SQL Data Definition Language

Overview

- **Data Types for Attributes/Fields**
- DDL SQL Statements
- Database Integrity
- DML SQL Statements - insert, update, delete data in tables

MySQL TEXT Types (I)

- **CHAR(size)**
  - fixed length string (letters, numbers, and special characters)
  - the fixed size is specified in parenthesis - up to 255 chars
- **VARCHAR(size)**
  - variable length string (letters, numbers, and special characters)
  - the maximum size is specified in parenthesis - up to 255 chars
  - if you put a greater value than 255 it will be converted to a TEXT type
- **TINYTEXT**
  - holds a string with a maximum length of 255 characters
- **TEXT**
  - holds a string with a maximum length of 65,535 characters
- **BLOB**
  - for BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
- **MEDIUMTEXT**
  - holds a string with a maximum length of 16,777,215 characters

MySQL TEXT Types (II)

CITY UNIVERSITY LONDON
EST 1894

- **MEDIUMBLOB**
  - for BLOBs (Binary Large OBjects) - up to 16,777,215 bytes
- **LONGTEXT**
  - holds a string with a max length of 4,294,967,295 chars
- **LONGBLOB**
  - for BLOBs (Binary Large OBjects) - up to 4,294,967,295 bytes
- **ENUM(x, y, z, etc.)**
  - lets you enter a list of possible values - up to 65535
  - enter these possible values with: ENUM('X', 'Y', 'Z')
- **SET**
  - similar to ENUM - SET may contain up to 64 list items

MySQL EXACT NUMBER Types

CITY UNIVERSITY LONDON

EST 1894

- **TINYINT(size)**
  - 1 byte: -128 to 127 signed or 0 to 255 UNSIGNED
  - maximum number of digits may be specified in parenthesis
  - **BOOL** and **BOOLEAN** (i.e., TRUE or FALSE) - synonyms for TINYINT(1).
- **SMALLINT(size)**
  - 2 bytes: -32768 to 32767 signed or 0 to 65535 UNSIGNED.
- **MEDIUMINT(size)**
  - 3 bytes: -8388608 to 8388607 signed or 0 to 16777215 UNSIGNED.
- **INT(size)**
  - 4 bytes
- **BIGINT(size)**
  - 8 bytes
- **DECIMAL(size, d)**
  - used to store values for which it is important to preserve exact precision, e.g., monetary data
  - size - max no. of digits; d – max no. of digits following decimal point
  - Fixed-point
  - Maximum number of digits - 65

MySQL APROX. NUMBER Types

CITY UNIVERSITY LONDON
EST 1894

- Floating point-numbers
- **FLOAT(size,d)**
  - a small number with a floating decimal point
  - 4 bytes
  - The max number of digits may be specified in the size parameter
  - The max number of digits to the right of the decimal point is specified in the d parameter
  - e.g., FLOAT(7, 4) – 999.9999
- **DOUBLE(size, d)** or **REAL**
  - a large number with a floating decimal point
  - 8 bytes

MySQL DATE & TIME Types

- **DATE**
  - format: YYYY-MM-DD
  - supported range is from '1000-01-01' to '9999-12-31'
- **TIME**
  - Format: HH:MM:SS
  - supported range is from '-838:59:59' to '838:59:59'
- **DATETIME**
  - date and time combination
  - format: YYYY-MM-DD HH:MM:SS.
  - supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
- **YEAR**
  - year in two-digit or four-digit format
  - values allowed in four-digit format: 1901 to 2155.
  - values allowed in two-digit format: 70 to 69 (from 1970 to 2069)
- Date and time functions: NOW(), CURRENT_DATE()

Using dates for your coursework

CITY UNIVERSITY LONDON
EST 1894

- I strongly advise you not to use ‘date’ data types for your coursework
- Instead define them as integer(6) and input the dates as a number in the form YYMMDD
- For example the 5th November 2020 would be 201105
- This will make it much simpler to program search queries for data in your database

