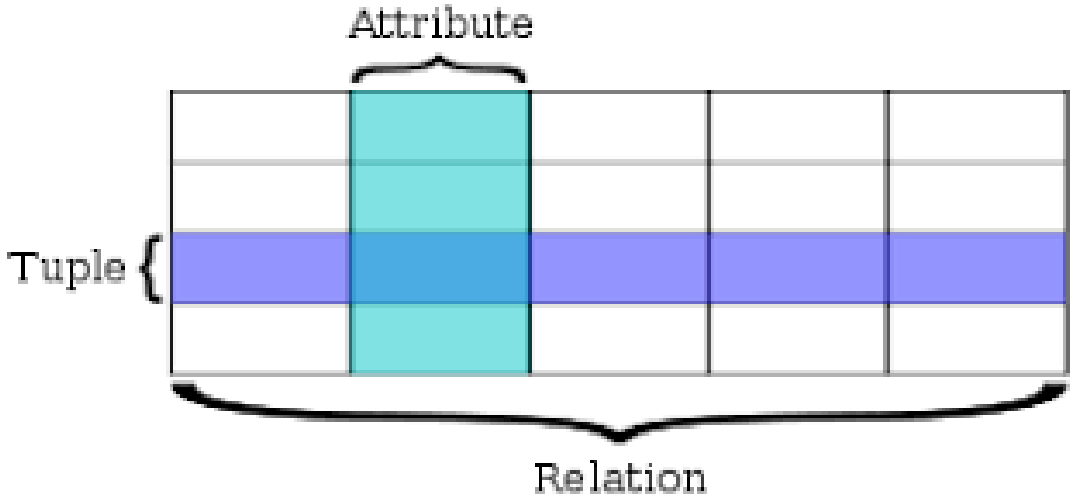
Database

panya

Relational Database

일련의 정형화된 테이블로 구성된 데이터 항목들의 집합



Key

- 해당 data를 고유하게 만들어주는 attribute(s)
- 하나 또는 여러 개의 attribute의 조합일 수 있다.
- Table의 각 row에는 저만의 고유 키가 있다.

+ table끼리 연결할 때 key를 사용

Key의 특성

1. Uniqueness (유일성) : 하나의 table에서 모든 tuple이 다른 값을 가지고 있어야 함
2. Minimality (최소성) : 최소 개수의 attributes들로 이루어져 있어야 함

About Relational Database

- **Super key**
  - 유일성을 만족하는 key
- **Candidate key**
  - 유일성 및 최소성을 만족하는 key
- **Primary Key**
  - Candidate key들 중, 기본적으로 사용하는 key
- **Alternate Key**
  - Primary key를 제외한 나머지 Candidate key
- **Foreign Key**
  - 다른 table을 참조하기 위한 key
  - 참조할 table의 primary key의 값과 match

