MySQL

Sunday, February 23, 2020 4:53 PM

Commands

USE	USE <database>	switch to a database
SELECT	SELECT DATABASE() * = show all	show the currently selected database
DROP	DROP DATABASE IF EXISTS <database> DROP COLUMN <column>	delete database Delete column
CREATE TABLE	CREATE TABLE. <table name>(…)	create table <table name>
VARCHAR	VARCHAR(<#>)	table entry has a variable number of characters with an estimate of # characters
CHAR	CHAR(<#>)	table entry has *exactly* # characters
DEFAULT	DEFAULT "<default value>"	set the default value in a table
MEDIUMINT		specifies medium sized integer in a table
DATE		specifies date in table
ENUM	ENUM('<entry>', '<entry>',…)	specifies an enumerated list to choose from for a table
TIMESTAMP		specifies a point in time in a table
FLOAT		specifies a float in a table
AUTO_INCREMENT		specifies that the number should be incremented with each new entry
SHOW TABLES		lists tables available
DESCRIBE	DESCRIBE <table>	shows a description of the table specified
INSERT	INSERT INTO <table> VALUE	add entry into table
ALTER	ALTER TABLE <table>	specifies that you want to alter a table

RENAME	RENAME TABLE	renames table names within a database
CONCAT	CONCAT(<list of strings>)	concatenates strings
BETWEEN		Find values between min and max
IN		Look within a pre-defined list of options

Types

Numeric types

- TINYINT [-128,127]
- SMALLINT [-32767, 32768]
- MEDIUMINT [-8388608, 8388608]
- INT $[2^{31} - 1, 2^{31}]$
- BIGINT $[-2^{63} - 1, 2^{63}]$
- FLOAT - decimal spaces, $[-1.1E38, 1.1E38]$
- DOUBLE - decimal spaces, $[-1.7E308, 1.7E308]$

String types

- CHAR - a character string with a fixed width
- VARCHAR - a character string with a length that is variable
- BLOB - can contain 2^{16} bytes of data
- ENUM - a character string that has a limited number of total values, which you must define
- SET - a list of legal possible character strings
    - Can contain multiple values, unlike enum
    - Should *not* use

Date types

- DATE: YYYY-MM-DD
- TIME: HH:MM:SS
- DATETIME: YYYY-MM-DD HH:MM:SS
- TIMESTAMP: YYYYMMDDHHMMSS
- YEAR: YYYY

Joins

- CROSS JOIN - matches each row from one database table to all rows of another
- INNER JOIN - return rows from both tables that satisfy the given condition
- OUTER JOIN - return all records matching from both tables

- LEFT JOIN - returns all the rows from the table on the left even if no matching rows have been found in the table on the right
- RIGHT JOIN - returns all the columns from the table on the right even if no matching rows have been found in the table on the left

Definitions

Primary key - unique ID for each entry in the table

- uniquely identify a row or record
- must be given a value when the row is created that cannot be NULL
- the original value cannot change
- It is *probably* best to auto-increment the key

Foreign key

- used to make references to primary keys from another table
- can have a *different* name from the primary key name
- *can* have the value of null
- does *not* have to be unique

Atomic Table & Table Templating

- Every table should focus on describing *only* one thing
- Decide what things you need to describe that one thing
- Write out all ways to describe the thing and if any of those things requires multiple inputs, pull them out
- Do not have multiple columns with the same sort of information
- do not include multiple values in one cell
- normalized tables

Steps

list all aspects needed for database
split up into database
i.e. test scores for students

- students database holding scores and ID info
- tests database for holding test information

Example

creating a table

CREATE TABLE student(
    first_name VARCHAR(30) NOT NULL

```
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    email VARCHAR(60) NOT NULL,
    city VARCHAR(40) NOT NULL,
    state CHAR(2) NOT NULL DEFAULT "NY",
    zip MEDIUMINT UNSIGNED NOT NULL,
    phone VARCHAR(20) NOT NULL,
    birth_date DATE NOT NULL,
    sex ENUM('M', 'F') NOT NULL,
    date_entered TIMESTAMP,
    lunch_cost FLOAT NULL,
    student_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);
```

inserting into a table

```
INSERT INTO student VALUE
    ('Dale', 'Cooper', 'dcooper@aol.com',
    '123 Main St.,
    'Yakima',
    'WA'
    98901,
    '792-223-8901',
    "1959-2-22",
    "M",
    NOW(),
    3.50,
    NULL);
```

add a row to a table

```
ALTER TABLE test
    ADD maxscore INT NOT NULL AFTER type
```

get a subset of entries

```
SELECT first_name, last_name, state
    FROM students
    WHERE state="WA";

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

SELECT first_name, last_name, birth_date
```

```
SELECT first_name, last_name, birth_date
        from students
        WHERE MONTH(birth_date) = 2 OR state="CA"
```
lists all students born in february or from CA

```
SELECT first_name, last_name
        FROM students
        ORDER BY last_name;
```

```
SELECT first_name, last_name, state
        FROM students
        ORDER BY state DESC, last_name ASC;
```
** lists all students by state in decreasing order, then, in the event of a tie, by last name in ascending order**

```
SELECT first_name, last_name
        FROM students
        LIMIT 5;
```
** limits the number of students listed to 5**

```
SELECT CONCAT(first_name, " ", last_name) AS 'Name',
        CONCAT(city, ", ", state) AS "Hometown"
        FROM students;
```
** lists first and last name together and town and state together**

```
SELECT first_name, last_name
        FROM students
        WHERE first_name LIKE 'D%' OR last_name LIKE '%n';
```
** gets all people whose last name ends in "n" or whose first name starts with "D" **
** "%" == any series of characters **
** "_" == any *single* character **

```
SELECT DISTINCT state
        FROM students
        ORDER BY state;
```
** list all *distinct* states **

```
SELECT COUNT(DISTINCT state)
        FROM students
```
** show number of distinct states **

```
SELECT COUNT(*)
        FROM students
** number of entries in table **

SELECT sex, count(8)
        FROM students
        GROUP BY sex
** print count of males and females **

SELECT state, COUNT(state) AS 'Amount'
        FROM students
        GROUP BY state
        HAVING Amount > 1;
** Print number of students from states having more than one student from that
state **

DELETE FROM absences
        WHERE student_id = 6;
** deletes row(s) containing student_id 6 **

ALTER TABLE absences
        MODIFY COLUMN test_taken ENUM('T', 'F') NOT NULL DEFAULT 'F';
** Change the test_taken row in absences to an ENUM('T', 'F') with default of 'F' **

SELECT scores.student_id, tests.date, scores.score
        tests.maxscore
        FROM tests, scores
        WHERE date='2014-08-25
        AND tests.test_id = scores.test_id;
** Get info from joined tables **
```

Sources

https://www.youtube.com/watch?v=yPu6qV5byu4&feature=emb_logo
https://www.guru99.com/joins.html