Overview

CITY UNIVERSITY LONDON
EST 1894

- Data Types for Attributes/Fields
- **DDL SQL Statements**
- Database Integrity
- DML SQL Statements - insert, update, delete data in tables

Main DDL Statements

CITY UNIVERSITY LONDON
EST 1894

CREATE	ALTER	DROP
• SCHEMA	• DOMAIN	• SCHEMA
• DOMAIN	• **TABLE**	• DOMAIN
• **TABLE**		• **TABLE**
• VIEW		• VIEW

CREATE TABLE: a basic SQL syntax

CITY UNIVERSITY LONDON
EST 1894

```
CREATE TABLE TableName (
  {colName dataType [NOT NULL] [UNIQUE]
    [DEFAULT defaultOption]
    [CHECK searchCondition] [,...]}
  [PRIMARY KEY (listOfColumns),]
  {[UNIQUE (listOfColumns),] [...,]}
  {[FOREIGN KEY (listOfFKColumns)
    REFERENCES ParentTableName [(listOfFKColumns)]
    [ON UPDATE referentialAction]
    [ON DELETE referentialAction ]] [,...]}
  {[CHECK (searchCondition)] [,...] }
);
```

DDL SQL Statements

Lets see how to use these statements:

- CREATE
- ALTER
- DROP

Create a table

```sql
CREATE TABLE journey (
  ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
  DISTANCE INTEGER,
  JOURNEYDATE DATE,
  TICKET CHAR(1)
);
```

Inspect tables

```
SHOW tables;

DESC journey;
```

Alter tables

We can redefine a structure: changing field type, adding/deleting column (also DROP COLUMN)

We forgot time!

```
ALTER TABLE journey ADD time INTEGER;

DESC journey;

ALTER TABLE journey MODIFY COLUMN time TIME;

DESC journey;

ALTER TABLE journey MODIFY COLUMN time TIME NOT NULL;

DESC journey;
```

Drop tables

CITY UNIVERSITY LONDON
EST 1894

We can delete a table in the database

```
CREATE TABLE dummy (ID INTEGER);

DESC dummy;

DROP TABLE dummy;

DESC dummy;
```

Overview

CITY UNIVERSITY LONDON

EST 1894

- Data Types for Attributes/Fields
- DDL SQL Statements
- **Database Integrity**
- DML SQL Statements - insert, update, delete data in tables

Database Integrity

- Database integrity constraints protect DB from becoming inconsistent
- Five types of integrity constraints:
    1. Required data
    2. Domain constraints
    3. Entity integrity
    4. Referential integrity
    5. General constraints

These constraints can be defined in **CREATE** and **ALTER TABLE** statements

Integrity 1. Required data

CITY UNIVERSITY LONDON
EST 1894

- Some attributes / columns cannot take **NULL** values
  - NOT NULL: if specified the DBMS rejects attempts to insert NULLs into the attribute
  - NULL: the default - if nothing is specified, the system accepts NULLs

