Database Management System Lab (CS-2094)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

1 Credit

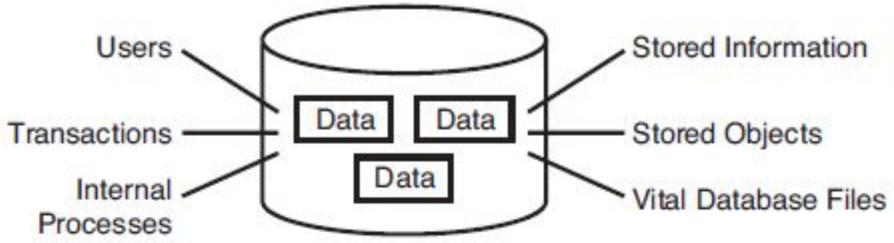
Dr. Jayanta Mondal

Lab Contents

Sr #	Major and Detailed Coverage Area	Lab#
1	An introduction to SQL Introduction to database management systems Overview of some basic terms and concepts	1

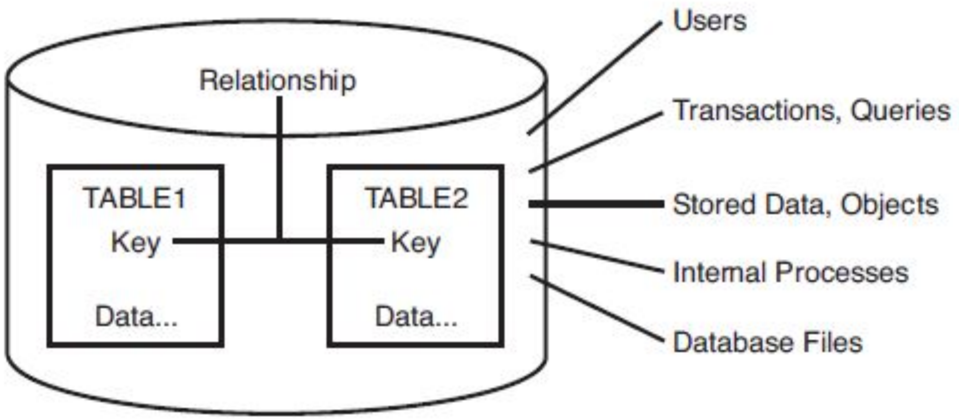
Recap on DBMS

In simple terms, a **database** is a collection of data. A phone book is a database. The data contained consists of individuals' names, addresses, and telephone numbers. Every modern-day business has data, which requires some organized method or mechanism for maintaining and retrieving the data. When the data is kept within a database, this mechanism is referred to as a **database management system** (DBMS). The database has to be maintained. As people move to different cities or states, entries might have to be added or removed from the phone book. Likewise, entries have to be modified for people changing names, addresses, telephone numbers, and so on.



Recap on RDBMS

The modern wave of information management is primarily carried out through the use of a relational database management system (**RDBMS**), derived from the traditional DBMS. A relational database is a database divided into logical units called tables, where tables are related to one another within the database. A relational database allows data to be broken down into logical, smaller, manageable units, enabling easier maintenance and providing more optimal database performance according to the level of organization.



SQL

The relational algebra provides a concise, formal notation for representing queries. However, commercial database systems require a query language that is more user friendly. SQL can do much more than just query a database. It can define the structure of the data, modify data in the database and specify security constraints.

What is SQL?

Structured Query Language (SQL) is the standard language used to communicate with a relational database. The prototype was originally developed by IBM using Dr. E.F. Codd’s paper (“A Relational Model of Data for Large Shared Data Banks”) as a model. In 1979, not long after IBM’s prototype, the first SQL product, ORACLE, was released by Relational Software, Incorporated (which was later renamed Oracle Corporation). Today it is one of the distinguished leaders in relational database technologies.

