SQL FOR DATABASE CONSTRUCTION AND APPLICATION PROCESSING

revised by 김태연

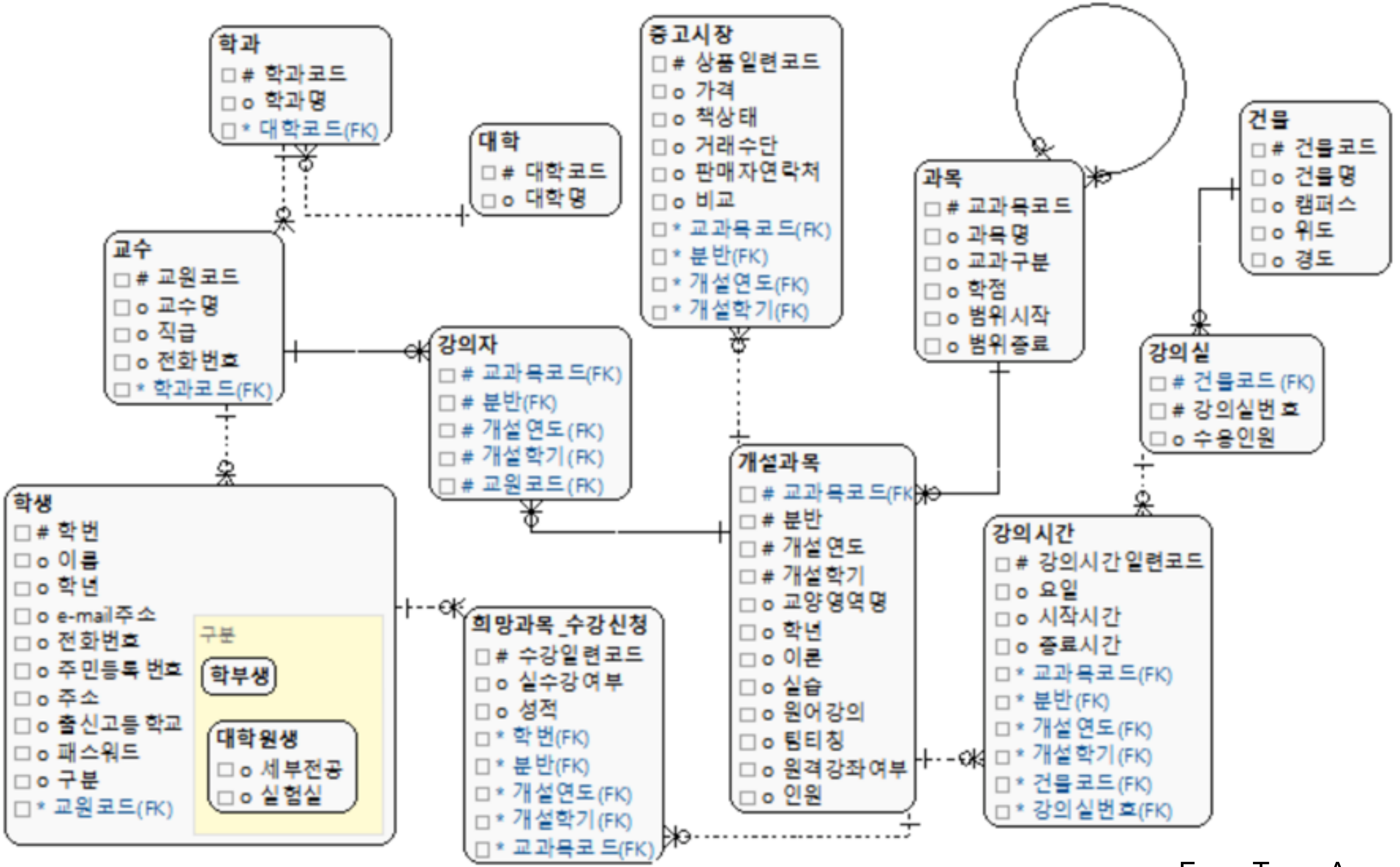
OBJECTIVES

- To create and manage table structures using SQL statements
- To understand how referential integrity actions are implemented in SQL statements
- To create and use SQL constraints
- To understand several uses for SQL views
- To use SQL statements to create and use views
- To gain an understanding of how SQL is used in an application program
- To understand how to create and use triggers
- To understand how to create and use stored procedures

SQL CATAGORIES

- SQL statements can be divided into five categories:
  - Data definition language (DDL)
    - Used for creating tables, relationships, and other structures
    - Covered in this chapter Chapter 7
  - SQL/Persistent Stored Modules (SQL/PSM) statements
    - Add procedural programming capabilities: Variables and Control-of-flow statements
    - Covered in this chapter Chapter 7 and 10C (MySQL 5.6)
  - Data manipulation language (DML) statements
    - Used for queries and data modification
    - Covered in this chapter and Chapter 2
  - Transaction control language (TCL) statements
  - Data control language (DCL) statements

ENROLLMENT DATABASE DESIGN



학과
- □ # 학과코드
- □ o 학과명
- □ * 대학코드(FK)

대학
- □ # 대학코드
- □ o 대학명

중고시장
- □ # 상품일련코드
- □ o 가격
- □ o 책상태
- □ o 거래수단
- □ o 판매자연락처
- □ o 비교
- □ * 교과목코드(FK)
- □ * 분반(FK)
- □ * 개설연도(FK)
- □ * 개설학기(FK)

과목
- □ # 교과목코드
- □ o 과목명
- □ o 교과구분
- □ o 학점
- □ o 범위시작
- □ o 범위종료

건물
- □ # 건물코드
- □ o 건물명
- □ o 캠퍼스
- □ o 위도
- □ o 경도

교수
- □ # 교원코드
- □ o 교수명
- □ o 직급
- □ o 전화번호
- □ * 학과코드(FK)

