SQL

Part I (DDL)

420-983-VA Introduction to Databases using Oracle Adrian Onet

SQL

Structured Query Language

Structured Query Language (SQL) is the ISO-ANSI standard data definition language and data manipulation language for relational database management systems (DBMSs). Individual relational database systems use slightly different dialects of SQL syntax and naming rules. Used in the physical implementation of relational database models.

Year	SQL Standard
SQL/74	Original SQL (SEQUEL).
SQL/86	SQL became a standard by ANSI (American National Standards Institute) and ISO (International Standards Organization).
SQL/96	Major modification (ISO 9075).
SQL/99	Added many features including recursive queries, triggers, procedural and control-of-flow statements, and some object-oriented structures.
SQL/2003	Introduced XML-related features.
SQL/2006	Defined ways for importing and storing XML data in database.
SQL/2008	Added TRUNCATE TABLE statement and INSTEAD OF triggers.
SQL/2011	Time Period Definitions, temporal primary keys, temporal referential integrity.
SQL/2016	JSON support, date and time formatting and parsing. New data type DECFLOAT.

SQL

Structured Query Language

SQL

(Data Definition Language) DDL

- CREATE
- ALTER
- DROP
- RENAME

(Data Manipulation Language) DML

- SELECT
- UPDATE
- DELETE
- INSERT
- TRUNCATE
- CTE

(Data Control Language) DCL

- GRANT
- REVOKE

SQL

Structured Query Language

Main Relational Database Management Systems that implement SQL standard.

SQL

Data Definition Language (DDL)

Managing Databases

To create a database (works with most RDBMS's: MySQL, SQL Server, Sybase, PostgreSQL, DB2)

```
CREATE DATABASE database_name;
```

After creating a database, before using it, in DB2 you have also to run:

```
ACTIVATE database_name;
```

In Oracle the user acts as the database owner. All objects created by that user are considered part of the user database. To create a database user:

```
CREATE USER database_user IDENTIFIED BY password;
```

```
GRANT CONNECT, DBA TO database_user;
```

SQL

Data Definition Language (DDL)

Managing Databases

To select a database you'll have to run

```
USE database_name;
```

```
CONNECT TO database_name;
```

In Oracle you have to prefix the object name by its owner, thus no database selection is necessary.

SQL

Data Definition Language (DDL)

Managing Databases

To delete a database you'll have to run

```
DROP DATABASE database_name;
```

```
DROP USER database_user CASCADE;
```

In DB2 you have the option to deactivate the database without deleting its data and log files.

```
DEACTIVATE database_name;
```

SQL

Data Definition Language (DDL)

Managing Databases

Example: with **or any other MySQL client**

Create database:

```
CREATE DATABASE students;
```

or:

```
CREATE DATABASE IF NOT EXISTS students;
```

Drop database:

```
DROP DATABASE students;
```

or:

```
DROP DATABASE IF EXISTS students;
```

SQL

Data Definition Language (DDL)

Managing Databases

List existing databases:

```
LIST DB DIRECTORY;
```

```
System Database Directory

 Number of entries in the directory = 3

Database 1 entry:

 Database alias                       = SAMPL
 Database name                        = SAMPLE
 Local database directory             =
/db2home/db2inst1
 Database release level               = 10.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            =
 Alternate server port number         =

Database 2 entry:
…
```

```
SHOW DATABASES;
```

```
SELECT username FROM sys.all_users;
```

```
SELECT name FROM master.dbo.sysdatabases;
```

```
SELECT datname FROM pg_database
WHERE datistemplate=false;
```

SQL

Data Definition Language (DDL)

Managing Tables

To create a table in the current database (or under the current user for ORACLE)

```
CREATE TABLE <table name>
(<table element> [{, <table element>}...])
```

```
<table element>::=
<column definition>
| {[CONSTRAINT <constraint name>] PRIMARY KEY
(<column name> [{, <column name>}...])}
| {[CONSTRAINT <constraint name>] FOREIGN KEY [<index name>]
(<column name> [{, <column name>}...]) REFERENCES
     <table_name> (<column name> [{, <column name>}...])
| {[CONSTRAINT <constraint name>] UNIQUE [<index name>]
(<column name> [{, <column name>}...])}
```

```
<column definition>::=
<column name> <type> [NOT NULL | NULL] [DEFAULT <value>]
[PRIMARY KEY]
```

SQL

Data Definition Language (DDL)

Managing Tables

Example:

Database name (students) optional if database is selected before (`USE students`)

```
CREATE TABLE students.students (
                  student_id INT NOT NOLL,
                  name VARCHAR(50) NULL,
      CONSTRAINT PK_STUDENTS PRIMARY KEY (student_id)
   );
```

