Databases – Introduction to SQL

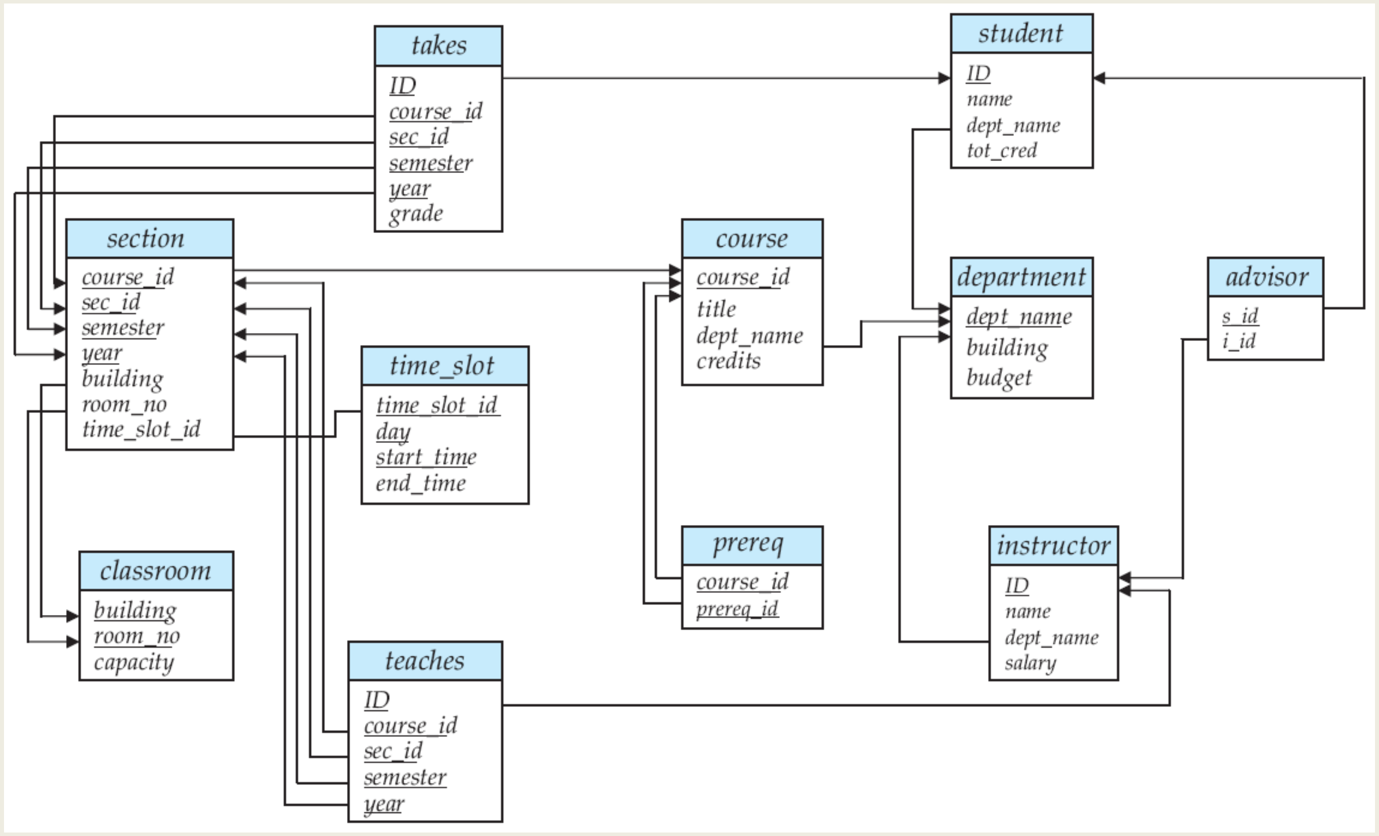
Jaeyong Choi
Dept. of Software, Gachon University

Overview

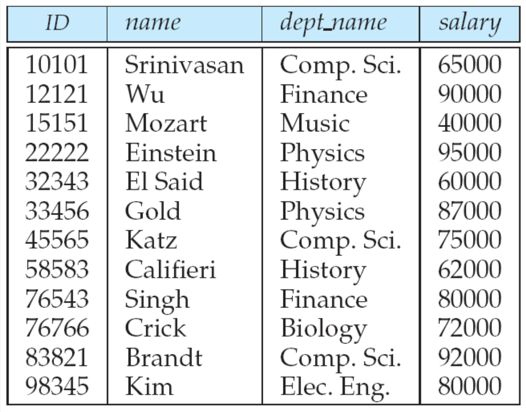
- History
  - IBM *Sequel* language developed as part of System R project at the IBM San Jose Research Laboratory
  - Renamed *Structured Query Language* (SQL)
  - ANSI and ISO standard SQL:
    - SQL-86, SQL-89, SQL-92
    - SQL:1999, SQL:2003, …, SQL: 2019
  - Commercial systems offer most of SQL-92 features
    - Plus varying feature sets from later standards and special proprietary features
    - NOTE: Some examples here may not work on your particular system

Sample Database

- University database

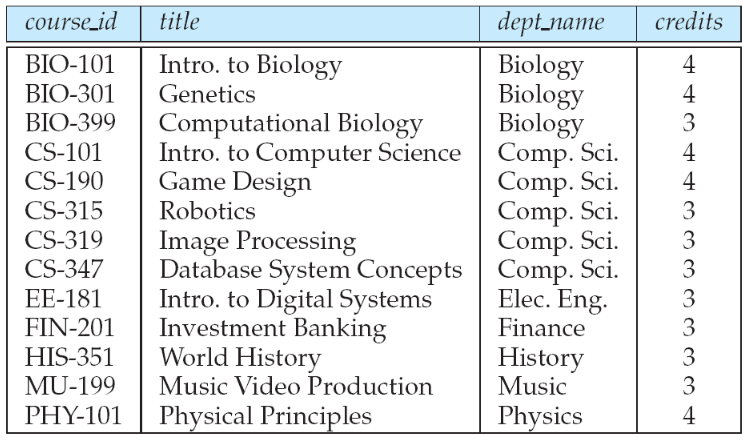


University database *cont'd*



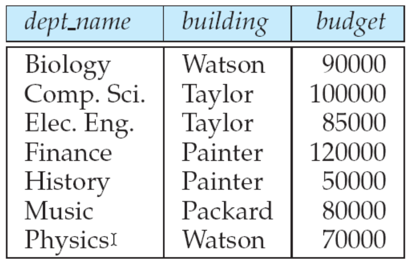
ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.



course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

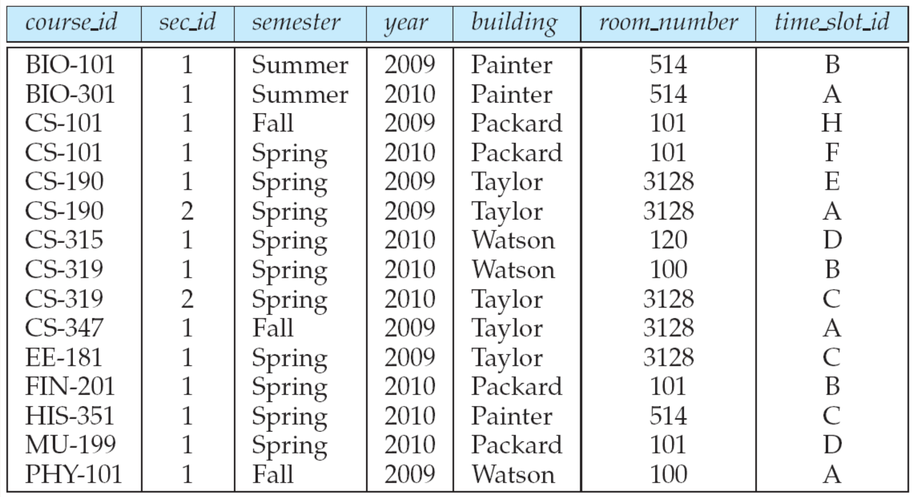
Figure 2.2 The *course* relation.



dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

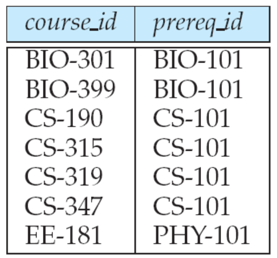
Figure 2.5 The *department* relation.