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

<학생 Table>

학번	주민등록번호	성명
A11	111-123	홍길동
A12	112-789	김철수
A13	119-753	박영희
A14	115-951	홍길동

<수강 Table>

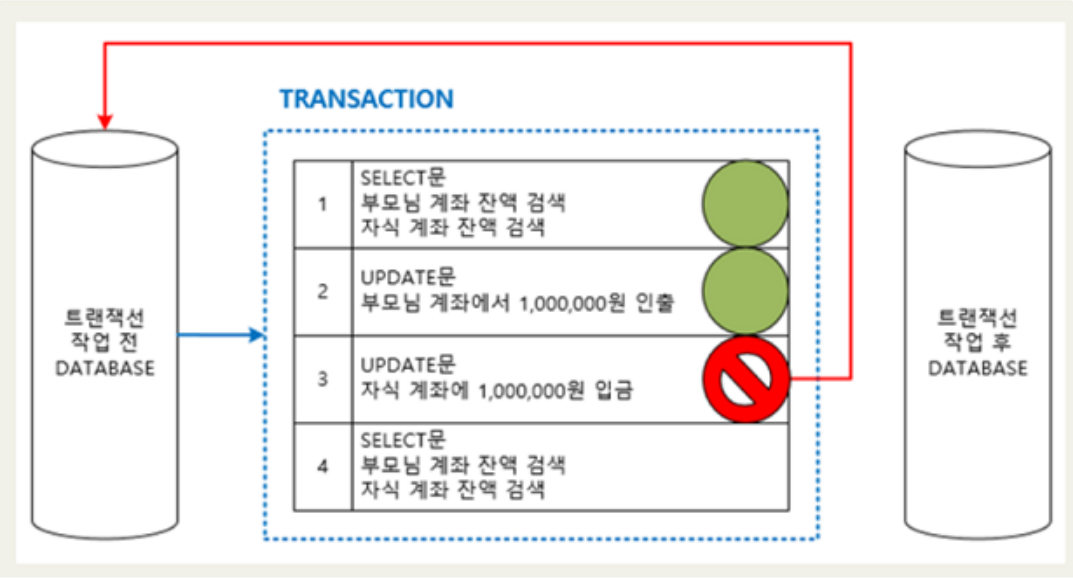
학번	과목명
A11	영어
A11	수학
A13	영어
A13	수학

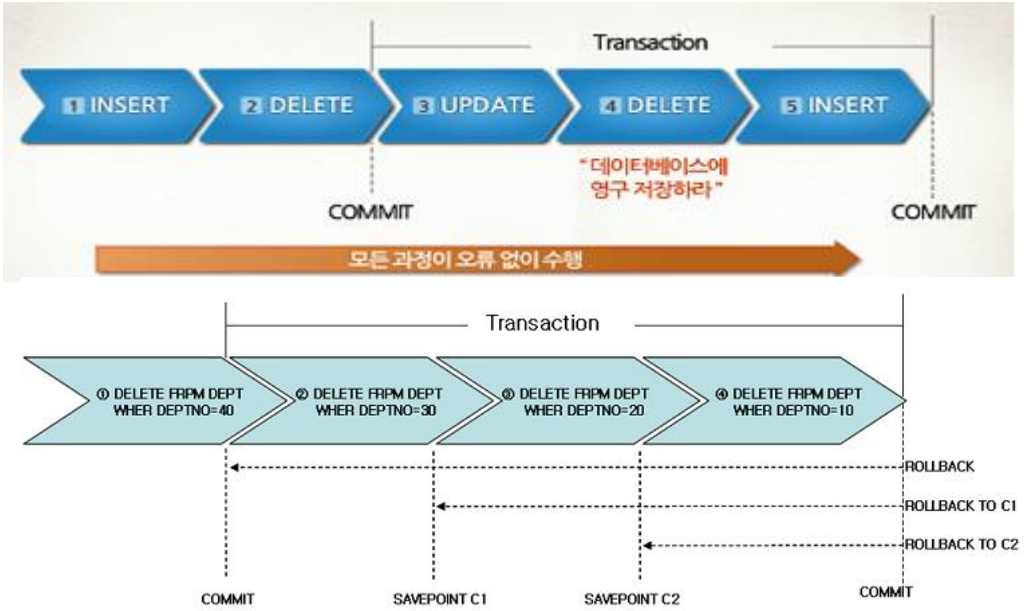
Transaction

하나의 논리적 단위를 구성하는 데이터베이스 연산의 모임

데이터의 일관성을 유지하면서 안정적으로 데이터를 복구하기 위함

Commit, Rollback, Savepoint





About Relational Database

ACID 원칙 데이터베이스 트랜잭션이 안전하게 수행된다는 것을 보장하기 위한 성질

- **Atomicity (원자성)**
  - Transaction을 구성하는 모든 명령이 실행되거나 어떠한 명령도 실행되지 않아야 한다.
- **Consistency (일관성)**
  - Transaction이 끝난 후 DB는 일관성을 유지해야 한다.
- **Isolation (독립성)**
  - 모든 Transaction은 동시에 일어나는 다른 Transaction과 상관없이 데이터베이스에 대해 일관된 뷰를 가지고 있다.
- **Durability (지속성)**
  - Transaction이 끝난 후, 데이터베이스는 데이터를 정확히 저장하고 정전이나 그 외의 이상 상황으로부터 데이터를 보호해야 한다.

