Mater Dei College
Tubigon, Bohol
2nd Semester, A.Y. 2022-2023
CC 105 – IM 101 (INFORMATION MANAGEMENT 1)

Topic 7
Database Design Phase 3- Implementation & Data Manipulation Language

Database Design Phase 3: Implementation

The implementation phase is where you install the DBMS on the required hardware, optimize the database to run best on that hardware and software platform, and create the database and load the data. The initial data could be either new data captured directly or existing data imported from a MariaDB database or another DBMS. You also establish database security in this phase and give the various users that you've identified access applicable to their requirements. Finally, you also initiate backup plans in this phase.

The following are steps in the implementation phase:

1. Install the DBMS.
2. Tune the setup variables according to the hardware, software and usage conditions.
3. Create the database and tables.
4. Load the data.
5. Set up the users and security.
6. Implement the backup regime.

Database Design Example Phase 3: Implementation

With the design complete, it's time to install MariaDB and run the CREATE statements, as follows:

```
CREATE DATABASE poets_circle;

CREATE TABLE poet (
  poet_code INT NOT NULL,
  first_name VARCHAR(30),
  surname VARCHAR(40),
  address VARCHAR(100),
  postcode VARCHAR(20),
  email VARCHAR(254),
  PRIMARY KEY(poet_code)
);


CREATE TABLE poem(
  poem_code INT NOT NULL,
  title VARCHAR(50),
  contents TEXT,
  poet_code INT NOT NULL,
  PRIMARY KEY(poem_code),
  INDEX(poet_code),
  FOREIGN KEY(poet_code) REFERENCES poet(poet_code)
);

CREATE TABLE publication(
  publication_code INT NOT NULL,
  title VARCHAR(100),
  price MEDIUMINT UNSIGNED,
  PRIMARY KEY(publication_code)
);


CREATE TABLE poem_publication(
  poem_code INT NOT NULL,
  publication_code INT NOT NULL,
  PRIMARY KEY(poem_code, publication_code),
  INDEX(publication_code),
  FOREIGN KEY(poem_code) REFERENCES poem(poem_code),
  FOREIGN KEY(publication_code) REFERENCES publication(publication_code)
);
```

```
CREATE TABLE sales_publication(
  sales_code INT NOT NULL,
  publication_code INT NOT NULL,
  PRIMARY KEY(sales_code, publication_code)
);


CREATE TABLE customer(
  customer_code INT NOT NULL,
  first_name VARCHAR(30),
  surname VARCHAR(40),
  address VARCHAR(100),
  postcode VARCHAR(20),
  email VARCHAR(254),
  PRIMARY KEY(customer_code)
);


CREATE TABLE sale(
  sale_code INT NOT NULL,
  sale_date DATE,
  amount INT UNSIGNED,
  customer_code INT NOT NULL,
  PRIMARY KEY(sale_code),
  INDEX(customer_code),
  FOREIGN KEY(customer_code) REFERENCES customer(customer_code)
);
```

SOURCES: https://mariadb.com/kb/en/database-design-phase-3-implementation/
https://mariadb.com/kb/en/database-design-example-phase-3-implementation/

DML

Data Manipulating Language (DML) Statements – allow you to modify the contents of the tables. There are three DML statements:

- INSERT – allows you to add tuple to a table
- UPDATE – allows you to change a tuple
- DELETE – allows you to remove tuples.

The SQL INSERT INTO Statement
The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax
It is possible to write the INSERT INTO statement in two ways.
The first way specifies both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

Demo Database
Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

INSERT INTO Example
The following SQL statement inserts a new record in the "Customers" table:

Example
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

Insert Data Only in Specified Columns
It is also possible to only insert data in specific columns.
The following SQL statement will insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

Example

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

The SQL UPDATE Statement
The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

Example

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

UPDATE Multiple Records
It is the WHERE clause that determines how many records will be updated.
The following SQL statement will update the contactname to "Juan" for all records where country is "Mexico":

Example

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

Update Warning!
Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

Example

```
UPDATE Customers
SET ContactName='Juan';
```

The SQL DELETE Statement
The DELETE statement is used to delete existing records in a table.

DELETE Syntax

```
DELETE FROM table_name WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

SQL DELETE Example
The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

Example

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

Delete All Records
It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name;
```

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

Example

```
DELETE FROM Customers;
```

SOURCE: https://www.w3schools.com/sql/sql_delete.asp