SQL Command Types

The basic categories of commands used in SQL to perform various functions are

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Query Language (DQL)

Will cover in detail

- Data Control Language (DCL)
- Transactional control commands

Will cover the very basics

DDL & DML

Data Definition Language (DDL) is the part of SQL that enables a database user to create and restructure database objects, such as the creation or the deletion of a table. Some of the most fundamental DDL commands include:

- Create Table
- Alter Table
- Drop Table
- Create Index
- Alter Index
- Drop Index
- Create View
- Alter View
- Drop View

Data Manipulation Language (DML) is the part of SQL used to manipulate data within objects of a relational database. The three basic DML commands are:

- INSERT
- UPDATE
- DELETE

DQL and DCL

DQL - Though comprised of only one command, **Data Query Language** (DQL) is the most concentrated focus of SQL for modern relational database users. The base command is SELECT. This command, accompanied by many options and clauses, is used to compose queries against a relational database. A query is an inquiry to the database for information. A query is usually issued to the database through an application interface or via a command-line prompt. You can easily create queries, from simple to complex, from vague to specific.

DCL - Data control commands in SQL enable you to control access to data within the database. These **Data Control Language** (DCL) commands are normally used to create objects related to user access and also control the distribution of privileges among users. Some data control commands are as follows:

- ALTER PASSWORD
- GRANT
- REVOKE
- CREATE SYNONYM

Transactional Control Commands

In addition to the previously introduced categories of commands, there are commands that enable the user to manage database transactions:

- COMMIT—Saves database transactions
- ROLLBACK—Undoes database transactions
- SAVEPOINT—Creates points within groups of transactions in which to

Database Objects

A database object is any defined object in a database that is used to store or reference data. Some examples of database objects include tables, views, clusters, sequences, indexes, and synonyms. The table is the focus because it is the primary and simplest form of data storage in a relational database.

What Is a Schema?

Shema refers to the structure of a database object. Broadly, schema can be a collection of database objects normally associated with one particular database username. This username is called the schema owner, or the owner of the related group of objects. You may have one or multiple such schemas in a database. Based on a user’s privileges within the database, the user has control over objects that are created, manipulated, and deleted. A schema can consist of a single table and has no limits to the number of objects that it may contain, unless restricted by a specific database implementation.

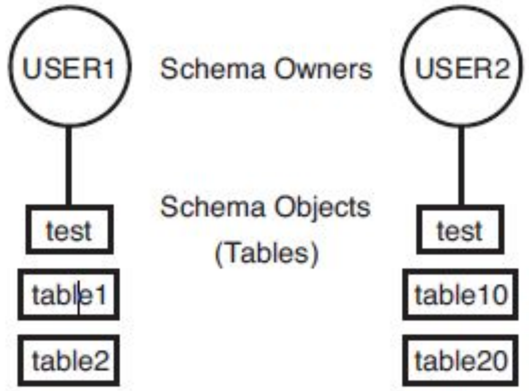
Schema cont...

Say you have been issued a database username and password by the database administrator. Your username is C##USER1. Suppose you log on to the database and then create a table called EMPLOYEE. According to the database, your table's actual name is C##USER1.EMPLOYEE. The schema name for that table is C##USER1, which is also the owner of that table. You have just created the first table of a schema. The good thing about schemas is that when you access a table that you own (in your own schema), you do not have to refer to the schema name. For instance, you could refer to your table as either one of the following:

- EMPLOYEE
- C##USER1.EMPLOYEE

The first option is preferred because it requires fewer keystrokes. If another user were to query one of your tables, the user would have to specify the schema as C##USER1.EMPLOYEE

Schema cont...



In this example, both users have a table called **TEST**. Tables can have the same names in a database as long as they belong to different users. If you look at it this way, table names are always unique in a database because the schema owner or username is actually part of the table name. For instance, USER1.TEST is a different table than USER2.TEST. If you do not specify a schema with the table name when accessing tables in a database, the database server looks for a table that you own by default. That is, if USER1 tries to access TEST, the database server looks for a USER1-owned table named TEST before it looks for other objects owned by USER1, such as synonyms to tables in another schema.