Normalization

관계형 데이터베이스의 설계에서 중복을 최소화하게 데이터를 구조화하는 프로세스

정해진 규칙에 의해 규격화

- 응용프로그램 단에서 불필요한 로직을 없앨 수 있다.
- 올바른 데이터만 얻을 수 있다. (변칙 방지)
- 불필요한 쿼리(예를 들면, 서브쿼리) 제거로 성능 향상

1~5차 정규화 + BCNF의 6단계 존재

보통 3단계까지 사용

Normalization

1차 정규형 (1NF)

원자적 데이터로 구성된 열은 그 열에 같은 타입의 데이터를 여러개 가질 수 없다.

원자적 데이터로 구성된 테이블은 같은 타입의 데이터를 여러개 가질 수 없다.

Customer

Customer ID	First Name	Surname	Tel. No. 1	Tel. No. 2	Tel. No. 3
123	Robert	Ingram	555-861-2025		
456	Jane	Wright	555-403-1659	555-776-4100	555-403-1659
789	Maria	Fernandez	555-808-9633		

Customer Name

Customer ID	First Name	Surname
123	Robert	Ingram
456	Jane	Wright
789	Maria	Fernandez

Customer Telephone Number

Customer ID	Telephone Number
123	555-861-2025
456	555-403-1659
456	555-776-4100
789	555-808-9633

Normalization

2차 정규형 (2NF)

1NF여야 한다.

부분적 함수 의존이 없다.

종업원의 기술

종업원	기술	근무지
Jones	Typing	114 Main Street
Jones	Shorthand	114 Main Street
Jones	Whittling	114 Main Street
Bravo	Light Cleaning	73 Industrial Way
Ellis	Alchemy	73 Industrial Way
Ellis	Flying	73 Industrial Way
Harrison	Light Cleaning	73 Industrial Way

종업원

종업원	근무지
Jones	114 Main Street
Bravo	73 Industrial Way
Ellis	73 Industrial Way
Harrison	73 Industrial Way

종업원의 기술

종업원	기술
Jones	Typing
Jones	Shorthand
Jones	Whittling
Bravo	Light Cleaning
Ellis	Alchemy
Ellis	Flying
Harrison	Light Cleaning

Normalization

3차 정규형 (3NF)

2NF여야 한다.

이행적 종속 관계가 없다.

대회 우승자

대회	연도	우승자	우승자 생년 월일
Des Moines Masters	1998	Chip Masterson	14 March 1977
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

대회 우승자

대회	연도	우승자
Des Moines Masters	1998	Chip Masterson
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

우승자 생년 월일

우승자	우승자 생년 월일
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

SQL(Structured Query Language)

관계형 DBMS를 관리하기 위해 설계된 프로그래밍 언어
많은 수의 데이터베이스 관련 프로그램들이 SQL을 표준으로 채택하고 있음

DBMS 처리 언어의 종류

- 데이터 정의어 (DDL : Data Definition Language)
  - 데이터베이스를 생성하는데 사용하기 위한 언어
  - CREATE, ALTER, DROP, TRUNCATE(DROP → CREATE)
- 데이터 조작어(DML : Data Manipulation Language)
  - 데이터의 검색, 수정, 삭제, 삽입 등 데이터를 다루기 위한 언어
  - SELECT, INSERT, UPDATE, DELETE
- 데이터 제어어(DCL : Data Control Language)
  - 데이터베이스를 공유하기 위한 데이터 제어를 정의하고 기술하는 언어
  - COMMIT, ROLLBACK, GRANT, REVOKE

-> SQL은
모든 기능을 수행할 수 있다.