강의자
- □ # 교과목코드(FK)
- □ # 분반(FK)
- □ # 개설연도(FK)
- □ # 개설학기(FK)
- □ # 교원코드(FK)

강의실
- □ # 건물코드(FK)
- □ # 강의실번호
- □ o 수용인원

학생
- □ # 학번
- □ o 이름
- □ o 학년
- □ o e-mail주소
- □ o 전화번호
- □ o 주민등록 번호
- □ o 주소
- □ o 출신고등학교
- □ o 패스워드
- □ o 구분
- □ * 교원코드(FK)

구분

학부생

대학원생
- □ o 세부전공
- □ o 실험실

희망과목_수강신청
- □ # 수강일련코드
- □ o 실수강여부
- □ o 성적
- □ * 학번(FK)
- □ * 분반(FK)
- □ * 개설연도(FK)
- □ * 개설학기(FK)
- □ * 교과목코드(FK)

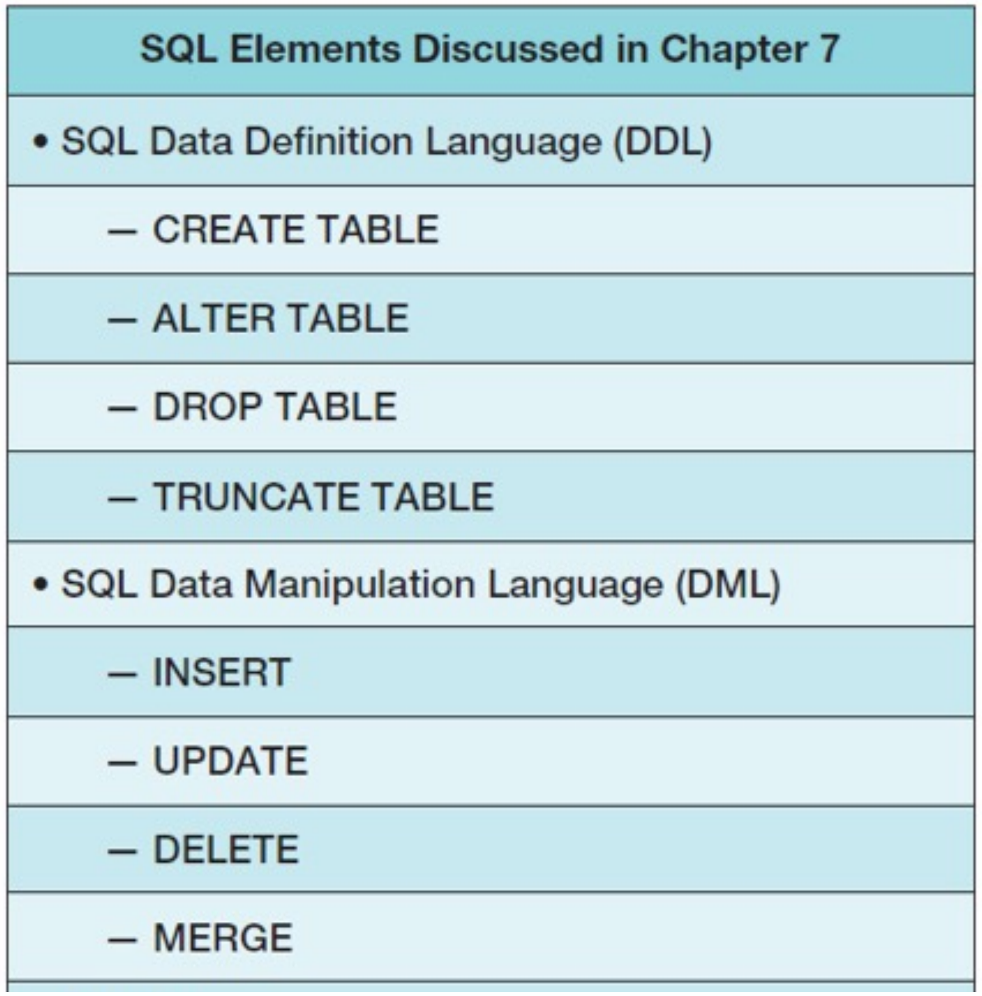
개설과목
- □ # 교과목코드(FK
- □ # 분반
- □ # 개설연도
- □ # 개설학기
- □ o 교양영역명
- □ o 학년
- □ o 이론
- □ o 실습
- □ o 원어강의
- □ o 팀티칭
- □ o 원격강좌여부
- □ o 인원

강의시간
- □ # 강의시간일련코드
- □ o 요일
- □ o 시작시간
- □ o 종료시간
- □ * 교과목코드(FK)
- □ * 분반(FK)
- □ * 개설연도(FK)
- □ * 개설학기(FK)
- □ * 건물코드(FK)
- □ * 강의실번호(FK)

