**SQL Data Definition Language (DDL) and Data Control Language (DCL)** (last updated 3/30/2020)

|  |  |
| --- | --- |
| **Data Definition Language (DDL)**  The DDL is used to create/alter/drop databases, tables, indexes, views, and other database objects. In order to execute DDL SQL statements, the users must have appropriate privileges.  **CREATE** statements create objects (e.g., database, table).  **ALTER** statements make changes to existing objects. For example, we may want to alter an existing table by adding a column to it.  **DROP** statements remove objects. For a table, drop doesn't just remove the table. Instead, it removes the table and its indexes from the catalog. It also removes all the rows. | **Example 1: Create Student Table DDL statement**  CREATE TABLE `abc123db`.`Student` (  `abc123Id` CHAR(6) NOT NULL,  `name` VARCHAR(30) NOT NULL,  `classification` CHAR(1) DEFAULT 'U',  `totGradePoints` INT DEFAULT 0,  `totGradeHours` INT DEFAULT 0,  `major` CHAR(3) DEFAULT NULL,  PRIMARY KEY (`abc123Id`));  Notes:  1. The names of the database, table, and columns are surrounded by back tick characters (below the tilde on most keyboards) in MySQL.  2. The columns in a primary key usually do not allow NULL Values.  3. The primary key can be a single column or a composite of multiple columns.  4. MySQL automatically creates an index on the primary key. |
| **Create DDL Statement**  CREATE TABLE *database.tableName* (  *columnNm1* *dataType* *nullDefault*,  ...  *columnNmN* *dataType* *nullDefault*,  PRIMARY KEY (*columnList*));  Notes:   1. The database is optional and is defaulted to the current database which can be set using the USE statement. 2. Each column has its own column name. Column names are case sensitive. 3. The column data types are explained below. 4. The *nullDefault* represents how to handle a column if it wasn't included in an SQL INSERT statement:   NOT NULL – this column must be given a value and cannot be NULL.  DEFAULT NULL – this column is defaulted to NULL  DEFAULT *value* – this column's value is defaulted to the specified *value*  DEFAULT AUTO\_INCREMENT – a surrogate key is provided  DEFAULT CURRENT\_TIMESTAMP – defaults a timestamp to the current timestamp   1. Below, we will also explain the AUTO\_INCREMENT clause. | **Example 2: Examining Example 1 and an INSERT statement with missing columns**  INSERT INTO Student (`abc123Id`, `name`)  VALUES ("AAA111", "Ava Kashun");  After executing that statement, selecting from that table would give:  mysql> select \* from Student;  +----------+------------+----------------+----------------+---------------+-------+  | abc123Id | name | classification | totGradePoints | totGradeHours | major |  +----------+------------+----------------+----------------+---------------+-------+  | AAA111 | Ava Kashun | U | 0 | 0 | NULL |  +----------+------------+----------------+----------------+---------------+-------+   1. Since we didn't specify the values for classification, totGradePoints, totGradeHours, and major, the default values were automatically assigned. 2. abc123Id and name must be given values. MySQL prefers primary key columns to not allow NULL values. 3. In example 1, classification, totGradePoints, and totGradeHours are defaulted to the specified values if the column wasn't included in an INSERT. 4. If the insert had left off the `**name**` column:   INSERT INTO Student (`abc123Id`) VALUES("AAA111");  we would receive this error:  ERROR 1364 (HY000): Field 'name' doesn't have a default value |
| **Column Data Types**  CHAR(k) fixed-length character string of k characters. If a value shorter than k is provided, it pads with spaces on the right. FIXED  VARCHAR(k) variable-length character string with a maximum of k characters. TAKES BYTE FOR LINK  INT 4-byte integer; like int in Java  TIME time value using a 24-hour format: "hh:mm:ss"  Example: "14:30:00" is 2:30 PM  DATE date value in the format: "YYYY-MM-DD" Example: "2018-04-01" is April 1, 2018  TIMESTAMP Timestamp value in the format (MySQL): "YYYY-MM-DD hh:mm:ss" Some implementations (e.g., DB2) also include fractions of seconds: "YYYY-MM-DD-hh.mm.ss.nnnnnn"  DECIMAL(p,s) p is the precision or total digits and s is the scale; Example: DECIMAL(5,2) has 3 digits to the left and 2 digits to the right of the decimal point.  