MySQL

- 오픈소스 RDBMS(관계형 데이터베이스 관리 시스템)
- MySQL AB -> SUN -> Oracle
- When Oracle aquired SUN in 2010, MariaDB was created from a forked MySQL project
- Part of LAMP stack(Linux, Apache, MySQL, PHP/Python/Perl)
- Used by many websites
- www.mysql.com

MySQL Engine

Engine

- **서버 엔진**
  - 클라이언트(또는 사용자)가 Query를 요청했을때, Query Parsing과, 스토리지 엔진에 데이터를 요청하는 작업을 수행
- **스토리지 엔진**
  - 물리적 저장장치에서 데이터를 읽어오는 역할
  - DB에서 데이터를 어떠한 방식으로 저장하고 접근할 것인지에 대한 기능을 제공
  - 엔진 종류 마다 동작원리가 다르고, 트랜잭션, 성능과 같은 주요 이슈와 밀접하게 연관

Storage Engine

```
mysql> SHOW ENGINES;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
```

MySQL Storage Engine

- **ISAM**
  - MySQL 3.x 버전까지의 테이블
  - 테이블 최대 용량은 4GB
  - 5.0 버전부터 사라졌다.
- **MyISAM**
  - MySQL 4.x 버전의 기본 스토리지 엔진
  - ISAM의 확장이다.
  - 용량은 OS에 의해 결정된다.
  - 작은 규모의 Database에 적합하다.
- **InnoDB**
  - MySQL 5.5 부터 기본 스토리지 엔진
  - Transaction, Foreign key, locking, row-level 지원
  - 용량 제한이 (거의) 없다.
  - 대규모의 Database에 적합하다.
  - 현재 ara에서 사용중.

Foreign Key Constraint

- **MyISAM**
  - Foreign key Constraint, Transaction 지원 X
- **InnoDB** (ara에 사용)
  - Foreign key Constraint, Transaction 지원 O

- Constraint : 제약
- Foreign Key : 외부 테이블에서 참조한 값
- Foreign Key constraint
  - 어떤 attribute 값을 외부 테이블에서 가져올 수 없다는 제약
  - 못쓴다는 얘기는? 외부에서 값을 가져올 수 없다.

MySQL Setting

MySQL 설치	apt-get install mysql-server
MySQL 시작	/etc/init.d/mysql start (=service mysql start)
MySQL 종료	/etc/init.d/mysql stop (=service mysql stop)
MySQL 재시작	/etc/init.d/mysql restart (=service mysql restart)
MySQL 상태	/etc/init.d/mysql status (=service mysql status)

