UNIT 12 QUERY LANGUAGE

Structure

12.0 OBJECTIVES

Structured Query Language (SQL) is a query language used in RDBMS environment, which allows the users to define and describe and manipulate data in a DBMS.

After going through this unit you will be able to:

- differentiate among SQL commands;
- define database tables using SQL;
- manipulate database tables using SQL; and
- make queries using SQL.

12.1 INTRODUCTION

The utility of query languages are for database manipulation. There are different query languages which may be used, depending upon the nature of the database management system (DBMS). A few of them are ISBL, SEQUEL, SQUARE, Query-By-Example, SQL, AQL etc. As you know, the relational database management system (RDBMS) is a view of DBMS and is widely accepted. The details of RDBMS have been detailed in Block 3 of Information systems. Structured query language (SQL) is the well accepted query language along with RDBMS. For searching in Internet, new query languages for HTML, XML documents are evolving. XML-QL is one among them to mention.

Structured Query Language (SQL) is not a complete programming language. It is used to interrogate and process data in a relational database environment. SQL commands can work interactively with a database or be embedded in programming languages such as C, COBOL or Java. SQL was developed by IBM for using in mainframes.

SQL is a non-procedural, fourth generation language that allows users to access data in relational database management systems, such as Oracle, Sybase, Informix, Microsoft SQL Server, Access, and others, by allowing users to describe the data that he or she wishes to see. SQL also allows users to define the data in a database, and manipulate that data. This unit will describe how to use SQL, with examples. The SQL is in both ANSI and ISO standards. The SQL described in this unit is standard SQL. Some SQL features of specific database management systems are mentioned in the Nonstandard SQL.

12.2 DATA DEFINITION LANGUAGE

The data definition language (DDL) in a RDBMS is for defining a relational table in the context of a RDBMS. It specifies the attributes, data types and constraints of data if any. CREATE TABLE command is used for DDL. Let us see with an example how a table is created.

Creating New Tables

CREATE TABLE statement is used to create a new database table in the predefined format. The General CREATE TABLE statement is:

CREATE TABLE *<Table Name>*

(<Column Name> <Data Type> [(<Size>)] <Column Constraint>,

...other columns);

The CREATE command had two major parts.

1) The clause CREATE TABLE plus the name of the table.
2) A list of columns, including their names, types, lengths, and whether null values allowed. The entire list is within a pair of parentheses.

Some common generic data types are:

Char(x)	A column of characters, where x is a number designating the maximum number of characters allowed (maximum length) in the column. For example for the name of the publisher if a maximum of 100 characters are specified, it means that the table can hold only up to 100 characters for publisher name.
Integer	A column of whole numbers, positive or negative. For example, the number of copies of a book or the number of library cards for a member, etc., can be defined as integer.
Decimal(x, y)	A column of decimal numbers, where x is the maximum length in digits of the decimal numbers in this column, and y

	is the maximum number of digits allowed after the decimal point. The maximum number that can be fit into (4, 2) is 99.99.
Date	A date column in a DBMS-specific format. Each Date is having three components that are day, month, and year. Date data type includes all these components in number format. For example Issue date can be expressed as a date data type. It means that it should have all the three elements in number formats, i.e., day, month and year, say 02/10/2002.
Logical	A column that can hold only two values: TRUE or FALSE. For example the information whether a library member paid his fee or not can be defined as a logical data type. This will contain the data as either Yes or No which is equivalent to True or False respectively.

Example :

```
CREATE TABLE publisher

(PublisherID INTEGER NOT NULL, Name  CHAR(100) NOT NULL,

Place CHAR(25), Country CHAR(20) NOT NULL);
```

This statement gives the table name and tells the DBMS about each column in the table. This statement will create a new table namely publisher with the information fields such as publisher Identification number, publisher name, publisher's place, and the country of the publisher. The 'NOT NULL' phrase means that the column *must have a value in each row*. If NULL was used or nothing is specified, that column may be left empty in a given row.

PublisherID	Name	Place	Country

Altering Tables

The ALTER TABLE statement allows to add or delete a column or columns from a table, or to change the specification (data type, etc.) on column of an existing table. The general format of ALTER TABLE statement is:

```
ALTER TABLE <Table Name> ADD|DROP|MODIFY ((<Column Name>
<Data Type> [(<Size>)] <Column Constraint>, … other columns)
```

Only one option can be performed for each ALTER TABLE statement --either ADD, DROP, or MODIFY in a single statement.

Example :

a) ALTER TABLE Publisher ADD (Pincode CHAR (10),

   Phone Number CHAR(10));

This statement will add two additional information fields to the publisher table created earlier by using Create statement, i.e., pin code and phone number.

b) ALTER TABLE Publisher MODIFY (Name CHAR (120));

   This statement will change the size of the field name from 100 to 120.

c) ALTER TABLE Publisher DROP (Pincode CHAR (10));

This statement will remove the field pin code from the publisher table.

Self Check Exercise

1) A CREATE TABLE command must contain four elemens. What are they?

2) What does the specification NOT NULL indicate?

3) Write SQL statements for:

   a) Define a table of relation ISSUERECEIPT with Journal ID, Volume Number, Issue Number, Publication Date, Receiving Date, and Accompanying Material.

   b) Include Number of pages and Language to the above table

   c) Remove the Accompanying Material from the table.

   d) Define a table for Items Details of a store with Item code, name of the item, unit Price of the item, quantity available

   e) Include reorder level of the item in the above table

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

....

....

....

....

....

12.3 DATA MANIPULATION LANGUAGE

The Data manipulation Language allows adding, modifying and deleting the content of a database table. The SQL commands used for this are INSERT INTO, UPDATE and DELETE FROM respectively.

