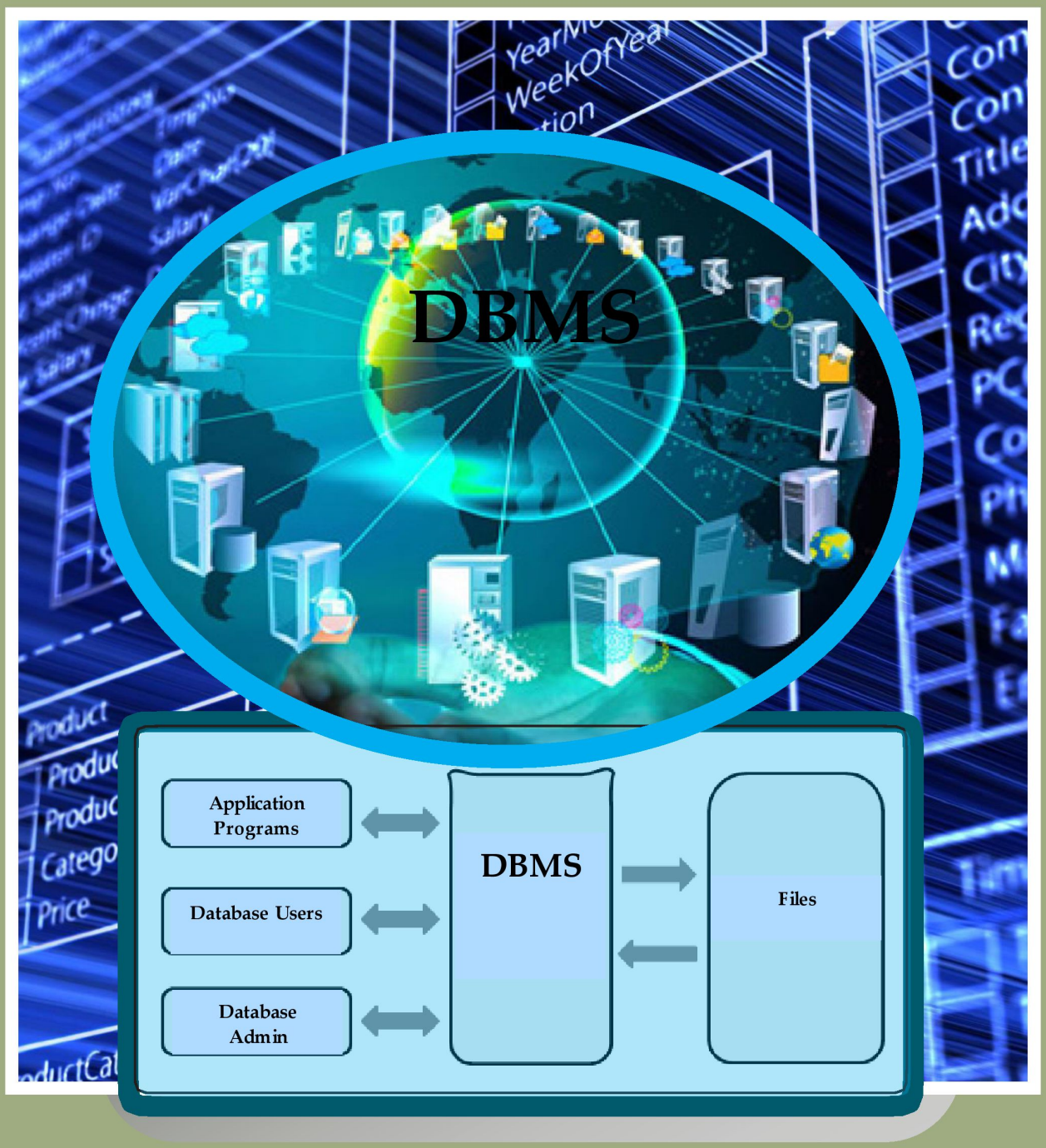
Indira Gandhi National Open University
School of Computer and Information Sciences (SOCIS)

BCS-092 Introduction of Databases



Structured Query Language And Database Development

3

Indira Gandhi
National Open University
School of Computer and
Information Sciences

BCS-092
Introduction to Databases

Block

3

STRUCTURED QUERY LANGUAGE AND DATABASE DEVELOPMENT

Course Coordinator (for the year 2021)

Mr. Akshay Kumar, Associate Professor
SOCIS, IGNOU, New Delhi-110068

In all the Units of this Block, Check Your Progress and Answers to Check Your Progress Questions are written and added in 2021 BY

Mr. Akshay Kumar, Associate Professor
SOCIS, IGNOU, New Delhi-110068

PRODUCTION

Mr. Tilak Raj
Asst. Registrar (Pub.)
MPDD, IGNOU, New Delhi

Mr. Yashpal
Section Officer (Pub.)
MPDD, IGNOU, New Delhi

January, 2021

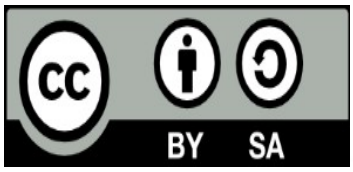
Laser Typesetting : Akashdeep Printers, 20-Ansari Road, Daryaganj, New Delhi-110002

IGNOU is one of the participants for the development of courses of this Programme

Copyright

This course has been developed as part of the collaborative advanced ICT course development project of the Commonwealth of Learning (COL). COL is an intergovernmental organization created by Commonwealth Heads of Government to promote the development and sharing of open learning and distance education knowledge, resources and technologies.

The Open University of Tanzania (OUT) is a fully fledged, autonomous and accredited public University. It offers its certificate, diploma, degree and postgraduate courses through the open and distance learning system which includes various means of communication such as face-to-face, broadcasting, telecasting, correspondence, seminars, e-learning as well as a blended mode. The OUT's academic programmes are quality-assured and as centrally regulated by the Tanzania Commission for Universities (TCU).



The Open University of Tanzania
P. O. Box 23409,
Kinondoni,
Dar Es Salaam,
Tanzania
Phone: +255 22 2668992
Fax: +255 22 2668756
Email: hod.ict@out.ac.tz
Website: www.out.ac.tz

Commonwealth of Learning
4710 Kingsway, Suite 2500,
Burnaby V5H 4M2,
British Columbia,
Canada
Phone: +1 604 775 8200
Fax: +1 604 775 8210
Email: info@col.org
Website: www.col.org

ACKNOWLEDGEMENTS By AUTHORS

The Open University of Tanzania (OUT), Faculty of Science, Technology and Environmental Studies, (FSTES) and the ICT Department wish to thank those below for their contribution to the production of this course material and video lectures:

Authors:
Ms. Grace W. Mbwete (Assistant Lecturer, ICT Department, OUT)
Mr. Elia A. Lukwaro (Tutorial Assistant, ICT Department, OUT)

Copy Editor:
Mr. Justin Kimaro (Principal Editor, IEMT, OUT)

Reviewers:
Ms. Lilian C. Mutalemwa (Assistant Lecturer, ICT Department, OUT)
Ms. Regina Monyemangene (Instructional Designer, IEMT, OUT)

Video Production:
Mr. Othman Mwinchoum (Multimedia studio, IEMT, OUT)
Ms. Grace W. Mbwete (Assistant Lecturer, ICT Department, OUT)
Ms. Lilian C. Mutalemwa (Assistant Lecturer, ICT Department, OUT)

This course has been developed with the support of the ***Commonwealth of Learning, Canada.***

BLOCK INTRODUCTION

You have already learnt about the basic concepts of database system in Block 1, including the database model. Block2 focused on database design and concepts like Entity-relationship model, and normalization. This block addresses one important concept of creating the database using SQL and inserting and manipulating data in the database system. Structured Query Language (SQL) is one of the standard languages across DBMSs for creating database, manipulating data in a database and retrieval if data from the database systems. This Block also discusses the data development process.

Unit 11 introduces the data definition language of SQL. It also explains the SQL commands for creating database constraints.

Unit 12 explains the commands for data manipulation and data retrieval.

Unit 13 discusses the join operation on two tables. Join is an important operation for a normalized database.

Unit14 introduces the overall process of database development.

You must perform all the activities listed in this Block to appreciate the process of database creation, manipulation and information retrieval.

ignou
THE PEOPLE'S
UNIVERSITY

UNIT 11 INTRODUCTION TO SQL

Structure

11.1 INTRODUCTION

Structured Query Language (SQL) is the main data definition language used for the creation and maintenance of databases. In this unit, we will learn the basic SQL syntax, including some data definition commands, how to define different types of constraints and different user defined data types in the application environment of databases.

11.2 OBJECTIVES

Upon completion of this unit you will be able to:

- *Describe* different SQL - DDL commands.
- *Identify* where and when to use the appropriate SQL DDL commands.
- *Create* database by using SQL DDL commands.
- *Create* set of database tables using DDL commands.
- *Describe* the various types of integrity constraints and how they are used.

11.3 TERMINOLOGIES

SQL : Database language designed for managing data held in a relational database management system.

DDL : Data Definition Language; standard for commands that define the different structures in a database

DML : Set of syntax elements used for selecting, inserting, deleting and updating data in a database.

Constraints : Rules enforced on data columns on table

11.4 INTRODUCTION TO SQL

Structured Query Language (SQL) is a database language designed for managing data held in a relational database management system. SQL was initially developed by IBM in the early 1970s . The initial version, called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's quasi-relational database management system, System R. Then in the late 1970s, Relational Software Inc., which is now Oracle Corporation, introduced the first commercially available implementation of SQL, Oracle V2 for VAX computers.

Many of the currently available relational DBMSs, such as Oracle Database, Microsoft SQL Server (shown in Figure 11.1), MySQL, IBM DB2, IBM Informix and Microsoft Access, use SQL.

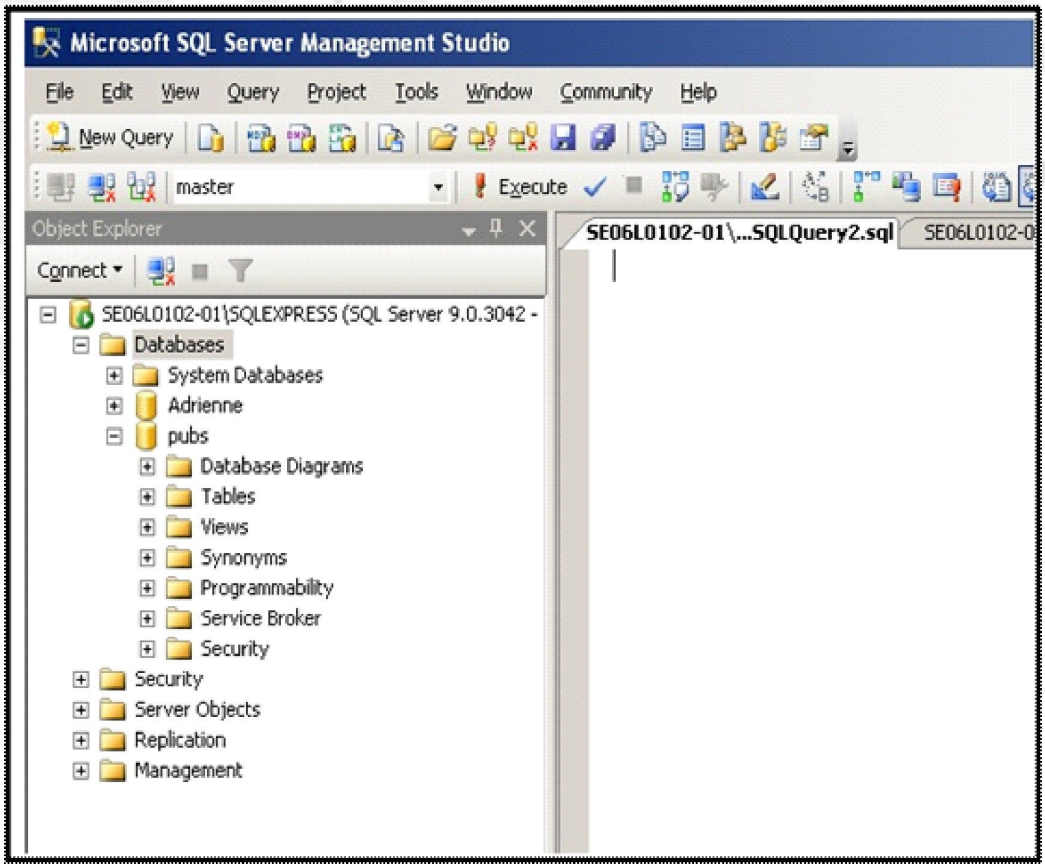


Figure 11.1: Example of Microsoft SQL Server, by A. Watt.