University database *cont'd*



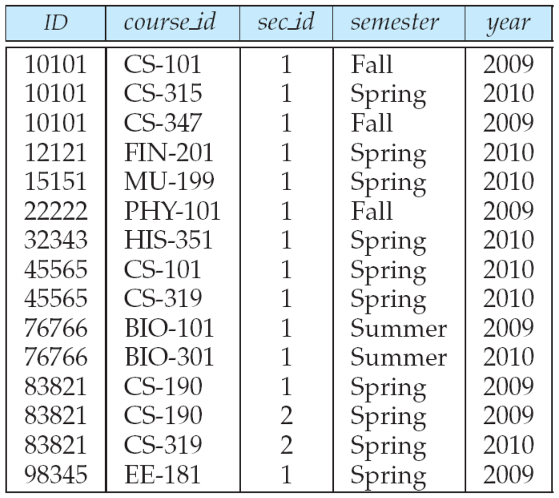
course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.



course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

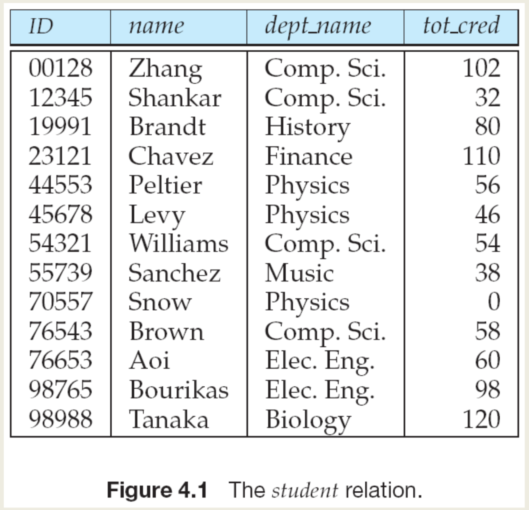
Figure 2.3 The *prereq* relation.



ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

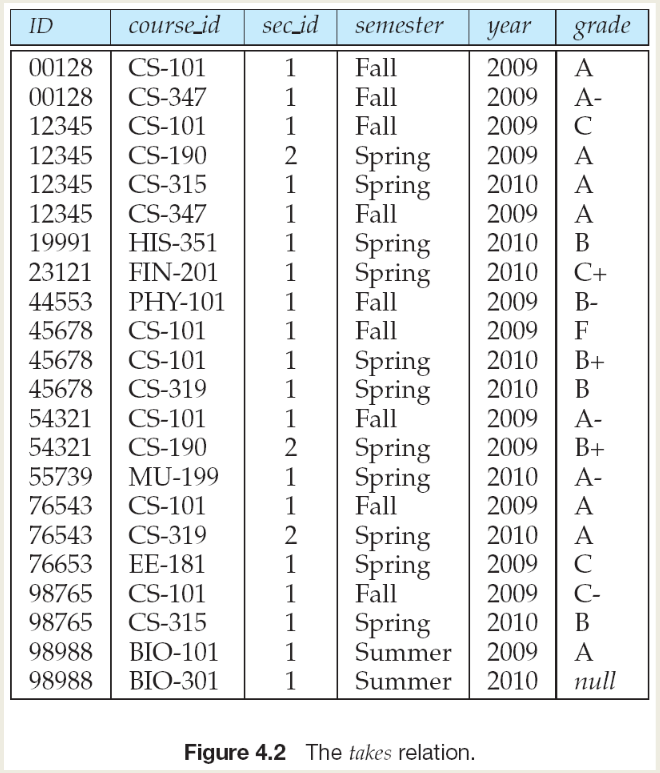
Figure 2.7 The *teaches* relation.

University database *cont'd*



ID	*name*	*dept_name*	*tot_cred*
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Figure 4.1 The *student* relation.

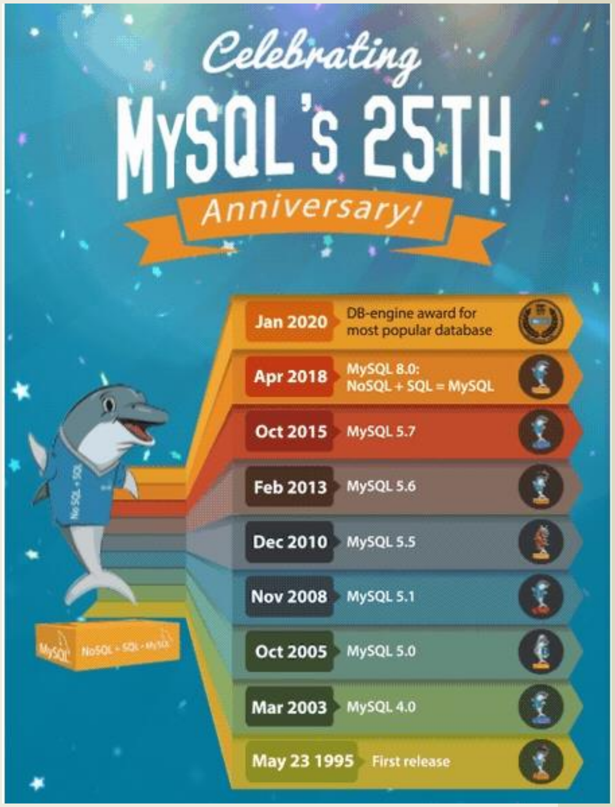


ID	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	CS-101	1	Fall	2009	A
00128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	*null*

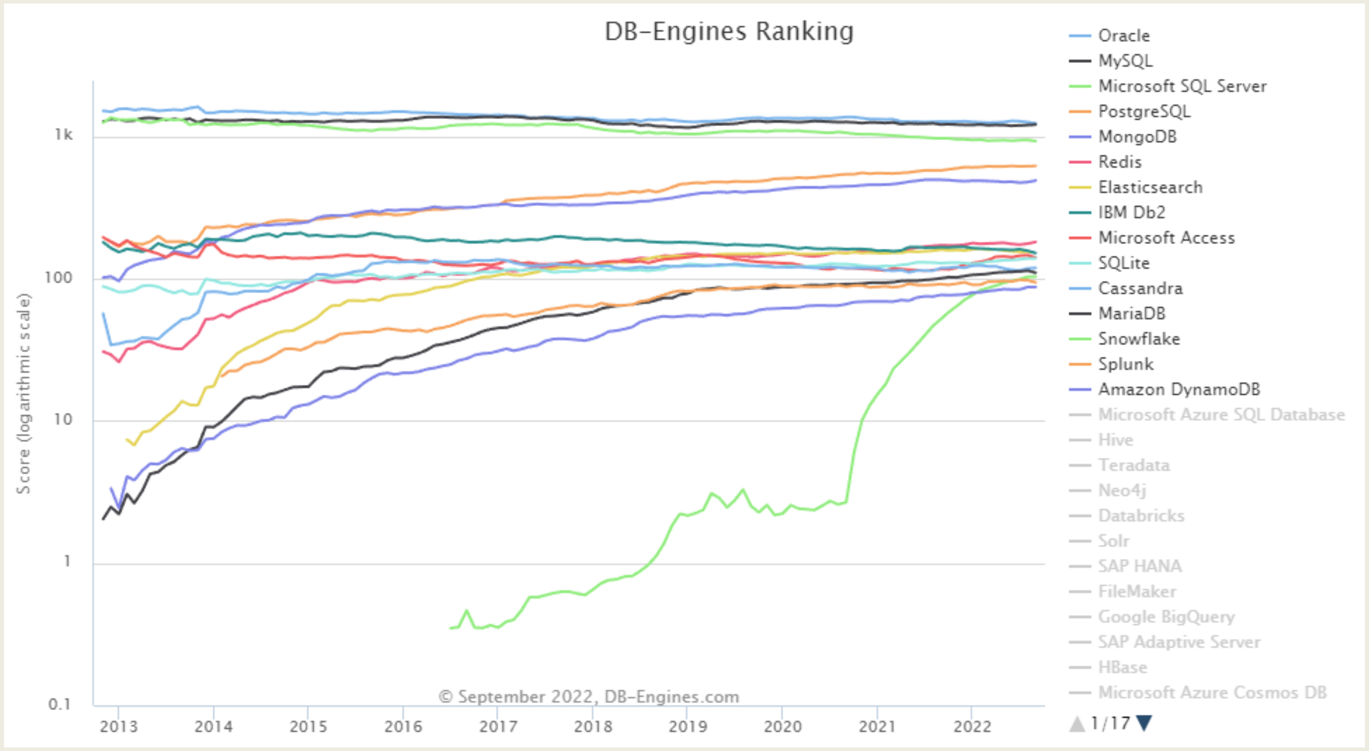
Figure 4.2 The *takes* relation.

