UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECD2523 DATABASE

Structured Query Language (SQL) 1| Data Definition Language (DDL)

Content adapted from Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education.

LECTURE LEARNING OUTCOME

By the end of this lecture, students should be able to:

01 Perform basic operations in a DBMS software

02 Construct SQL statements to:

- Create and delete tables
- Perform changes to table structure
- Include integrity constraints to tables

Introduction to DBMS

- DBMS:
  - A software system that enables users to **define**, **create**, and maintain the database and that provides controlled access to this database.
  - DBMS perform the tasks through commands (aka queries) written in Structured Query Language (SQL)
  - Example of DBMS
    - mySQL (what we will be using for this course)
    - Oracle

Structured Query Language

A database language that allows users to:

- Create database and relation structures.
- Perform basic data management tasks (insertion, modification and deletion of data from relations)
- Perform simple and complex queries.

Two major components:

- **Data Definition Language (DDL)**
- **Data Manipulation Language (DML)**

Data Definition Language (DDL)

- Definition:
  - A language that allows DBA or user to **describe** and **name** the entities, attributes, and relationships required for the application, together with any associated **integrity** and **security** constraints.
- DDL allows database objects (schemas, domains, tables, views, etc) to be **created**, **modified** or **deleted**. Examples of DDL statements:
  - CREATE SCHEMA … DROP SCHEMA …
  - CREATE TABLE … DROP TABLE …
  - ALTER TABLE …

Integrity Enhancement Feature

- SQL provides some facilities for integrity control to protect the database from becoming inconsistent.
  - Required Data
    - Using the **NOT NULL** constraint for column to ensure the column must contain a valid value.
  - Entity Integrity
    - Using **PRIMARY KEY** constraint for a column with unique, non-null value. Only ONE primary key per table.
    - Use **UNIQUE** constraint for column with unique values. Allow to be NULL but can assign to multiple columns.
  - Referential Integrity
    - Using **FOREIGN KEY** constraint to link to parent table that containing the matching attribute. A table can have multiple foreign keys.

Data Integrity Constraints

Constraints	Description
NOT NULL	The column cannot contain a null value
UNIQUE	The values for a column or a combination of columns must be unique for all rows in the table, allows NULL value
PRIMARY KEY	The column (or a combination of columns) must contain the unique **AND IS NOT NULL** value for all rows
FOREIGN KEY	The column (or a combination of columns) must establish and enforce a reference to a column or a combination of columns in another (or the same) table
CHECK	A condition must be true when the query is executed

Constraint Guidelines (1)

- Name a constraint (otherwise the mySQL server generates one)
- Constraints are easier to reference if given a meaningful name (e.g.: *emp_deptNo fk* or *dept_deptNo pk*)
- Create a constraint at either of the following times:
  - At the same time as the creation of the table
  - After the creation of the table
- Define a constraint at the column or table level
- View a constraint in the data dictionary

Constraint Guidelines (2)

- Column-level constraints are included when the column is defined
- Table-level constraints are defined at the end of the table definition, and must refer to the column or columns on which the constraints pertains
- Functionally, a column-level constraint is the same as a table-level constraint
- **NOT NULL** constraints can be defined only at the column level
- Constraints that apply to more than one column must be defined at the table level

Dealing with Tables

- Define database structure and control access to data

1. **CREATE TABLE**:
   - to create table.
2. **ALTER TABLE**:
   - to modify the structure of the existing tables.
3. **DROP TABLE**:
   - to delete the existing tables.

Creating Tables

- General SQL syntax for creating a table:

```
CREATE TABLE TableName
        {(columnName dataType [NOT NULL] [UNIQUE]
        [DEFAULT defaultOption] [CHECK (searchCondition)] [, . . .]}
        [PRIMARY KEY (listOfColumns),]
        {[UNIQUE (listOfColumns)] [, . . .]}
        {[FOREIGN KEY (listOfForeignKeyColumns)
        REFERENCES ParentTableName [(listOfCandidateKeyColumns)]
            [MATCH {PARTIAL | FULL}
            [ON UPDATE referentialAction]
            [ON DELETE referentialAction]] [, . . .]}
        {[CHECK (searchCondition)] [, . . .]})
```

Source: Connolly, 2015

[ ] : optional

Common SQL Data Type

- Overview

TABLE 7.1 ISO SQL data types.

DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit†	BIT	BIT VARYING			
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION		
datetime	DATE	TIME	TIMESTAMP		
interval	INTERVAL				
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT		

†BIT and BIT VARYING have been removed from the SQL:2003 standard.

Source: Connolly, 2015

Example (CREATE TABLE)

- Given the structure of the Department table as follows:

Department

Attribute	Datatype
deptNo (Primary Key)	Integer (5)
deptName – requires values	Varchar(20)
address	Varchar(30)
city	Varchar(15)

Employees

Attribute	Datatype
empID (Primary Key)	Integer
fName – requires values	Varchar(20)
lName – requires values	Varchar(30)
email – unique	Varchar(30)
salary – must be more than 1,700	Decimal (10,2)
deptNo (Foreign key) – deptNo is an attribute in Department relation	Integer(5)

CREATE TABLE without constraint

- Creating a table **without constraint**.

```
CREATE TABLE tableName (
columnName dataType [ DEFAULT value ]
[, column2Name datatype [ DEFAULT value]]);
```

[ ] : optional

Example (CREATE TABLE) without constraint

- Create table without constraint:

```
CREATE TABLE Department (
        deptNo INT(5),
        deptName VARCHAR(20),
        address VARCHAR(30),
        city VARCHAR(15)
);
```

```
CREATE TABLE Employees (
        empID INT,
        fName VARCHAR(10),
        lName VARCHAR(20),
        deptNo INT(5)
);
```

- In this case, constraints (e.g.: Primary Key) can be added using **ALTER TABLE** command.

CREATE TABLE (WITH CONSTRAINTS AT COLUMN LEVEL)

- Creating a table **with constraints**.
- Constraints at **column level**.

```
CREATE TABLE tableName
(columnName dataType [CONSTRAINT constraintName]
constraintType [DEFAULT value]
[, column2Name datatype [CONSTRAINT constraintName]
constraintType [ DEFAULT value]]);
```

[ ] : optional

CREATE TABLE (WITH CONSTRAINTS AT TABLE LEVEL)

- Creating a table **with constraints**.
- Constraints at **table level**.

```
CREATE TABLE tableName
(columnName dataType [DEFAULT value]
[, column2Name datatype[ DEFAULT value]]
[, CONSTRAINT constraintName constraintType
(columnName,…)]);
```

[ ] : optional

PRIMARY KEY Constraint

- A **PRIMARY KEY** constraint creates a primary key for the table
- Only **one** Primary Key can be created for each table
- The PRIMARY KEY constraint is **a column** or **a set of columns** that uniquely identifies each row in a table
- **No column** that is part of the primary key can contain a **NULL value**
- A **composite primary key** must be created at the **table level**

Example (CREATE TABLE) with constraints

- **Column-level constraint**

```
CREATE TABLE Department (
        deptNo INT(5) CONSTRAINT dept_deptNo_pk PRIMARY KEY,
        deptName VARCHAR(20) NOT NULL,
        address VARCHAR(30),
        city INT(15)
);
```

- **Table-level constraint**

```
CREATE TABLE Department (
        deptNo INT(5),
        deptName VARCHAR(20) NOT NULL,
        address VARCHAR(30),
        city INT(15),
        CONSTRAINT dept_deptNo_pk PRIMARY KEY (dept_No)
);
```

UNIQUE Constraint

- A **UNIQUE** key integrity constraint requires that every value in a column or a set of columns be unique
- If the **UNIQUE** constraint has more than one column, that group of columns is called a **composite unique key**
- UNIQUE constraints **enable** the input of NULLs
- A **NULL** in a column (or in all columns of a composite UNIQUE key) always satisfies a UNIQUE constraint

CHECK Constraint

- It defines a condition that each row must satisfy
- It cannot reference columns from other tables