11.5 DATADEFINITION LANGUAGE (DDL) COMMANDS

In a DBMS, the SQL database language is used to:

- Create the database and table structures
- Perform basic data management chores (add, delete and modify)
- Perform complex queries to transform raw data into useful information

In this unit, we will focus on using SQL to create the database and table structures, mainly using SQL as a data definition language (DDL). In the following unit; unit 12 we will use SQL as a data manipulation language (DML) to insert, delete, select and update data within the database tables.

11.5.1 Create Database Command

The major SQL DDL statements are CREATE DATABASE and CREATE/DROP/ ALTER TABLE. The SQL statement CREATE is used to create the database and table structures. The SQL syntax for creating a database is as follows:-

```
CREATE DATABASE dbname;
```

Example: CREATE DATABASE university

A new database named 'university' is created by the SQL statement;

```
CREATE DATABASE university;
```

In order for your database to be created within a RDBMS; you must adhere to the following conditions, which are:

- Making sure you have admin privilege before creating any database.
- The SQL command (CREATE DATABASE) is in capital letters and the command ends with a semicolon sign (;).
- Ensuring that the database name should be unique within the RDBMS.

After creating the database 'university'; you can now check your created database in the list of databases by typing the following SQL statement;

```
SHOW DATABASES;
```

The result of this SQL statement should list down the databases that have been created with 'university' database which has just been crated; as follows:-

Database
Shop
Customer
university

```
3 rows in set (0.00 sec)
```

11. 5.2 Create Table Command

A new database named 'university' is created by the SQL statement CREATE DATABASE university. Once the database is created, the next step is to create the database tables. The general format for the CREATE TABLE command is:

```
CREATE TABLE <tablename>
(
ColumnName, Datatype, Optional Column Constraint,
ColumnName, Datatype, Optional Column Constraint,
Optional table Constraints
);
```

Tablename is the name of the database table such as Employee. Each field in the CREATE TABLE has three parts (see above):

- ColumnName
- Data type
- Optional Column Constraint

i) ColumnName

The ColumnName must be unique within the table. Some examples of ColumnNames are FirstName and LastName.

ii) Data Type

The data type, as described below, must be a system data type or a user-defined data type. Many of the data types have a size such as CHAR(35) or Numeric(8,2).

1. **Bit** - Integer data with either a 1 or 0 value
2. **Int** - Integer (whole number) data from -2^{31} (-2,147,483,648) through 2^{31} - 1 (2,147,483,647)
3. **Smallint** - Integer data from 2^{15} (-32,768) through 2^{15} - 1 (32,767)
4. **Tinyint** - Integer data from 0 through 255
5. **Decimal** - Fixed precision and scale numeric data from -10^{38} -1 through 10^{38}
6. **Numeric** - A synonym for decimal
7. **Timestamp** - Adatabase-wide unique number
8. **Uniqueidentifier** - A globally unique identifier (GUID)
9. **Money** - Monetary data values from -2^{63} (-922,337,203,685,477.5808) through 2^{63} - 1 (+922,337,203,685,477.5807), with accuracy to one-ten-thousandth of a monetary unit
10. **Smallmoney** - Monetary data values from -214,748.3648 through +214,748.3647, with accuracy to one-ten-thousandth of a monetary unit
11. **Float** - Floating precision number data from -1.79E + 308 through 1.79E + 308

12. **Real** - Floating precision number data from -3.40E + 38 through 3.40E + 38
13. **Datetime** - Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of one-three-hundredths of a second, or 3.33 milliseconds
14. **Smalldatetime** - Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute
15. **Char** - Fixed-length non-Unicode character data with a maximum length of 8,000 characters
16. **Varchar** - Variable-length non-Unicode data with a maximum of 8,000 characters
17. **Text** - Variable-length non-Unicode data with a maximum length of 2^{31} - 1 (2,147,483,647) characters
18. **Binary** - Fixed-length binary data with a maximum length of 8,000 bytes
19. **Varbinary** - Variable-length binary data with a maximum length of 8,000 bytes
20. **Image** - Variable-length binary data with a maximum length of 2^{31} - 1 (2,147,483,647) bytes

iii) Optional Column Constraints

The Optional Column Constraints are NULL, NOT NULL, UNIQUE, PRIMARY KEY and DEFAULT, used to initialize a value for a new record. The column constraint NULL indicates that null values are allowed, which means that a row can be created without a value for this column. The column constraint NOT NULL indicates that a value must be supplied when a new row is created.

To illustrate, we will use the SQL statement CREATE TABLE Students to create the students table with 6 attributes or fields.

```
USE university
CREATE TABLE Students
(
StudentRegNo CHAR(10) NOT NULL UNIQUE,
StudentFName CHAR(30)NOT NULL,
StudentMName CHAR(25)NOT NULL,
StudentLName CHAR(25)NOT NULL,
YearEnrolledDATE NOT NULL,
BirthDate DATE NOT NULL,
CONSTRAINT students_PK PRIMARY KEY(StudentRegNo)
);
```

The first field is 'Student Reg No' with a field type of CHAR. For this field, the field length is 10 characters, and the user cannot leave this field empty (NOT NULL).

Similarly, the second field is StudentFName with a field type CHAR of length 30. After all the table columns are defined, a table constraint, identified by the word CONSTRAINT, is used to create the primary key:

```
CONSTRAINT StudentsPK PRIMARY KEY(StudentRegNo)
```

We will discuss the constraint property further later in this unit. Likewise, we can create two more tables a 'course' table and an 'AcademicProgram' table using the CREATE TABLE DDL command of SQL as shown in the below example.

```
USE university
CREATE TABLE COURSE
(
CourseTitle Char(35)  NOT NULL,
CourseCode Char(30) NOT NULL,
CourseUnitsInt  NOT NULL,
CONSTRAINT COURSE_PK PRIMARY KEY(CourseCode)
);
```

In this example, an 'AcademicProgram' table is created with three fields: ProgramID, ProgramName and CreditPoints.

```
USE university
CREATE TABLE AcademicPrograms
(
ProgramID Int  NOT NULL IDENTITY,
ProgramName Char(50) NOT NULL,
CreditPoints Int  NOT NULL,
CONSTRAINT AcademicPrograms_PK PRIMARY KEY(ProgramID)
);
```

You can use ALTER TABLE statements to add and drop constraints. ALTER TABLE allows columns to be removed.

When a constraint is added, all existing data are verified for violations.

```
ALTER TABLE table_name
ADD column_name datatype
```

In the example below, we will use the ALTER TABLE statement to the alter table AcademicProgram by adding another column/field called 'YearCommenced".

```
USE university
GO
ALTER TABLE  AcademicProgram
ADD YearCommencedDateTime NOT NULL,
```

The above SQL ALTER TABLE example will add a column called *YearCommenced* to the *AcademicProgram* table.

We can also use alter table use the ALTER TABLE statement to modify a column in a table by using the following SQL statement;

```
ALTER TABLE table_name

MODIFY  column_namecolumn_type
```

Look at the following example below where we will modify the column ProgramName by changing its data type.

```
ALTER TABLE AcademicProgram

MODIFY ProgramName VARCHAR(100) NOT NULL;
```

You can confirm these changes by using SQL statement DESC table_name to verify the data structure changes.

```
DESC AcademicProgram;
```

11.5.3 DROP TABLE

The DROP TABLE will remove a table from the database. Make sure you have the correct database selected.

```
DROP TABLE table_name
```

For example; if we want to drop/delete table AcademicProgram, we write the SQL statement as follows;

```
DROP TABLE AcademicProgram
```

Executing the above SQL DROP TABLE statement will remove the table AcademicProgram from the database university.

Check Your Progress 1

Q1: Why is learning SQL important for DBMS?

.....

.....

.....

.....

Q2: Define the following relation using SQL in a Database named Bank.

Account Holder (Customer id, Account no, date of account opening, balance)

.....

.....

.....

.....

Q3: What are the commands for the following?

(a) Dropping a table

(b) Displaying table structure

.....

.....

.....

.....

11.6 TABLE CONSTRAINTS

Table constraints are identified by the CONSTRAINT keyword and can be used to implement various constraints described below.

11.6.1 IDENTITY constraint

We can use the optional column constraint IDENTITY to provide a unique, incremental value for that column. Identity columns are often used with the PRIMARY KEY constraints to serve as the unique row identifier for the table. The IDENTITY property can be assigned to a column with a tinyint, smallint, int, decimal or numeric data type. This constraint:

- Generates sequential numbers
- Does not enforce entity integrity
- Only one column can have the IDENTITY property
- Must be defined as an integer, numeric or decimal data type
- Cannot update a column with the IDENTITY property
- Cannot contain NULL values
- Cannot bind defaults and default constraints to the column

For IDENTITY [(seed, increment)]

- Seed - the initial value of the identity column
- Increment - the value to add to the last increment column

We will use another database example to further illustrate the SQL DDL statements by creating the table tblHotel in this HOTEL database.

```
CREATE TABLE  tblHotel
(
HotelNo Int IDENTITY (1,1),
Name Char(50) NOT NULL,
Address Char(50) NULL,
City Char(25) NULL,
);
```

11.6.2 UNIQUE constraint

The UNIQUE constraint prevents duplicate values from being entered into a column.

- Both PK and UNIQUE constraints are used to enforce entity integrity.
- Multiple UNIQUE constraints can be defined for a table.
- When a UNIQUE constraint is added to an existing table, the existing data is always validated.
- A UNIQUE constraint can be placed on columns that accept nulls. Only one row can be NULL.
- A UNIQUE constraint automatically creates a unique index on the selected column.

This is the general syntax for the UNIQUE constraint:

```
[CONSTRAINT  constraint_name]
UNIQUE  [CLUSTERED  |  NONCLUSTERED]
(col_name  [,  col_name2  […,  col_name16]])
[ON  segment_name]
```

This is an examle using the UNIQUE constraint.

```
CREATE  TABLE  Students
(
StudentRegNo   CHAR(10)    NOT  NULL  UNIQUE,
)
```

11.6.3 FOREIGN KEY Constraint

The FOREIGN KEY (FK) constraint defines a column, or combination of columns, whose values match the PRIMARY KEY (PK) of another table.

- Values in an FK are automatically updated when the PK values in the associated table are updated/changed.
- FK constraints must reference PK or the UNIQUE constraint of another table.
- The number of columns for FK must be same as PK or UNIQUE constraint.
- If the WITH NOCHECK option is used, the FK constraint will not validate existing data in a table.
- No index is created on the columns that participate in an FK constraint.

This is the general syntax for the FOREIGN KEY constraint:

```
[CONSTRAINT  constraint_name]
[FOREIGN KEY (col_name [, col_name2 […, col_name16]])]
REFERENCES [owner.]ref_table [(ref_col [, ref_col2
[…,  ref_col16]])]
```

In this example, the field HotelNo in the tblRoom table is a FK to the field HotelNo in the tblHotel table shown previously.

```
USE HOTEL
GO
CREATE TABLE  tblRoom
(
HotelNo Int NOT NULL ,
RoomNo  Int NOT NULL,
Type  Char(50)NULL,
Price  Money NULL,
PRIMARY KEY (HotelNo, RoomNo),
FOREIGN KEY (HotelNo) REFERENCES tblHotel
)
```

11.6.4 CHECK constraint

The CHECK constraint restricts values that can be entered into a table.