MySQL

- MySQL
  - A free open-source database management system (DBMS)
    - Pronounced "My S-Q-L" or "My Sequel"
  - A popular choice as the database system for use with web applications (a component of LAMP)
    - **L**inux – **A**pache – **M**ySQL – **P**HP
  - Widely used in various web sites
    - Facebook, Google, Wikipedia, Twitter, Flickr, and YouTube

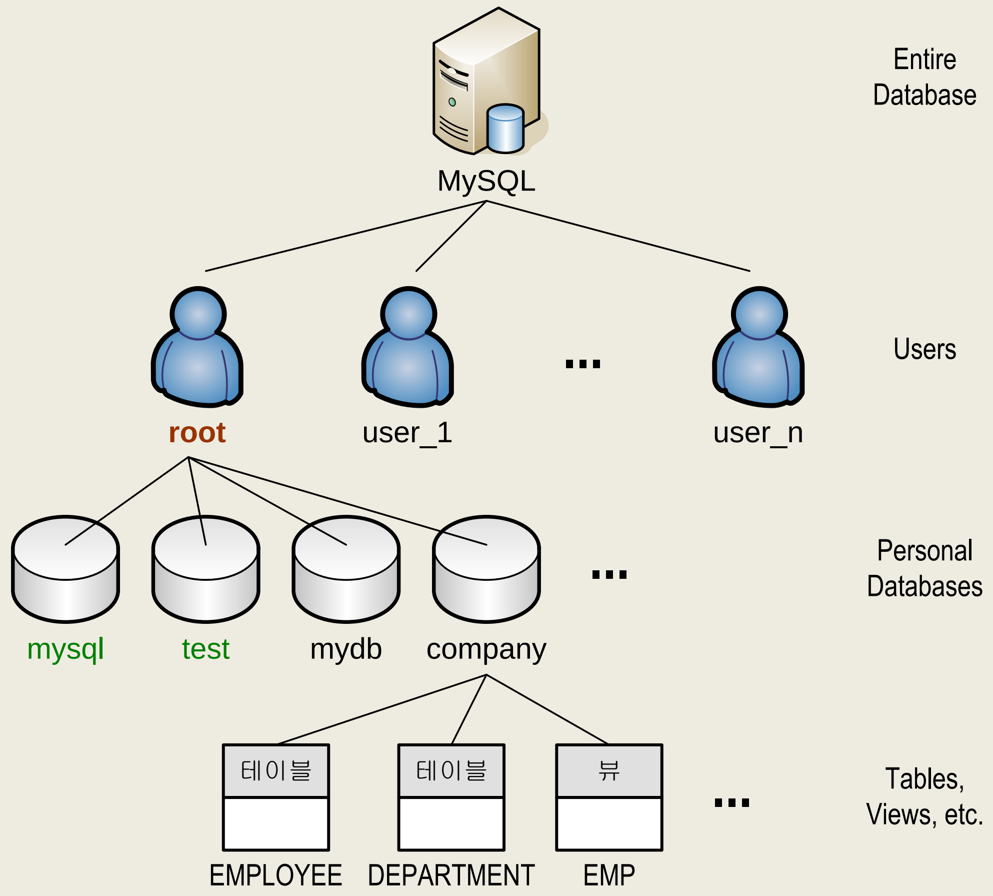


- MySQL usage
  - Ranked second after Oracle; first among open source databases (as of Sept. 2022)



http://db-engines.com/en/ranking

MySQL database hierarchy



Using MySQL

- Connect to MySQL server

```
C:\> mysql -u root -p mysql
C:\> mysql -u root -p mydb
C:\> mysql -u root -p
Enter password: 12345
```

Using MySQL

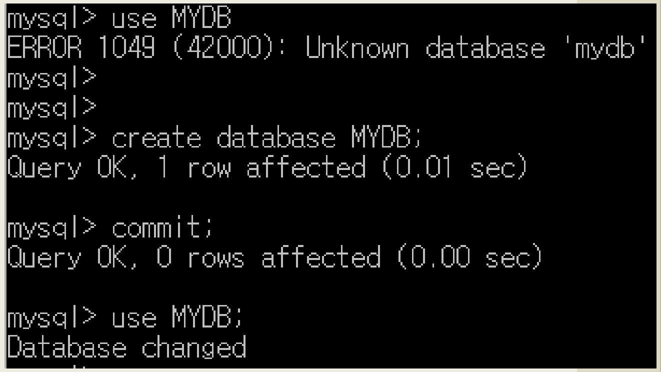
- **Create a database**

  mysql> **CREATE** DATABASE MYDB;

  mysql> **COMMIT**;

  mysql> **USE** MYDB;

  (Note: MySQL commands and SQL database/table/attribute names are case-insensitive!)



```
mysql> use MYDB
ERROR 1049 (42000): Unknown database 'mydb'
mysql>
mysql>
mysql> create database MYDB;
Query OK, 1 row affected (0.01 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> use MYDB;
Database changed
```

- **Example**

  mysql> **CREATE** DATABASE MYDB;

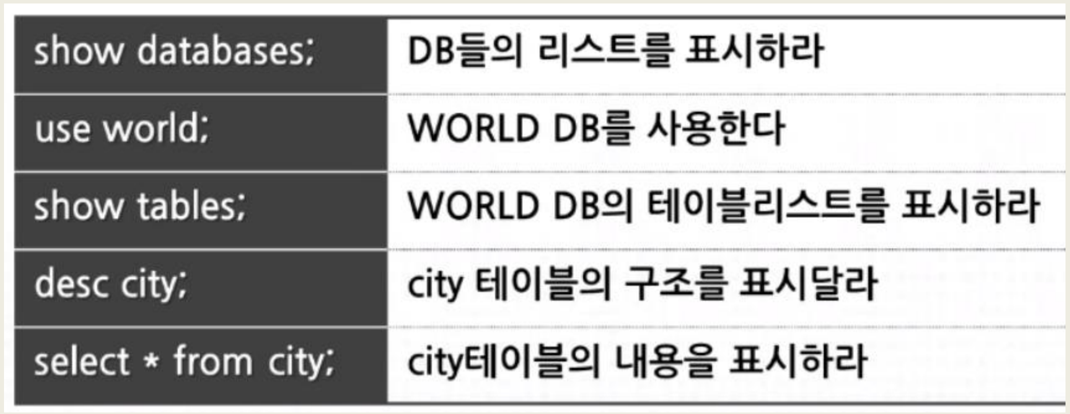
  mysql> **CREATE** TABLE MyTable (칼럼명1 data_type, 칼럼명2 data_type);

Using MySQL

- Update root password

```
mysql> USE MYSQL;
mysql> UPDATE USER
          SET PASSWORD=PASSWORD('12345')
          WHERE USER='root';
mysql> FLUSH PRIVILEGES;
```

- Others



show databases;	DB들의 리스트를 표시하라
use world;	WORLD DB를 사용한다
show tables;	WORLD DB의 테이블리스트를 표시하라
desc city;	city 테이블의 구조를 표시달라
select * from city;	city테이블의 내용을 표시하라

Using MySQL

❑ Manage databases

mysql> **SHOW** DATABASES;

mysql> **USE** MYDB;

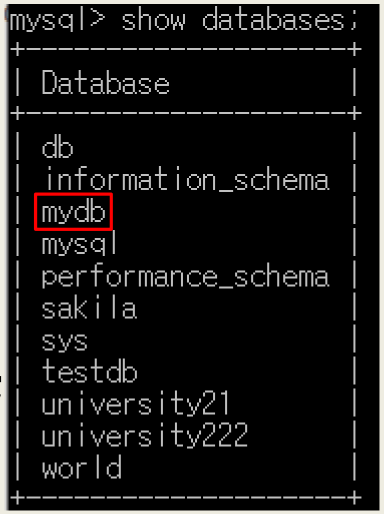
mysql> **DELETE** DATABASE MYDB;

mysql> **TRUNCATE** DATABASE MYDB;

