Python Notes

Steven Wang

March 28, 2016

```
-- Author: Steven Wang  Date: 20150118

# single line comment in mysql
-- single line comment in standard SQL and newer versions of mysql
*** multiline comments ###
Database is based on tales.
Tables(relations) are made of columns and rows.
Columns(attributes) are like categories.
Each row is like a entry.
All tables should have primary keys.(you can assign a column as primary
   key)
Primary key is a column that is unique.

shell> mysql --host=localhost --user=myname --password=mypass mydb
shell> mysql -h localhost -u myname -p mypass mydb

quit
show databases;
show tables;

-- Normal SQL text commands use uppercase.
-- SQL statements (SQL is not case sensitive; semicolon at the end of
   each SQL statement)

USE database_name;
SELECT DATABASE(); -- show currently selected database

DROP DATABASE database_name; -- delete database database_name
DROP DATABASE IF EXISTS database_name;

CREATE TABLE student (
first_name VARCHAR(30) NOT NULL,    -- "first_name" is the name of column
   ; "VARCHAR" is its data type; "NOT NULL" makes sure the data is
   entered(be in the database)
```

```
last_name VARCHAR(30) NOT NULL,
email VARCHAR(60) NULL, -- not necessarily be filled in
state CHAR(2) NOT NULL DEFAULT "MA", -- by default the value is "MA"
zip MEDIUMINT UNSIGNED NOT NULL,
birth_date DATE NOT NULL,
sex ENUM('M', 'F') NOT NULL,
date_entered TIMESTAMP, -- gives data and time
lunch_cost FLOAT NULL,
student_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);   -- with "
   AUTO_INCREMENT", no need to input; set "student_id" as the PRIMARY KEY

*** 1. Primary keys is unique.
    2. It cannot be NULL.
    3. Original value cannot be changed.
    4. ? It is probably best to auto-increment the value of the key.

    Tables:
    1. Every table should just focus on one thing.
    2. Then decide the things you need to describe this one thing.
    3. If one description needs multiple inputs, pull it out.(So, we call
    them atomic tables.)
    4. Don't have multiple columns with same sort of information.(eg,
   travel_history_1, travel_histroy_2)
    5. Don't include multiple values in one cell.(eg. jobs: Goldman,
   Blackstone)
    6. Normalized tables.(Database normalization is the process of
   organizing the columns and tables of a relational database to minimize
    data redundancy.)
###

-- Datatypes:
-- numeric types:
TINYINT:    127 ~ -128
SMALLINT:   32768 ~ -36767
MEDIUMINT:  8388608 ~ -8388608
INT:        2^31 ~ (-2^31-1)
BIGINT:     2^31 ~ (-2^63-1)

FLOAT:      Decimal spaces: 1.1E38 ~ -1.1E38
DOUBLE:     Decimal spaces: 1.7E308 ~ -1.7E308

--string types:
CHAR:       fixed length character string
VARCHAR:    A character string with a variable length
BLOB:       Binary Large Object (BLOB): can contain 2^16 bytes of data
```

```
70
71 ENUM:        limited number of total values.(eg. "male" or "femail")
72 SET:         list of values
73
74 --time types:
75 DATE:        YYYY-MM-DD
76 TIME:        HH:MM:SS
77 DATETIME:    YYYY-MM-DD HH:MM:SS
78 TIMESTAMP:   YYYYMMDDHHMMSS
79 YEAR:        YYYY
80
81 DESCRIBE table_name
82
83 INSERT INTO student VALUE
84 ('Steven', 'Wang', '1@gmail.com', 'MA',02134,19910101,'M',NOW(),3.5,NULL)
      ; --"NOW()" returns present time
85
86 SELECT * FROM student;
87
88 CREATE TABLE class (
89 name VARCHAR(30) NOT NULL,
90 class_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);
91
92 INSERT INTO class VALUES ('English', NULL), ('Chinese', NULL), ('Physics'
      , NULL);
93
94 CREATE TABLE test(
95 date DATE NOT NULL,
96 type ENUM('T', 'Q') NOT NULL,
97 class_id INT UNSIGNED NOT NULL, -- "class_id" is the foreign key
98 test_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);
99
100 *** Foreign Key:
101     1. Used to make references to the Primary Key of another table.
102     2. The foreign Key can have a different name from the Primary Key
       name.
103     3. The value of a Foreign Key can have the value of NULL.
104     4. A Foreign Key does not have to be unique.
105 ###
106
107 CREATE TABLE score(
108 student_id INT UNSIGNED NOT NULL,
109 event_id INT UNSIGNED NOT NULL,
110 score INT NOT NULL,
111 PRIMARY KEY(event_id, student_id)); -- combine the two id to make sure
```

```
   score is unique

CREATE TABLE absence(
student_id INT UNSIGNED NOT NULL,
date DATE NOT NULL,
PRIMARY KEY (student_id, date));

ALTER TABLE test
ADD maxscore INT UNSIGNED NOT NULL AFTER type;
-- To add a new column "maxscore" in the "test" table; "AFTER": add it
   after "type" column

INSERT INTO test VALUES
('2014-8-25','Q',15,1,NULL),
('2014-8-27','Q',15,1,NULL),
('2014-8-29','T',30,1,NULL),
('2014-8-29','T',30,2,NULL),
('2014-8-27','Q',15,4,NULL),