- It can contain search conditions similar to a WHERE clause.
- It can reference columns in the same table.
- The data validation rule for a CHECK constraint must evaluate to a Boolean expression.
- It can be defined for a column that has a rule bound to it.

This is the general syntax for the CHECK constraint:

```
[CONSTRAINT constraint_name]
CHECK [NOT FOR REPLICATION] (expression)
```

In this example, the Type field is restricted to have only the types 'Single', 'Double', 'Suite' or 'Executive'.

```
USE HOTEL
GO
CREATE TABLE  tblRoom
(
HotelNo Int NOT NULL,
RoomNo Int NOT NULL,
Type Char(50) NULL,
Price Money NULL,
PRIMARY KEY (HotelNo, RoomNo),
```

```
FOREIGN KEY (HotelNo) REFERENCES tblHotel
CONSTRAINT Valid_Type
CHECK (Type IN ('Single', 'Double', 'Suite', 'Executive'))
)
```

In this second example, the employee hire date should be before January 1, 2004, or have a salary limit of $300,000.

```
GO
CREATE TABLE SALESREPS
(
Empl_num Int Not Null
CHECK (Empl_num BETWEEN 101 and 199),
Name Char (15),
Age Int CHECK (Age >= 21),
Quota Money CHECK (Quota >= 0.0),
HireDate DateTime,
CONSTRAINT QuotaCap CHECK ((HireDate< "01-01-2004")
OR (Quota <=300000))
);
```

11.6.5 DEFAULT constraint

The DEFAULT constraint is used to supply a value that is automatically added for a column if the user does not supply one.

- A column can have only one DEFAULT.
- The DEFAULT constraint cannot be used on columns with a timestamp data type or identity property.
- DEFAULT constraints are automatically bound to a column when they are created.

The general syntax for the DEFAULT constraint is:

```
[CONSTRAINT constraint_name]
DEFAULT {constant_expression | niladic-function | NULL}
[FOR col_name]
This example sets the default for the city field
to 'Vancouver'.
USE HOTEL
ALTER TABLE tblHotel
Add CONSTRAINT df_city DEFAULT 'Vancouver' FOR City
```

11.7 USER DEFINED TYPES

User defined types are always based on system-supplied data type. They can enforce data integrity and they allow nulls.

To create a user-defined data type in SQL Server, choose types under "Programmability" in your database. Next, right click and choose 'New' ->'User-defined data type' or execute the sp_addtype system stored procedure. After this, type:

```
sp_add type ssn, 'varchar(11)', 'NOT NULL'
```

This will add a new user-defined data type called SIN with nine characters. In this example, the field EmployeeSIN uses the user-defined data type SIN.

```
CREATE TABLE SINTable
(
EmployeeID INT Primary Key,
EmployeeSIN SIN,
CONSTRAINT CheckSIN
CHECK (EmployeeSIN LIKE
'[0-9][0-9][0-9] - [0-9][0-9] [0-9] - [0-9][0-9][0-9]')
);
```

Check Your Progress 2

Q1: You want to assign successive odd numbers to a field named security no (starting form 1).

Write SQL Command to do so.

.....

.....

.....

Q2: Create the following table with UNIQUE, PRIMARY KEY and FOREIGN KEY constraints:

STUDENT (Student id, Name, aadhar number, Programme code)

Where Student id is Primary key

Aadhar number is an alternate key, and

Programme code is a foreign key to a table named PROGRAMME.

.....

.....

.....

Q3: How will you check that Marital Status field is contains the value 'SINGLE' or 'MARRIED' and salary is between 15,000 and 2,00,000.

.....

.....

.....

11.8 VIDEO LECTURE

Introduction to SQL - https://tinyurl.com/zlh99lo

DDL Commands - https://tinyurl.com/j992obd

CREATE TABLE Command and Table Constraints: https://tinyurl.com/jegmsh3

INSERT COMMAND: https://tinyurl.com/zn4ko8m

11.9 ACTIVITY

Activity 11.1

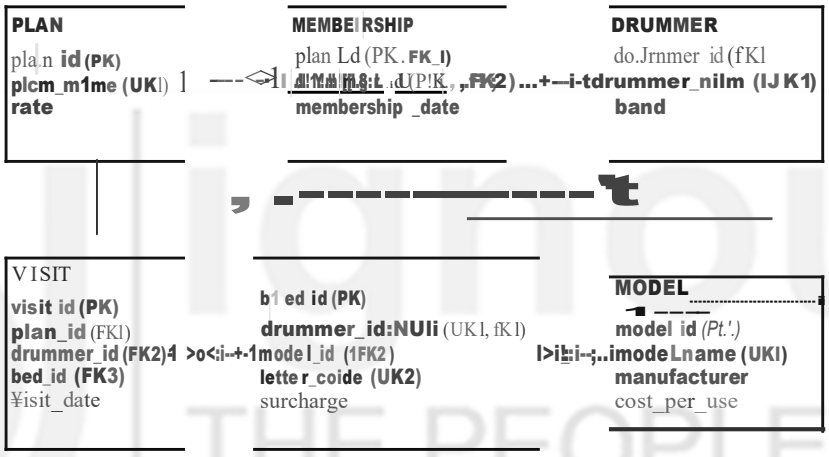
Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQLDBMS or its equivalent.

What to do:

Create a database called 'Emporium' with the attached structure below in a ERD and populate it with tanning-related data. UNIQUE constraints were used to guarantee unique values for the identifiers from the entity-relationship model. The BED table used an additional UNIQUE constraint on the drummer id column to make sure that more than one bed can't be named for the same drummer. All currency columns used the FLOAT datatype.



Duration: Expect to spend about 2 hours on this activity.

Feedback: This activity should be submitted to the course Instructor for an assessment and feedback.

Activity 11.2

Introduction to SQL Exercises

Aim: To become conversant with basic SQLDDL operations.

Resources: Unit 11 Learning materials and MySQL DBMS or its equivalent.

What to do:

Populate the table (MODEL) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

MODEL table

model_id	model_name	manufacturer	cost_per_use
1	Super Crisp	CrispTanCorp	10
2	Super Crisp Deluxe	CrispTanCorp	12
3	Roentgen Radiator	CrispTanCorp	14.5

Duration: Expect to spend about 1 hour on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 11.3

Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQL DBMS or its equivalent.

What to do:

Populate the table (DRUMMER) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

DRUMMER table

drummer_id	drummer_name	band
1	Keith Moon	Who, The
2	John Bonham	Led Zepplin
3	Charlie Watts	Rolling Stones, The

Duration: Expect to spend about 2 hours on this activity.

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 11.4

Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQLDBMS or its equivalent.

What to do:

Populate the table (PLAN) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

PLAN Table

plan_id	plan_name	rate
1	Walk-In	25
2	Lobster	15
3	Bronze	10

Duration: Expect to spend about 30 minutes on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 11.5

Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQL DBMS or its equivalent.

What to do:

Populate the table (MEMBERSHIP) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

MEMBERSHIP table

plan_id	drummer_id	membership_date
1	**1**	1976-05-02
2	**2**	1976-05-14
1	**3**	1976-05-28
3	**1**	1976-06-15

Duration: Expect to spend about 2 hours on this activity.

Feedback: This activity should be submitted to the course Instructor for a feedback.

Activity 11.6

Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQLDBMS or its equivalent.

What to do:

Populate the table (BED) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

BED table

bed_id	drummer_id	model_id	letter_code	surchage
1	NULL	1	A	0
2	3	2	B	1.25
3	NULL	1	C	0

Duration: Expect to spend about 2 hours on this activity.

Feedback: This activity should be submitted to the course Instructor for a feedback.

Activity 11.7

Introduction to SQL Exercises

Aim: To become conversant with basic SQL operations.

Resources: Unit 11 Learning materials and MySQLDBMS or its equivalent.

What to do:

Populate the table (BED) with data as detailed below. Please pay attention to the order in which your tables are populated: primary key columns must be populated before dependent foreign key columns.

VISIT table

visit_id	plan_id	drummer_id	bed_id	visit_date
1	1	1	2	1976-05-02
2	1	1	3	1976-05-10
3	2	2	1	1976-05-14
4	1	3	2	1976-05-28
5	1	3	2	1976-06-15
6	3	1	1	1976-06-15

Duration: Expect to spend about 1 hour on this activity

Feedback: This activity should be submitted to the course Instructor for a feedback.

11.10 SUMMARY

In this unit you learned about SQL which is divided into two main categories; DDL and DML. These categories show how to create a database and tables in that particular database. For this unit, the emphasis was on data definition language by using SQL where we had a chance of creating database and tables; within their respective data types and required constraints. In summary, this unit has covered how to define structure of a database and its associated tables and how to manipulate the data within.

11.11 ANSWERS

Check Your Progress 1

Ans 1: SQL is a standard language across DBMS, and is used for defining data, manipulating data and retrieving data as per the need of the user.

Ans 2:
```
CREATE  DATABASE Bank;

USE Bank

CREATE TABLE AccountHolder (

Cust_id CHAR (20) NOT NULL UNIQUE,

Account_no CHAR (20) Not NULL,

Dataofaccountopening Date time NOT NULL,

Balance Money NOT NULL,

CONSTRAINT AcHolder_PK PRIMARY KEY (Cust_id, Account_no)

);
```

Ans 3: (a) DROP TABLE tablename;

(b) DESC table name;

Check Your Progress 2

Ans 1:
```
CREATE TABLE test

(

    Securityno Int IDENTITY (1, 2),

    Name CHAR(50)

)
```

Ans 2: You may use the following command provided. You have executed SQL command to use the appropriate databases and created the PROGRAMME table with a field name Programme-code which has same data type as of STUDENT table.

```
CREATE TABLE STUDENT
(
id CHAR(10) PRIMARY KEY,
name CHAR(50) NOT NULL,
Aadharno CHAR(12) UNIQUE,
Programme-code CHAR(10),
FOREIGN KEY(Programme-code) REFERENCES PROGRAMME
```

Ans 3:

```
...
maritalstatus CHAR(10),
Salary Int,
CONSTRAINT Valid-maritial-status
CHECK(maritalstatus IN ('SINGLE','MARRIED'))
CONSTRAINT salary-range CHECK ((salary>15,000) AND
(salary<2,00,000))
…
```

11.12 REVIEW QUESTIONS FROM AUTHORS

Aim : These questions are aimed at assessing student knowledge on SQL operation skills

Duration : Expect to spend 2hours for this activity.

Task 1

Create the database named 'Organization' then a table 'Employee' with the following attributes and its corresponding data declaration:-

ATTRIBUTE (FIELD) NAME	DATA DECLARATION
EMP_NUM	CHAR(3)
EMP_LNAME	VARCHAR(15)
EMP_FNAME	VARCHAR(15)
EMP_INITIAL	CHAR(1)
EMP_HIREDATE	DATE
JOB_CODE	CHAR(3)

Task 2

Populate the table created in task 1 with the following data.

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-96	500
104	Ramoras	Anne	K	15-Nov-87	501
105	Johnson	Alice	K	01-Feb-93	502
106	Smithfield	William		22-Jun-04	500
107	Alonzo	Maria	D	10-Oct-93	500
108	Washington	Ralph	B	22-Aug-91	501
109	Smith	Larry	W	18-Jul-97	501

Task 3

Write the SQL code to change the job code to 501 for the person whose personnel number is 107. After you have completed the task, examine the results, and then reset the job code to its original value.

Task 4

Assuming that the data shown in the Employee table have been entered, write the SQL code that lists all attributes for a job code of 502.

