ANALYZING BIG DATA - I

Week 3 – Basic Searches

Agenda

- Homework 2: Brief revisit just to make sure we are all in the same plain field
- Interacting with relational DB's:
  - Weeks 1 and 2: **Input** data:
    - We use the database to store data
      - creating tables, creating
  - Week 3 : **Output** data:
    - Commands: Select, Having,
    - Handling Null Values
    - String manipulation

Referential Integrity

- Suppose there is a link between CAR and DRIVER via OWNER.
- If there is a value in OWNER, then this value must also appear somewhere in DRIVER.
- If you change a driver's name in DRIVER, you must make sure the same change is made in OWNER of CAR.
- The DBMS enforces the rules.
- If you try to break the rules the DBMS reports the problem as a **REFERENTIAL INTEGRITY** error.

Supporting our study of data structures

- Numerous books – *basics* have not changed in quite some time
- Our approach: http://db.grussell.org/index.html
- Our focus: *Relational database model*

RETRIEVING DATA

MySQL Comments

- To give you the ability to make notes in queries you are allowed to have comments.
- Comments are not executed
- A comment starts with -- and ends with a newline
- They are only permitted within a query.

```
SELECT regno -- The registration number
FROM car -- The car storage table;
```

CONNECTING AND DESCRIBE

```
USE db_countries;


SHOW TABLES;


DESCRIBE continents;
DESCRIBE cities;
DESCRIBE countries;
DESCRIBE attractions;
```

SELECT: ALL COLUMNS

```
SELECT * FROM continents;
SELECT * FROM cities;
SELECT * FROM countries;
```

SELECT: RESTRICTING COLUMNS

```
SELECT SURFACE FROM countries;
```

```
SELECT countries.SURFACE  FROM countries;
```

SELECT : AS

```
SELECT SURFACE AS `SIZE IN MILES` FROM continents;
SELECT * FROM cities;
SELECT * FROM countries;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT surface, population
FROM   countries
WHERE  ID_COUNTRY=1;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT surface, population
FROM   countries
WHERE  name='Spain';
```

```
SELECT surface, population
FROM   countries
WHERE  name=‘SPAIN';
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM   countries
WHERE  population > 20;
```

```
SELECT name,    population
FROM   countries
WHERE  0.0001*population > 20;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM countries
WHERE population BETWEEN 0 AND 200000;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM countries
WHERE population BETWEEN 0 AND 200000
ORDER BY population;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM countries
WHERE population BETWEEN 0 AND 200000
ORDER BY population DESC;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM countries
WHERE population BETWEEN 0 AND 200000
ORDER BY population DESC LIMIT 10;
```

SELECT: RESTRICTING ROWS (WHERE)

```
SELECT name,    population
FROM countries
WHERE population BETWEEN 0 AND 200000
ORDER BY population DESC LIMIT 3
OFFSET 3;
```

SELECT with AGGREGATORS

```
SELECT continent, MAX(surface)
FROM countries;
```

SELECT with AGGREGATORS

```
SELECT continent, SUM(surface)
FROM countries
GROUP BY continent;
```

SELECT with AGGREGATORS

```
SELECT continent, MIN(surface)
FROM countries
GROUP BY continent;
```

SELECT with AGGREGATORS

```
SELECT continent, AVG(surface)
FROM countries
GROUP BY continent;
```

SELECT with AGGREGATORS

```
SELECT continent, VARIANCE(surface)
FROM countries
GROUP BY continent;
```

SELECT with AGGREGATORS

```
SELECT continent, MIN(surface)
FROM countries
GROUP BY continent;
```

SELECT with AGGREGATORS

```
SELECT continent, ROUND(1.86*surface) AS SURFACE_IN_K
FROM countries
GROUP BY continent;
```

SELECT with String Manipulations

```
SELECT *
FROM attractions
WHERE name LIKE '%square%';

SELECT * FROM attractions WHERE name LIKE 'A%';
SELECT * FROM attractions WHERE name LIKE '_%';
SELECT * FROM attractions WHERE name LIKE '______%';
SELECT * FROM attractions WHERE name LIKE '______';
SELECT * FROM countries_of_the_world WHERE Country REGEXP 'mar';
SELECT * FROM countries_of_the_world WHERE Country REGEXP '^Den';
SELECT * FROM countries_of_the_world WHERE Country REGEXP '^[AT]';
```

SELECT with String Manipulations

```
SELECT *
FROM attractions
WHERE name LIKE 'A%';
```

SELECT with String Manipulations

```
SELECT *
FROM attractions
WHERE name LIKE '_%';
```

SELECT with String Manipulations

```
SELECT *
FROM attractions
WHERE name LIKE '_____%';

SELECT *
FROM attractions
WHERE name LIKE '_____';
```

SELECT with String Manipulations

```
SELECT *
FROM countries_of_the_world
WHERE Country REGEXP 'mar';

SELECT *
FROM countries_of_the_world
WHERE Country REGEXP '^Den';

SELECT *
FROM countries_of_the_world
WHERE Country REGEXP '^[AT]';
```

SELECT with String Manipulations

```
SELECT *
FROM countries_of_the_world
WHERE Country REGEXP 'mar';

SELECT *
FROM countries_of_the_world
WHERE Country REGEXP '^Den';

SELECT *
FROM countries_of_the_world
WHERE Country REGEXP '^[AT]';
```

SELECT WITH NULL VALUES

SELECT regno from car
WHERE OWNER is null

REGNO
SC04 BFE

SELECT regno from car
WHERE OWNER is not null

REGNO
F611 AAA
J111 BBB
A155 BDE
K555 GHT
SC04 BFE

SQL Basic Commands

- Basic SQL statements include
  - CREATE – a data structure
  - SELECT – read one or more rows from a table
  - INSERT – one of more rows into a table
  - DELETE – one or more rows from a table
  - UPDATE – change the column values in a row
  - DROP – a data structure

NULL

- NULL indicates that something has no value
- It is not a value, and you cannot use normal comparison operators.
- For instance, looking for cars without owners…

Wrong: SELECT regno from car where owner = NULL
Wrong: SELECT regno from car where owner = 'NULL'

- Instead there are two special operators, IS NULL, and IS NOT NULL

G. Russell, Database e-Learning

LIKE

- Sometimes you want to have a rule involving partial strings, substrings, or wildcards
- LIKE does this, and is a slot-in replacement for '='
- If the string contains '%' or '_', LIKE uses them to support wildcards.
  - % - Matches 0 or more characters in the string
  - _ - Matches exactly 1 character in the string

More Simple Examples

• Name LIKE 'Jim Smith'	e.g. Jim Smith
• Name LIKE '_im Smith'	e.g. Tim Smith
• Name LIKE '___ Smith'	e.g. Bob Smith
• Name LIKE '% Smith'	e.g. Frank Smith
• Name LIKE '% S%'	e.g. Brian Smart
• Name LIKE 'Bob %'	e.g. Bob Martin
• Name LIKE '%'	i.e. match anyone

- LIKE is more expensive than =
- If you are not using wildcards, always use = rather than LIKE.