('2014-8-29','T',30,4,NULL);

ALTER TABLE score CHANGE event_id test_id -- change the name of "event_id
   " to "test_id"
INT UNSIGNED NOT NULL; -- define the data type for "test_id"

SELECT FIRST_NAME, last_name FROM student;

RENAME TABLE
absence TO absences,
class TO classes,
score TO scores,
student TO students,
test TO tests;

SELECT first_name, last_name, state
FROM students
WHERE state="MA"; -- WHERE: limit the data

SELECT first_name, last_name, birth_date
FROM students
WHERE YEAR(birth_date)>=1965;

-- compare: =, >, <, >=, <=, !=

SELECT first_name, last_name, birth_date
FROM students
```

```
WHERE MONTH(birth_date) = 2 OR state='MA';

-- AND && OR || NOT !

SELECT first_name, last_name, birth_date
FROM students
WHERE DAY(birth_date) >= 12 && (state="CA" || state="NV");

-- NULL: To check the value is NULL or not, we cannot use "=" or ">", can
    only use IS NULL or IS NOT NULL

SELECT last_name
FROM students
WHERE last_name IS NOT NULL;

-- order:
SELECT first_name, last_name
From students
ORDER BY last_name;

-- reverse order:
ORDER BY col_name DESC;
ORDER BY state DESC, last_name ASC;

SELECT first_name, last_name
From students
LIMIT 5; -- Get the first 5 results; Limit the data you are going to get.

SELECT first_name, last_name
From students
LIMIT 5, 10; -- Get the next 5 results.

SELECT CONCAT(first_name, " ", last_name) AS 'Name',
CONCAT(city, " ", state) AS 'Hometown'
FROM students;

-- LIKE
SELECT first_name, last_name
From students
WHERE first_name LIKE 'D%' OR last_name LIKE '%n'; -- "%" stands for any
   sequence of characters

SELECT first_name, last_name
From students
WHERE first_name LIKE '____y'; -- "_" stands for any single character
```

```
197
198 -- DISTINCT
199 SELECT DISTINCT state
200 FROM students
201 ORDER BY state;
202
203 -- COUNT
204 SELECT COUNT(DISTINCT state)
205 FROM students;
206 -- count the number
207
208 SELECT COUNT(*)
209 FROM students;
210
211 SELECT COUNT(*)
212 FROM students
213 WHERE sex="M";
214
215 SELECT sex, COUNT(*)
216 FROM students
217 GROUP BY sex; -- count by group
218
219 SELECT MONTH(birth_date) AS 'Month', COUNT(*)
220 FROM students
221 GROUP BY Month
222 ORDER BY Month;
223
224 SELECT state, COUNT(state) AS 'Amount'
225 FROM students
226 GROUP BY state
227 HAVING Amount > 1; -- show the results of month that have more than 1
      count
228
229 -- math
230 SELECT test_id AS 'Test',
231 MIN(score) AS min,
232 MAX(score) AS max,
233 MAX(score) - MIN(score) AS 'range',
234 SUM(score) AS total,
235 AVG(score) AS average -- !! NO COMMA
236 FROM scores
237 GROUP BY test_id;
238
239 ***
240 ABS()
```

```
ACOS(), ASIN(), ATAN(), ATAN2()  COS(), SIN(), TAN()
AVG()
CEILING()
COUNT()
DEGREES()
EXP()
FLOOR()
LOG()
MAX()
MIN()
MOD()
PI()
POWER()
RADIANS()
RAND()
ROUND()
SQRT()
STD()
SUM()
TRUNCATE
###

DELETE FROM absences
WHERE student_id=6;

ALTER TABLE absences
ADD COLUMN test_taken CHAR(1) NOT NULL DEFAULT 'F'
AFTER student_id;

ALTER TABLE absences
MODIFY COLUMN test_taken ENUM('T','F') NOT NULL DEFAULT 'F';

ALTER TABLE absences
DROP COLUMN test_taken;


UPDATE scores SET score=25  -- change value of attribute
WHERE student_id=4 AND test_id=3;

SELECT first_name
FROM students
WHERE birth_date
BETWEEN '1960-1-1' AND '1970-1-1'; -- between a minimum and maximum

SELECT first_name
```

```
FROM students
WHERE first_name IN ('Alice', 'Bob', 'Steven'); -- predefined list of possible options

-- Join 2 tables
SELECT student_id, date, score, maxscore -- the same as: SELECT scores.student_id, tests.date, scores.score, test.maxscore)
FROM tests,scores
WHERE date='2014-8-25'
AND tests.test_id=scores.test_id; -- The same column in both tables

--Join 3 tables
SELECT CONCAT(students.first_name, " ", students.last_name) AS Name,
tests.date, scores.score, tests.maxscore
FROM tests, scores, students
WHERE date='2014-08-25'
AND tests.test_id=scores.test_id
AND scores.student_id=students.student_id;

SELECT students.student_id,
CONCAT(students.first_name, " ",students.last_name) AS Name,
COUNT(absences.date) AS Absences    --no space between "COUNT" and "("
FROM students, absences
WHERE students.student_id=absences.student_id
GROUP BY students.student_id;

-- LEFT JOIN: show everything of the table on the left even the right one has no corresponding value
SELECT students.student_id,
COUNT(absences.date) AS Absences
FROM students LEFT JOIN absences
ON students.student_id=absences.student_id  -- The two things that both tables have in common
GROUP BY students.student_id;

-- INNER JOIN: Get all the data from both tables and join them together.
SELECT first_name
FROM students
INNER JOIN scores
ON students.student_id=scores.student_id
WHERE scores.score<=15
GROUP BY scores.test_id;
EOF
```

SQL