Task 5

Write the SQL code to delete the row for the person named William Smithfield, who was hired on June 22, 2004, and whose job code classification is 500. (Hint: Use logical operators to include all the information given in this problem.)

Task 6

Add the attributes EMP_PCT and PROJ_NUM to the Employee table. The EMP_PCT is the bonus percentage to be paid to each employee.

Task 7

Using a single command, write the SQL code that will enter the project number (PROJ_NUM) = 18 for all employees whose job classification (JOB_CODE) is 500.

Task 7

Using a single command, write the SQL code that will enter the project number (PROJ_NUM) = 25 for all employees whose job classification (JOB_CODE) is 502 or higher.

Task 8

Write the SQL code that will change the PROJ_NUM to 14 for those employees who were hired before January 1, 1994, and whose job code is at least 501. (You may assume that the table will be restored to its original condition preceding this question.)

11.13 FURTHER READING

1. Eng, N., & Watt, A. (2013). Database Design - 2nd Edition. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/sql-structured-query-language/

2. Intro to SQL: Querying and managing data. Retrieved September 12, 2016, from KhanAcademy, https://www.khanacademy.org/computing/computer-programming/sql?ref=resume_learning#sql-basics

3. Rogers, D. (2002). COMP210: Class notes. Retrieved May 15, 2016, from Yukon College, http://college.yukondude.com/2006_09_comp210/html/notes.php

11.14 ATTRIBUTION

The following material was written by Grace Mbwete:

1. Introduction
2. SQL Examples in creating Databases and column constraints.
3. Unit Summary

UNIT 12 SQL DATA MANIPULATION LANGUAGE

Structure

12.1 INTRODUCTION

In unit 11, you were introduced to Data Definition Language (DDL) which is a prerequisite for this unit. In this unit, you have an opportunity to learn about Data Manipulation Language (DML) whereby you will learn how to manipulate (query) a database by using different SQL. Please note that this unit is a practical based unit, where as you go along the content, you need to be hands on with a DBMS of your choice.

12.2 OBJECTIVES

Upon completion of this unit you should be able to:

- *Construct* basic queries using Distinct Clause.
- *Construct* basic queries using Where Clause.
- *Construct* basic queries using AND/OR operator.
- *Construct* basic queries using Aggregate Functions.

- *Construct* basic queries using Order By Clause.
- *Construct* basic queries using Special Operator.

12.3 TERMINOLOGIES

Distinct Clause : SQL operator which is used to return only distinct (different) values.

Where Clause : SQL operator which is used to filter records.

And/Or Operator : SQL operator which is used to filter records on one or more conditions.

SQLAggregate Function : SQL functions which perform a calculation on a set of values and return a single, or summary, value.

SQL Special Operator : SQL operators which filter records on special conditions.

12.4 DATA MANIPULATION LANGUAGE (DML)

The SQL Data Manipulation Language (DML) is used to query and modify database data. In this unit, we will describe how to use the SELECT, INSERT, UPDATE, and DELETE SQL DML command statements, defined below.

- *SELECT* - to query data in the database
- *INSERT* - to insert data into a table
- *UPDATE* - to update data in a table
- *DELETE* - to delete data from a table

In the SQL DML statement, please note that:

- Each clause in a statement should begin on a new line.
- The beginning of each clause should line up with the beginning of other clauses.
- If a clause has several parts, they should appear on separate lines and be indented under the start of the clause to show the relationship.
- Upper case letters are used to represent reserved words.
- Lower case letters are used to represent user-defined words.

12.5 INSERT STATEMENT

Before you can manipulate any data from a database, you need to populate that database with data. Consider the following example , where you are going to create a table called 'courses' and then insert data in it.

Note that, before creating the table you need to select particular database where your table will be created. You are going to use database named 'university' from previous unit and a table called courses which was created by using the following syntax. The following are the steps involved;

(i) Creating the table

```
USE university

CREATE TABLE courses

(

CourseCode VARCHAR(40) PRIMARY KEY,

CourseTitle VARCHAR(255),

CourseUnits INT NOT NULL,

CourseCredits INT NOT NULL

);
```

(ii) Confirming the structure of the table

In order to confirm the structure of your table, type the statement

```
DESC courses;
```

The following should be the output;

```
+----------------+-------------+------+-----+---------+------ +
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+------ +
| CourseCode     | varchar(40) | NO   | PRI | NULL    |       |
| CourseTitle    | varchar(255)| YES  |     | NULL    |       |
| CourseUnits    | int(11)     | NO   |     | NULL    |       |
| CourseCredits  | int(11)     | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+------ +
```

4 rows in set (0.06 sec)

(iii) Inserting Data into the table 'courses'

```
INSERT INTO courses VALUES ("OIT214", "OO Programming", 3, 30);

INSERT INTO courses VALUES (OIT215, "Data Communication", 3, 30);

INSERT INTO courses VALUES ("OIT216", "Introduction to E-Business", 2, 10);

INSERT INTO courses VALUES ("OIT217", "Database Design", 2, 10);

INSERT INTO courses VALUES ("OIT218", "Computer Security", 1, 5);

INSERT INTO courses VALUES ("OIT219", "Software Design", 1, 20);
```

Now, you have learned how to insert/populate data into our tables. Please note, you have to be careful, with SQL syntax while creating and pulating the database. One, spelling mistake, then the statement will not go through.

12.6 SELECT STATEMENT

The SELECT statement, or command, allows the user to extract data from tables in a selected database. Please, note the syntax of the SELECT statement below:

- In a circumstance that you only want to select few selected columns from that particular table, the following syntax applies.

```
SELECT  column_name,column_name
FROM  table_name;
```

- In a circumstance that you only want to select all columns from that particular table, the following syntax applies.

```
SELECT *
FROM  table_name;
```

Example 1:

Referring to the table 'courses' that you had inserted data into; you are going to have a look at what data the table holds after inserting the data in the previous section.

```
SELECT *
FROM courses;
```

The following should be the output;

CourseCode	CourseTitle	CourseUnits	CourseCredits
OIT219	Software Design	1	20
OIT214	OO Programming	3	30
OIT215	Data Communication	3	30
OIT216	Introduction to E-business	2	10
OIT217	Database Design	2	10
OIT218	Computer Security	1	5

Example 2:

If you want to select one or more column from a particular table; the following syntax holds;

```
SELECT  column_name,column_name
FROM  table_name;
```

Following, out table courses we had populated with data in the previous section;

```
SELECT  CourseTitle
FROM  courses;
```

The following should be the result;

```
+-----------------------------+
| CourseTitle                 |
+-----------------------------+
| Software Design             |
| OO Programming              |
| Data Communication          |
| Introduction to E-business  |
| Database Design             |
| Computer Security           |
+-----------------------------+
```

6 rows in set (0.00 sec)

The following sections, you are going to learn more conditions that are used with the SELECT statement in querying the database.

12.7 WHERE CLAUSE

In this section you are going to learn more on the SELECT statements by filtering the data in the database. The WHERE clause is mainly used to filter records. Its syntax is as follows:

```
SELECT  column_name,column_name
FROM  table_name
WHERE  column_name  operator  value;
```

Consider the following example below, where we will be using our university database and 'courses' tables from previous unit and section;

Example 1

If we want to know Course Title courses which have 2 units, the syntax will be as follows:

```
SELECT  CourseTitle  from  courses
WHERE  CourseUnits  =  2;
```

The following should be the output;

```
+-----------------------------+
| coursetitle                 |
+-----------------------------+
```

| Introduction to E-business |

| Database Design |

+----------------------------+

2 rows in set (0.00 sec)

Example 2

If you want know how many courses have 2 or more units, we would use the following syntax:-

```
SELECT CourseTitle from courses
   WHERE CourseUnits>= 2;
```

The following should be the output;

+----------------------------+

| coursetitle |

+----------------------------+

| OO Programming |

| Data Communication |

| Introduction to E-business |

| Database Design |

+----------------------------+

4 rows in set (0.00 sec)

Check Your Progress 1

Q1: What is the purpose of SELECT, UPDATE and DELETE commands in SQL?

.....

.....

.....

.....

Q2: Consider the courses table in university database (given in section 12.5)

How will you insert a record containing the following data into the courses table?

CourseCode	BCS091
CourseTitle	DBMS
CourseUnits	3
CourseCredits	30

What will happen if you try to enter CourseCode and CourseTitle only?

.....

.....

.....

.....

Q3: Using the same table as above, list those CourseCode and CourseTitle for which CourseCredits are less than or equal to 20.

.....

.....

.....

.....

12.8 SPECIAL OPERATORS

The WHERE clause can be used with the following operators in querying the database as illustrated in the examples above;

S/N	Operator	Description
1	BETWEEN	Between an inclusive range
2	LIKE	Search for a certain pattern
3	IN	Specifies multiple possible values for a field
4	<>	Not equal.
5	>	Greater than
6	<	Less than
7	>=	Greater than or equal
8	<=	Less than or equal

12.9 AND /OR OPERATORS

AND & OR operators are used with the WHERE clause filter records based on more than one condition. Consider the following example below:

Example 1: AND Operator

We are going to select all courses which have 3 unit and 30 credits from our table courses from previous section.

```
SELECT * FROM Courses
WHERE CourseUnits=3
AND CourseCredits=30;
```

The result should be as follows:

```
+---------------+-------------------- +-----------------+------------------+
| CourseCode    | CourseTitle         | CourseUnits     | CourseCredits    |
+---------------+-------------------- +-----------------+------------------+
| OIT214        | OO Programming      |               3 |               30 |
| OIT215        | Data Communication  |               3 |               30 |
+---------------+-------------------- +-----------------+------------------+
```

2 rows in set (0.00 sec)

Example 2:OR Operator

We are going to select all courses which have 3 unit and 30 credits from our table courses from previous section.

```
SELECT * FROM Courses
WHERE CourseUnits=2
OR CourseCredits=20;
```

The following should be the results;

```
+---------------+------------------------ +--------------- +--------------+
| CourseCode    | CourseTitle             | CourseUnits     | CourseCredits |
+---------------+------------------------ +--------------- +--------------+
| OIT 219       | Software Design         |               1 |           20 |
| OIT216        | Introduction to E-business |            2 |           10 |
| OIT217        | Database Design         |               2 |           10 |
+---------------+------------------------ +--------------- +--------------+
```

3 rows in set (0.00 sec)

For more querying of data, AND operator and OR operator can be combined.

12.10 SQL AGGREGATE FUNCTIONS

Aggregate functions perform a calculation on a set of values and return a single, or summary, value. Table 12.2 lists these functions.

Table 12.2: SQLAggregate Functions

S/N	Function	Description
1	AVG	Returns the average of all the values, or only the DISTINCT values, in the expression.
2	COUNT	Returns the number of non-null values in the expression. When DISTINCT is specified, COUNT finds the number of unique non-null values.

3	COUNT(*)	Returns the number of rows. COUNT(*) takes no parameters and cannot be used with DISTINCT.
4	MAX	Returns the maximum value in the expression. MAX can be used with numeric, character and datetime columns, but not with bit columns. With character columns, MAX finds the highest value in the collating sequence. MAX ignores any null values.
5	MIN	Returns the minimum value in the expression. MIN can be used with numeric, character and datetime columns, but not with bit columns. With character columns, MIN finds the value that is lowest in the sort sequence. MIN ignores any null values.
6	SUM	Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM can be used with numeric columns only.