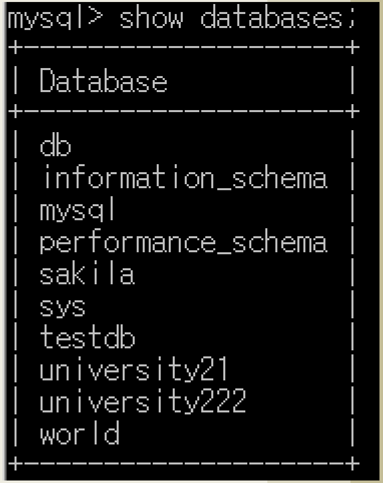
mysql> **DROP** DATABASE MYDB;

mysql> **COMMIT**;

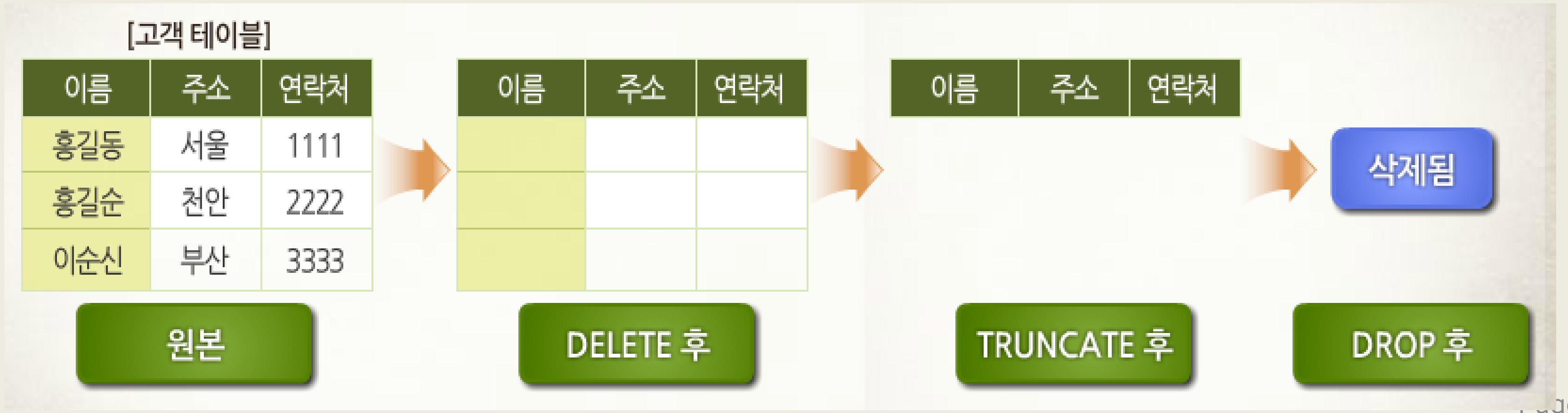
mysql> **DROP** DATABASE MYDB;



```
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| db                   |
| information_schema   |
| mydb                 |
| mysql                |
| performance_schema   |
| sakila               |
| sys                  |
| testdb               |
| university21         |
| university222        |
| world                |
+----------------------+
```



```
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| db                   |
| information_schema   |
| mysql                |
| performance_schema   |
| sakila               |
| sys                  |
| testdb               |
| university21         |
| university222        |
| world                |
+----------------------+
```



[고객 테이블]

이름	주소	연락처
홍길동	서울	1111
홍길순	천안	2222
이순신	부산	3333

원본

이름	주소	연락처

DELETE 후

이름	주소	연락처

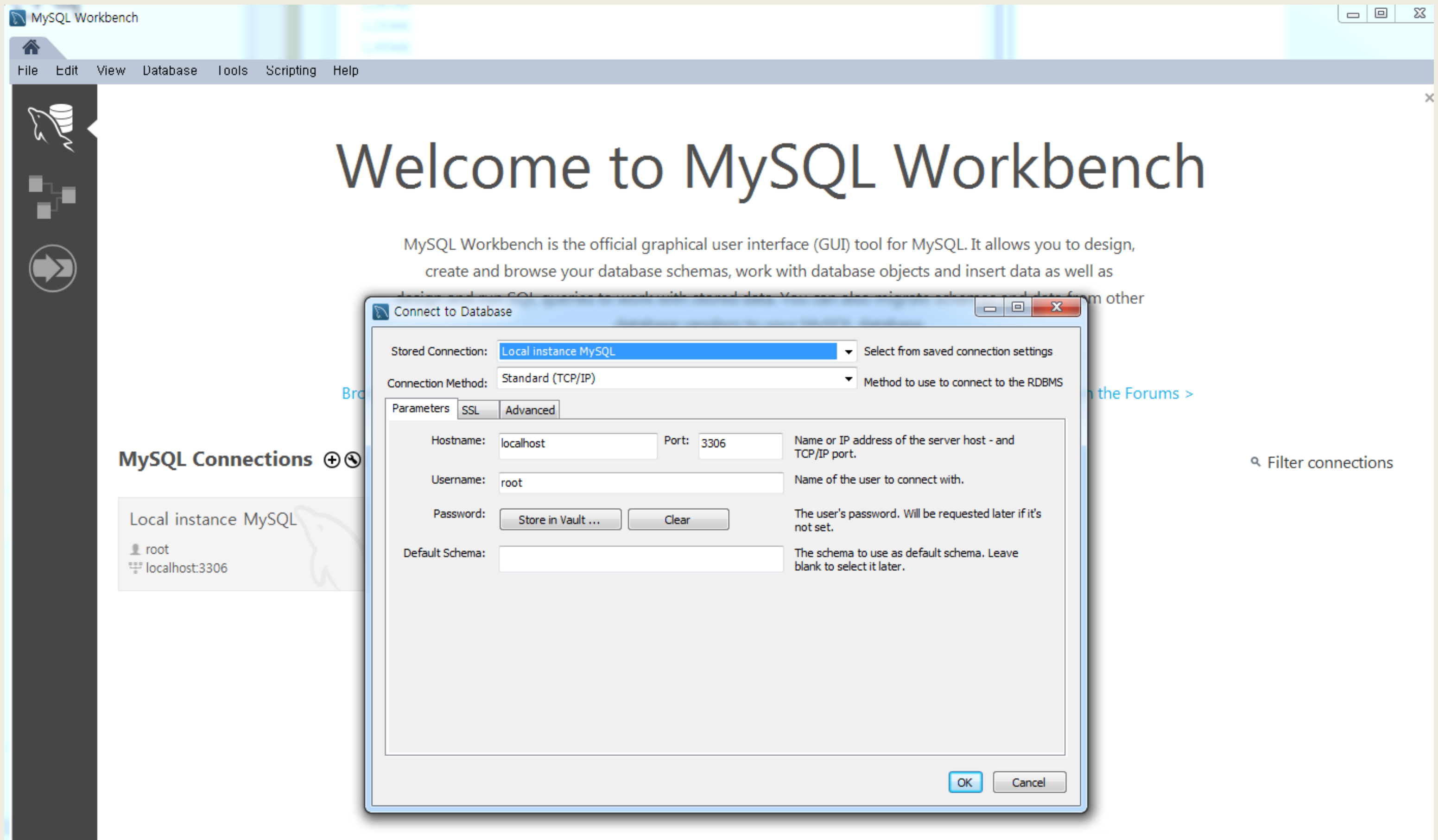
TRUNCATE 후

삭제됨

DROP 후

Exercise

- Program → MySQL workbench 8.0 CE 선택
- Database → connect to database



Connect to Database

Stored Connection:	Local instance MySQL	Select from saved connection settings
Connection Method:	Standard (TCP/IP)	Method to use to connect to the RDBMS