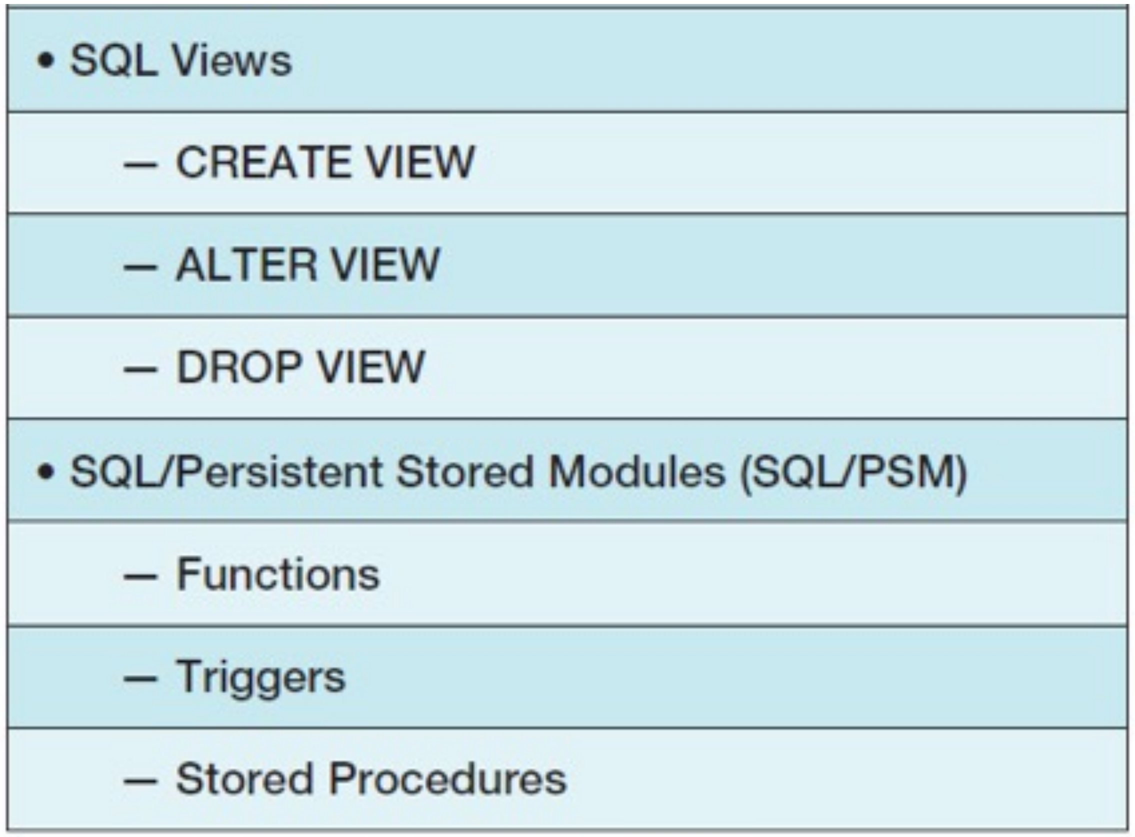
From Team A

SQL ELEMENTS

- ➤ The figure summarizes the new SQL DDL and DML statements described in this chapter.