SMALLINT 2-byte integer; similar to short in C  BIGINT 8-byte integer; like long in Java  FLOAT 4-byte floating point  DOUBLE 8-byte floating point  There are also BLOB and TEXT types for large binary and text values. | **Example 3: Create Section Table DDL statement**  CREATE TABLE `abc123db`.`Section` (  `semester` CHAR(7) NOT NULL,  `courseNr` CHAR(7) NOT NULL,  `sectNr` INT(3) NOT NULL,  `prof` VARCHAR(20) DEFAULT NULL,  `classDays` CHAR(4) DEFAULT NULL,  `classTime` TIME DEFAULT NULL,  `room` VARCHAR(20) DEFAULT NULL,  PRIMARY KEY (`semester`, `courseNr`, `sectNr`));  Notes:  1. The primary key is a composite of multiple columns.  2. For sectNr, INT(3) doesn't specify the stored size in bytes (which is 4 bytes for INT). Instead, it specifies the default display width of 3 characters.  3. The default value for each of the last four columns is NULL. If an INSERT statement doesn't include the column, its value is defaulted to NULL. |
| **Exercise #1: Show an SQL create statement to create an Employee table.**   * We want its primary key to be a six-character employee ID. Employees have a first name, middle initial, and last name. First and last name must be provided for each employee. If not specified, middle initial should be a blank. First and last name can each be a maximum of 40 characters. * Most employees are assigned to departments which are three characters. * Employees have a one-character employee status which can be A – active, T – terminated, R – retired. If not specified, it should be default to 'A'. * Employees also have a salary and number of exemptions. Salary is defaulted to 0, but number of exemptions is defaulted to 1. | **Exercise #1: POTENTIAL FINAL EXAM QUESTION**  CREATE TABLE Employee (  `empId` CHAR(6) NOT NULL,  `firstNm` VARCHAR(40) NOT NULL,  `lastNm` VARCHAR(40) NOT NULL,  `midInit` CHAR(1) DEFAULT ‘ ’,  `department` CHAR(3) NOT NULL,  `empStatus` CHAR(1) DEFAULT ‘A’,  `salary` DOUBLLE DEFAULT 0.0,  `exemptionCount` INT DEFAULT 1,  PRIMARY KEY (‘empid’) ); |
| **Indexes and Hashing**  MySQL doesn't support hashing. It uses B-trees for its indexes.  CREATE uniqueINDEX *indexName* ON *tableName*  (*indexColumn1*, *indexColumn2*, …);  Automatically, the primary key is the clustering index. You cannot change to another index as the clustering index. | **Example 4: Create Enrollment Table and Create Index.**  Create the Enrollment table having abc123Id, semester, courseNr, and sectNr as the composite primary key. To obtain the roster for a section, we also need a secondary key on semester, courseNr, and sectNr. If we also include the abc123Id in that secondary key, it will be unique.  CREATE TABLE `abc123db`.`Enrollment` (  `abc123Id` CHAR(6) NOT NULL,  `courseNr` CHAR(7) NOT NULL,  `semester` CHAR(7) NOT NULL,  `sectNr` INT(3) NOT NULL,  `grade` CHAR(1) DEFAULT ' ',  PRIMARY KEY (`abc123Id`, `semester`, `courseNr`, `sectNr`));  CREATE UNIQUE INDEX semIdx ON `abc123db`.`Enrollment`  (`semester`, `courseNr`, `sectNr`, `abc123Id`); |
| **Surrogate Key (aka, Auto Increment)**  In examples 1, 3, and 4, we used **natural keys** (aka, business keys) for Student and Section. A **surrogate key** is a key that is generated for a table instead of an INSERT statement providing the value**. A surrogate key is not derived from a natural key.**  When an insertion is done without specifying the surrogate key column, the DBMS will automatically generate the surrogate key.  It may be necessary to reference a surrogate as part of a key for another table. The value LAST\_INSERT\_ID() can be referenced to obtain the last generated surrogate key for the current connection.  We will discuss AUTO\_INCREMENT further when we discuss JDBC. | **Example 5: Auto Increment**  Suppose a banking application needs a bank account, transaction type (e.g., deposit, withdrawal, penalty), amount, and timestamp; however, there isn't a reasonable natural key. We can use AUTO INCREMENT to generate the key.  