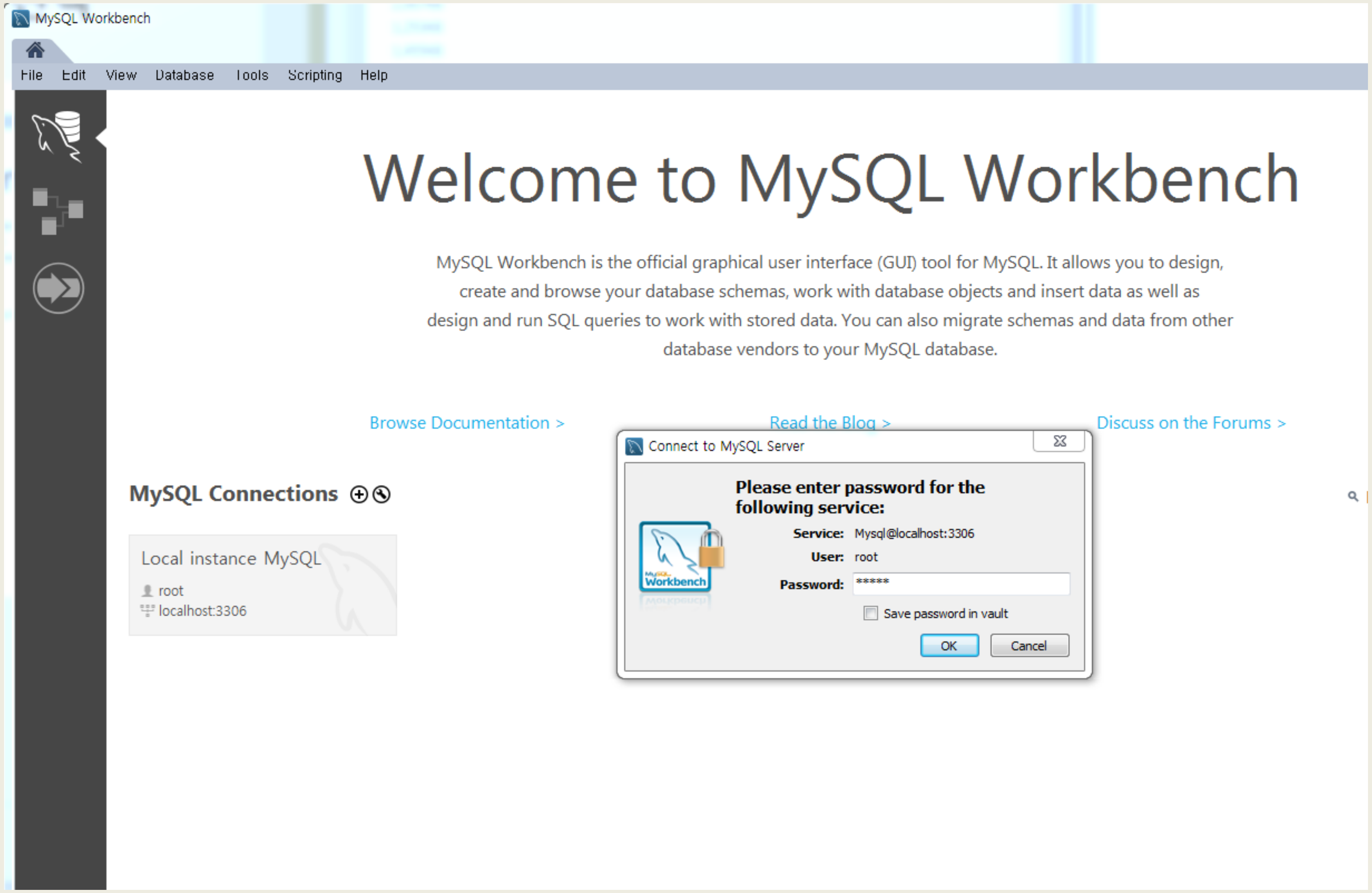
Parameters | SSL | Advanced

Hostname:	localhost Port: 3306	Name or IP address of the server host - and TCP/IP port.
Username:	root	Name of the user to connect with.
Password:	Store in Vault ... Clear	The user's password. Will be requested later if it's not set.
Default Schema:		The schema to use as default schema. Leave blank to select it later.

MySQL Workbench
File Edit View Database Tools Scripting Help
Welcome to MySQL Workbench
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as
m other
the Forums >
MySQL Connections
Filter connections
Local instance MySQL
root
localhost:3306
OK
Cancel

Exercise

- Password 입력

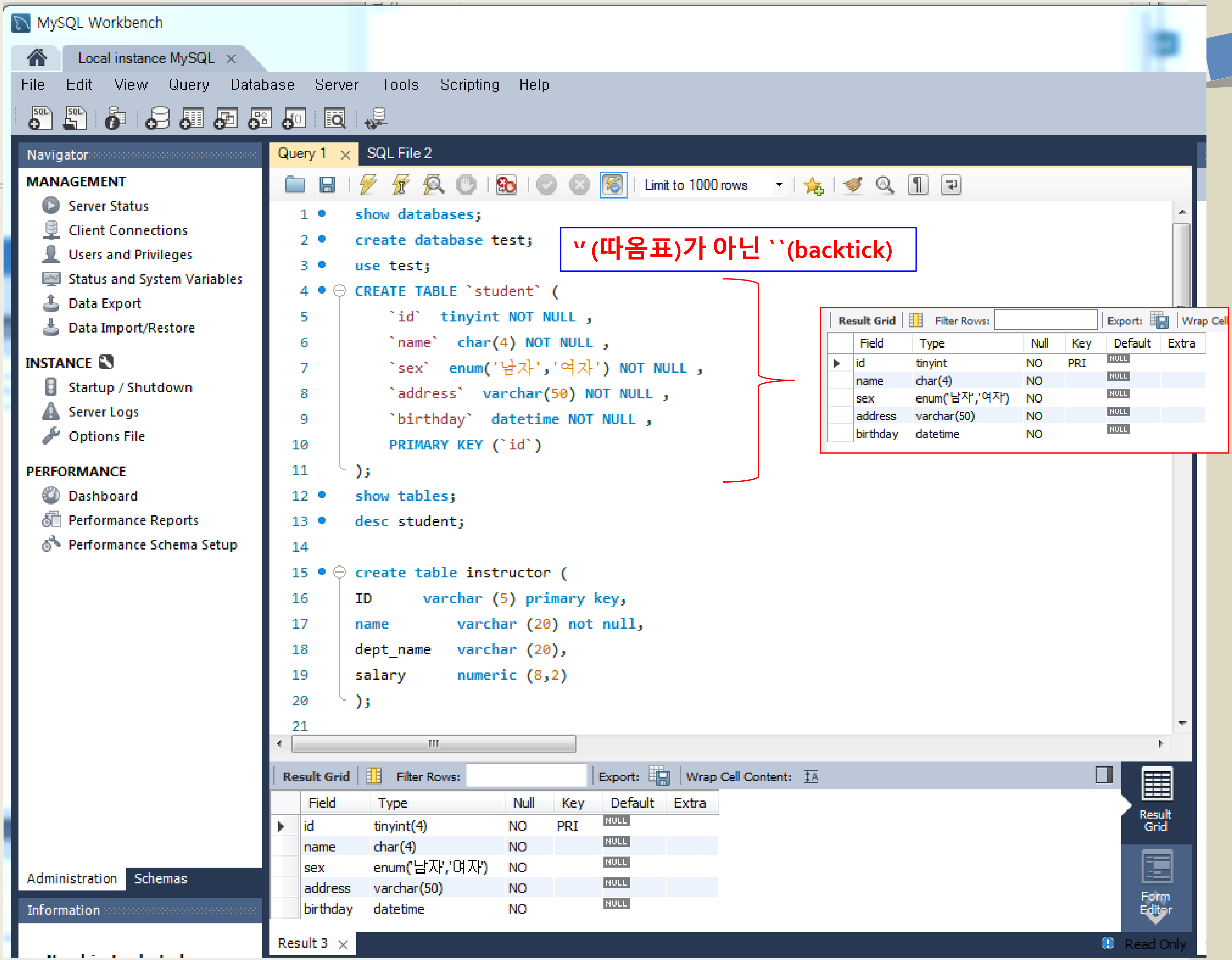


Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Please enter password for the following service:

Service: Mysql@localhost:3306
User: root
Password: *****
Save password in vault



```
show databases;
create database test;
use test;
CREATE TABLE `student` (
    `id` tinyint NOT NULL ,
    `name`  char(4) NOT NULL ,
    `sex`  enum('남자','여자') NOT NULL ,
    `address`  varchar(50) NOT NULL ,
    `birthday`  datetime NOT NULL ,
    PRIMARY KEY (`id`)
);
show tables;
desc student;

create table instructor (
ID       varchar (5) primary key,
name         varchar (20) not null,
dept_name    varchar (20),
salary       numeric (8,2)
);
```

