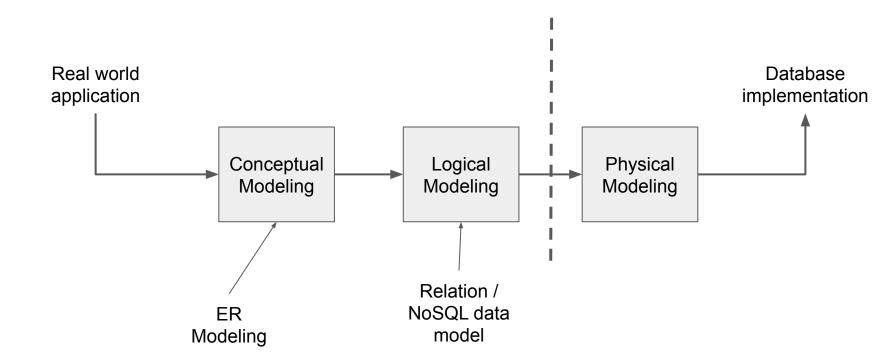
Database and Big Data Systems

Lab 3

Today

- Recap
- SQL

Recap



- Declarative language
- Data definition
 - o Create database, tables
- Data manipulation
 - o Insert, update, delete
- Query
 - o Select ... From ... Where

SQL - Data Definition

Create database and tables

Payroll (UserID, Name, Job, Salary)

```
create table Payroll (
    UserID integer,
    Name varchar(100),
    Job varchar(100),
    Salary integer
);
```

create database if not exists university;

Case insentive (except for Table name).

But *please* don't capitalize everything.

SQL - Data Manipulation

Add new data into table

| UserID | Name | Job | Salary |
|--------|---------|------|--------|
| 123 | Jack | TA | 50000 |
| 345 | Allison | TA | 60000 |
| 567 | Magda | Prof | 90000 |
| 789 | Dan | Prof | 100000 |
| 001 | Anh | Prof | 10000 |
| 002 | Cyrille | TA | 10000 |

```
insert into Payroll values (001, "Anh", "Prof", 10000);
insert into Payroll values (002, "Cyrille", "TA", 10000);
```

SQL - Query



- 1. Take product of input relation R1, R2,...
- 2. Apply selection condition
- 3. Take specific column Col1, Col2, ...

Today

- Recap
- SQL

- MySQL doesn't support set subtraction
- Write a query that returns R(<u>a</u>) S(<u>a</u>)
 - Table R has one attribute a (primary key)
 - Table S has one attribute a (primary key)

- MySQL doesn't support set subtraction
- Write a query that returns R(<u>a</u>) S(<u>a</u>)
 - Table R has one attribute a (primary key)
 - Table S has one attribute a (primary key)

Select a from R Where a **not in** (select a from S);

Select a from R Where not exist (select a from S where R.a = S.a);

select R.a from R left join S on R.a = S.a where S.a is null;

Launch an instance from the following image (US-East1)

ami-09f7eb2d6268ba83a

 If you are not using AWS, a local Linux/MacOS machine with MySQL installed works too.

Data schema

 Exercises: create the tables

| Trip | [table] |
|--------------------|---------------------|
| id, integer not no | all |
| duration, integ | er |
| start_time, tim | estamp |
| start_station_ | name, text |
| start_station_ | id, smallint |
| end_time, time | estamp |
| end_station_n | name, text |
| end_station_i | d , smallint |
| bike_id, smallir | nt |
| PRIMARY_KE | Y(id) |

Station [table] station_id, smallint not null station_name, text latitude, real longitude, real dock_count, smallint city, text installation_date, date zip_code,text PRIMARY_KEY(station_id)

| Weather | [table] |
|-------------------------|------------------|
| date, date not null | |
| max_temp, real | |
| mean_temp, real | |
| min_temp, real | |
| max_visibility_miles | , real |
| mean_visibility_mile | es , real |
| min_visibility_miles | , real |
| max_wind_speed_m | ph , real |
| mean_wind_speed_1 | mph, real |
| max_gust_speed_mp | ph , real |
| cloud_cover, real | |
| events, text | |
| wind_dir_degrees, re | eal |
| zip_code, text not null | |

PRIMARY KEY(date, zip code)

- Insert new data in
- Check for constraints
 - Primary key constraint
 - Foreign key constraint
- Delete tables and database

| Trip | [table] |
|----------------------|----------|
| id, integer not null | |
| duration, integer | |
| start_time, timesta | mp |
| start_station_nan | ne, text |
| start_station_id, s | mallint |
| end_time, timestan | ıp |
| end_station_name | e, text |
| end_station_id, sn | nallint |
| bike_id, smallint | |
| PRIMARY_KEY(id | l) |

| Station | [table] |
|------------------------|------------------|
| station_id, sn | nallint not null |
| station_name | e, text |
| latitude, real | |
| longitude , rea | 1 |
| dock_count, | smallint |
| city, text | |
| installation_o | date, date |
| zip_code,text | |
| PRIMARY_K | EY(station_id) |

| Weather | [table] |
|------------------------|-------------------|
| date, date not null | |
| max_temp, real | |
| mean_temp, real | |
| min_temp, real | |
| max_visibility_mile | es, real |
| mean_visibility_mi | les, real |
| min_visibility_mile | s, real |
| max_wind_speed_r | nph , real |
| mean_wind_speed_ | _mph, real |
| max_gust_speed_m | nph, real |
| cloud_cover, real | |
| events, text | |
| wind_dir_degrees, | real |
| zip_code, text not nul | 1 |
| PRIMARY_KEY(dat | te, zip_code) |

- Load data in with /home/ubuntu/setup.sql
- If you are using a local machine, you may download the setup.sql files (v5.7 and v8) here.

- [Q1] Count the number of cities (no duplicates)
- [Q2] Count the number of stations in each city. Output the city name, and station count.
- [Q3] Count the number of self-loop trips. A self-loop trip is one that starts and ends at the same station.

- How to create a temporary table from some SQL query
 - o create temporary table <name> <SQL query>
 - Useful for replacing nested queries
- [Q4] Print the ratio of self-loop trips over all trips.
- [Q5] Find the most popular city, in terms of percentage of trips belonging to the city.

- [Q6] Find all the bikes (their bike_id) that have been to more than 1 city. A
 bike has been to a city if its start or end station of one of its trips is in this city.
- [Q7] List the bikes (their bike_id) that have never been to Japantown station.