Tables

The **table** is the primary storage object for data in a relational database. In its simplest form, a table consists of row(s) and column(s), both of which hold the data. A table takes up physical space in a database and can be permanent or temporary.

Note:

- When you create a table, you must specify the table name, name of each column, data type of each column, and size of each column
- The table and column names can be up to 30 characters long
- Table or column name must begin with a letter
- The names are not case sensitive
- Spaces and hyphens are not allowed in a table or a column name; but $, _ and # are allowed

Columns

A **field**, also called a **column** in a relational database, is part of a table that is assigned a specific data type. The data type determines what kind of data the column is allowed to hold. This enables the designer of the table to help maintain the integrity of the data. Every database table must consist of at least one column. Columns are those elements within a table that hold specific types of data, such as a person's name or address. For example, a valid column in a customer table might be the customer's name. Generally, a column name must be one continuous string and can be limited to the number of characters used according to each implementation of SQL. It is typical to use underscores with names to provide separation between characters. For example, a column for the customer's name can be named CUSTOMER_NAME instead of CUSTOMERNAME. This is normally done to increase the readability of database objects. Columns also can be specified as **NULL** or **NOT NULL**, meaning that if a column is NOT NULL, something must be entered. If a column is specified as NULL, nothing has to be entered.

Rows

A row is a record of data in a database table. For example, a row of data in a customer table might consist of a particular customer's identification number, name, address, phone number, and fax number. A row is composed of fields that contain data from one record in a table. A table can contain as little as one row of data and up to as many as millions of rows of data or records.

Data Types

Data types are characteristics of the data itself, whose attributes are placed on fields within a table. For example, you can specify that a field must contain numeric values, disallowing the entering of alphanumeric strings. After all, you would not want to enter alphabetic characters in a field for a dollar amount. Defining each field in the database with a data type eliminates much of the incorrect data found in a database due to data entry errors. Field definition (data type definition) is a form of data validation that controls the type of data that may be entered into each given field.

Depending on the implementation of relational database management system (RDBMS), certain data types can be converted automatically to other data types depending upon their format. This type of conversion in known as an implicit conversion, which means that the database handles the conversion for you. An example of this is taking a numeric value of 1000.92 from a numeric field and inputting it into a string field. Other data types cannot be converted implicitly by the host RDBMS and therefore must undergo an explicit conversion. This usually involves the use of an SQL function, such as CAST or CONVERT. For example SELECT CAST('12/27/1974' AS DATETIME) AS MYDATE

Data Types cont...

The very basic data types, as with most other languages, are

- String types
- Numeric types
- Date and time types

String Types

- **char(n) –** The **fixed-length character** with user specified length n. The full form character can be used instead.
- **varchar(n)/varchar2(n) -** The **variable-length character** with user specified maximum length n. The full form character varying is equivalent.

Numeric Types

Numeric values are stored in fields that are defined as some type of number, typically referred to as NUMBER, INTEGER, REAL, DECIMAL, and so on.

Data Types cont...

Numeric Types

The following are the standards for SQL numeric values:

- BIT(n)
- BIT VARYING(n)
- DECIMAL(p,s)
- INTEGER
- SMALLINT
- BIGINT
- FLOAT(p, s)
- DOUBLE PRECISION(p, s)
- REAL(s)

Date and Time Types

Date and time data types are quite obviously used to keep track of information concerning dates and time. Standard SQL supports what are called DATETIME data types, which include the following specific data types:

- DATE
- TIME
- DATETIME
- TIMESTAMP

Other Data Types

NULL Types

NULL value is a missing value or a column in a row of data that has not been assigned a value. NULL values are used in nearly all parts of SQL, including the creation of tables, search conditions for queries, and even in literal strings.