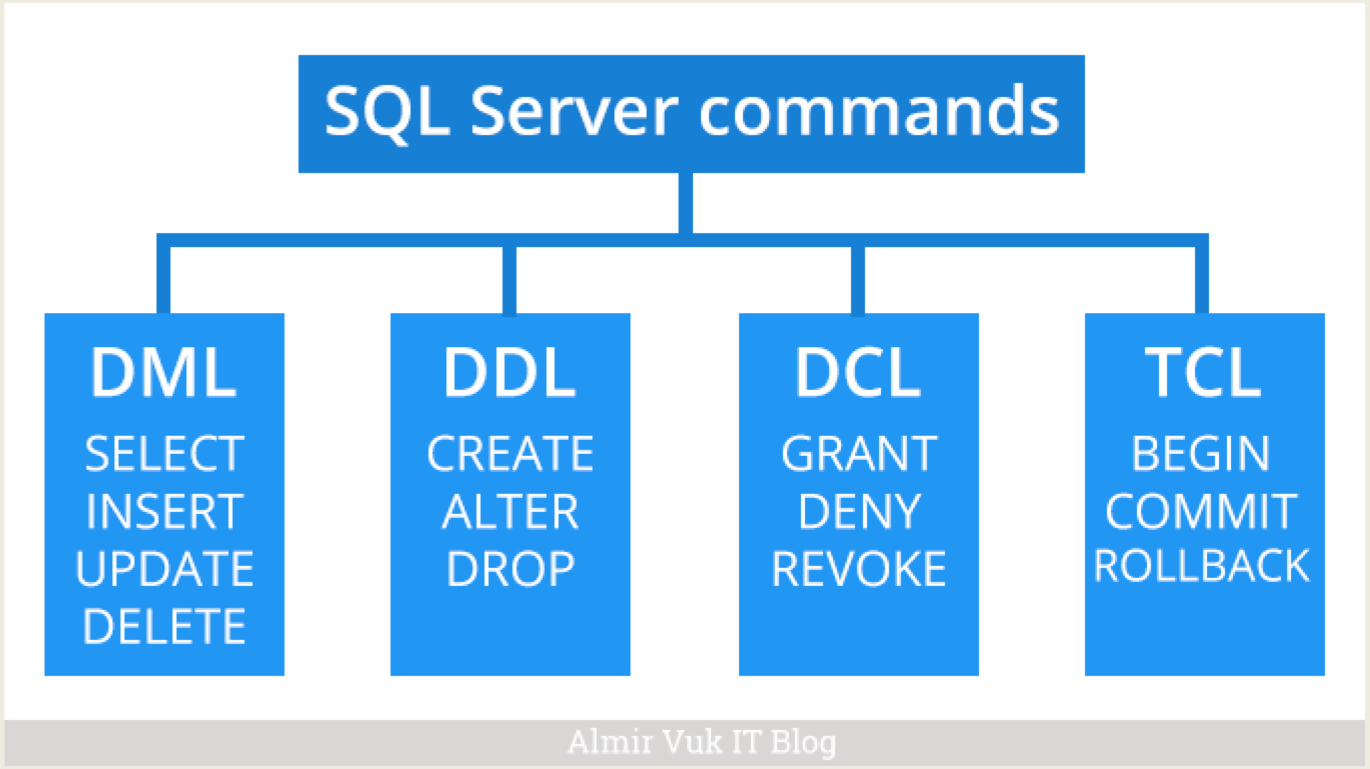
' (따옴표)가 아닌 ``(backtick)

Field	Type	Null	Key	Default	Extra
id	tinyint	NO	PRI	NULL	
name	char(4)	NO		NULL	
sex	enum('남자','여자')	NO		NULL	
address	varchar(50)	NO		NULL	
birthday	datetime	NO		NULL	

Field	Type	Null	Key	Default	Extra
id	tinyint(4)	NO	PRI	NULL	
name	char(4)	NO		NULL	
sex	enum('남자','여자')	NO		NULL	
address	varchar(50)	NO		NULL	
birthday	datetime	NO		NULL	

For more information

- Check the MySQL documentation
- https://dev.mysql.com/doc/refman/8.0/en/



SQL Data Definition

- **Data definition language (DDL)**
  - Allows the specification of information about relations
    - Schema for each relation
    - Domain of values associated with each attribute
    - Integrity constraints
  - Also other information such as:
    - Set of indices to be maintained for each relations
    - Security and authorization information for each relation
    - Physical storage structure of each relation on disk

Domain types in SQL

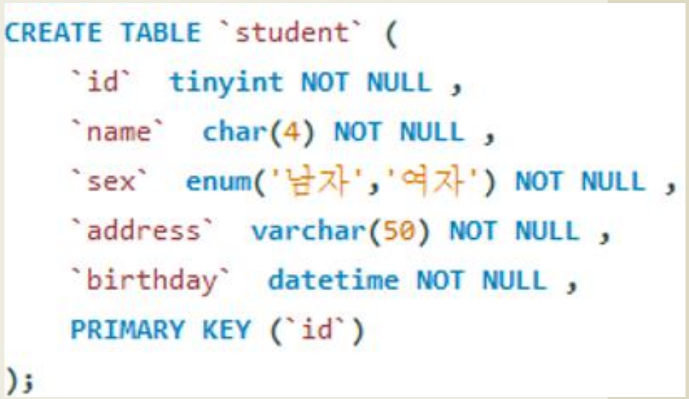
- **char(n)** – fixed length character string, with user-specified length *n*
- **varchar(n)** – variable length character strings, with user-specified maximum length *n*
- **int** – integer (a finite machine-dependent subset of integers)
- **tinyint** – 1 byte integer
- **numeric(p,d)** – fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point
  - E.g., numeric(3,1) allows 44.5, but not 444.5 or 0.32
- **real, double precision** – floating point and double-precision floating point numbers, with machine-dependent precision
- **float(n)** – floating point number, with user-specified precision of at least *n* digits
- and more

Examples of DDL commands

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database.