- Examples:
  - `position VARCHAR(10) NOT NULL`
  - `comment VARCHAR(50) NULL`

Integrity 2. Domain Constraints

CITY UNIVERSITY LONDON
EST 1894

- **CHECK** clause
  - `CHECK (searchCondition)`
  - `gender CHAR(1) NOT NULL CHECK (VALUE IN ('M', 'F', 'X'))`
  - `sex ENUM('M', 'F', 'X') NOT NULL`
- **CREATE DOMAIN** statement

  -
    ```
    CREATE DOMAIN GenderType AS CHAR(1)
        DEFAULT 'X'
        CHECK (VALUE IN ('M', 'F', 'X'));
    ```
  - `gender GenderType NOT NULL`
- **DROP DOMAIN** to remove a domain

Integrity 3. Entity

- The Primary Key (PK) of a table must contain a unique, non-null value for each row
- **PRIMARY KEY**
  - a single attribute PK: `PRIMARY KEY (branchNo)`
  - a composite PK: `PRIMARY KEY (clientNo, propertyNo)`
- SQL will reject any INSERT or UPDATE operation that attempts to create a duplicate value for the primary key

Integrity 4. Referential

- Foreign Key (FK) - a column or a set of columns that links each row in child table to row of parent table containing matching PK.
- Referential integrity means that, if FK contains a value, this must refer to an existing row in the parent table.
- SQL rejects inserts that do not meet this condition.

- Example:
  - `branchNo CHAR(4)`
  - `FOREIGN KEY (branchNo) REFERENCES Branch or …`
  - `FOREIGN KEY (branchNo) REFERENCES Branch (branchNo)`

Integrity 4. Referential Actions

- INSERT/UPDATE attempts to create FK value in the child table …
    - without matching value in parent - rejected
- UPDATE/DELETE attempts to remove a PK value in parent table …
    - with matching rows in child - depends on referential action
- Referential action specified with sub-clauses:
    - `ON UPDATE | ON DELETE`
    -
```
FOREIGN KEY staffNo REFERENCES Staff(staffNo)
          ON DELETE CASCADE | SET NULL | SET DEFAULT
```

CASCADE	deletes row from parent and matching rows in children
SET NULL	deletes row from parent and set to NULL in children
SET DEFAULT	deletes row from parent and set to DEFAULT in children
NO ACTION	the default - rejects delete from parent table

Integrity 5. General Constraints

- Adding rules governing real world transactions not related to **CREATE TABLE** ...

```
CREATE ASSERTION StaffNotHandlingTooMuch
    CHECK (NOT EXISTS (
            SELECT staffNo
            FROM PropertyForRent
            GROUP BY staffNo
            HAVING COUNT(*) > 100
          )
    )
```

Database Integrity (recap)

CITY UNIVERSITY LONDON

EST 1894

- Database integrity constraints protect DB from becoming inconsistent
- Five types of integrity constraints
    1. Required data
    2. Domain constraints
    3. Entity integrity
    4. Referential integrity
    5. General constraints

- These constraints can be defined in the **CREATE** and **ALTER TABLE** statements
- DDL language

Overview

CITY UNIVERSITY LONDON
EST 1894

- Data Types for Attributes/Fields
- DDL SQL Statements
- Database Integrity
- **DML SQL Statements - insert, update, delete data in tables**

DML SQL Statements (Syntax)

CITY UNIVERSITY LONDON
EST 1894

```
INSERT INTO table-name (field-names)
       VALUES (value-list);

INSERT INTO table-name SET field-name1=value1, field-name2=value2,
                           field-nameN=valueN;

UPDATE table-name SET field-name=value WHERE condition;

DELETE FROM table-name WHERE condition;
```

DML SQL Statements

Lets see how to use these statements:

- Insert
- Update
- Delete

Inserting data: INSERT command

CITY UNIVERSITY LONDON
EST 1894

```
INSERT INTO table-name (field-names)
            VALUES (value-list);

INSERT INTO journey (distance, journeydate, ticket, time)
            VALUES (2, '2017-03-14', 'o', '8:30:00');

SELECT * FROM journey; [to show data]
```

Inserting data: INSERT command

CITY UNIVERSITY LONDON
EST 1894

```
INSERT INTO journey SET
       distance=3, journeydate='2019-01-14', ticket='f',
       time='10:13:45';

INSERT INTO journey VALUES (7, '2020-10-07', 'f', '11:20:51');
```

Updating data: UPDATE command

What if you've made a mistake on one piece of data? You can use **UPDATE** to change it.

```
UPDATE journey SET ticket='p' WHERE ID=1;

SELECT * FROM journey; [to show data]
```

Deleting data: DELETE command

DELETE removes an entire row (record)

```
DELETE FROM journey WHERE ID=1;

SELECT * FROM journey; [to show data]
```

Very Important Information

CITY UNIVERSITY LONDON
EST 1894

- PhpMyAdmin **IS NOT A DATABASE!!!**
- It is just a nice web-interface to MySQL
- MySQL is the database behind
- The same happens with PostgreSQL and pgAdmin

Take-away

This session has covered the basics of using SQL to create database tables, and insert and update data.