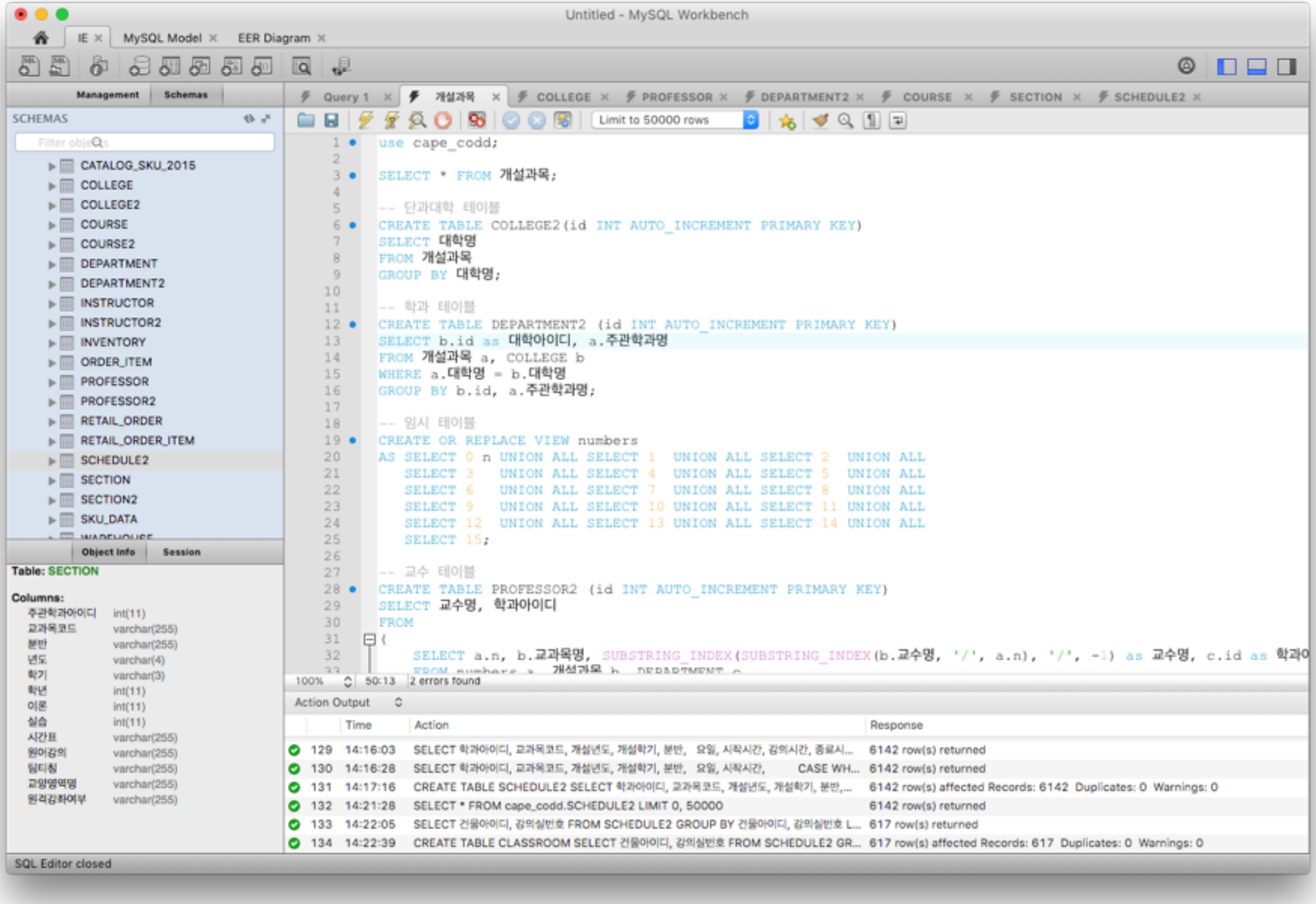


SQL Elements Discussed in Chapter 7
• SQL Data Definition Language (DDL)
— CREATE TABLE
— ALTER TABLE
— DROP TABLE
— TRUNCATE TABLE
• SQL Data Manipulation Language (DML)
— INSERT
— UPDATE
— DELETE
— MERGE



• SQL Views
— CREATE VIEW
— ALTER VIEW
— DROP VIEW
• SQL/Persistent Stored Modules (SQL/PSM)
— Functions
— Triggers
— Stored Procedures

ORACLE MYSQL WORKBENCH



```sql
use cape_codd;

SELECT * FROM 개설과목;

-- 단과대학 테이블
CREATE TABLE COLLEGE2(id INT AUTO_INCREMENT PRIMARY KEY)
SELECT 대학명
FROM 개설과목
GROUP BY 대학명;

-- 학과 테이블
CREATE TABLE DEPARTMENT2 (id INT AUTO_INCREMENT PRIMARY KEY)
SELECT b.id as 대학아이디, a.주관학과명
FROM 개설과목 a, COLLEGE b
WHERE a.대학명 = b.대학명
GROUP BY b.id, a.주관학과명;

-- 임시 테이블
CREATE OR REPLACE VIEW numbers
AS SELECT 0 n UNION ALL SELECT 1  UNION ALL SELECT 2  UNION ALL
   SELECT 3   UNION ALL SELECT 4  UNION ALL SELECT 5  UNION ALL
   SELECT 6   UNION ALL SELECT 7  UNION ALL SELECT 8  UNION ALL
   SELECT 9   UNION ALL SELECT 10 UNION ALL SELECT 11 UNION ALL
   SELECT 12  UNION ALL SELECT 13 UNION ALL SELECT 14 UNION ALL
   SELECT 15;

-- 교수 테이블
CREATE TABLE PROFESSOR2 (id INT AUTO_INCREMENT PRIMARY KEY)
SELECT 교수명, 학과아이디
FROM
(
    SELECT a.n, b.교과목명, SUBSTRING_INDEX(SUBSTRING_INDEX(b.교수명, '/', a.n), '/', -1) as 교수명, c.id as 학과아
    FROM numbers a, 개설과목 b, DEPARTMENT c
```

2 errors found

Table: SECTION

Columns:

주관학과아이디	int(11)
교과목코드	varchar(255)
분반	varchar(255)
년도	varchar(4)
학기	varchar(3)
학년	int(11)
이론	int(11)
실습	int(11)
시간표	varchar(255)
원어강의	varchar(255)
팀티칭	varchar(255)
교양영역명	varchar(255)
원격강좌여부	varchar(255)

```
     Time      Action                                                                           Response
✓ 129 14:16:03 SELECT 학과아이디, 교과목코드, 개설년도, 개설학기, 분반,   요일, 시작시간, 강의시간, 종료시...   6142 row(s) returned
✓ 130 14:16:28 SELECT 학과아이디, 교과목코드, 개설년도, 개설학기, 분반,   요일, 시작시간,        CASE WH...   6142 row(s) returned
✓ 131 14:17:16 CREATE TABLE SCHEDULE2 SELECT 학과아이디, 교과목코드, 개설년도, 개설학기, 분반,...   6142 row(s) affected Records: 6142  Duplicates: 0  Warnings: 0
✓ 132 14:21:28 SELECT * FROM cape_codd.SCHEDULE2 LIMIT 0, 50000                                 6142 row(s) returned
✓ 133 14:22:05 SELECT 건물아이디, 강의실번호 FROM SCHEDULE2 GROUP BY 건물아이디, 강의실번호 L...   617 row(s) returned
✓ 134 14:22:39 CREATE TABLE CLASSROOM SELECT 건물아이디, 강의실번호 FROM SCHEDULE2 GR...   617 row(s) affected Records: 617  Duplicates: 0  Warnings: 0
```