Create table construct

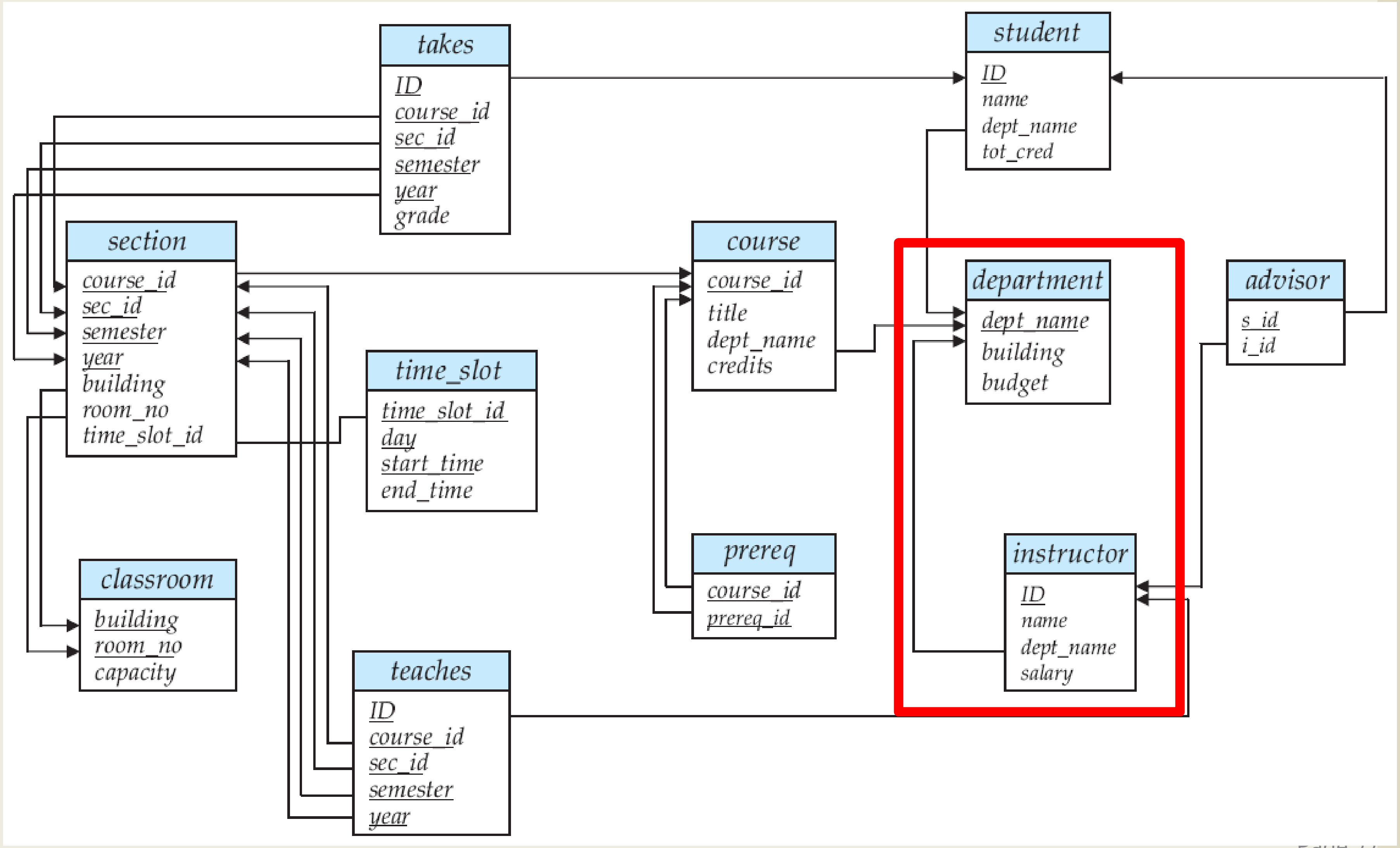
- An SQL relation is defined using the create table command:
  - **create table** *r* (A_1 D_1, A_2 D_2, ... , A_n D_n,
    <integrity-constraint$_1$>,
    ... ,
    <integrity-constraint$_k$>);
  - *r* is the name of the relation
  - Each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i



```
CREATE TABLE `student` (
    `id`  tinyint NOT NULL ,
    `name`  char(4) NOT NULL ,
    `sex`  enum('남자','여자') NOT NULL ,
    `address`  varchar(50) NOT NULL ,
    `birthday`  datetime NOT NULL ,
    PRIMARY KEY (`id`)
);
```

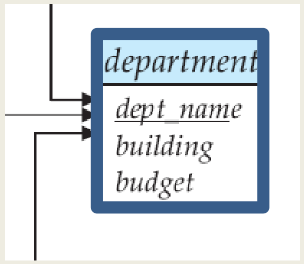
- Integrity constraints(무결성 제약조건) in create table
  - **not null** ➔ Required fields
  - **primary key** (A_1, ... , A_n)
  - **foreign key** (A_m, ... , A_n) **references** *r*

Example

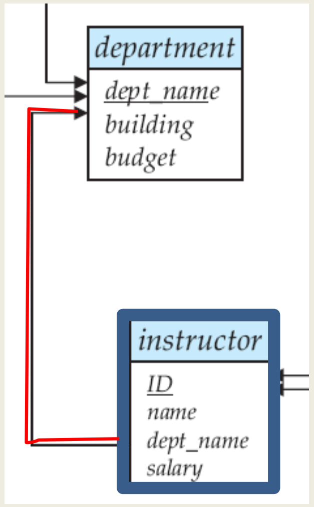

takes
ID
course_id
sec_id
semester
year
grade
student
ID
name
dept_name
tot_cred
section
course_id
sec_id
semester
year
building
room_no
time_slot_id
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
time_slot
time_slot_id
day
start_time
end_time
classroom
building
room_no
capacity
prereq
course_id
prereq_id
instructor
ID
name
dept_name
salary
teaches
ID
course_id
sec_id
semester
year

Example

- **create table** department (
```
        dept_name   varchar (20),
        building    varchar (15),
        budget      numeric (12,2),
        primary key (dept_name));
```



- **create table** instructor (
```
        ID          varchar (5),
        name        varchar (20) not null,
        dept_name   varchar (20),
        salary      numeric (8,2),
        primary key (ID),
        foreign key (dept_name)
                    references department (dept_name));
```



- **primary key** declaration automatically ensures not null

Example – alter the table

- **ALTER TABLE** changes the structure of a table
  - add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself.
  - Multiple ADD, ALTER, DROP, and CHANGE clauses are permitted in a single ALTER TABLE statement, separated by commas.

- **Examples**
  - Add and Drop column
    - alter table instructor add column [a] [int];
    - alter table instructor drop column [a];
  - Modify column
    - ALTER TABLE [table_name] MODIFY COLUMN [ex_column] [varchar(16)] NULL;

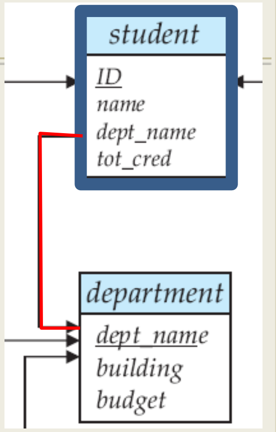
Example – alter the table

- Change name and definition (이름까지 변경)
  - **ALTER TABLE** t1 **CHANGE** a b **BIGINT NOT NULL**;
  - **ALTER TABLE** t1 **RENAME COLUMN** b **TO** a;

- Modify constraints
  - alter table instructor add foreign key (dept_name) references department(dept_name) ;
  - **ALTER TABLE** *t1* **DROP FOREIGN KEY** *fk_symbol*;

- Multiple column changes
  - **ALTER TABLE** t2 **DROP COLUMN** c, **DROP COLUMN** d;

More relation definitions

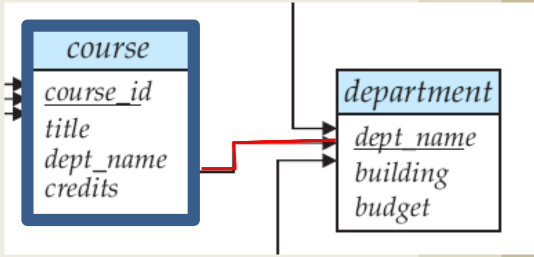


- **create table** student (

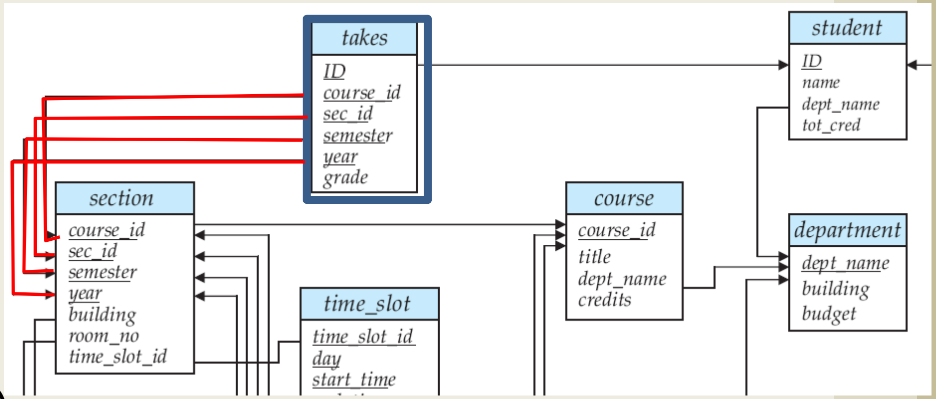
```
ID          varchar(5) primary key,
name        varchar(20) not null,
dept_name   varchar(20)
            references department (dept_name),
tot_cred    numeric(3,0));
```

- **create table** course (

```
course_id   varchar (7) primary key,
title       varchar (50),
dept_name   varchar (20)
            references department (dept_name),
credits              numeric (2,0));
```