CREATE TABLE `Transaction` (  `transId` INT(11) NOT NULL AUTO\_INCREMENT,  `acctNr` CHAR(12) NOT NULL,  `transType` CHAR(3) NOT NULL,  `transAmt` double NOT NULL,  `transTs` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`transId`));  -- insert a row without specifying transId and transTs  INSERT INTO `Transaction` (acctNr, transType, transAmt)  VALUES("123456789012", "DEP", 100.00);  -- insert a row into a related LogTrans table  CREATE TABLE LogTrans (  `transId` INT(11) NOT NULL,  `operation` CHAR(1) DEFAULT 'I',  `acctNr` CHAR(12) NOT NULL,  `transType` CHAR(3) NOT NULL,  `transAmt` double NOT NULL,  `transTs` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`transId`, `operation`));  INSERT INTO `LogTrans` (transId, acctNr, transType, transAmt, transTs)  SELECT transId, acctNr, transType, transAmt, transTs from `Transaction`  where transId = LAST\_INSERT\_ID();  mysql> select \* from LogTrans;  +---------+-----------+--------------+-----------+----------+---------------------+  | transId | operation | acctNr | transType | transAmt | transTs |  +---------+-----------+--------------+-----------+----------+---------------------+  | 1 | I | 123456789012 | DEP | 100 | 2018-03-05 10:25:50 |  +---------+-----------+--------------+-----------+----------+---------------------+ |
| **Miscellaneous DDL**  CREATE USER `*userId*`@`*server*` IDENTIFIED BY '*passWord'*; create the specified user and password.  DROP USER if exists `userId`@`server`; drops the specified user. Using the optional **if exists** clause, avoids an error if the user doesn't exist.  CREATE DATABASE *database*; create the specified *database*  DROP DATABASE if exists *database*; drops the specified database. Using the optional **if exists** clause, avoids an error if the database doesn't exist.  DROP TABLE if exists *database.tableName*; drops the specified table. Using the optional **if exists** clause, avoids an error if the table doesn't exist. All of the table's data is lost. WHOLE TABLE BE GONE  ALTER TABLE *tableNm* ADD COLUMN *colName dataType nullDefault*;  alters the specified table by adding the specified column. The *dataType* and *nullDefault* values were described earlier. Note that an added column must either allow null values or have a default value so that existing rows don't cause issues. | **Example 6: Misc DDL**  drop user IF EXISTS 'MYUSER'@'%';  create user 'MYUSER'@'%' IDENTIFIED BY 'PASSWD';  drop database IF EXISTS `MYDB` ;  create database `MYDB`;  alter table Course add column hasLab char(1) DEFAULT 'N'; |
| **Data Control Language (DCL) - GRANT**  The DCL is used to give users privileges and revoke those privileges. We will frist discuss GRANT and then REVOKE.  GRANT *privilege* ON *resource* TO *userId*;  GRANT *privilege* ON *resource* TO *userId*  with grant option;  Table privileges:  SELECT retrieve rows from the table  INSERT insert rows into the table  UPDATE update existing rows in the table  DELETE delete existing rows from the table  CREATE create a table  DROP drop a table  Database privileges:  USAGE reference the catalog tables for the specified db  *resource*:  *`database`.`tableName`* specific table  *`database`.\** All tables in the db  *userId*:  public privileges available to everyone  `userId`@`server` | **Example 7: GRANTs**  grant usage on `MYDB`.\* to 'MYUSER'@'%';  % = matching anything  -- In most implementations of SQL, this statement allows users to  -- create any tables in `MYDB`.  grant create on `MYDB`.\* to 'MYUSER'@'%';  -- The specified table must exist to do this grant  grant drop, select, insert, update, delete on `MYDB`.`tableNm` to 'MYUSER'@'%';  -- Amaxingly, if we specify the create privilege, we can grant those other  -- privileges in the same statement  grant create, drop, select, insert, update, delete  on `MYDB`.`tableNm` to 'MYUSER'@'%';  -- Give the specified user select privilege on the specified table and allow that  -- user to grant it to others.  grant select on `MYDB`.`tableNm` to 'MYUSER'@'%' with grant option; |
| **Data Control Language (DCL) - REVOKE**  REVOKE *privilege* ON *resource* FROM *userId*; | **Example 8: GRANTs**  revoke create on `MYDB`.\* from 'MYUSER'@'%'; |
|  |  |

**©2020 Larry W. Clark,** UTSA CS students may make copies for their personal use