LOB Types

Some variable-length data types need to hold longer lengths of data than what is traditionally reserved for a VARCHAR field. The **BLOB** and **TEXT** data types are two examples of such data types in modern database implementations. These data types are specifically made to hold large sets of data. The BLOB is a binary large object, so its data is treated as a large binary string (a byte string). A **BLOB** is especially useful in an implementation that needs to store binary media files in the database, such as images or MP3s. The **TEXT** data type is a large character string data type that can be treated as a large VARCHAR field. It is often used when an implementation needs to store large sets of character data in the database. An example of this would be storing HTML input from the entries of a blog site. Storing this type of data in the database enables the site to be dynamically updated.

Domains

A domain is a set of valid data types that can be used. A domain is associated with a data type, so only certain data is accepted.

DDL - Table Creation

The CREATE TABLE statement in SQL is used to create a table. Although the very act of creating a table is quite simple, much time and effort should be put into planning table structures before the actual execution of the CREATE TABLE statement. Carefully planning your table structure before implementation saves you from having to reconfigure things after they are in production. The basic syntax is:

```
CREATE TABLE table_name
( field1 data_type [ not null ],
field2 data_type [ not null ],
field3 data_type [ not null ],
field4 data_type [ not null ],
field5 data_type [ not null ] );
```

[not null] depicts optional

DDL - Create Table Example

```
CREATE TABLE EMPLOYEE
(EMP_ID NUMBER(9) NOT NULL,
EMP_NAME VARCHAR2 (40) NOT NULL,
EMP_ST_ADDR VARCHAR2 (20) NOT NULL,
EMP_CITY VARCHAR2 (15) NOT NULL,
EMP_ST VARCHAR(2) NOT NULL,
EMP_ZIP INTEGER,
EMP_PHONE INTEGER NULL,
EMP_PAGER INTEGER NULL);
```

Class Work – Create Tables

- ❑ STUDENT
- ❑ PRODUCT
- ❑ CUSTOMER
- ❑ ACCOUNT
- ❑ PERSON

DDL - Table Description

DESCRIBE statement is used for viewing table structure. The syntax is:

DESCRIBE table_name; or

DESC table_name;

For example: DESCRIBE EMPLOYEE;

Lab Exercise – Describe the following tables

- STUDENT
- PRODUCT
- CUSTOMER
- ACCOUNT
- PERSON

DDL - Creating Table from another Table

Creating Table from another Table

CREATE TABLE tablename(column1,column2) AS SELECT column1,column2 FROM tablename;

CREATE TABLE Person(Roll, Name, Age) AS SELECT Roll, Name, Age FROM Stud;

The SQL statement populates the target table with data from the source table

DDL - Dropping a Table

The syntax to drop a table follows:

DROP TABLE table_name;

Class Work – Drop the following tables

- ❑ STUDENT
- ❑ PRODUCT
- ❑ CUSTOMER
- ❑ ACCOUNT
- ❑ PERSON

DDL – Altering a Table

You can modify a table after the table has been created by using the **ALTER TABLE** command. You can add column(s), drop column(s), change column definitions, add and drop constraints.

Example

- ALTER TABLE EMPLOYEE MODIFY EMP_ID VARCHAR(10);
- ALTER TABLE EMPLOYEE ADD EMP_SEX VARCHAR(1);
- ALTER TABLE CUSTOMER DROP COLUMN CUSTOMER_NAME;
- ALTER TABLE CUSTOMER ADD (CUSTOMER_NAME VARCHAR2(45), CITY VARCHAR2(40));
- ALTER TABLE CUSTOMER MODIFY (CUSTOMER_NAME VARCHAR2(100) NOT NULL, CITY VARCHAR2(75));
- ALTER TABLE CUSTOMER RENAME COLUMN CUSTOMER_NAME TO CNAME;
- ALTER TABLE CUSTOMERS RENAME TO CONTACTS;

DDL – Altering a Table

Stud (roll, name, age)

Column	NULL?	Datatype
ROLL		NUMBER(6)
NAME		VARCHAR2(20)
AGE		NUMBER(2)

Adding a New Column

ALTER TABLE tablename ADD(column definition);

ALTER TABLE Stud ADD (address number(20));