MySQL Setting

MySQL설치

- AWS접속
- apt-get 패키지 목록 갱신

  $ sudo apt-get update

- mysql 서버 설치

  $ sudo apt-get install mysql-server

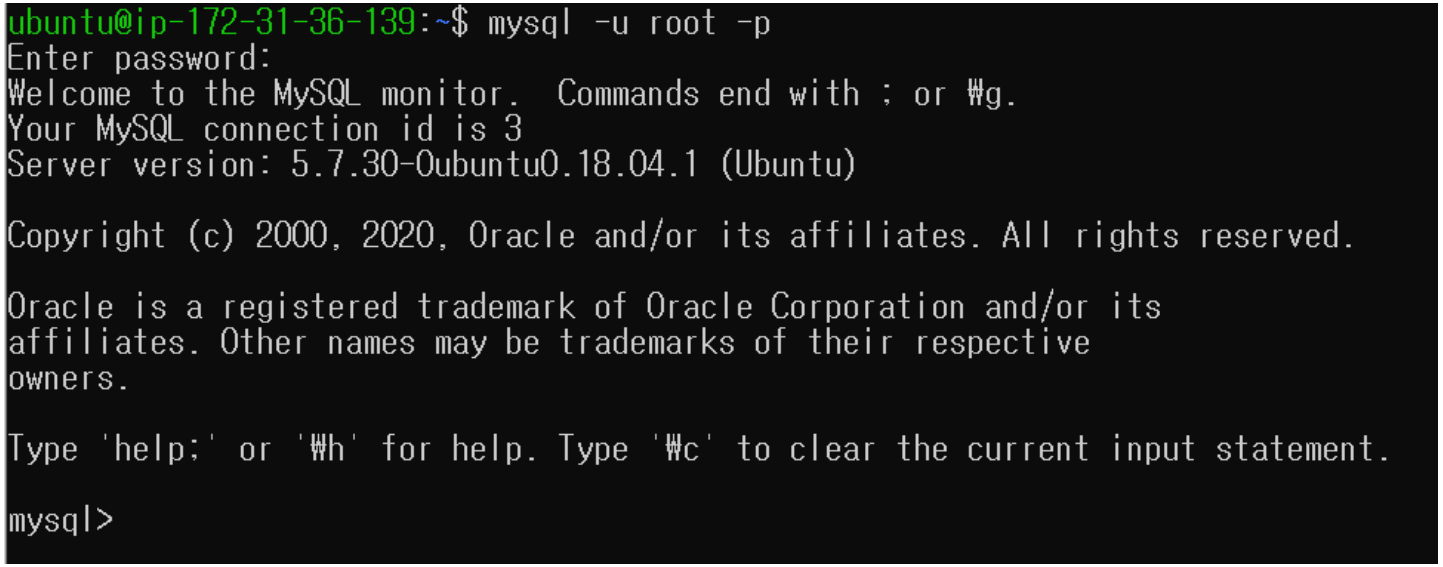
초기 설정

```
$ sudo mysql
mysql> update mysql.user set plugin='mysql_native_password' where user='root';
mysql> update mysql.user set authentication_string=PASSWORD('new_password') where user='root';
mysql> flush privileges;
mysql> quit
$ sudo service mysql restart
$ mysql -u root -p
```

MySQL 실행

- **mysql [-u/-p/-h/-P/-S]** *대소문자에 주의
  - -u : user id(-u [user id])
  - -p : user password(-p[password])

    ex) mysql –u root –p → 엔터 → password 입력
  - -h[host], -P[port], -S[socket address]을 사용하면 원격접속도 가능하다.
  - 언급을 하지 않으면 localhost로 인식한다.



(My)SQL의 기본적인 쿼리문

DB 보기	**SHOW** DATABASES;
DB 생성	**CREATE** DATABASE [DB name];
DB 사용	**USE** [DB name];
Selecting Data	**Select** "column1" from "tablename" [where "condition"];
Inserting Data	**Insert** into "tablename" (first_column,...last_column) values (first_value,...last_value);
Updating Data	**UPDATE** [table name] SET [field name] = [new value] WHERE ...
Deleting Data	**DELETE** FROM [table name] WHERE ...

(My)SQL의 기본적인 쿼리문

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| new_db             |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
mysql> CREATE DATABASE wheel;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| new_db             |
| performance_schema |
| sys                |
| wheel              |
+--------------------+
6 rows in set (0.00 sec)

mysql> USE wheel;
Database changed
```

Creating Table

- 반드시 use [DB name]; 입력 후 사용
- 단순보기: show tables;
- 생성:
  - Create table [table name] ([column1] [datatype] [constraint], [cloumn2] [datatype] [constraint],)
  - Varchar의 경우 괄호안에 크기 지정

- 예시:

```
create table employee(first varchar(15),
last varchar(20), age int(3),
address varchar(30));
```

Here are the most common Data types:

`char(size)`	Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
`varchar(size)`	Variable-length character string. Max size is specified in parenthesis.
`number(size)`	Number value with a max number of column digits specified in parenthesis.
`date`	Date value
`number(size,d)`	Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.

Creating Table

