드컵경기장	40245
C02	부산아사히경기장	30000
A03	강릉종합경기장	33000
A04	제주월드컵경기장	42256
A05	대구월드컵경기장	66422
F01	대구시민경기장	30000
F02	부산시민경기장	30000
F03	일산경기장	20000
F04	마산경기장	20000
F05	안양경기장	20000

2. 영향 없음

팀(Team) 테이블			
TEAM_ID	TEAM_NAME	REGION_NAME	STADIUM_ID
K05	현대모터스	전북	D03
K08	일화천마	성남	B02
K03	스틸러스	포항	C06
K07	드래곤즈	전남	D01
K09	FC서울	서울	B05
K04	유나이티드	인천	B01
K11	경남FC	경남	C05
K01	울산현대	울산	C04
K10	시티즌	대전	D02
K02	삼성블루윙즈	수원	B04
K12	광주상무	광주	A02
K06	아이파크	부산	C02
K13	강원FC	강원	A03
K14	제주유나이티드FC	제주	A04
K15	대구FC	대구	A05

1. 삭제

Cascade로 정의되었다면?

3. 삭제

PLAYER_ID	PLAYER_NAME	TEAM_ID
2012137	이고르	K06
2012136	오비나	K10
2012135	유원인	K02
2012134	페르난도	K04
2012133	레오	K03
2012132	실바	K07
2012131	무스타파	K04
2012130	에디	K01
2012129	알리 송	K01
2012128	자스민	K08
2012127	디디	K06

1. 삭제

STADIUM_ID	STADIUM_NAME	SEAT_COUNT
D03	전주월드컵경기장	28000
B02	성남종합운동장	27000
C06	포항스틸아드	25000
D01	광양전용경기장	20009
B05	서울월드컵경기장	66806
B01	인천월드컵경기장	35000
C05	창원종합운동장	27000
C04	울산문수경기장	46102
D02	대전월드컵경기장	41000
B04	수원월드컵경기장	50000
A02	광주월드컵경기장	40245
C02	부산아시아드경기장	30000
A03	강릉종합경기장	33000
A04	제주월드컵경기장	42256
A05	대구월드컵경기장	66422
F01	대구시민경기장	30000
F02	부산시민경기장	30000
F03	일산경기장	20000
F04	마산경기장	20000
F05	안양경기장	20000

2. 삭제

TEAM_ID	TEAM_NAME	REGION_NAME	STADIUM_ID
K05	현대 모터스	전북	D03
K08	일화천마	성남	B02
K03	스틸러스	포항	C06
K07	드래곤즈	전남	D01
K09	FC서울	서울	B05
K04	유나이티드	인천	B01
K11	경남FC	경남	C05
K01	울산현대	울산	C04
K10	시티즌	대전	D02
K02	삼성블루윙즈	수원	B04
K12	광주상무	광주	A02
K06	아이파크	부산	C02
K13	강원FC	강원	A03
K14	제주유나이티드FC	제주	A04
K15	대구FC	대구	A05

Cascade로 정의되었다면?

5. DROP TABLE

□ Syntax

```
DROP TABLE 테이블명 [RESTRICT | CASCADE];
```

- **RESTRICT**: FK를 통해, 해당 테이블의 튜플을 참조하는 다른 테이블의 튜플이 하나라도 존재하면, 테이블 제거 명령이 실행되지 못함.
- **CASCADE**: FK를 통해, 해당 테이블의 튜플을 참조하는 다른 테이블의 튜플이 하나라도 존재하면, 이 튜플들도 함께 함께 제거함.