Look at the examples below, illustrating the aggregate functions:

Example 1: MIN

MIN aggregate function returns the smallest value of the selected column. For example, lets us filter records to get the course with the lowest number of credits of all courses in the course table;

```
SELECT MIN(CourseCredits) AS 'Minimum Course Credits'
from  courses;
```

The result should be as follows;

```
+-------------------------+
| Minimum Course Credits  |
+-------------------------+
|                       5 |
+-------------------------+
```

1 row in set (0.00 sec)

Example 2: MAX

MAX aggregate function returns the largest value of the selected column. For example, lets us filter records to get the course with the highest number of credits of all courses in the course table;

```
SELECT MAX(CourseCredits) AS 'Maximum Course Credits'
from  courses;
```

The result should be as follows;

```
+-------------------------+
| Maximum Course Credits  |
```

```
+--------------------------+
|                  30      |
+--------------------------+
```

1 row in set (0.00 sec)

Example 3: COUNT

COUNT function returns the number of rows that matches specified criteria. Let's look at the example below from courses table where we are querying the total number of courses.

```
SELECT COUNT(*) AS 'Number of Courses' from courses;
```

The result should be as follows;

```
+----------------------+
| Number of Courses    |
+----------------------+
|                6     |
+----------------------+
```

1 row in set (0.00 sec)

Example 4: Avg

Avg function returns the average value of a numeric column. Let's look at the example below from courses table where we are querying the average of course credits in the course table.

```
SELECT Avg(CourseCredits) AS 'Average Course Credits'
from  courses;
```

The above statement should return the following output;

```
+--------------------------+
| Average Course Credits   |
+--------------------------+
|               17.5000    |
+--------------------------+
```

1 row in set (0.00 sec)

Example 5: Sum

SUM function returns the total sum of a numeric column. Let's look at the example below from courses table where we are calculating the total number of units of course units in the course table.

```
SELECT SUM(CourseUnits) AS 'Total Number of Units'
from  courses;
```

The result should be as follows;

```
+-------------------------+
| Total Number of Units   |
+-------------------------+
|                      12 |
+-------------------------+
```

1 row in set (0.00 sec)

12.11 ORDER BY CLAUSE

You use the ORDER BY clause to sort the records in the resulting list. Use ASC to sort the results in ascending order and DESC to sort the results in descending order.

Example 1:ASC:

```
select * from courses
ORDER BY CourseCode ASC;
```

The results should be as follows:-

```
+------------+-------------------------+-------------+---------------+
| CourseCode | CourseTitle             | CourseUnits | CourseCredits |
+------------+-------------------------+-------------+---------------+
| OIT214     | OO Programming          |           3 |            30 |
| OIT215     | Data Communication      |           3 |            30 |
| OIT216     | Introduction to E-business |        2 |            10 |
| OIT217     | Database Design         |           2 |            10 |
| OIT218     | Computer Security       |           1 |             5 |
| OIT219     | Software Design         |           1 |            20 |
+------------+-------------------------+-------------+---------------+
```

6 rows in set (0.00 sec)

Example 2: DESC:

```
select * from courses
ORDER BY CourseCode DESC;
```

The results should be as follows:

```
+------------+-------------------------+-------------+---------------+
| CourseCode | CourseTitle             | CourseUnits | CourseCredits |
+------------+-------------------------+-------------+---------------+
| OIT219     | Software Design         |           1 |            20 |
```

OIT218	Computer Security	1	5
OIT217	Database Design	2	10
OIT216	Introduction to E-business	2	10
OIT215	Data Communication	3	30
OIT214	OO Programming	3	30

6 rows in set (0.00 sec)

12.12 GROUP BY CLAUSE

The *GROUP BY* clause is used to create one output row per each group and produces summary values for the selected columns, as shown below. Run this query using My SQL and analyse the outcome.

Example:

Using our courses table, let's select courses whereby we will group our courses by course units.

```
select coursetitle AS 'Course Title', CourseUnits AS
'Course  Units'
from courses GROUP BY courseunits;
```

Course Title	Course Units
Computer Security	1
Introduction to E-business	2
OO Programming	3

3 rows in set (0.00 sec)

By using activities below, we should be more familiar with these SQL functions.

Check Your Progress 2

Using the courses table of university database, perform the following information retrieval operations using SQL:

Q1: List all the courses, where CourseUnits are less than 2 but CourseCredits are more than 10.

.....

.....

.....

.....

Q2: Find the total of CourseUnits of those courses whose CourseCredits are more than 10.

.....

.....

.....

.....

Q3: Consider the following database table named student & an instance of database.

Student-ID	Student-Name	Subject	Marks
S01	ABC	Database	75
S01	ABC	Physics	69
S02	XYZ	Database	80
S02	XYZ	Physics	90

Find the average marks of each student.

.....

.....

.....

.....

12.13 VIDEO LECTURE

Special Operators: https://tinyurl.com/zmr4fje

Aggregate functions: https://tinyurl.com/z62dfdk

12.14 ACTIVTIY

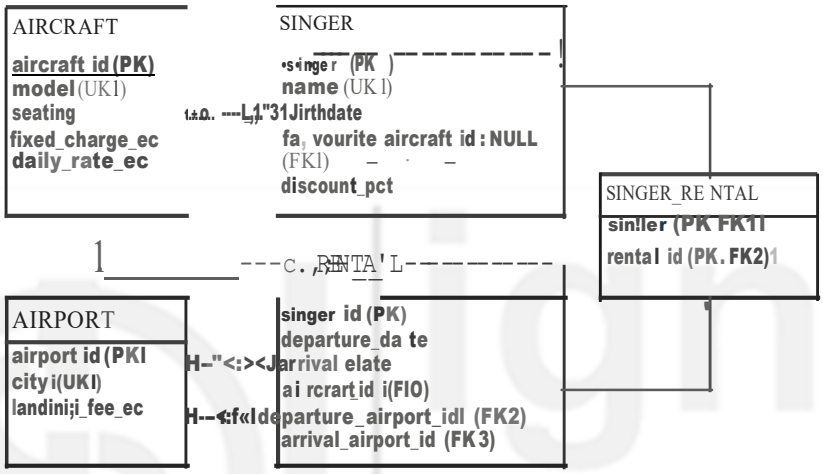
Activity 12.1

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12learning materials and MySQL DBMS or equivalent software.

What to do:

Create a database for the fictitious 'Singer Prop Rentals Company', which rents WWII-vintage fighter aircraft to a select group of singing clients: singers. The singers rent aircraft and fly between a numbers of European capital cities and are charged by the company based upon the model of aircraft and the length of the rental. The name of the database should be 'SingerProp'.



Duration: Expect to spend about 2 hours on this activity.

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 12.2

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12leaming materials and MySQL Database Software or similar DBMS.

What to do:

Populate the following table which was created in a database 'Singer Prop Rentals Company', in activity 12.1.

2	Corsair	1	275050	48280
3	Hurricane	1	199595	36050
4	Messers chmitt	1	120000	37595
5	Mustang	1	223700	39425
6	Stuka	2	149595	41000

Duration: Expect to spend about 1hour on this activity.

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 12.3

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12 learning materials and MySQL Database Software or similar DBMS.

What to do:

Populate the following table which was created in a database 'Singer Prop Rentals Company', in activity 12.1.

AIRPORT Table

AIRPORT_ID	CITY	LANDING_FEE_EC
1	Berlin	301000
2	London	359525
3	Madrid	150010
4	Paris	258600
5	Rome	422840
6	Warsaw	96000

Duration: Expect to spend about 30minutes on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 12.4

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12 learning materials and MySQL Database Software or similar DBMS.

What to do:

Populate the following table which was created in a database 'Singer Prop Rentals Company', in activity 12.1.

SINGER Table

Singer_Id	Name	BirthDate	Favourite_Aircraft_Id	Discount_pct
1	Justin	1942-04-24	2	15
2	Rohan	1981-12-02	NULL	5
3	Kunal	1946-05-20	6	12
4	Lata	1968-03-30	6	3
5	Kishore	1958-08-16	3	9
6	Rafi	1970-03-27	NULL	0
7	Asha	1918-02-06	NULL	2

Duration: Expect to spend about 1hour on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 12.5

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12 learning materials and MySQL Database Software or similar DBMS.

What to do:

Populate the following table which was created in a database 'Singer Prop Rentals Company', in activity 12.1.

SINGER _RENTAL Table

SingerId	Rental_Id
3	1
5	2
4	3
6	3
3	4
4	5
3	6
4	6
7	7
2	8
1	9

5	9
6	9
1	10
2	10
3	10
4	10
5	10
6	10
7	10
1	11
2	12
7	12

Duration: Expect to spend about 3 hour on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

Activity 12.6

Aim: To acquire practical skills in using basic SQL operations.

Resources: Unit 12 learning materials and MySQL Database Software or similar DBMS.

What to do:

Populate the following table which was created in a database 'Singer Prop Rentals Company', in activity 12.1.

RENTAL Table

Renta _Id	Depart _Date	Arrival _Date	Aircraft _Id	Deeparture _Airport_Id	Aircraft _Id
1	2006-07-18	2006-07-21	1	2	5
2	2006-07-20	2006-07-20	3	2	4
3	2006-07-25	2006-08-02	6	3	1
4	2006-07-31	2006-07-31	5	5	5
5	2006-08-02	2006-08-04	6	1	4
6	2006-08-15	2006-08-15	6	4	6
7	2006-08-07	2006-08-16	2	1	6

8	2006-08-12	2006-08-15	4	1	6
9	2006-08-10	2006-08-11	1	4	6
10	2006-08-17	2006-08-19	1	6	3
11	2006-08-21	2006-08-21	2	3	5
12	2006-08-22	2006-09-02	6	3	2

Duration: Expect to spend about 2 hours on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

12.15 SUMMARY

In this unit you learned about all the basic SQL data manipulation commands. This was mainly hands-on based unit where we have learned how to use different SQL operators like WHERE clause, Group By, order By, Aggregate functions and special operators. All these functions are aimed at helping the database end user to filter their data as per their specific requirements.

12.16 ANSWERS

Check Your Progress 1

Ans 1: SELECT - for retrieving formation from one or more SQL tables.

UPDATE - updating the data values stored in certain field or fields of a table

DELETE - to delete one or several records from a table.

Ans 2: USE university ; INSERT INTO courses VALUES ("BCS091","DBMS",3,30);

Try to enter the data, most likely you will get an error message. Try to find out why?

Ans 3: SELECT CourseCode, CourseTitle

FROM courses

WHERE CourseCredits <=20;

Check Your Progress 2

Ans 1:
```
SELECT *
FROM courses
WHERE CourseUnits < 2 AND CourseCredits > 10
```

Ans 2:
```
SELECT    sum (CourseUnits)
FROM      Courses
WHERE     CourseCredits > 10
```

Ans 3:
```
SELECT Student-ID, AVG (Marks)
FROM Student
GROUP BY Student-ID
```

12.17 REVIEW QUESTIONS FROM AUTHORS

Aim: These questions are aimed at examining learners knowledge's of SQL basic 'select statements. Students have to get the exact answer which will prove that they have done the activity section correctly.

Duration: Expect to spend 2hours on this activity.

By using 'SingerProp' database in the activity section please answer the following questions;

1. What is the average aircraft fixed charge for all of the models of aircraft?

   Answer: €2,042.40

2. What is the average aircraft fixed charge for all of the rentals?

   Answer: €2,053.56

3. What price would each singer pay, including discounts, of a 3 day rental of the Mustang? Sort the results in descending order by price. *(Hint: you'll need to use both the AIRCRAFT and SINGER table in your query, but you do not INNER JOIN them together.)*

   Answer: 7 prices ranging from €3419.75 for Rafi down to €2906.79 for Justin

4. What is the average price a Singer would pay, including discounts, of a 5 day rental of the Hurricane?

   Answer: €3,548.84

5. What is the total landing fee incurred for any rentals of aircraft that have the letter "c" (uppercase or lowercase) in the model name?

   Answer: €15,422.90

6. Excluding Singer discounts, what is the total charge (fixed and daily rate) and landing fee for each of the rentals? (*Hint: calculate the number of days for each rental by subtracting the departure date from the arrival date and then adding 1.*)

   Answer: 12 rows, total charge ranges from €1,905.95 to €7,578.50 and landing fee ranges from €960.00 to €4,228.40

7. How many times has each model of aircraft been rented? Sort the results so that the most-often-rented model appears first.