STUDENTS

student_id	name

SQL

Data Definition Language (DDL)

Managing Tables

Example:

Database name (students) optional if database is selected before (**USE** students)

```
CREATE TABLE students.courses (
            course_id int NOT NULL PRIMARY KEY,
            course_name varchar(100) NOT NULL,
            hours_number int NOT NULL DEFAULT 60
         );
```

SQL

Data Definition Language (DDL)

Managing Tables

Dropping tables

```
DROP TABLE [db_name.]<table_name>;
```

```
DROP TABLE [IF EXISTS] [db_name.]<table_name>;
```

SQL

Data Definition Language (DDL)

Managing Tables

List Database Tables

```
SHOW TABLES;
```

```
SELECT table_name FROM user_tables;
```
→ Current user tables.

```
SELECT owner, table_name FROM all_tables;
```
→ Tables accessible by current user.

```
SELECT owner, table_name FROM dba_tables;
```
→ All tables.

```
SELECT table_schema, table_name FROM information_schema.tables
WHERE table_schema='models';
```

```
SELECT owner, name FROM SYSIBM.SYSTABLES WHERE type='T';
LIST TABLES;
```

```
SELECT user_name(uid), name FROM sysobjects WHERE type='U';
```

SQL

Data Definition Language (DDL)

Managing Tables

Add/Modify/Drop columns

```
ALTER TABLE <table_name> ADD [ COLUMN ]
                      <column_name> <column_definition>;

ALTER TABLE <table_name> ALTER [ COLUMN ]
                      <column_name> <column_definition>;

ALTER TABLE <table_name> MODIFY [ COLUMN ]
                      <column_name> <column_definition>;

ALTER TABLE <table_name> DROP [ COLUMN ]
                      <column_name>;
```

SQL

Data Definition Language (DDL)

Managing Tables

Add/Modify/Drop columns. Example:

```
ALTER TABLE students ADD gender CHAR(1) NOT NULL;
```

STUDENTS

student_id	name	gender

SQL

Data Definition Language (DDL)

Managing Tables

List table columns.

MySQL

```
DESCRIBE employees;
```

```
SELECT column_name, data_type FROM information_schema.columns
 WHERE table_schema='models' AND table_name='employees';
```

PostgreSQL

SQL Server

```
SELECT name, type_name(xtype) as data_type FROM syscolumns
 WHERE id=object_id('employees');
```

SAP Sybase

```
SELECT C.name, T.name as date_type
   FROM syscolumns C INNER JOIN systypes T ON C.type=T.type
 WHERE id=object_id('employees');
```

ORACLE

```
SELECT column_name, data_type FROM sys.all_tab_colummns
 WHERE table_name='employees';
```

IBM DB2

```
SELECT colname,type_name as data_type FROM syscat.columns
WHERE tabname='employees';
```

SQL

Data Definition Language (DDL)

Managing Constraints

Add/Drop Primary key. Note that all below constraints can be added at table creation.
See table creation section.

```
ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name>
         PRIMARY KEY (<column name> [{, <column name>}...]);
```

```
ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>;
```

```
ALTER TABLE <table_name> DROP INDEX 'PRIMARY';
```

SQL

Data Definition Language (DDL)

Managing Constraints

Add/Drop Foreign keys. Note that all below constraints can be added at table creation.
See table creation section.

```
ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name>
FOREING KEY (<column name> [{, <column name>}...])
REFERENCES <table_name> (<column name> [{, <column name>}...]);
```

```
ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>;
```

```
ALTER TABLE <table_name> DROP FOREIGN KEY <constraint_name>;
```

SQL

Data Definition Language (DDL)

Managing Constraints

Add/Drop Unique keys. Note that all below constraints can be added at table creation.
See table creation section.

```
ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name>
         UNIQUE (<column name> [{, <column name>}...]);
```

```
ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>;
```

```
ALTER TABLE <table_name> DROP INDEX <constraint_name>;
```

SQL

Data Definition Language (DDL)

Managing Indexes

Add/Drop Unique keys. Note that all below constraints can be added at table creation.
See table creation section.

```
CREATE [ UNIQUE ] INDEX index_name
          ON <object> ( column [ ASC | DESC ] [ ,...n ] );
```

```
DROP INDEX index_name;
```

SQL

Data Control Language (DCL)

Grant User/Group Access

Following statement grant user (also group or role) access to database objects.

```
        { ALL |
        EXECUTE |
        SELECT |
GRANT   INSERT |    ON <object_name> TO user [, user] …;
        UPDATE |
        DELETE }
```

SQL

Data Control Language (DCL)

Revoke User/Group Access

Following statement revoke user (also group or role) access to database objects.

```
         { ALL     |
          EXECUTE  |
          SELECT   |
REVOKE    INSERT   |   ON <object_name> FROM user [, user] …;
          UPDATE   |
          DELETE   }
```

SQL

Data Manipulation Language (DML)

Set Auto commit ON/OFF.