CREATE TABLE STATEMENT IN MYSQL

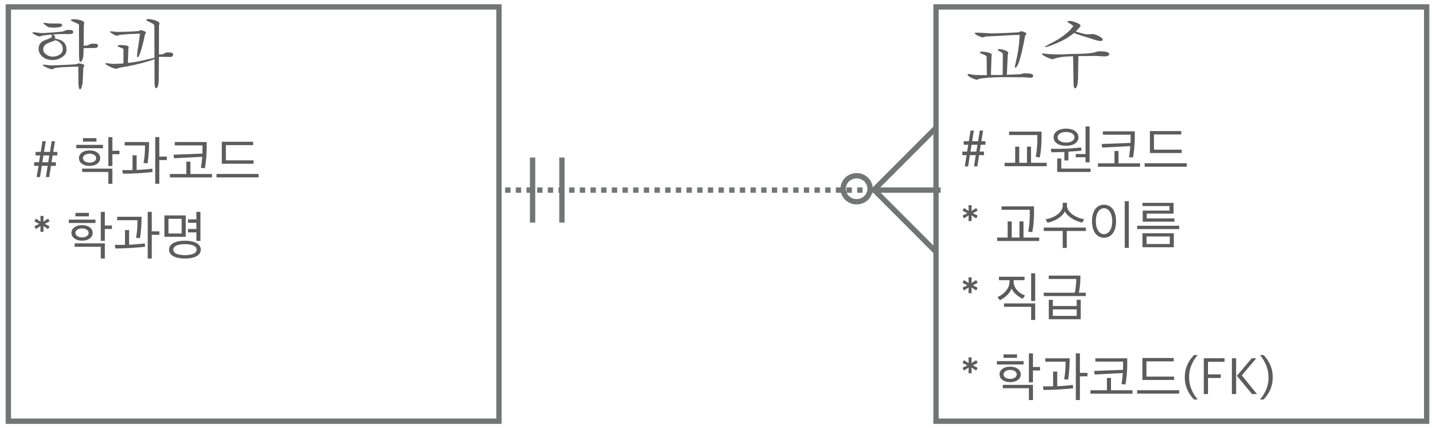
```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    ...
```

- Variations in SQL Data Types (INT, VARCHAR, DATETIME)
  - [MySQL 5.6 Data types] (http://dev.mysql.com/doc/refman/5.6/en/data-types.html)
- Optional Constraint (NOT NULL, DEFAULT, AUTO_INCREMENT)
  - [MySQL 5.6 Data Properties] (http://dev.mysql.com/doc/refman/5.6/en/create-table.html#create-table-types-attributes)
- Optional Table constraint (PRIMARY KEY, UNIQUE, FOREIGN KEY, INDEX)
  - [MySQL 5.6 Indexes and Foreign Keys] (http://dev.mysql.com/doc/refman/5.6/en/create-table.html#create-table-indexes-keys)

CONSTRAINTS

- Constraints can be defined within the CREATE TABLE statement, or they can be added to the table after it is created using the ALTER table statement.
- Five types of constraints:
  - PRIMARY KEY may not have null values
  - UNIQUE may have null values
  - NULL/NOT NULL
  - FOREIGN KEY
  - CHECK
    - The CHECK clause is parsed but ignored by all storage engines in MySQL 5.6.

INDEX



학과

# 학과코드

* 학과명

교수

# 교원코드

* 교수이름

* 직급

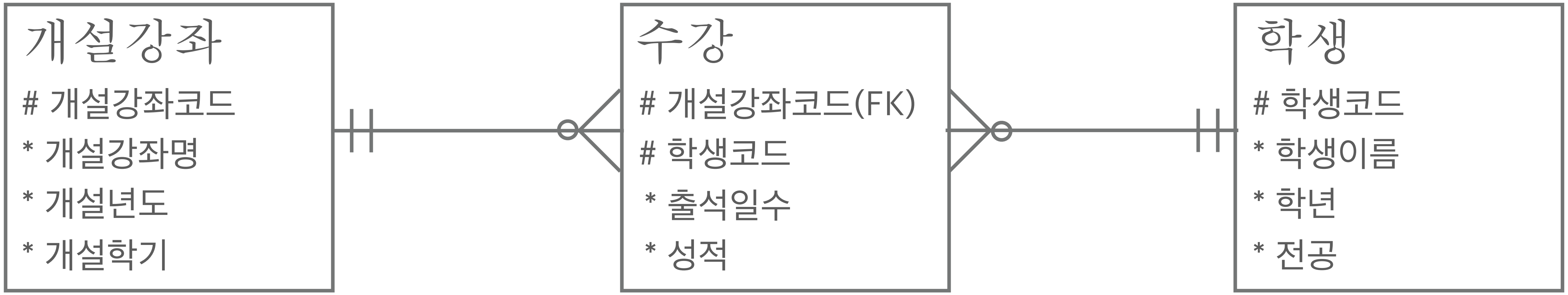
* 학과코드(FK)

```
SELECT a.교수이름, a.직급, b.학과명
FROM 교수 a, 학과 b
WHERE a.학과코드 = b.학과코드
```

Index

학과 **PRIMARY KEY** 학과코드

교수 **PRIMARY KEY** 교원코드



개설강좌

# 개설강좌코드

* 개설강좌명

* 개설년도

* 개설학기

수강

# 개설강좌코드(FK)

# 학생코드

* 출석일수

* 성적

학생

# 학생코드

* 학생이름

* 학년

* 전공

```
SELECT b.학생이름, b.직급, b.학과명
FROM 수강 a, 학생 b
WHERE b.전공 = ‘산업공학과’
AND a.성적 > ’90’ and a.학생코드 = b.학생코드 (+)
```