□ 예제

```
DROP TABLE 과목;
```

- 과목 테이블의 삭제를 시도하면, 삭제가 진행되지 않음.
 - ◆ 등록 테이블의 학수번호가 FK로서 과목 테이블을 참조하고 있음. 따라서 등록 테이블에 과목을 참조하는 튜플이 존재하므로, 삭제가 진행되지 않음.
 - ◆ 과목 테이블의 학수번호에 대한 FK 제약조건을 먼저 제거(ALTER TABLE 사용)하거나, 등록 테이블에 과목을 참조하는 튜플이 없으면 제거 가능.
- 다음과 같이 **CASCADE** 옵션을 사용하면, 과목을 참조하는 등록 테이블의 모든 튜플들도 같이 제거됨.

```
DROP TABLE 과목 CASCADE;
```

6. ALTER TABLE

□ 기능

- 컬럼의 추가/삭제/수정
- 제약조건의 추가/삭제

□ Syntax

```
ALTER TABLE 테이블명  
{ {ADD 컬럼명 Datatype [NOT NULL] [DEFAULT 묵시값]} |  
  {DROP COLUMN 컬럼명 [RESTRICT | CASCADE]} |  
  {MODIFY 컬럼명 Datatype [NOT NULL] [DEFAULT 묵시값]} |  
  {ADD CONSTRAINT 조건명 제약조건} |  
  {DROP CONSTRAINT 조건명}  
};
```

ADD 컬럼

□ Syntax

```
ALTER TABLE 테이블명  
ADD 컬럼명 Datatype;
```

□ Oracle

```
ALTER TABLE      PLAYER  
ADD                (ADDRESS VARCHAR2(80));
```

컬럼	NULL 가능	데이터 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) ▣ 추가된 열

□ SQL Server

```
ALTER TABLE    PLAYER
ADD            ADDRESS VARCHAR(80);
```

칼럼이름	데이터 유형	길이	NULL 가능
PLAYER_ID	CHAR(7)	7	NO
PLAYER_NAME	VARCHAR(20)	20	NO
TEAM_ID	CHAR(3)	3	NO
E_PLAYER_NAME	VARCHAR(40)	40	YES
NICKNAME	VARCHAR(30)	30	YES
JOIN_YYYY	CHAR(4)	4	YES
POSITION	VARCHAR(10)	10	YES
BACK_NO	TINYINT	1	YES
NATION	VARCHAR(20)	20	YES
BIRTH_DATE	DATE	3	YES
SOLAR	CHAR(1)	1	YES
HEIGHT	SMALLINT	2	YES
WEIGHT	SMALLINT	2	YES
ADDRESS	VARCHAR(80)	80	YES

추가된 열

DROP COLUMN 컬럼

□ Syntax

```
ALTER TABLE 테이블명  
DROP COLUMN 컬럼명 [RESTRICT | CASCADE];
```

□ Oracle / SQL Server

```
ALTER TABLE PLAYER  
DROP COLUMN ADDRESS;
```

DESC PLAYER;

컬럼	NULL 가능	데이터 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)

ANSI/IS



국민대학교
KOOKMIN UNIVERSITY

MODIFY 컬럼

□ 고려사항

- 해당 컬럼의 크기를 늘릴 수는 있지만 줄이지는 못한다. 이는 기존의 데이터가 훼손될 수 있기 때문이다.
- 해당 컬럼이 NULL 값만 가지고 있거나 테이블에 아무 행도 없으면 컬럼의 폭을 줄일 수 있다.
- 해당 컬럼이 NULL 값만을 가지고 있으면 데이터 유형을 변경할 수 있다.
- 해당 컬럼의 DEFAULT 값을 바꾸면 변경 작업 이후 발생하는 행 삽입에만 영향을 미치게 된다.
- 해당 컬럼에 NULL 값이 없을 경우에만 NOT NULL 제약조건을 추가할 수 있다.

□ Oracle

```
ALTER TABLE 테이블명  
MODIFY ( {컬럼명 Datatype [NOT NULL] [DEFAULT 값],}+ );
```

```
ALTER TABLE TEAM_TEMP  
MODIFY (ORIG_YYYY VARCHAR2(8) NOT NULL DEFAULT '20020129' );
```

컬럼	NULL 가능	데이터 유형
TEAM_ID	NOT NULL	CHAR(3)
REGION_NAME	NOT NULL	VARCHAR2(8)
TEAM_NAME	NOT NULL	VARCHAR2(40)
E_TEAM_NAME		VARCHAR2(50)
ORIG YYYY	NOT NULL	VARCHAR2(8) 기본값 '20020129'
STADIUM_ID	NOT NULL	CHAR(3)
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)