```
mysql> show tables;
Empty set (0.00 sec)

mysql> create table users (name varchar(10), num int(3), uid varchar(20));
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+-----------------+
| Tables_in_wheel |
+-----------------+
| users           |
+-----------------+
1 row in set (0.00 sec)

mysql> desc users;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| name  | varchar(10) | YES  |     | NULL    |       |
| num   | int(3)     | YES  |     | NULL    |       |
| uid   | varchar(20) | YES  |     | NULL    |       |
+-------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Inserting Data

INSERT INTO [tablename] ([first_column],...,[last_column])
VALUES ([first_column],...,[last_value]);

```
mysql> insert into users (name, num, uid) values ("Jiyeon", 18, "panya");
Query OK, 1 row affected (0.01 sec)

mysql> insert into users (name, num, uid) values ("Seunghyuk", 19, "hyuk");
Query OK, 1 row affected (0.00 sec)

mysql> insert into users (name, num, uid) values ("Yohan", 19, "nenw");
Query OK, 1 row affected (0.00 sec)
```

Selecting Data

- select "column1“ from "tablename" {where "condition“};
  {} = optional
- Where:

=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal to
LIKE	*See note below

- Like: like 뒤에 쓴 row만 가져온다

Selecting Data

```
mysql> select * from users;
+-----------+------+-------+
| name      | num  | uid   |
+-----------+------+-------+
| Jiyeon    |   18 | panya |
| Seunghyuk |   19 | hyuk  |
| Yohan     |   19 | nenw  |
+-----------+------+-------+
3 rows in set (0.00 sec)

mysql> select name, num from users;
+-----------+------+
| name      | num  |
+-----------+------+
| Jiyeon    |   18 |
| Seunghyuk |   19 |
| Yohan     |   19 |
+-----------+------+
3 rows in set (0.00 sec)
```

```
mysql> select name from users where name="Jiyeon";
+--------+
| name   |
+--------+
| Jiyeon |
+--------+
1 row in set (0.01 sec)

mysql> select name, uid from users where name LIKE "J%";
+--------+-------+
| name   | uid   |
+--------+-------+
| Jiyeon | panya |
+--------+-------+
1 row in set (0.00 sec)
```

Updating Data

- **Update**

  UPDATE [table name]

  SET [field name] = [new value]

  WHERE …

```
mysql> select * from users;
+-----------+------+-------+
| name      | num  | uid   |
+-----------+------+-------+
| Jiyeon    |   18 | panya |
| Seunghyuk |   19 | hyuk  |
| Yohan     |   19 | nenw  |
+-----------+------+-------+
3 rows in set (0.00 sec)
```

```
mysql> update users set num=21 where num=18;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from users;
+-----------+------+-------+
| name      | num  | uid   |
+-----------+------+-------+
| Jiyeon    |   21 | panya |
| Seunghyuk |   19 | hyuk  |
| Yohan     |   19 | nenw  |
+-----------+------+-------+
3 rows in set (0.00 sec)
```

Deleting Data

- **Delete**

  DELETE FROM [table name]

  WHERE …

```
mysql> select * from users;
+-----------+------+-------+
| name      | num  | uid   |
+-----------+------+-------+
| Jiyeon    |   18 | panya |
| Seunghyuk |   19 | hyuk  |
| Yohan     |   19 | nenw  |
+-----------+------+-------+
3 rows in set (0.00 sec)
```

```
mysql> delete from users where name="Jiyeon";
Query OK, 1 row affected (0.01 sec)

mysql> select * from users;
+-----------+------+------+
| name      | num  | uid  |
+-----------+------+------+
| Seunghyuk |   19 | hyuk |
| Yohan     |   19 | nenw |
+-----------+------+------+
2 rows in set (0.00 sec)
```

참고

- 문자는 ‘ 또는 “로 감싸져 있어야 한다.
  - Ex) ‘hello world’
- 숫자는 감쌀 필요가 없다. 그냥 쓴다.
  - Ex) 0x41, 1234, …
- AND, OR을 자유롭게 사용 가능하다.
- #은 주석을 의미한다.
- If, sleep와 같은 작동이 가능하다

실습

- MySQL에서 'wheel' 데이터베이스를 만든 다음 그 DB를 선택한다.
- 데이터베이스에서 아래와 같은 테이블('presenters')을 만든다.