Index

학생 **PRIMARY KEY** 학생코드

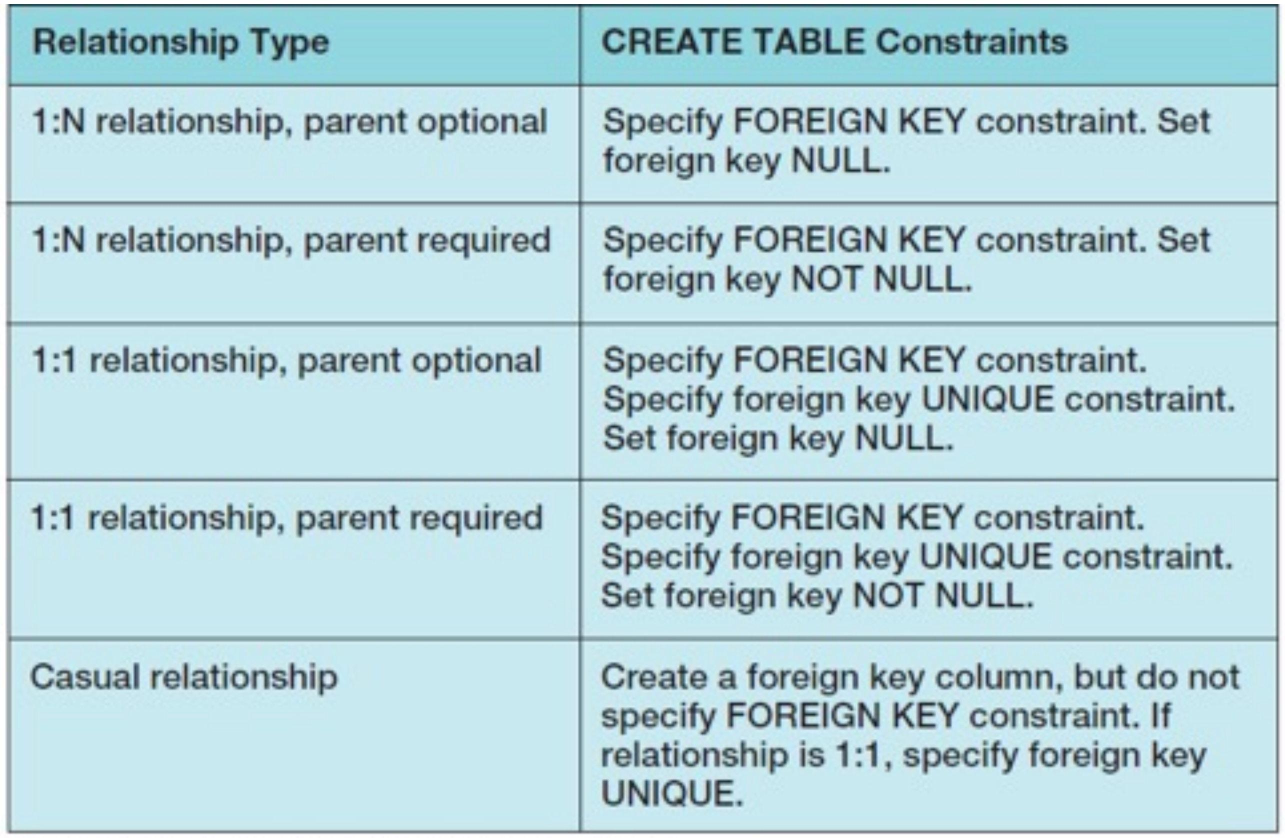
학생 **INDEX** 전공

수강 **PRIMARY KEY**
(개설강좌코드, 학생코드)

수강 **INDEX** (학생코드, 성적)

RELATIONSHIP

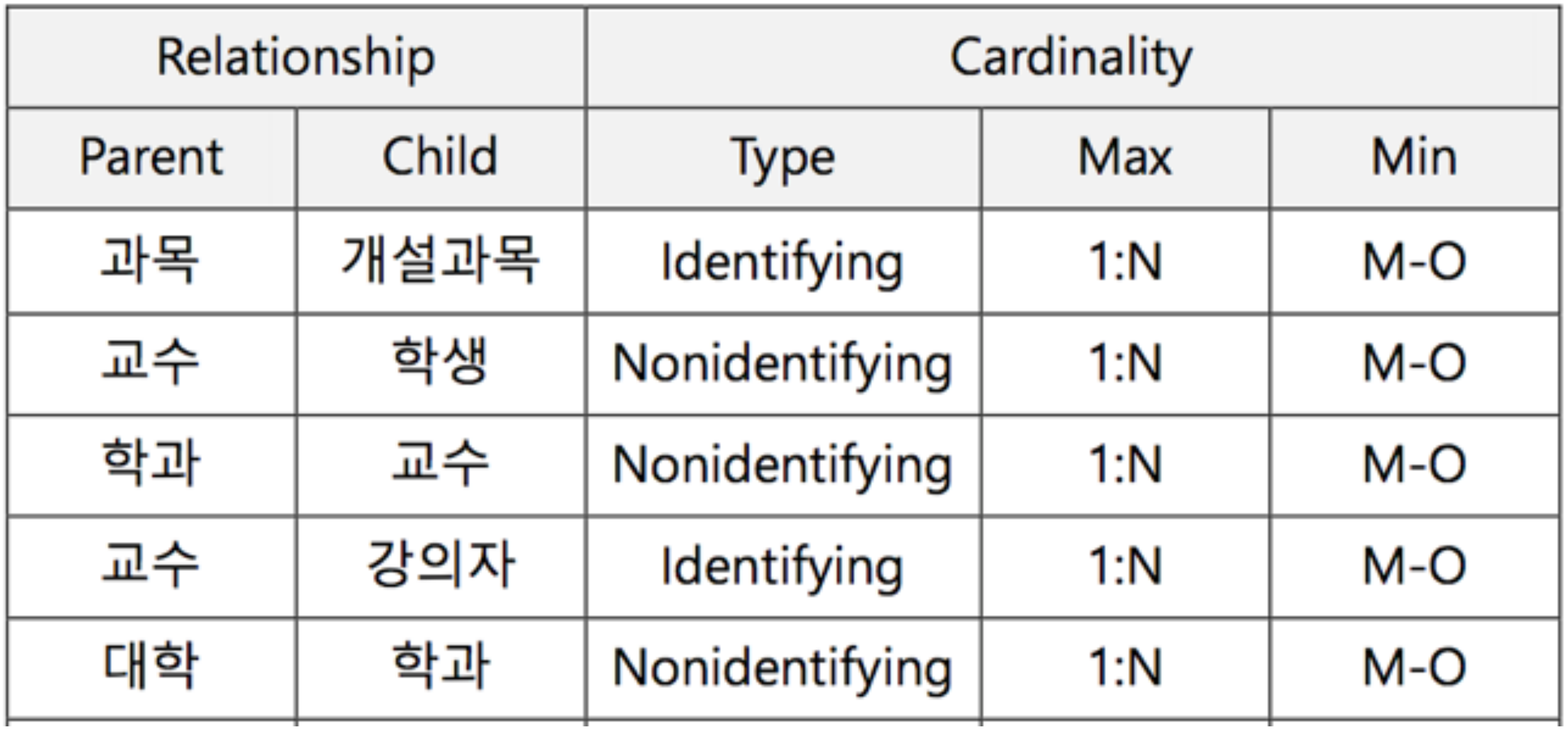
- The parent were required: you must define the referential integrity constraint and set the foreign key to NOT NULL in the child table.
- The parent were not required: you would specify the foreign key in the child table as NULL.
- It is appropriate to create a foreign key column but not specify a FOREIGN KEY constraint.