   Answer: 6 rows, the Stuka has been rented 4 times

8. How many times has each airport been visited either as a departure or arrival city? Sort the results so that the most-often-visited airport appears first. (*Hint: use an OR in the join condition between AIRPORT and RENTAL.*)

   Answer: 6 cities, Warsaw has been visited 5 times

9. What is the total amount charged per aircraft model (fixed charge and daily rate), excluding singer discounts, for all of the rentals? Sort the results so that the model producing the most income appears first. (*Hint: starting with your query from exercise #6 makes this a snap.*)

Answer: 6 models, total charge ranges from €16,233.80 to €2,356.45

10. What is the average profit or loss per aircraft model (total charge minus landing fees), excluding singer discounts, for all of the rentals? Sort the results so that the least profitable model appears first.

    Answer: 6 models, average profit ranges from €-1,597.15 to €2,811.70

11. On what dates have airplanes departed from London?

    Hint: (2 dates)

12. Which singers have favorite airplanes? And what is the model name and seating capacity of those aircraft?

    Hint: (4 singers)

13. Which singer have flown to Berlin (arrivals only)? And on which dates did they arrive?

    Hint: (2 singers)

14. Which models of aircraft (together with their seating capacities) have visited Paris (arrivals or departures)?

    Hint: (3 models, but one visited Paris twice)

15. Which singers have flown in an aircraft that is not their favorite model, other than the Catalina (whether or not they have a favorite)? Display both the singer name and model in your result set.

16. First try to answer the question without the "favorite model" restriction. That is, start off by just finding the aircraft models in which each singer has flown.

    Hint: (4 different singers in 4 different models, 6 rows in total)

12.18 FURTHER READING

1. Eng, N., & Watt, A. (2013). Database Design - 2nd Edition. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/sql-structured-query-language/

Download this book for free at http://open.bccampus.ca

2. Intro to SQL: Querying and managing data. Retrieved September 12, 2016, from KhanAcademy, https://www.khanacademy.org/computing/computer-programming/sql?ref=resume_learning#sql-basics

Rogers, D. (2002). COMP210: Class notes. Retrieved May 15, 2016, from Yukon College, http://college.yukondude.com/2006_09_comp210/html/notes.php

12.19 ATTRIBUTION

Download this book for free at http://open.bccampus.ca

The following material was written by Grace Mbwete:

1. Introduction
2. All SQL examples in the unit for manipulating data in the Databases.
3. Unit Summary

UNIT 13 SQL - JOIN STATEMENT

Structure

13.1 INTRODUCTION

Programmers frequently join data from a number of different tables in order to obtain more information. In this unit, you will learn about SQL Joins, which allow us to create complex queries, combine data from different tables, and obtain a new result set that can provide you with a better understanding of the data and maximize database flexibility.

13.2 OBJECTIVES

Upon completion of this unit you should be able to:

- *Explain* the different methods for joining tables.
- *Construct* advanced queries of two or more tables using join operations.

13.3 TERMINOLOGIES

Inner Join : SQL operator which connects two tables on a column with the same data type.

Left Outer Join : SQL operator which specifies that all left outer rows be returned.

Right Outer Join : SQL operator which includes all rows from the right table in its result set.

Group By : Aggregate functions in the SELECT clause <select> list provide information about each group instead of individual rows.

13.4 SQL JOIN STATEMENTS

Joining two or more tables is the process of comparing the data in specified columns and using the comparison results to form a new table from the rows that qualify. Ajoin statement:

- Specifies a column from each table
- Compares the values in those columns row by row
- Combines rows with qualifying values into a new row

Although the comparison is usually for equality - values that match exactly - other types of joins can also be specified. All the different joins such as inner, left (outer), right (outer), and cross join will be described below.

13.5 INNER JOINS

An *inner join* connects two tables on a column with the same data type. Only the rows where the column values match are returned; unmatched rows are discarded. The following is the syntax which illustrate how inner join operator works:-

```
SELECT  column_name(s)
FROM  table1
INNER  JOIN  table2
ON  table1.column_name=table2.column_name;
```

Check Your Progress 1

Q1: What is the need of Join Operation?

.....

.....

.....

.....

For Questions 2 & 3.

Consider the following tables

Student		Subject		
S-id	**Subject-code**	**Subject-code**	**Title**	**Credits**
S001	DBMS	DBMS	Database Management System	3
S002	DBMS	OS	Operating System	3

S001	OS	DS	Data Structure	4
S002	OS	CO	Computer Organisation	4
S003	DS			
S003	OS			

Q2: What is the necessary condition of joining these two tables?

.....

.....

.....

.....

Q3: Write the INNER JOIN command as well as show the result of INNER JOIN.

.....

.....

.....

.....

13.6 OUTER JOINS

Inner Join combines the two tables, but leaves those records, which does not have a matching record in the other table. In some cases, you require these records, therefore outer joins are needed.

13.6.1 Left Outer Join

A left outer join specifies that all left outer rows be returned in addition to rows returned by INNER JOIN. All rows from the left table that did not meet the condition specified are included in the results set, and output columns from the other table are set to NULL.

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

13.6.2 Right Outer Joins

A right outer join includes, in its result set, all rows from the right table that did not meet the condition specified in addition to rows retuned by INNER JOIN. Output columns that correspond to the other table are set to NULL.Below is an example using the new syntax for a right outer join.

```
SELECT  column_name(s)
FROM  table1
RIGHT  JOIN  table2
ON  table1.column_name=table2.column_name;
```

Check Your Progress 2

Consider the tables of Check Your Progress 1 and answers the following questions:

Q1: What would be the command and output of LEFT OUTER JOIN?

.....

.....

.....

.....

Q2: What would be the command and output of RIGHT OUTER JOIN?

.....

.....

.....

.....

13.7 VIDEO LECTURE

https://tinyurl.com/zlpvaom

13.8 ACTIVITY

Activity 13.1

Aim: To acquire practical skills in using basic SQL Join operations.

Resources: Unit 12& 13learning materials and MySQL Database Software or similar DBMS.

What to do:

By using the database created in unit 12; please attempt the following:-

1. Show the list of all drummers, and the code of the tanning bed that has been named for each, or "N/A" if a bed has not been named for the drummer.

   Hint: 7 drummers, 3 have not had beds named after them

2. Show the list of all tanning beds, and the name of the drummer for which each has been named, or just the letter code again for unnamed beds.

   Hint: 6 beds, two have not been named

3. What has each drummer paid (rate + surcharge) for their visits, broken down by the bed they used? If a drummer uses a bed named for himself, he does not pay the surcharge. (*Hint: You'll need an extra copy of BED. Use COALESCE() to "refund" the surcharge. You may run into trouble using ROUND( ), so skip it.*)

   Hint: 16 drummer and bed combinations, Charlie pays $67.25 for B, Keith pays $45.00 for D

How to do it:

By using 'Emporium' database created in unit 11; attempt the above referred tasks.

Duration: Expect to spend about 1 hour on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

13.9 SUMMARY

In this unit you learned about inner and outer join function which connects two tables in which it allows creation of a new table with comparison of rows that have qualified..

13.10 ANSWERS

Check Your Progress 1

Ans 1: It combines the related rows of two or more tables. This operation is needed as during normalization process, you decompose tables into several tables; join recombines those tables to give you combined information. Please note JOIN does not create a new table unless specifically asked, it is just an information retrieval query.

Ans 2: At least ONE column/attribute should have the same domain. In the tables given above Subject-code is the joining attribute.

Ans 3: SELECT S-id, Student.Subject-code, Title, Credits

FROM Student

INNER JOIN Subject

ON Student.Subject-code = Subject.Subject-code

* You may notice the redundancy in the output.

Output

S-id	Subject-code	Title	Credits
S001	DBMS	Database Management System	3
S002	DBMS	Database Management System	3
S001	OS	Operating System	3
S002	OS	Operating System	3
S003	DS	Data Structure	4
S003	OS	Operating System	3

Check Your Progress 2

Ans 1. SELECT S-id, Student.Subject-code, Title, Credits

FROM Student

LEFT JOIN Subject

ON Student, Subjet-code = Subject.Subject-code.

Result s exactly same as of answer of CYP-1, Q3. as all rows from left are already in the result.

Ans 2. SELECT S-id, Subject.Subject-code, Title, Credits

FROM Student

RIGHT JOIN Subject

ON Student.Subject-code = Subject.Subject-code

S-id	Subject-code	Title	Credits
-	CO	Computer Organisation	4
S001	DBMS	Database Management System	3
S002	DBMS	Database Management System	3
S001	OS	Operating System	3
S002	OS	Operating System	3
S003	DS	Data Structure	4
S003	OS	Operating System	3

13.11 CASE STUDY

Aim: This activity is a continuation of unit 11and 12; where you have to be competent with the basic SQL functions and complement them with OUTER and INNER join statements.

By using 'SingerProp' database created in unit 12; attempt the following tasks as part of this unit assessment:-

1. The company institutes a new promotional incentive so that singers that fly on their favorite aircraft get a bonus 5% discount on each rental that applies even if the rental is shared (but that's the maximum discount even if more than one singer fly together in their favorite model). Create a view called INCENTIVE_DISCOUNT that contains the columns rental_id and incentive_factor (0.95) for each of the rentals in the RENTAL table. This step doesn't require an outer join.

   i. **Hint: rentals 2, 3, 5, 6, 11**

2. What is the income, expense, discount factor, incentive factor, and profit/loss (including discounts and incentives) for each of the rentals? (Hint: Use the RENTAL_SUMMARY, RENTAL_DISCOUNT (from last week), and INCENTIVE_DISCOUNT views. Use the COALESCE() function to calculate an incentive factor of 1.00 when it doesn't apply to a particular rental.)

   i. **Hint: 12 rentals, profit ranges from €-1,912.90 €6,466.93**

3. What is the total and average profit/loss for all rentals, including diva discounts and incentives?

   i. **Hint: total profit of €13,938.10,**

   ii. **average per-rental profit of €1,161.51**

4. How many times has each diva flown on a model of aircraft that is not her favorite?

   a. (Hint: first try to answer the question without the bit about it not being her favorite aircraft. No outer join is needed, but you will need to use the COALESCE() function to force the SINGER .favourite_aircraft_id column to be an impossible value when it is NULL (e.g. 0).)

   13 rows;

13.12 FURTHER READING

1. Eng, N., & Watt, A. (2013). Database Design - 2nd Edition. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign01/chapter/sql-structured-query-language/

Download this book for free at http://open.bccampus.ca

2. Intro to SQL: Querying and managing data. Retrieved September 12, 2016, from KhanAcademy, https://www.khanacademy.org/computing/computer-programming/sql?ref=resume_learning#sql-basics

Rogers, D. (2002). COMP210: Class notes. Retrieved May 15, 2016, from Yukon College, http://college.yukondude.com/2006_09_comp210/html/notes.php

13.13 ATTRIBUTION

The content of this unit of the Learning Material - Introduction to Databases (including its images, unless otherwise noted) is a derivative copy of materials from the book Database Design by Adrienne Watt and Nelson Eng licensed under **Creative Commons Attribution 4.0 International License.**

Download this book for free at http://open.bccampus.ca

The following material was written by Grace Mbwete:

1. Introduction
2. Unit Summary

UNIT 14 DATABASE DEVELOPMENT PROCESS

Structure

14.1 INTRODUCTION

After learning all the basic theories and practical techniques of database design in all previous thirteen units of this course; you will now have the required skills of designing and implement a database given user requirements. This unit will cover the process of software development life cycle, database development process which will assist you in achieving design and implementation of database for a particular case.