□ SQL Server

```
ALTER TABLE 테이블명
ALTER COLUMN ( {컬럼명 Datatype [NOT NULL] [DEFAULT 값],}+ );
```

```
ALTER TABLE TEAM_TEMP
ALTER COLUMN ORIG_YYYY VARCHAR(8) NOT NULL ;
```

```
ALTER TABLE TEAM_TEMP
ADD CONSTRAINT DF_ORIG_YYYY DEFAULT '20020129' FOR ORIG_YYYY;
```

컬럼이름	데이터 유형	길이	NULL 가능
TEAM_ID	CHAR(3)	3	NO
REGION_NAME	VARCHAR(8)	8	NO
TEAM_NAME	VARCHAR(40)	40	NO
E_TEAM_NAME	VARCHAR(50)	50	YES
ORIG_YYYY	CHAR(8)	8	NO
STADIUM_ID	CHAR(3)	3	NO
ZIP_CODE1	CHAR(3)	3	YES
ZIP_CODE2	CHAR(3)	3	YES
ADDRESS	VARCHAR(80)	80	YES
DDD	VARCHAR(3)	3	YES
TEL	VARCHAR(10)	10	YES
FAX	VARCHAR(10)	10	YES
HOMEPAGE	VARCHAR(50)	50	YES
OWNER	VARCHAR(10)	10	YES

constraint_type	constraint_name	constraint_keys
DEFAULT on column ORIG_YYYY	DF_ORIG_YYYY	('20020129')

ANSI/ISO SQL

RENAME COLUMN 컬럼

□ Oracle 등 일부 DBMS에서만 제공

```
ALTER TABLE 테이블명  
RENAME COLUMN 컬럼명 TO 컬럼명;
```

```
ALTER TABLE PLAYER  
RENAME COLUMN PLAYER_ID TO TEMP_ID;
```

□ SQL Server

– 저장 프로시저 sp_rename 제공

```
sp_rename 'dbo.TEAM_TEMP.TEAM_ID', 'TEAM_TEMP_ID', 'COLUMN';
```

ADD CONSTRAINT

□ Syntax

```
ALTER TABLE    테이블명  
ADD CONSTRAINT  조건명  제약조건;
```

□ 예제

```
ALTER TABLE    PLAYER  
ADD CONSTRAINT  PLAYER_FK  
                FOREIGN KEY (TEAM_ID) REFERENCES TEAM(TEAM_ID);
```

DROP CONSTRAINT

□ Syntax

```
ALTER TABLE 테이블명  
DROP CONSTRAINT 조건명;
```

□ 예제

```
ALTER TABLE PLAYER  
DROP CONSTRAINT PLAYER_FK;
```

7. RENAME 테이블

- Oracle 등 일부 DBMS에서만 제공

```
RENAME 테이블명 TO 테이블명;
```

```
RENAME TEAM TO TEAM_BACKUP;
```

- SQL Server

- 저장 모듈(프로시저)인 sp_rename 제공

```
sp_rename 'dbo.TEAM', 'TEAM_BACKUP';
```

8. TRUNCATE TABLE

□ 테이블 관련 삭제 연산

– DROP TABLE 테이블명

- ◆ 테이블 정의 + 데이터(모든 행) 삭제
- ◆ 로그를 남기지 않으므로 시스템에 부하가 적음.
- ◆ 복구 불가

– TRUNCATE TABLE 테이블명

- ◆ 테이블 구조는 유지, 데이터(모든 행)만 삭제
- ◆ 로그를 남기지 않으므로 시스템 부하가 적음.
- ◆ 복구 불가.

– DELETE FROM 테이블명

- ◆ 테이블 구조는 유지, 데이터(모든 행)만 삭제.
- ◆ 로그를 남기므로 시스템에 부하를 줌
- ◆ rollback을 통한 복구 가능.

□ DROP TABLE, TRUNCATE TABLE을 DDL로 분류하는 이유

- 테이블 데이터를 메모리에 로딩하지 않고, 하드디스크에서 그대로 실행함. 따라서 로그를 통한 commit과 rollback을 할 수 없음

□ Syntax

```
TRUNCATE TABLE 테이블명;
```

□ 예제

```
TRUNCATE TABLE TEAM;
```