Column	NULL?	Datatype
ROLL		NUMBER(6)
NAME		VARCHAR2(20)
AGE		NUMBER(2)
ADDRESS		NUMBER(20)

DDL – Altering a Table

Modifying an Existing Column

ALTER TABLE tablename MODIFY(column definition);

ALTER TABLE Stud MODIFY(address varchar2(20));

Column	NULL?	Datatype
ROLL		NUMBER(6)
NAME		VARCHAR2(20)
AGE		NUMBER(2)
ADDRESS		VARCHAR2(20)

Dropping a Column

ALTER TABLE tablename DROP COLUMN columnname;

ALTER TABLE Stud DROP COLUMN address;

Column	NULL?	Datatype
ROLL		NUMBER(6)
NAME		VARCHAR2(20)
AGE		NUMBER(2)

DDL – Altering a Table

Renaming a Column

```
ALTER TABLE tablename RENAME COLUMN oldname to
newname;
```

```
ALTER TABLE Stud RENAME COLUMN roll to id;
```

Column	NULL?	Datatype
ID		NUMBER(6)
NAME		VARCHAR2(20)
AGE		NUMBER(2)

DDL – Renaming a Table

- THROUGH ALTER STATEMENT

  ALTER TABLE CUSTOMERS RENAME TO CONTACTS;

- DIRECTLY THROUGH RENAME STATEMENT

  RENAME CONTACTS TO CUSTOMERS;

SYNTAX: RENAME OLD_TABLE_NAME TO NEW_TABLE_NAME;

DML - Inserting data into a Table

INSERT statement is to insert new data into a table. The basic syntax to begin:

INSERT INTO TABLE_NAME VALUES (‘value1’, ‘value2’, [NULL]);

Example: Let the table structure for “PRODUCT” is as follows –

COLUMN Name	Null?	DATA TYPE
PROD_ID	NOT NULL	VARCHAR(10)
PROD_DESC	NOT NULL	VARCHAR(25)
COST	NOT NULL	NUMBER(6,2)

INSERT INTO PRODUCT VALUES (‘7725’, ’LEATHER GLOVES’, 24.99);

Class Work - Insert 2 distinct rows to the following tables

- STUDENT
- PRODUCT
- CUSTOMER
- ACCOUNT
- PERSON

DML - Entering NULL values

Implicit method: Here, column name is omitted from the column list in the INSERT statement

Explicit method: Here, NULL is used as a value for a numeric column, and an empty string (“”) is used for date or character columns

DML - Substitution variables

- Substitution variables enable you to create an interactive SQL script, which prompts you to enter a value for the substitution variable
- In command line version, & character is used before the substitution variable in the query; whereas : character is used in graphical version
- Substitution variables for character and date columns are enclosed within a pair of single quotation marks
- For more records, press /

DML - Different ways of inserting a row

- INSERT INTO STUDENT(Roll, Name, Gender, Age, CGPA) VALUES (705129, ’Uday’, ’M’, 19, 9.2);
- INSERT INTO STUDENT VALUES (705129, ’Uday’, ’M’, 19, 9.2);
- INSERT INTO STUDENT(Roll, Name, CGPA) VALUES (705129, ’Uday’, 9.2);
- INSERT INTO STUDENT VALUES (&Roll, ’&Name’, ’&Gender’, &Age, &CGPA);
- INSERT INTO STUDENT (Roll, Name, Gender, Age) VALUES(&Roll, ’&Name’, ’&Gender’, &Age);

DML - Date/Time Insertion

Time can be inserted by using TO_DATE()

INSERT INTO STUDENT(dob) VALUES (TO_DATE(
'12-JAN-1990 10:34:45 P.M.', 'DD-MON-YYYY HH:MI:SS
P.M.'));

• If only the date value is entered in a date-type column, the time value is set to the midnight (12:00A.M.)

• If only the time value is entered in a date-type column, the date is set to first of the current month

DQL - Selecting data from a Table

The SELECT statement is used in conjunction with the FROM clause to extract data from the database in an organized, readable format. The SELECT part of the query is for selecting the data you want to see according to the columns in which they are stored in a table.