FOREIGN KEY Constraint

- The **FOREIGN KEY** (or referential integrity) constraint designates **a column** or **a combination of columns** as a foreign key
- Establishes a relationship with a primary key in the same table or a different table
- Here are the guidelines for foreign key constraints:
  - A foreign key value must match an existing value in the parent/base table or be **NULL**
  - Foreign keys are based on data values and are purely logical, rather than physical, pointers

FOREIGN KEY Constraint: Keywords

Keyword	Description
FOREIGN KEY	Defines the column in the child table at the table-constraint level
REFERENCES	Identifies the table and column in the parent table
ON DELETE CASCASDE	Deletes the dependent rows in the child table when a row in the parent/base table is deleted
ON DELETE SET NULL	Converts dependent foreign key values to null

Example (CREATE TABLE) with constraints NOT NULL Constraints

- **Column-level constraint**

```
CREATE TABLE Employees (
        empID INT PRIMARY KEY,
        fName VARCHAR(10) NOT NULL,
        lName VARCHAR(20) NOT NULL,
        email VARCHAR(30) CONSTRAINT emp_email_uk UNIQUE,
        salary DECIMAL(10,2) CHECK (salary >= 1700),
        deptNo INT(5) CONSTRAINT emp_fk_deptNo FOREIGN KEY (deptNo) REFERENCES
Department(deptNo)
);
```

- **Table-level constraint**

```
CREATE TABLE Employees (
        empID INT,
        fName VARCHAR(10) NOT NULL,
        lName VARCHAR(20) NOT NULL,
        email VARCHAR(30),
        deptNo INT(5),
        salary DECIMAL(10,2),
        CONSTRAINT emp_pk_empID PRIMARY KEY (empID),
        CONSTRAINT emp_email_uk UNIQUE (email),
        CONSTRAINT emp_chk_salary_min CHECK (salary >= 1700),
        CONSTRAINT emp_fk_deptNo FOREIGN KEY (deptNo) REFERENCES Department(deptNo)
);
```

NOT NULL Constraint can **ONLY** be defined at the column level

Altering Tables

- To change the structure of a table once it has been created.
- General Syntax:

```
ALTER TABLE tableName
ADD columnName dataType );
```

Add new column

```
ALTER TABLE tableName
ADD CONSTRAINT constraintName
constraintType(columnName);
```

Add new constraint

```
ALTER TABLE tableName
MODIFY columnName newdataType/size/defaultvalue;
```

Change data type, data size, default values, constraints

```
ALTER TABLE tableName
DROP COLUMN columnName;
```

Delete column

Altering Tables

- General syntax for altering tables

```
ALTER TABLE TableName
[ADD [COLUMN] columnName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption] [CHECK (searchCondition)]]
[DROP [COLUMN] columnName [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [ConstraintName]] tableConstraintDefinition]
[DROP CONSTRAINT ConstraintName [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT defaultOption]
[ALTER [COLUMN] DROP DEFAULT]
```

Source: Connolly, 2015

[ ] : optional

ALTER TABLE - Examples

- **Add** new column “Address” into Employee table.

```
ALTER TABLE Employee
ADD Address VARCHAR(40);
```

- **Change** datatype of “Address” to VARCHAR(20) and set the column as NOT NULL

```
ALTER TABLE Employee
MODIFY Address VARCHAR(20) NOT NULL;
```

- **Delete** the column “Address”

```
ALTER TABLE Employee
DROP COLUMN Address;
```

ALTER TABLE - Examples

- Adding constraint after a table has been created.
- Set the attribute of “EmployeeID” as the PRIMARY KEY for the Employee table.

```
ALTER TABLE Employee
ADD CONSTRAINT emp_pk_EmployeeID
PRIMARY KEY(EmployeeID);
```

Optional

Example (ALTER TABLE)

- If both tables are created without constraint.
- Constraints must be added using **ALTER TABLE**

Department

Attribute	Datatype
deptNo (Primary Key)	Integer (5)
deptName – requires values	Varchar(20)
address	Varchar(30)
city	Varchar(15)

Employees

Attribute	Datatype
empID (Primary Key)	Integer
fName – requires values	Varchar(20)
lName – requires values	Varchar(30)
email – unique	Varchar(30)
salary – must be more than 1,700	Decimal (10,2)
deptNo (Foreign key) – deptNo is an attribute in Department relation	Integer(5)