Tells the server to commit each DML statements separately (ON) or commit only when the transaction is committed (OFF).

`SET AUTOCOMMIT ON;`	ON	`SET autocommit=1;`
`SET AUTOCOMMIT OFF;`	OFF	`SET autocommit=0;`
`EXPORT DB2OPTIONS='-c'`	ON	`SET AUTOCOMMIT TO ON;`
`EXPORT DB2OPTIONS='+c'`	OFF	`SET AUTOCOMMIT TO OFF;`

SQL

Data Manipulation Language (DML)

Adding rows to existing table.

Inserting one row in the table.

```
INSERT INTO table_name [(column1, column2, column3, …)]
VALUES (value1, value2, value3, ...);
```

Example:

```
INSERT INTO students (student_id, name, gender)
VALUES (1, 'John', 'M');
```

STUDENTS

student_id	name	gender
1	John	M

SQL

Data Manipulation Language (DML)

Adding rows to existing table.

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

Inserting multiple rows in a table.

```
INSERT INTO table_name [(column1, column2, …)]
VALUES (value1, value2, value3, ...)
[,(value1, value2, value3, ...) [, …]];
```

Example:

```
INSERT INTO students (student_id, name, gender)
VALUES (2, 'Mike', 'M'), (3, 'Marry', 'F');
```

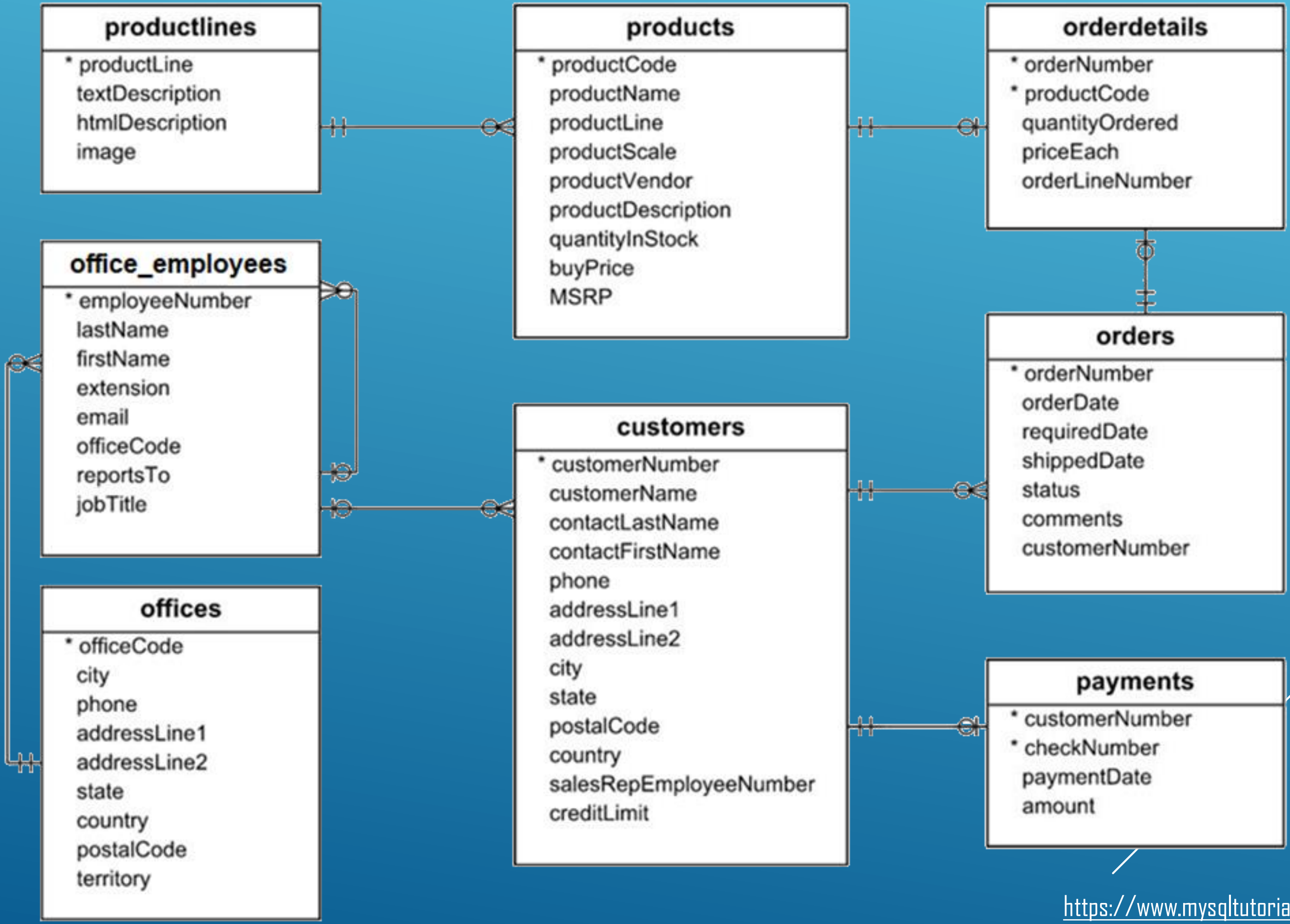
ORACLE

```
INSERT ALL
    INSERT INTO students (student_id, name, gender)
                                        VALUES (2, 'Mike', 'M')
    INSERT INTO students (student_id, name, gender)
                                        VALUES (3, 'Marry', 'F')
SELECT 1 FROM DUAL
```