Name	Num	UID	gender	age
Seunghyuk	19	hyuk	M	21
Jiyeon	18	panya	F	21
Yohan	19	nenw	M	20

- 자신의 정보를 테이블에 추가하세요.
- Seunghyuk회원의 나이가 잘못되었습니다. 21에서 20으로 바꿔주세요.
- 성별이 'F'인 사람을 테이블에서 제거하세요.
- 전체 테이블을 확인하고 캡처해서 slack에 올려주세요!

SQL Joins

- 두 개 이상의 테이블에 대해서 결합하여 나타낼 때 사용
- 즉, 두 개 이상의 테이블 사이에서 관계를 찾아야 할 때 사용

Orders

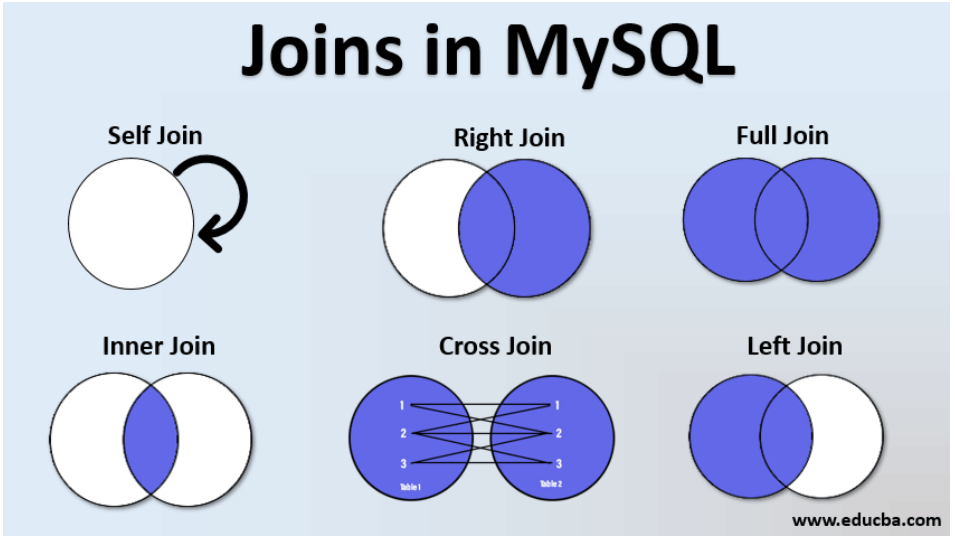
OrderID	CustomerID	Order Date
10308	2	2019-06-27
10309	33	2019-04-05
10310	13	2018-12-25

Customers

CustomerID	First	Last	Country
1	James	Bond	United States
2	Juan	Antonio	Mexico
3	JiSung	Park	South Korea

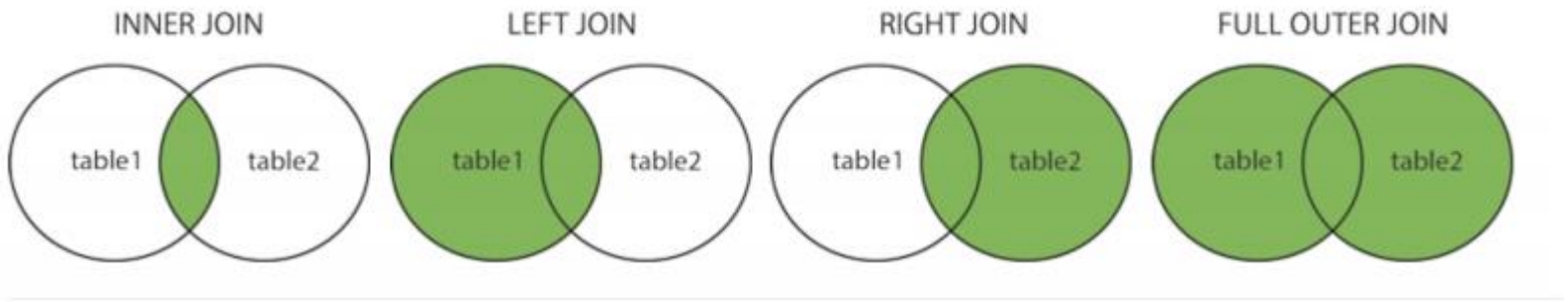
```
SELECT Orders.OrderID,
Customers.First,
Customers.Last
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID=
Customers.CustomerID;
```