More relation definitions *cont'd*



- **create table** takes (

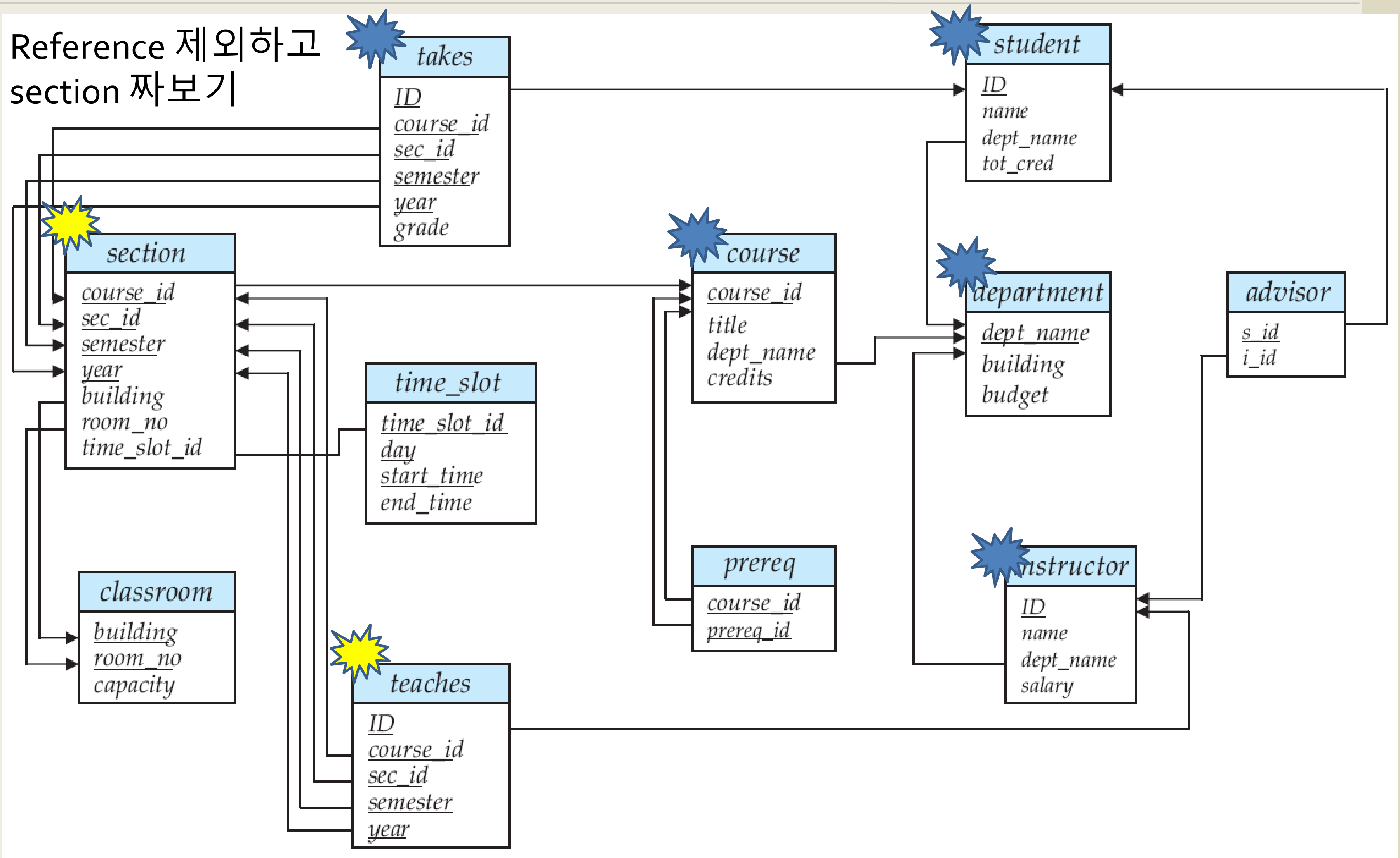
```
    ID            varchar(5),
    course_id     varchar(8),
    sec_id        varchar(8),
    semester      varchar(6),
    year          numeric(4,0),
    grade         varchar(2),
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (ID) references  student (ID),
    foreign key (course_id, sec_id, semester, year)
            references section (course_id, sec_id, semester, year));
```

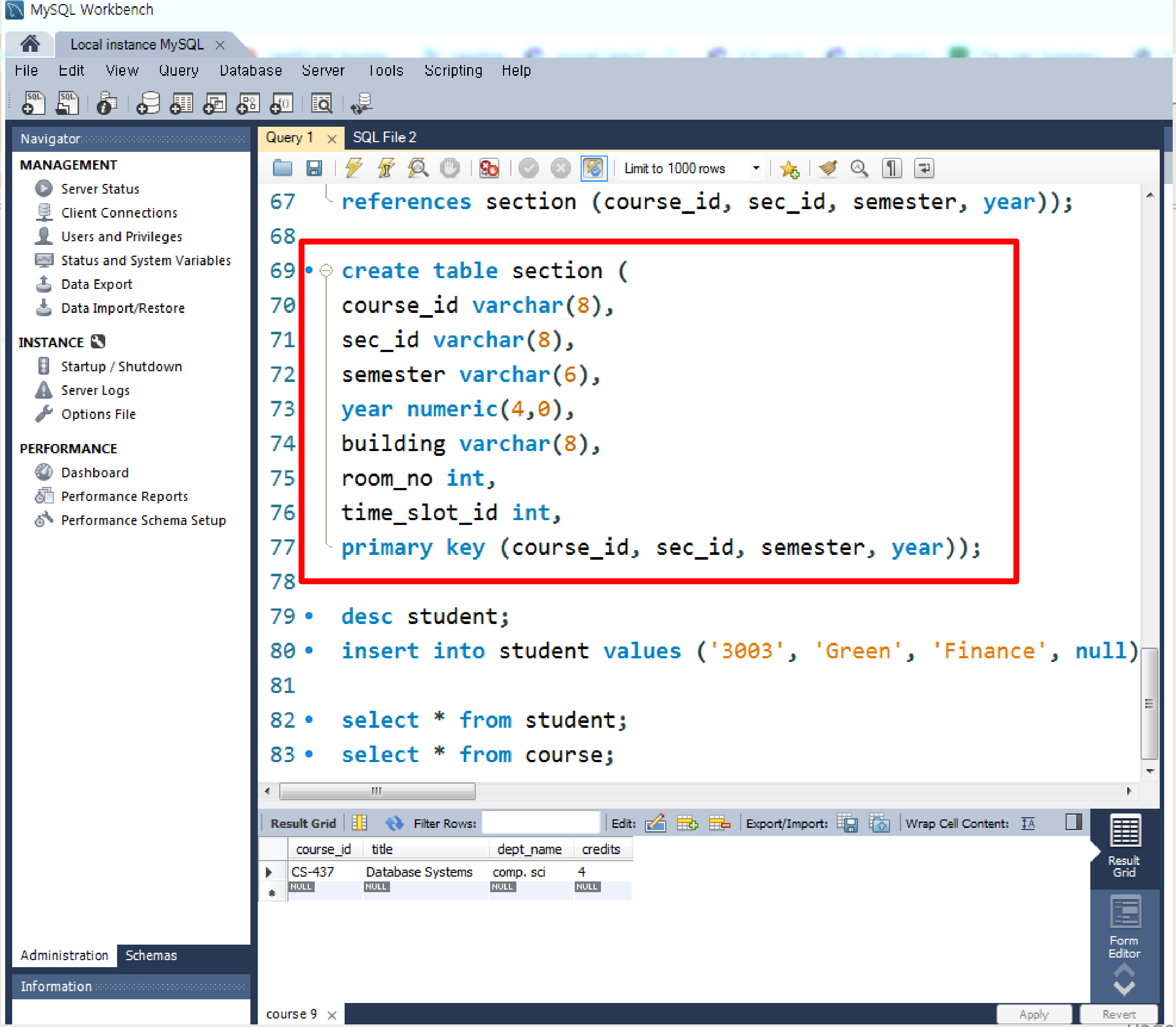
- Note: *sec_id(분반)* can be dropped from primary key above, to ensure a student cannot be registered for two sections of the same course in the same semester

ID	Course_id	Sec_id	semester	Year
20191123	1111	01	Fall	2019
20191123	1111	02	Fall	2019

Exercise -Let's try!

Reference 제외하고
section 짜보기





```
references section (course_id, sec_id, semester, year));

create table section (
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
building varchar(8),
room_no int,
time_slot_id int,
primary key (course_id, sec_id, semester, year));

desc student;
insert into student values ('3003', 'Green', 'Finance', null)

select * from student;
select * from course;
```

course_id	title	dept_name	credits
CS-437	Database Systems	comp. sci	4
NULL	NULL	NULL	NULL