The syntax for a simple SELECT statement is as follows:

*Either * (means all columns) or Individual columns*

[,TABLE2] depicts optional

SELECT [* | COLUMN1, COLUMN2] FROM TABLE1 [, TABLE2];

The SELECT keyword in a query is followed by a list of columns that you want displayed as part of the query output. The asterisk (*) denotes that all columns in a table should be displayed as part of the output.

Example - SELECT * FROM PRODUCT;

Class Work - select all rows from the following tables

- STUDENT
- CUSTOMER
- PERSON
- PRODUCT
- ACCOUNT

DQL - Selecting data from a Table

*SELECT * FROM STUDENT;*

Roll	Name	Gender	Age	CGPA
705129	Uday	M	19	9.2
705170	Ram	M	20	
705171	Kim	F	19	8.6
705172	Raji		20	7.5

NULL value means the value is unknown or doesn’t exist

DQL - Selecting data from a Table

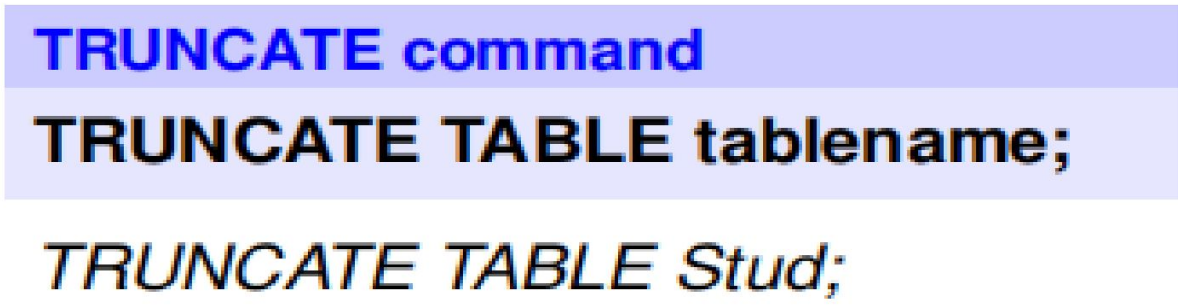
Querying Data...

SELECT Roll, Name, CGPA FROM STUDENT;

Roll	Name	CGPA
705129	Uday	9.2
705170	Ram	
705171	Kim	8.6
705172	Raji	7.5

DDL - Truncate Statement

TRUNCATE TABLE Command is used to delete complete data from an existing table.



TRUNCATE command

```
TRUNCATE TABLE tablename;
```

```
TRUNCATE TABLE Stud;
```

Viewing all user Objects

```
SELECT * FROM TAB;
```

Viewing all user Tables

```
SELECT table_name FROM user_tables;
```

Thank You
End of Lab 1

Assignment

Write PL/SQL queries for the following -

1. Create tables for - Student(student_id, first_name, last_name, dept, Date_of_birth, gender, religion), Employee, Product, Customer, and Account. Identify relevant attributes for each table make sure each table has atleast four columns. Ensure each table has a _ID column e.g. Employee should have EMPLOYEE_ID column, Student should have STUDENT_ID column etc.
2. Describe each table.
3. Insert at least 5 distinct rows to each table.
4. Fetch all data from the respective tables.
5. Fetch Employee ids and their names from Employee table.
6. Create table YOUTH (f_name, l_name, sex, DOB) from the Student table.
7. Delete all data from the customer table.
8. Delete the Account table.

Assignment

9. Fetch the f_name and DOB from YOUTH table.
10. Insert a new record into the Youth table. And keep NULL value in the l_name column.
11. Insert a new record into the Employee table. And keep NULL value in the employee_id column.
12. Change the name of the employee table to workers.
13. Increase the size of the dept field in the student table by 10.
14. Add a column ph_no in the student table.
15. Drop the religion attribute from the student table.
16. Rename the student_id field to roll_no in the student table.
17. Change the datatype and size of the product id column in the product table.