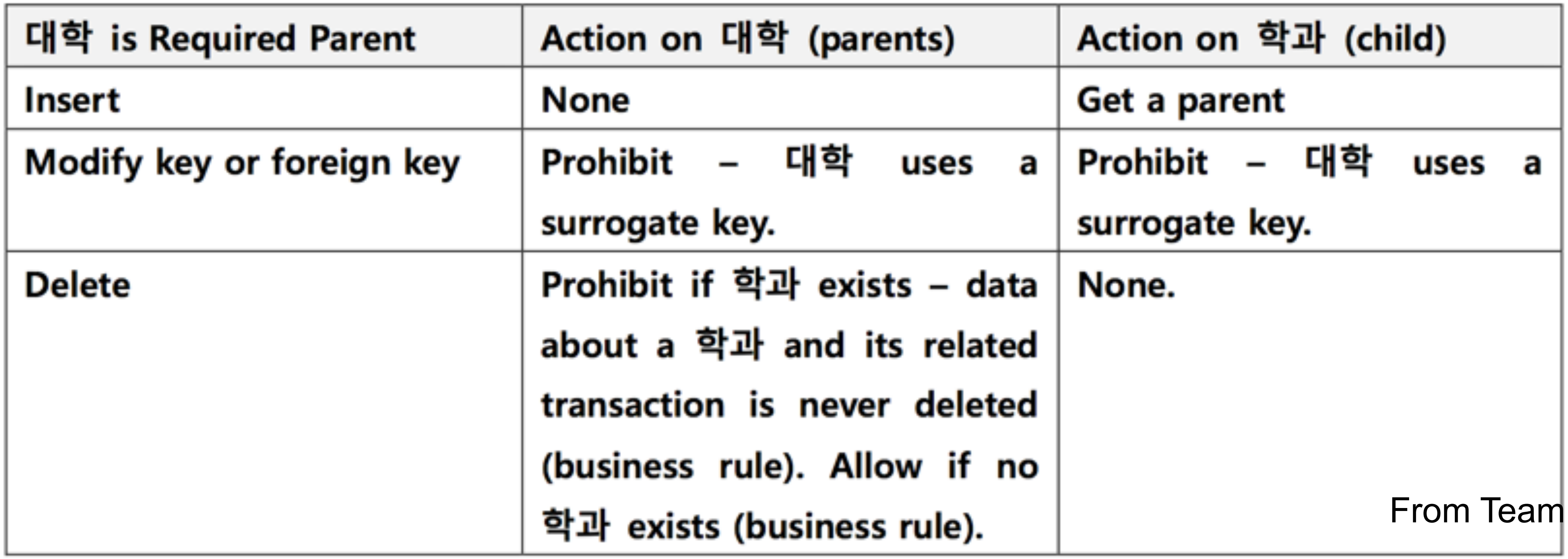
Relationship Type	CREATE TABLE Constraints
1:N relationship, parent optional	Specify FOREIGN KEY constraint. Set foreign key NULL.
1:N relationship, parent required	Specify FOREIGN KEY constraint. Set foreign key NOT NULL.
1:1 relationship, parent optional	Specify FOREIGN KEY constraint. Specify foreign key UNIQUE constraint. Set foreign key NULL.
1:1 relationship, parent required	Specify FOREIGN KEY constraint. Specify foreign key UNIQUE constraint. Set foreign key NOT NULL.
Casual relationship	Create a foreign key column, but do not specify FOREIGN KEY constraint. If relationship is 1:1, specify foreign key UNIQUE.

Copyright © 2016, by Pearson Education, Inc.,

DATABASE DESIGN RELATIONSHIP



Relationship		Cardinality		
Parent	Child	Type	Max	Min
과목	개설과목	Identifying	1:N	M-O
교수	학생	Nonidentifying	1:N	M-O
학과	교수	Nonidentifying	1:N	M-O
교수	강의자	Identifying	1:N	M-O
대학	학과	Nonidentifying	1:N	M-O



대학 is Required Parent	Action on 대학 (parents)	Action on 학과 (child)
Insert	None	Get a parent
Modify key or foreign key	Prohibit – 대학 uses a surrogate key.	Prohibit – 대학 uses a surrogate key.
Delete	Prohibit if 학과 exists – data about a 학과 and its related transaction is never deleted (business rule). Allow if no 학과 exists (business rule).	None.