14.2 OBJECTIVES

Upon completion of this unit you will be able to:

- *Understand* the basic concepts of software development life cycle.
- *Analyze* user requirements for database design and implementation process.
- *Convert* user requirements into a logical design.
- *Realize* the logical design and implement the database.
- *Test and maintain the implemented database*.

14.3 TERMINOLOGIES

Analysis : Starts by considering the statement of requirements and finishes by producing a system specification.

Data requirements document : Used to confirm the understanding of requirements with the user.

Design : Begins with a system specification, produces design documents and provides a detailed description of how a system should be constructed.

Establishing requirements : Involves consultation with, and agreement among, stakeholders as to what they want from a system; expressed as a statement of requirements.

Implementation : The construction of a computer system according to a given design document.

Maintenance : Involves dealing with changes in the requirements or the implementation environment, bug fixing or porting of the system to new environments.

Requirements gathering : A process during which the database designer interviews the database user to understand the proposed system and obtain and document the data and functional requirements.

Software Development Life Cycle (SDLC) : The series of steps involved in the database development process.

14.4 OVERVIEW

A core aspect of software engineering is the subdivision of the development process into a series of phases, or steps, each of which focuses on one aspect of the development. The collection of these steps is sometimes referred to as the software development life cycle (SDLC). The software product moves through this life cycle (sometimes repeatedly as it is refined or redeveloped) until it is finally retired from use. Ideally, each phase in the life cycle can be checked for correctness before moving on to the next phase.

14.5 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

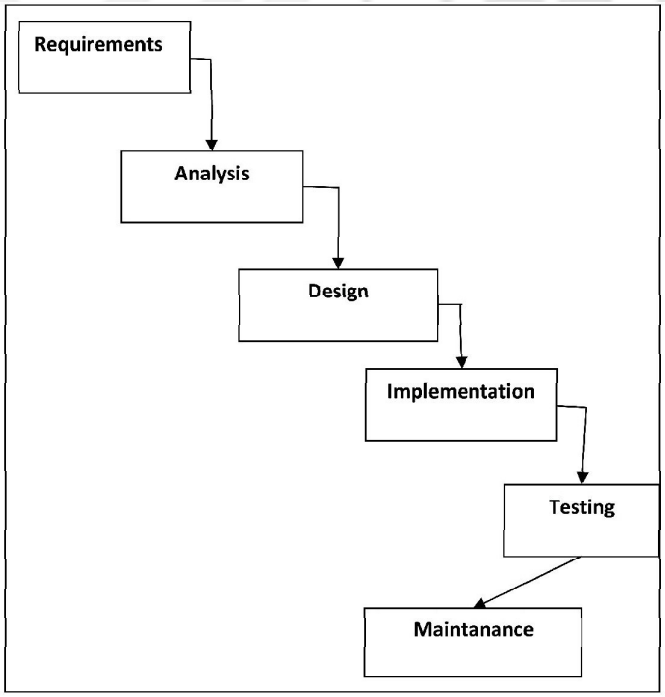


Figure 14.1: Illustration of *waterfall* model, by G.Mbwete

Let us start with an overview of the waterfall model such as you will find in most software engineering textbooks. This waterfall figure, seen in Figure 14.1, illustrates a general waterfall model that could apply to any computer system development. It shows the process as a strict sequence of steps where the output of one step is the input to the next and all of one step has to be completed before moving onto the next.

Waterfall model is used to illustrate the tasks that are required, together with the input and output for each activity. What is important is the scope of the activities, which can be summarized as follows:

i. Establishing requirements involves consultation with, and agreement among, stakeholders about what they want from a system, expressed as a statement of requirements.

ii. Analysis starts by considering the statement of requirements and finishes by producing a system specification. The specification is a formal representation of what a system should do, expressed in terms that are independent of how it may be realized.

iii. Design begins with a system specification, produces design documents and provides a detailed description of how a system should be constructed.

iv. Implementation is the construction of a computer system according to a given design document and taking into account the environment in which the system will be operating (e.g., specific hardware or software available for the development). Implementation may be staged, usually with an initial system that can be validated and tested before a final system is released for use.

v. Testing compares the implemented system against the design documents and requirements specification and produces an acceptance report or, more usually, a list of errors and bugs that require a review of the analysis, design and implementation processes to correct (testing is usually the task that leads to the waterfall model iterating through the life cycle).

vi. Maintenance involves dealing with changes in the requirements or the implementation environment, bug fixing or porting of the system to new environments (e.g., migrating a system from a standalone PC to a UNIX workstation or a networked environment). Since maintenance involves the analysis of the changes required, design of a solution, implementation and testing of that solution over the lifetime of a maintained software system, the waterfall life cycle will be repeatedly revisited.

Check Your Progress 1

Q1: What is SDLC?

.....

.....

.....

.....

Q2: Can design be completed before Analysis phase?

.....

.....

.....

.....

Q3: What is the difference between SDLC and Database Life Cycle?

.....

.....

.....

.....

14.6 DATABASE LIFE CYCLE

We can use the waterfall cycle as the basis for a model of database development that incorporates three assumptions:

i. We can separate the development of a database - that is, specification and creation of a schema to define data in a database - from the user processes that make use of the database.

ii. We can use the three-schema architecture as a basis for distinguishing the activities associated with a schema

iii. We can represent the constraints to enforce the semantics of the data once within a database, rather than within every user process that uses the data.

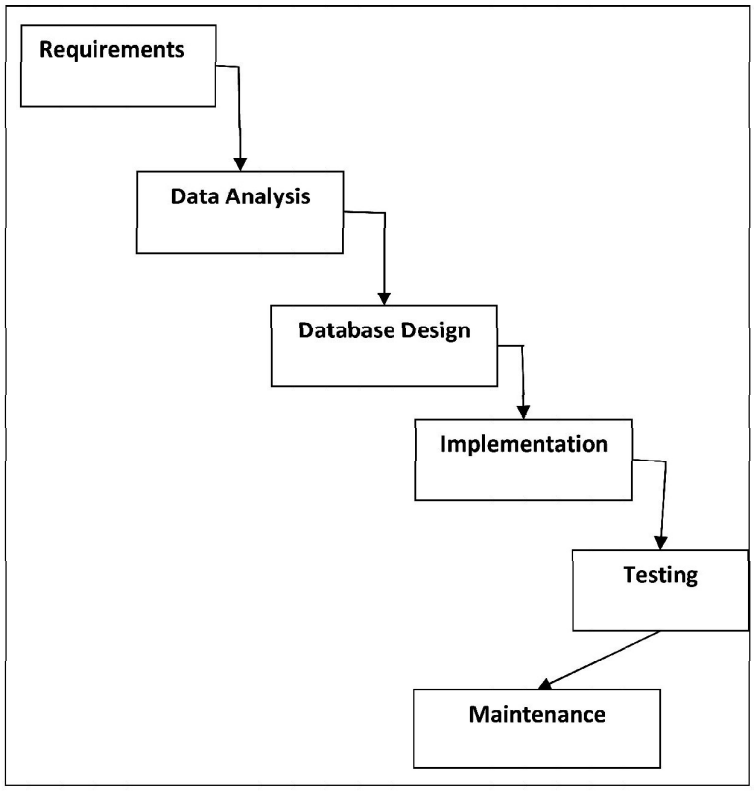


Figure 14.2: Illustration of *waterfall* model for databases, by G.Mbwete

Using these assumptions and Figure 14.2, we can see that this diagram represents a model of the activities and their outputs for database development. It is applicable to any class of DBMS, not just a relational approach. Database application development is the process of obtaining real-world requirements, analyzing requirements, designing the data and functions of the system, and then implementing the operations in the system.

14.6.1 Requirements Gathering

The first step is *requirements gathering*. During this step, the database designers have to interview the customers (database users) to understand the proposed system and obtain and document the data and functional requirements. The result of this step is a document that includes the detailed requirements provided by the users.

Establishing requirements involves consultation with, and agreement among, all the users as to what persistent data they want to store along with an agreement as to the

meaning and interpretation of the data elements. The data administrator plays a key role in this process as they overview the business, legal and ethical issues within the organization that impact on the data requirements.

The data requirements document is used to confirm the understanding of requirements with users. To make sure that it is easily understood, it should not be overly formal or highly encoded. The document should give a concise summary of all users' requirements - not just a collection of individuals' requirements - as the intention is to develop a single shared database.

The requirements should not describe how the data is to be processed, but rather what the data items are, what attributes they have, what constraints apply and the relationships that hold between the data items.

14.6.2 Analysis

Data analysis begins with the statement of data requirements and then produces a conceptual data model. The aim of analysis is to obtain a detailed description of the data that will suit user requirements so that both high and low level properties of data and their use are dealt with. These include properties such as the possible range of values that can be permitted for attributes (e.g., in the school database example, the student course code, course title and credit points).

The conceptual data model provides a shared, formal representation of what is being communicated between clients and developers during database development - it is focused on the data in a database, irrespective of the eventual use of that data in user processes or implementation of the data in specific computer environments. Therefore, a conceptual data model is concerned with the meaning and structure of data, but not with the details affecting how they are implemented.

The conceptual data model then is a formal representation of what data a database should contain and the constraints the data must satisfy. This should be expressed in terms that are independent of how the model may be implemented. As a result, analysis focuses on the questions, "What is required?" not "How is it achieved?"

14.6.3 Logical Design

Database design starts with a conceptual data model and produces a specification of a logical schema; this will determine the specific type of database system (network, relational, object-oriented) that is required. The relational representation is still independent of any specific DBMS; it is another conceptual data model.

We can use a relational representation of the conceptual data model as input to the logical design process. The output of this stage is a detailed relational specification, the logical schema, of all the tables and constraints needed to satisfy the description of the data in the conceptual data model. It is during this design activity that choices are made as to which tables are most appropriate for representing the data in a database. These choices must take into account various design criteria including, for example, flexibility for change, control of duplication and how best to represent the constraints. It is the tables defined by the logical schema that determine what data are stored and how they may be manipulated in the database.

Database designers familiar with relational databases and SQL might be tempted to go directly to implementation after they have produced a conceptual data model. However,

such a direct transformation of the relational representation to SQL tables does not necessarily result in a database that has all the desirable properties: completeness, integrity, flexibility, efficiency and usability. A good conceptual data model is an essential first step towards a database with these properties, but that does not mean that the direct transformation to SQL tables automatically produces a good database. This first step will accurately represent the tables and constraints needed to satisfy the conceptual data model description, and so will satisfy the completeness and integrity requirements, but it may be inflexible or offer poor usability. The first design is then flexed to improve the quality of the database design. Flexing is a term that is intended to capture the simultaneous ideas of bending something for a different purpose and weakening aspects of it as it is bent.

Figure 14.3 summarizes the iterative (repeated) steps involved in database design, based on the overview given. Its main purpose is to distinguish the general issue of what tables should be used from the detailed definition of the constituent parts of each table - these tables are considered one at a time, although they are not independent of each other. Each iteration that involves a revision of the tables would lead to a new design; collectively they are usually referred toas second-cut designs, even if the process iterates for more than a single loop.