OrderID	First	Last
10308	Juan	Antonio



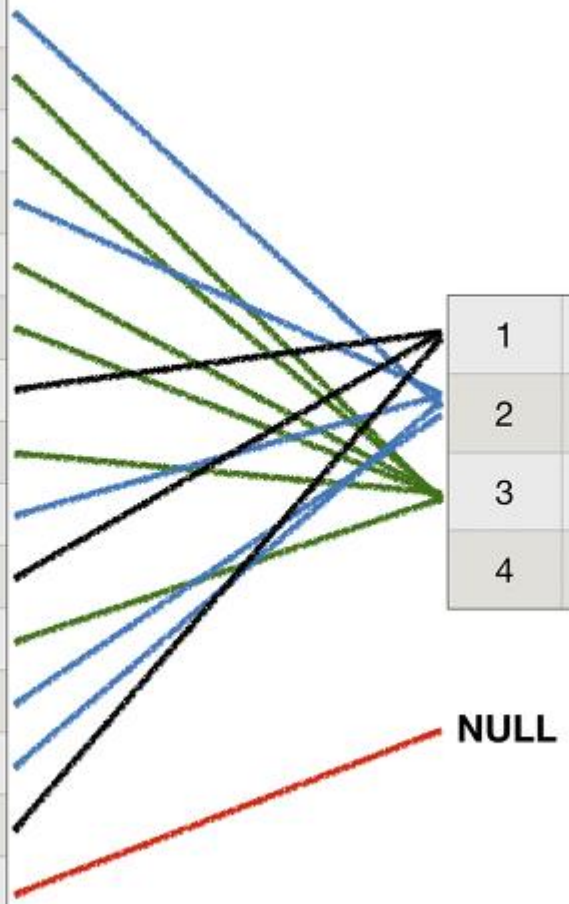
SQL Joins

- **(INNER) JOIN**
  - Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**
  - Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**
  - Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**
  - Returns all records when there is a match in either left or right table



SQL Joins

1	최범우	20
2	김범수	30
3	장태훈	30
4	최상우	20
5	변봉중	30
6	송원철	30
7	문서연	10
8	장웅	30
9	최상범	20
10	이명근	10
11	서은혜	30
12	이철진	20
13	한태범	20
14	김광우	10
15	문정진	NULL



1	10	경리부	서울
2	20	인사부	인천
3	30	영업부	대전
4	40	전산부	부천

NULL

Inner Join

	empName	deptName
1	최범우	인사부
2	김범수	영업부
3	장태훈	영업부
4	최상우	인사부
5	변봉중	영업부
6	송원철	영업부
7	문서연	경리부
8	장웅	영업부
9	최상범	인사부
10	이명근	경리부
11	서은혜	영업부
12	이철진	인사부
13	한태범	인사부
14	김광우	경리부

Full (Outer) Join

	empName	deptName
1	최범우	인사부
2	김범수	영업부
3	장태훈	영업부
4	최상우	인사부
5	변봉중	영업부
6	송원철	영업부
7	문서연	경리부
8	장웅	영업부
9	최상범	인사부
10	이명근	경리부
11	서은혜	영업부
12	이철진	인사부
13	한태범	인사부
14	김광우	경리부
15	문정진	NULL
16	NULL	전산부