

Comp2350/6350 Week8: Practical Supplement

(Getting familiar with DDL commands in SQL)

SQL DDL Commands

The acronym DDL means Data Definition Language. DDL commands are used to create database objects and modify their structure.

Examples

- **CREATE** – Create an object i.e. create a database, table, triggers, index, functions, stored procedures, etc.
- **DROP** – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc.
- **ALTER** – Used to alter the existing database or its object structures.
- **TRUNCATE** – This SQL DDL command removes records from tables

Creating a Table (CREATE TABLE):

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows:

```
CREATE TABLE [IF NOT EXISTS] table_name(  
    column_1_definition,  
    column_2_definition,  
    ...,  
    table_constraints  
) ENGINE=storage_engine;
```

- First, you specify the name of the table that you want to create after the **CREATE TABLE** keywords. The table name must be unique within a database. The **IF NOT EXISTS** is optional. It allows you to check if the table that you create already exists in the database. If this is the case, MySQL will ignore the whole statement and will not create any new table.
- Second, you specify a list of columns of the table in the **column_list** section, columns are separated by commas.
- Third, you can optionally specify the storage engine for the table in the **ENGINE** clause. You can use any storage engine such as InnoDB and MyISAM. If you don't explicitly declare a storage engine, MySQL will use InnoDB by default.

The following shows the syntax for a column's definition:

<code>column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT] column_constraint;</code>

Here are the details:

- The `column_name` specifies the name of the column. Each column has a specific data type and optional size e.g., `VARCHAR(255)`
- The `NOT NULL` constraint ensures that the column will not contain `NULL`. Besides the `NOT NULL` constraint, a column may have additional constraint such as `CHECK`, and `UNIQUE`.
- The `DEFAULT` specifies a default value for the column.
- The `AUTO_INCREMENT` indicates that the value of the column is incremented by one automatically whenever a new row is inserted into the table. Each table has a maximum one `AUTO_INCREMENT` column.

The remaining clauses are known as **table constraints** and can optionally be preceded with the clause:

CONSTRAINT ConstraintName

which allows the constraint to be dropped by name using the `ALTER TABLE` statement

The **PRIMARY KEY** clause specifies the column or columns that form the primary key for the table. If this clause is available, it should be specified for every table created. By default, `NOT NULL` is assumed for each column that comprises the primary key. Only one `PRIMARY KEY` clause is allowed per table. SQL rejects any `INSERT` or `UPDATE` operation that attempts to create a duplicate row within the `PRIMARY KEY` column(s). In this way, SQL guarantees the uniqueness of the primary key.

The **FOREIGN KEY** clause specifies a foreign key in the (child) table and the relationship it has to another (parent) table. This clause implements referential integrity constraints. The clause specifies the following:

- A *listOfForeignKeyColumns*, the column or columns from the table being created that form the foreign key.
- A `REFERENCES` subclause, giving the parent table; that is, the table holding the matching candidate key. If the *listOfCandidateKeyColumns* is omitted, the foreign key is assumed to match the primary key of the parent table. In this case, the parent table must have a `PRIMARY KEY` clause in its `CREATE TABLE` statement.

There can be as many `FOREIGN KEY` clauses as required.

1. Creating a Table with Primary Key Constraint:

The screenshot shows a SQL IDE window with a query editor and an action output pane. The query editor contains the following SQL code:

```
1 CREATE TABLE IF NOT EXISTS Worker (  
2     workerid varchar(10) NOT NULL,  
3     workername VARCHAR(255) NOT NULL,  
4     joining_date DATE NOT NULL,  
5     city VARCHAR(255),  
6     salary INT(20) NOT NULL,  
7     PRIMARY KEY (workerid)  
8 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

The action output pane shows the execution of the query:

#	Time	Action	Message
1	19:36:32	CREATE TABLE IF NOT EXISTS Worker (workerid varchar(10) NOT NULL, workername VARCHAR(255) ...	0 row(s) affected

Describing the newly created table:

The screenshot shows the same SQL IDE window with a query editor and a result grid. The query editor contains the following SQL code:

```
1 desc Worker;  
2
```

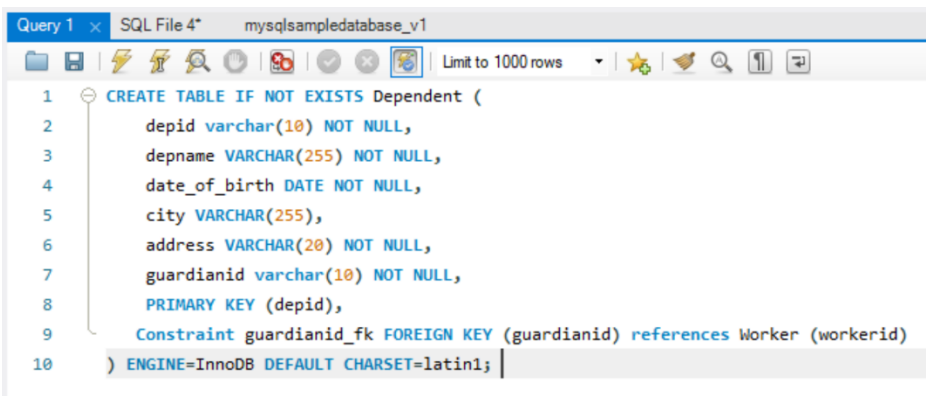
The result grid shows the table structure:

Field	Type	Null	Key	Default	Extra
workerid	varchar(10)	NO	PRI	NULL	
workername	varchar(255)	NO		NULL	
joining_date	date	NO		NULL	
city	varchar(255)	YES		NULL	
salary	int(20)	NO		NULL	

Creating a table with both Primary and Foreign Key Constraints:

```
1 CREATE TABLE IF NOT EXISTS Dependent (  
2     depid varchar(10) NOT NULL,  
3     depname VARCHAR(255) NOT NULL,  
4     date_of_birth DATE NOT NULL,  
5     city VARCHAR(255),  
6     address VARCHAR(20) NOT NULL,  
7     guardianid varchar(10) NOT NULL,  
8     PRIMARY KEY (depid),  
9     FOREIGN KEY (guardianid) references Worker (workerid)  
10 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Where the primary key of the *Dependent* table references the primary key of the *Worker* table.



```
Query 1 x SQL File 4* mysqlsampledatabase_v1  
1 CREATE TABLE IF NOT EXISTS Dependent (  
2     depid varchar(10) NOT NULL,  
3     depname VARCHAR(255) NOT NULL,  
4     date_of_birth DATE NOT NULL,  
5     city VARCHAR(255),  
6     address VARCHAR(20) NOT NULL,  
7     guardianid varchar(10) NOT NULL,  
8     PRIMARY KEY (depid),  
9     Constraint guardianid_fk FOREIGN KEY (guardianid) references Worker (workerid)  
10 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

This query assigns a constraint name i.e. '*guardianid_fk*' to the foreign key constraint which makes it easy to drop or modify a constraint by name when working with ALTER statements.

Changing a Table Definition (ALTER TABLE):

The ALTER TABLE statement is used for changing the structure of a table once it has been created. The definition of the ALTER TABLE statement in the ISO standard consists following options to:

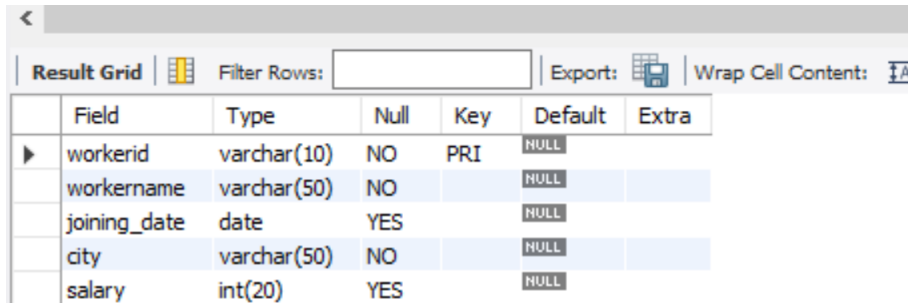
- add a new column to a table;
- drop a column from a table;
- add a new table constraint;
- drop a table constraint;

Adding a Column:

Let's suppose you want to record 'leaving date' for the workers who have either resigned from their services or were terminated. We need to add a new column to the following table and it should accept null values as well for the employees who are still serving in the company.

The general syntax of adding a column to the table is:

```
ALTER TABLE table_name  
ADD new_column column_definition;
```



	Field	Type	Null	Key	Default	Extra
▶	workerid	varchar(10)	NO	PRI	NULL	
	workername	varchar(50)	NO		NULL	
	joining_date	date	YES		NULL	
	city	varchar(50)	NO		NULL	
	salary	int(20)	YES		NULL	

Example:

```
ALTER TABLE Worker ADD COLUMN end_of_Service DATE NULL;
```

Dropping a Column in Table:

The syntax to drop a column in a table is:

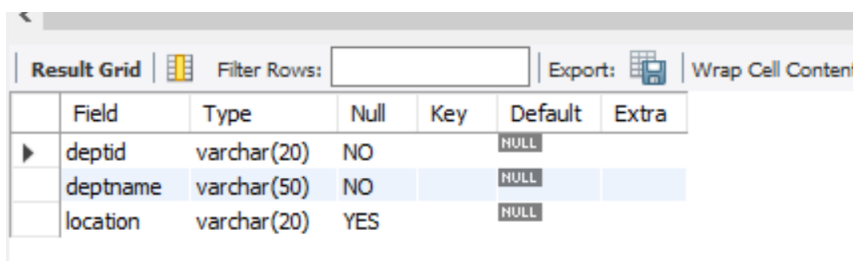
```
ALTER TABLE table_name  
DROP column column_name;
```

Example:

```
ALTER TABLE Worker DROP COLUMN end_of_Service;
```

Adding a Constraint:

The ADD CONSTRAINT command is used to create a constraint after a table is already created. Consider the following *departments* table that doesn't have any primary key:



	Field	Type	Null	Key	Default	Extra
▶	deptid	varchar(20)	NO		NULL	
	deptname	varchar(50)	NO		NULL	
	location	varchar(20)	YES		NULL	

The following SQL adds a constraint named "PK_id" that is a PRIMARY KEY constraint on multiple columns (deptid):

```
ALTER TABLE departments  
ADD CONSTRAINT PK_id PRIMARY KEY (deptid);
```

Dropping a Constraint:

The DROP CONSTRAINT command is used to delete a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint.

The general syntax for dropping a constraint is:

```
ALTER TABLE table_name  
DROP PRIMARY KEY/FOREIGN KEY constraint_name;
```

Example:

```
ALTER TABLE departments  
DROP primary key;
```

Removing a Table (DROP TABLE):

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

NOTE – You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

The basic syntax of this DROP TABLE statement is as follows:

```
DROP TABLE table_name;
```

Example:

Let's drop the table that we created in CREATE TABLE's example.
Upon executing:

```
DROP TABLE Worker;
```

You should get:


 11 20:02:40 Drop table Worker Error Code: 1217. Cannot delete or update a parent row: a foreign key constraint fails

In order to drop the Parent table, you need to drop the child table where the primary key of the parent table is being used as a foreign key in the child table. So we first need to drop Dependent table:


```
Drop table Dependent;
```

And then

```
Drop table Worker;
```

 15 20:12:07 drop table Worker 0 row(s) affected

You will get the following error if you try selecting data from the Worker table:

 17 20:13:19 select * from Worker LIMIT 0, 1000 Error Code: 1146. Table 'mq45063192.Worker' doesn't exist

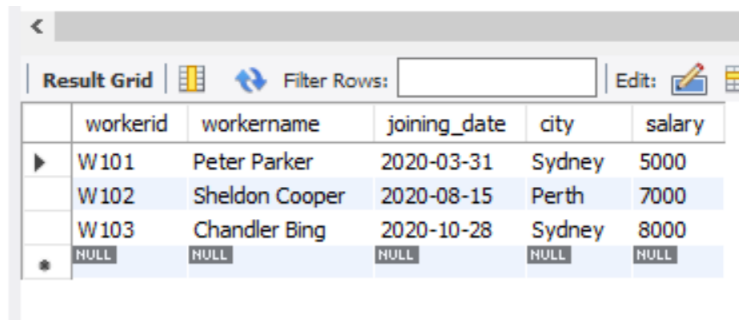
Truncating Table:

The SQL TRUNCATE TABLE statement is used to remove all records from a table. It performs the same function as a DELETE statement without a WHERE clause. The general syntax of truncate command is as follows:

TRUNCATE TABLE *table_name*;

Example:

If you have a following table called *Worker*:

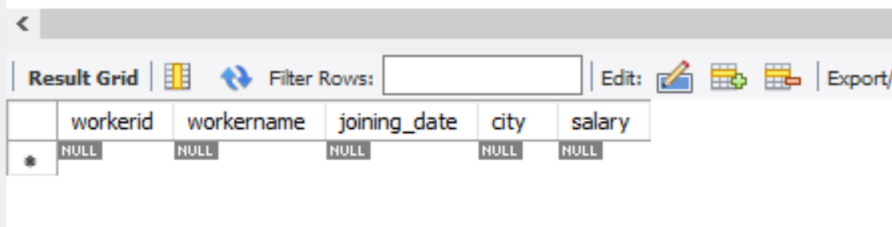


	workerid	workername	joining_date	city	salary
▶	W101	Peter Parker	2020-03-31	Sydney	5000
	W102	Sheldon Cooper	2020-08-15	Perth	7000
	W103	Chandler Bing	2020-10-28	Sydney	8000
*	NULL	NULL	NULL	NULL	NULL

Then after executing:

Truncate table Worker;

These are the results that you should see when you try to select/retrieve data from the Worker table:



	workerid	workername	joining_date	city	salary
*	NULL	NULL	NULL	NULL	NULL

Points to Remember:

- You need to create parent/primary table first before creating the child/secondary table so you can refer to parent table's primary key into the child table.
- You need to drop/truncate child table first in order to remove all the dependencies/references of parent table on the child table.

Content Credit:

- <https://www.w3schools.com/sql/>
- <https://www.tutorialgateway.org/sql-dml-ddl-dcl-and-tcl-commands/>
- <https://www.tutorialspoint.com/sql/index.htm>
- Connolly, Thomas & Begg, Carolyn. *Database Systems: A Practical Approach to Design, Implementation, and Management* (4th Edition)