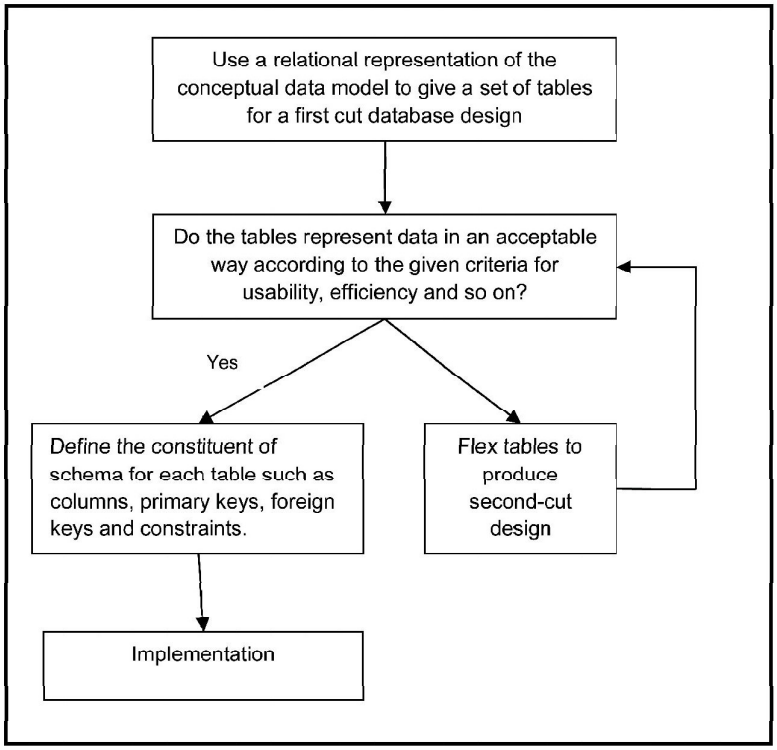


Figure 14.3: Asummary of the iterative steps involved in database design, by A. Watt.

First, for a given conceptual data model, it is not necessary that all the user requirements it represents be satisfied by a single database. There can be various reasons for the development of more than one database, such as the need for independent operation in different locations or departmental control over "their" data. However, if the collection of databases contains duplicated data and users need to access data in more than one database, then there are possible reasons that one database can satisfy multiple requirements, or issues related to data replication and distribution need to be examined.

Second, one of the assumptions about database development is that we can separate the development of a database from the development of user processes that make use

of it. This is based on the expectation that, once a database has been implemented, all data required by currently identified user processes have been defined and can be accessed; but we also require flexibility to allow us to meet future requirements changes. In developing a database for some applications, it may be possible to predict the common requests that will be presented to the database and so we can optimize our design for the most common requests.

Third, at a detailed level, many aspects of database design and implementation depend on the particular DBMS being used. If the choice of DBMS is fixed or made prior to the design task, that choice can be used to determine design criteria rather than waiting until implementation. That is, it is possible to incorporate design decisions for a specific DBMS rather than produce a generic design and then tailor it to the DBMS during implementation.

It is not uncommon to find that a single design cannot simultaneously satisfy all the properties of a good database. So it is important that the designer has prioritized these properties (usually using information from the requirements specification); for example, to decide if integrity is more important than efficiency and whether usability is more important than flexibility in a given development.

At the end of our design stage, the logical schema will be specified by SQL data definition language (DDL) statements, which describe the database that needs to be implemented to meet the user requirements.

14.6.4 Implementation

Implementation involves the construction of a database according to the specification of a logical schema. This will include the specification of an appropriate storage schema, security enforcement, external schema and so on. Implementation is heavily influenced by the choice of available DBMSs, database tools and operating environment. There are additional tasks beyond simply creating a database schema and implementing the constraints - data must be entered into the tables, issues relating to the users and user processes need to be addressed, and the management activities associated with wider aspects of corporate data management need to be supported. In keeping with the DBMS approach, we want as many of these concerns as possible to be addressed within the DBMS. We look at some of these concerns briefly now.

In practice, implementation of the logical schema in a given DBMS requires a very detailed knowledge of the specific features and facilities that the DBMS has to offer. In an ideal world, and in keeping with good software engineering practice, the first stage of implementation would involve matching the design requirements with the best available implementing tools and then using those tools for the implementation. In database terms, this might involve choosing vendor products with DBMS and SQL variants most suited to the database we need to implement. However, we don't live in an ideal world and more often than not, hardware choice and decisions regarding the DBMS will have been made well in advance of consideration of the database design. Consequently, implementation can involve additional flexing of the design to overcome any software or hardware limitations. The following are the stages involved in the implementation of database:

i) Realizing the Design

After the logical design has been created, we need our database to be created according to the definitions we have produced. For an implementation with a relational DBMS,

this will probably involve the use of SQL to create tables and constraints that satisfy the logical schema description and the choice of appropriate storage schema (if the DBMS permits that level of control).

One way to achieve this is to write the appropriate SQL DDL statements into a file that can be executed by a DBMS so that there is an independent record, a text file, of the SQL statements defining the database. Another method is to work interactively using a database tool like SQL Server Management Studio or Microsoft Access. Whatever mechanism is used to implement the logical schema, the result is that a database, with tables and constraints, is defined but will contain no data for the user processes.

ii) Populating the Database

After a database has been created, there are two ways of populating the tables - either from existing data or through the use of the user applications developed for the database.

For some tables, there may be existing data from another database or data files. For example, in establishing a database for a hospital, you would expect that there are already some records of all the staff that have to be included in the database. Data might also be brought in from an outside agency (address lists are frequently brought in from external companies) or produced during a large data entry task (converting hard-copy manual records into computer files can be done by a data entry agency). In such situations, the simplest approach to populate the database is to use the import and export facilities found in the DBMS.

Facilities to import and export data in various standard formats are usually available (these functions are also known in some systems as loading and unloading data). Importing enables a file of data to be copied directly into a table. When data are held in a file format that is not appropriate for using the import function, then it is necessary to prepare an application program that reads in the old data, transforms them as necessary and then inserts them into the database using SQL code specifically produced for that purpose. The transfer of large quantities of existing data into a database is referred to as a bulk load. Bulk loading of data may involve very large quantities of data being loaded, one table at a time so you may find that there are DBMS facilities to postpone constraint checking until the end of the bulk loading.

iii) Supporting Users and User Processes

Use of a database involves user processes (either application programs or database tools) which must be developed outside of the database development. In terms of the three-schema architecture we now need to address the development of the external schema. This will define the data accessible to each user process or group of user processes. In reality, most DBMSs, and SQL itself, do not have many facilities to support the explicit definition of the external schema. However, by using built-in queries and procedures, and with appropriate security management, it is possible to ensure access to data by a user process is limited to a tailored subset of the entire database content.

In addition to ensuring that appropriate data access for each user process is achieved, the database developer needs to address other user-related issues. Examples of such issues include: reporting of error conditions, constraint enforcement, automated processing using triggers, grouping of activities into transactions, defining database procedures and functions and data security (in addition to the general database and user process access control).

Check Your Progress 2

Q1: What is the need of Logical design in Database?

.....

.....

.....

.....

Q2: Why is the design iterative?

.....

.....

.....

.....

Q3: What are the steps of implementation?

.....

.....

.....

.....

14.6.5 Testing

The aim of testing is to uncover errors in the design and implementation of the database, its structure, constraints and associated user and management support. Testing is usually considered to involve two main tasks - validation and verification. Without adequate testing users will have little confidence in their data processing.

Validation answers the question: has the right database been developed to meet the requirements? It attempts to confirm that the right database has been constructed, with the right characteristics to meet the specified requirements.

Verification answers the question: has the database design been implemented correctly? Verification ensures that the processing steps, constraints and other 'programmed' components of the database (security, backup, recovery, audit trails, etc.) have been correctly implemented and contain no errors in program logic or execution sequences.

Of course, testing does not just take place only after all the above development steps are complete. It is usually applied throughout the stages in the development processes and includes appropriate reviews to scrutinise the outputs of the development activities. The aim is to identify errors as soon as possible in the development life cycle.

14.6.6 Maintenance

Databases are one of the more enduring **software engineering artefacts**; it is not uncommon to find database implementations whose use can be traced back for 15 years or more. Consequently, maintenance of the database is a key issue. Maintenance can take three main forms:

i) **Operational maintenance**, where the performance of the database is monitored. If it falls below some acceptable standard, then reorganization of the database, usually in the form of adjustments to the storage schema, takes place to ensure that performance is maintained at an acceptable level.

ii) **Porting** and implementation maintenance, in which the DBMS, the user processes, the underlying computer system or some other aspect undergoes changes that require the database implementation to be revised. Of course, the extreme form of porting occurs when the database is to be entirely replaced - in which case the entire development life cycle is usually followed using the existing database and its limitations as the starting point for requirements analysis. Adjustments to the storage schema are almost inevitable as new data storage capabilities become available. This can involve both restructuring and reorganization, depending on the scope of the changes taking place.

iii) **Requirements change**, where the original requirement specification changes, usually because databases are frequently used for purposes for which they were not originally designed. This involves restructuring and typically involves a 'mini life cycle' related to the development of changes to meet the new requirements.

If we follow the three-schema architecture approach we would hope to minimize the impact of change and build systems which are easily maintained.

Check Your Progress 3

Q1: Why is testing done?

.....

.....

.....

.....

Q2: What are different types of maintenance?

.....

.....

.....

.....

14.7 VIDEO LECTURE

https://tinyurl.com/juu9f2e

14.8 ACTIVITY

Activity 14.0

Database Development Process

Motivation: To become conversant with basic concepts of database development life cycle.

Resources: Internet access and Unit 14 learning materials.

What to do: Provide answers for the following questions:

1. Explain in detail, three software development methodologies that would be suitable for databases.
2. What does the acronym SDLC mean, and what does an SDLC portray?
3. What needs to be modified in the waterfall model to accommodate database design?
4. Provide the iterative steps involved in database design.

Duration: Expect to spend about 1 hour on this activity

Feedback: This activity should be submitted to the course Instructor for assessment and feedback.

14.9 SUMMARY

In this unit you learned about Database Development Process. In summary, relational database systems underpin the majority of the managed data storage in computer systems. In this unit we have considered database development as an instance of the waterfall model of the software development life cycle. We have seen that the same activities are required to develop and maintain databases that meet user requirements.

14.10 ANSWERS

Check Your Progress 1

Ans 1: SDLC is the process of development of software. The software develpment goes through these cycles till it is replaced.

Ans 2: No, design requires extensive detailing of what is to be developed, that can be done only if proper analysis has been done.

Ans 3: Database life cycle is the process of analysis, design and development of a database for a system.

Check Your Progress 2

Ans 1: It provides a specification for logical schema, which can be directly transformed to tables.

Ans 2: It is a creative process and detailing and discussion at each step makes design robust.

Ans 3: Realizing the design, populating the database, supporting users & user processes.

Check Your Progress 3

Ans 1: Testing uncover errors.

Ans 2: Operational maintenance; Porting; Requirement changes.

14.11 REVIEW QUESTIONS FROM AUTHORS

Aim: The following review questions are aimed at assessing the students' knowledge and understanding of software development life cycle and database development process.

1. Explain in details, the meaning of Software Development Life Cycle.
2. Describe the importance of Software Development Life Cycle.
3. What is SDLC model? Mention three types of the most common used SDLC models.
4. What is a waterfall model? Outline the steps in waterfall model.
5. What are the advantages of using waterfall model?
6. Explain when to use waterfall model.
7. What are the disadvantages of using waterfall model?
8. Explain in details the steps involved in Database Life Cycle.
9. Describe the tasks involved during implementation stage of database life cycle.
10. Outline the activities which take place during testing and evaluation of a database.

14.12 FURTHER READING

1. Watt, A., &Eng, N. (2013). Database Design - 2nd Edition. Retrieved May 23, 2016, from BC Open TextBooks, https://opentextbc.ca/dbdesign/chapter/chapter-13-database-development-process/

   Download this book for free at http://open.bccampus.ca

2. Open, T. (2016). The Database Development Life Cycle. Retrieved April 15, 2016, from OpenLearn - The Open University, http://www.open.edu/openlearn/science-maths-technology/computing-and-ict/information-and-communication-technologies/the-database-development-life-cycle/content-section-1.1.1

14.13 ATTRIBUTION

The content of this unit (Database Development Process) of the Learning Material - Introduction to Databases (including its images, unless otherwise noted) is from Chapter

The following material of this unit was written by Grace Mbwete:

1. Introduction
2. Activity 14.0
3. Unit Summary

Video 20 - Review Questions and Activities

https://tinyurl.com/j59x9rz