From Team A

CREATE TABLE STATEMENT

```
CREATE TABLE `대학` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `대학명` varchar(255) NOT NULL DEFAULT '미정',
  PRIMARY KEY (`id`),
  UNIQUE KEY `대학명_UNIQUE` (`대학명`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `학과` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `대학아이디` int(11) NOT NULL DEFAULT '0',
  `주관학과명` varchar(45) NOT NULL DEFAULT '미정',
  PRIMARY KEY (`id`),
  UNIQUE KEY `주관학과명_UNIQUE` (`주관학과명`),
  KEY `fk_DEPARTMENT_COLLEGE_idx` (`대학아이디`),
  CONSTRAINT `fk_DEPARTMENT_COLLEGE` FOREIGN KEY (`대학아이디`) REFERENCES
`COLLEGE` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=165 DEFAULT CHARSET=utf8;
```

ALTER TABLE STATEMENT

- ALTER TABLE statement changes table structure, properties, or constraints after it has been created.

```
ALTER TABLE ASSIGNMENT
    ADD CONSTRAINT EmployeeFK
        FOREIGN KEY (EmployeeNumber)
        REFERENCES EMPLOYEE (EmployeeNumber)
            ON UPDATE CASCADE ON DELETE NO ACTION;
```

- Adding and Dropping Columns
  - The following statement will add a column named MyColumn to the CUSTOMER table:

```
ALTER TABLE CUSTOMER DROP COLUMN MyColumn;
```

  - You can drop an existing column with the statement:

```
ALTER TABLE CUSTOMER ADD MyColumn Char(5) NULL;
```

REMOVING TABLES

- SQL DROP TABLE:

```
DROP TABLE TRANS;
```

- If there are constraints:

```
ALTER TABLE CUSTOMER_ARTIST_INT
        DROP CONSTRAINT  Customer_Artist_Int_CustomerFK;
ALTER TABLE TRANS
        DROP CONSTRAINT  TransactionCustomerFK;
DROP TABLE CUSTOMER;
```

- Removing Data Only

```
TRUNCATE TABLE TRANS;
```

  - Cannot be used with a table that is referenced by a foreign key constraint.

SQL DML—INSERT, UPDATE, DELETE

- SQL INSERT statement:

```
INSERT INTO ARTIST (LastName, FirstName, Nationality, DateOfBirth, DateDeceased)
VALUES ('Tamayo', ‘Rufino', 'Mexican', 1899, 1991);
```

- Bulk INSERT:

```
INSERT INTO ARTIST (LastName, FirstName, Nationality, DateOfBirth)
SELECT LastName, FirstName, Nationality, DateOfBirth
FROM    IMPORTED_ARTIST;
```

- SQL UPDATE statement:

```
UPDATE  CUSTOMER SET City = 'New York City’ WHERE CustomerID = 1000;
```

- SQL DELETE statement:

```
DELETE FROM CUSTOMER WHERE   CustomerID = 1000;
```

  - If you omit the WHERE clause, you will delete every row in the table.

VIEW

```
CREATE VIEW view_name [(column_list)]
    AS select_statement
```

- An SQL view is a virtual table that is constructed from other tables or views.
- It has no data of its own, but obtains data from tables or other views.
- SELECT statements are used to define views:
  - A view definition may not include an ORDER BY clause.
- SQL views are a subset of the external views:
  - They can be used only for external views that involve one multivalued path through the schema.
- CREATE VIEW Syntax in MySQL 5.6 (http://dev.mysql.com/doc/refman/5.6/en/create-view.html)

SQL VIEWS

- SQL views can be used to hide columns to simplify results or to prevent the display of sensitive data.
- Another purpose of views is to show the results of computed columns without requiring the user to enter the computation expression.
- Developers need not enter a complex SQL statement when they want a particular result.
- You can, however, construct a view that computes a variable and then write an SQL statement on that view that uses the computed variable in a WHERE clause.
- SQL Views have three other important uses.
  - To isolate source data tables from application code
  - To give different sets of processing permissions to the same table
  - to enable the definition of multiple sets of triggers on the same data source

USER-DEFINED FUNCTIONS

```
CREATE FUNCTION function_name (arguments)
     RETURNS data_type
```

- SPLIT Function (UDF) in MySQL 5.6

```
CREATE FUNCTION `SPLIT`(
  x VARCHAR(255),
  delimiter VARCHAR(12),
  pos INT
) RETURNS VARCHAR(255) CHARSET utf8
RETURN REPLACE(SUBSTRING(SUBSTRING_INDEX(x, delimiter, pos),
       CHAR_LENGTH(SUBSTRING_INDEX(x, delimiter, pos -1)) + 1),
       delimiter, '')
```

- CREATE FUNCTION Syntax in MySQL 5.6 (http://dev.mysql.com/doc/refman/5.6/en/create-function-udf.html)

TRIGGERS

- A trigger is a stored program that is executed by the DBMS whenever a specified event occurs on a specified table or view.
- Three trigger types:
  BEFORE, INSTEAD OF, and AFTER
  - Each type can be declared for Insert, Update, and Delete.
  - Resulting in a total of nine trigger types.
  - MySQL 5.6 supports two trigger types (BEFORE and AFTER).
- The four uses are as follows:
  - Providing default values
  - Enforcing data constraints
  - Updating SQL views
  - Performing referential integrity actions

STORED PROCEDURES

- A stored procedure is a program that is stored within the database and is compiled when used.
  - In Oracle, it can be written in PL/SQL or Java.
  - In SQL Server, it can be written in TRANSACT-SQL.
- Stored procedures can receive input parameters and they can return results.
- Stored procedures can be called from many program languages
- Greater security as stored procedures are always stored on the database server
- Decreased network traffic
- SQL can be optimized by the DBMS compiler
- Code sharing resulting in:
  - Less work
  - Standardized processing
  - Specialization among developers