Examples (ALTER TABLE)

- For Department table, primary key and the NOT NULL constraint for deptName attribute

```
ALTER TABLE Department
ADD CONSTRAINT pk_Department PRIMARY KEY(deptNo)
MODIFY deptName VARCHAR(20) NOT NULL;
```

- For Employees table, primary key, NOT NULL constraints and foreign key to be added.

```
ALTER TABLE Employees
MODIFY fName VARCHAR(20) NOT NULL
MODIFY lName VARCHAR(30) NOT NULL
ADD CONSTRAINT pk_Employees PRIMARY KEY(empID)
ADD CONSTRAINT fk_EmpDept FOREIGN KEY (deptNo)
REFERENCES Department(deptNo);
```

Deleting Table

- General Syntax:

```
DROP TABLE tableName;
```

- Example: delete the table named “Employee”

```
DROP TABLE Employee;
```

To CREATE a Database

- To create a database, use the **CREATE DATABASE**

```
CREATE DATABASE database_name;
```

- To avoid errors if the database already exists, you can use the **IF NOT EXISTS** clause

```
CREATE DATABASE IF NOT EXISTS database_name;
```

To Use a specific Database

- To use a specific database, use the **USE** database_name

```
USE database_name;
```

To DELETE a Database

- To **DELETE** a database, use the **DROP DATABASE** database_name

```
DROP DATABASE database_name;
```

- To avoid errors if the database does not exists, you can use the **IF EXISTS** clause

```
DROP DATABASE IF EXISTS database_name;
```

- Important Notes:
  - **Permanent Deletion**: DROP DATABASE is **permanent** and **cannot be undone**. Make sure you have backed up any necessary data before running this command
  - **Use with caution**: In a production environment, be especially careful when using a DROP DATABASE to avoid accidental loss

VIEW TABLE STRUCTURE DESCRIBE

- Viewing table structures using **DESC** or **DESCRIBE**.
  - Syntax:
```
DESCRIBE tableName;
```
  - Example: View the table structure of Employee table.
```
DESCRIBE Employee;
```

- Rename a table using **RENAME**
  - Syntax:
```
RENAME tableName TO newtableName;
```
  - Example: Rename the Employee table to “Workers”
```
RENAME Employee TO Workers;
```

CREATE TABLE FROM EXISTING TABLES (using AS – does not copy constraints)

- Create table from existing database tables, using the **AS** clause and subqueries
- Syntax:

```
CREATE TABLE tableName [(columnName,…)]
AS (…subquery…);
```

- Example: create a new table based on the HR Employee table.

```
CREATE TABLE NewEmployee
AS (SELECT * FROM hr.Employees);
```

Subquery

CREATE TABLE FROM EXISTING TABLES (using LIKE – copy with constraints)

- Create table an empty table based on the definition of an existing table.
- Includes any column attributes and indexes defined in the original table. Use the **LIKE** clause Syntax:

```
CREATE TABLE tableName
LIKE existingTableName;
```

- Example: create a new table based on the HR Employee table.

```
CREATE TABLE NewEmployee
LIKE hr.Employees;
```

Exercise

- Using mySQL server, do following:
  1. Create the tables with constraint based on the following information:

Student

Attribute	Datatype	Constraints
studID	Varchar(10)	Primary key
name	Varchar(50)	Requires values
tel	Varchar(15)	
nationality	Varchar(30)	

CourseReg

Attribute	Datatype	Constraints
studID	Varchar(10)	Combination of both is the primary key
courseID	Varchar(10)	
RoomNo	Number with size of 4	Requires values

NOTE: studID in CourseReg is also a foreign key that refers to Student (studID)

Exercise

2. Add a new column called “**DOB**” with the datatype of DATE to the “Student” table
3. Delete column **roomNo** from “CourseReg” table.
4. Add new column called “**Availability**” that holds variable character data (with the size of 5) to CourseReg table and set the default value to ‘NA’

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Innovating Solutions
Menginovasi Penyelesaian