SQL

Data Manipulation Language (DML)

Sample Models Schema. Describes an automotive models manufacturer and its sales.



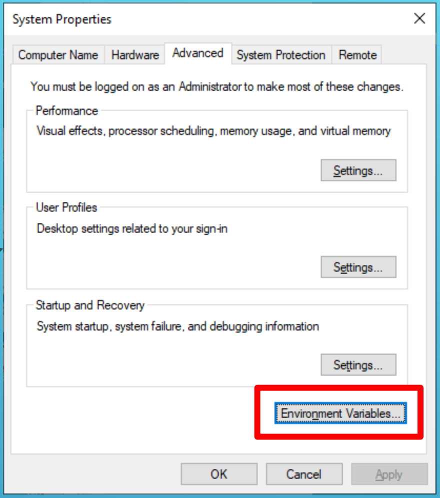
https://www.mysqltutorial.org/mysql-sample-database.aspx

SQL

Data Manipulation Language (DML)

Loading Sample Data Into MySQL

1. Add mysql executable into system Path.



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

```
export PATH=/usr/bin/:$PATH
```

2. Run mysql client from command line

```
$ mysql -u root -p
password: ********
```

SQL

Data Manipulation Language (DML)

Loading Sample Data into MySQL

3. Load schema and data into the database.

```
mysql> source c:/downloaded/mysqlsampledatabase.sql
```

```
Query OK, 1 row affected (0.01 sec)

Database changed
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.04 sec)

Query OK, 23 rows affected (0.01 sec)
Records: 23  Duplicates: 0  Warnings: 0

…
```

SQL

Data Manipulation Language (DML)

Delete rows

Deleting rows from the database.

STUDENTS

student_id	name	gender
2	Mike	M
3	Marry	F

```
DELETE FROM <table_name> [ WHERE <condition> ];
```

Deleting all rows students data.

```
DELETE FROM students WHERE student_id=1;
```

SQL

Data Manipulation Language (DML)

Delete rows

MySQL to delete rows without where condition or where condition not on key values you have to disable safe updates.

```
DELETE FROM students WHERE name='Mike';
```
Error!

```
SET SQL_SAFE_UPDATES = 0;
```

```
DELETE FROM students WHERE name='Mike';
```
Success!

SQL

Data Manipulation Language (DML)

Delete all rows

Deleting all rows without logging.

```
TRUNCATE TABLE <table_name>;
```

Note, in MySQL even if safe updates is enabled, truncate table will work!!!

SQL

Data Manipulation Language (DML)

Update rows

Simple Update.

```
UPDATE <table_name> SET field1=expression_1
[, field2=expression_2 [, …]] [WHERE condition];
```

STUDENTS

student_id	name	gender
2	Adam	M
3	Marry	F

Example.

```
UPDATE students SET name='Adam' WHERE student_id=2;
```

SQL

Data Manipulation Language (DML)

Update rows

In MySQL to update rows without where condition or where condition not on key values you have to disable safe updates.

```
UPDATE students SET name='Mike1'
WHERE name='Mike';
```

Error!

```
SET SQL_SAFE_UPDATES = 0;
```

```
UPDATE students SET name='Mike1'
WHERE name='Mike';
```

Success!