The DML also allows retrieving the content of the table. The most important command of these is the SELECT statement, which allows the data retrieval on various conditions and style by using various clauses along with it such as WHERE, LIKE, ORDER BY etc. Some places the SELECT statement is referred as **QML** stands for **Query Manipulation Language.**

Adding Data

To add rows into a table, the *INSERT* statement is used. The general format of this statement is

INSERT INTO *<Table Name> [(<Column List>)]*

VALUES *(<Value List>);*

Examples:

a) INSERT INTO Publisher (101, 'IGNOU', 'New Delhi', 'India', '6961845');

This inserts the data into the table, as a new row, column-by-column, in the pre-defined order. The attribute whose values are not mentioned will have null value. In order to change the order of the columns or to enter values in some selected columns, the columns name should be mentioned.

PublisherID	Name	Place	Country	Pincode
101	IGNOU	New Delhi	India	6961845

b) INSERT INTO Publisher (Publisher Id, Country, Name)

VALUES (102, 'India', 'Young Printers');

The values of Publisher place, pin code and phone number are left blank.

PublisherID	Name	Place	Country	Pincode
101	IGNOU	New Delhi	India	6961845
102	Young Printers		India	

Deleting Data

To remove rows from a table, The DELETE statement is used. The general format of this statement is:

DELETE FROM *Table Name* WHERE *Condition.*

Example:

DELETE FROM Publisher

WHERE Place = 'New Delhi';

This will delete all the rows that contain 'New Delhi', as publisher place.

PublisherID	Name	Place	Country	Pincode
102	Young Printers		India	

In order to delete all the records in the table, don't mention the WHERE condition. For example:

DELETE FROM publisher;

This will remove all the rows from publisher table.

PublisherID	Name	Place	Country	Pincode

Updating Data

The UPDATE statement is used to update or change records that match the specified criteria. This is accomplished by carefully constructing a where clause.

UPDATE *<Table Name>*

SET *<Column name> = <value>,[,<Column name> = <value>,..]*

WHERE Condition

Example:

UPDATE Publisher SET PhoneNumber = '1234567'

WHERE Name = "IGNOU";

PublisherID	Name	Place	Country	Pincode
101	IGNOU	New Delhi	India	1234567
102	Young Printers		India	

If the WHERE clause is left out, all rows will be updated according to the SET statement. For Example

UPDATE Publisher SET Place="New Delhi";

This will insert "New Delhi" in place column of all rows.

PublisherID	Name	Place	Country	Pincode
101	IGNOU	New Delhi	India	6961845
102	Young Printers	New Delhi	India	

Self Check Exercise

1) Write SQL statements to
   a) Insert data into the above example table
   b) Insert data into the IssueReciept table of self exercise
   c) Insert data into the ItemDetails table of self exercise
   d) Change the publisher place into 'Kolkata' instead of Calcutta
   e) Change the publisher place into 'Chennai' instead of Madras
   f) Give a initial quantity of all items as 10 in Item Details

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

....

....

....

12.3.1 SELECT Statement

The SELECT statement is used to query the database and retrieve selected data that match the criteria that you specify. The general format of the SELECT statement is:

```
SELECT [ALL | DISTINCT] column1 [, column2]
FROM table1 [, table2]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions"]
[ORDER BY "column-list" [ASC | DESC]]
```

Where **ASC|DESC** allows the ordering to be done in Ascending or Descending order.

The SELECT statement has five main clauses to choose from, although, FROM is the only required clause. Each of the clauses has a vast selection of options, parameters, etc.

Consider the table "serials detail" given below as an example, relate ISSN, Title of the serial, Publisher of the serial, Subject, Keywords and Price.

Serials Detail table

ISSN	Title	Publisher	Subject	Keywords	Price
0001 253X	ASLIB Proceedings	Portland Press Ltd.	Library Science	Conference	270.00
0002-7969	American Libraries	SOE Human st.	Library Science	Library automation, information technology	70.00
1041-7915	Computer Libraries	Information Today Inc.	Science	Information Technology.	92.95
0340-0352	IFLA Journal	KG saur Verlag	Library Science	Library Associations	80.85
1057-2317	The International information & library review	Academic Press	Information science	Library, information, technology, Bibliometrics	195.00
1532-2882	JASIST	At.Wiley Interscience	Science	Technology, Information Science, Management	1490
3300-2887	Scientrometrics	Kluwer Academic Publisher	Scientro-metrics	Bibliometrics, Citation	1214
0018-0528	Herald of Library Science	Lucknow University	Library Science	Citation Analysis, Library automation	10
0018-8441	IASLIC Bulletin	ISALIC	Library	Library Associations, Management	5
0970-47728	ILA bulletin	ILA	Library Association	Library Science, Library Management, Associations	55

Now, to see the subject and keywords of each serial, use the SELECT statement as:

```
SELECT Title, subject, keywords
FROM serialdetail;
```

The following is the results of your *query:*

Title	**Subject**	**Keyword**
ASLIB Proceedings	Library Science	Conference
American Libraries	Library Science	Library automation, information technology
Computer Libraries	Science	Information Technology.
IFLA Journal	Library Science	Library Associations
The International information & library review	Information science	Library, information, technology
JASIST	Science	Technology, Information Science, Management
Scientrometrics	Scientrometrics	Bibliometrics, Citation
Herald of Library Science	Library Science	Citation Analysis, Library automation
IASLIC Bulletin	Library	Library Associations, Management
ILA bulletin	Library Association	Library Science, Library Management, Associations

To get all columns of a table without typing all column names the asterisks (wild card) can be used. For example:

SE`LECT * FROM *TableName;*

Conditional Selection or WHERE Clause

Consider another table of library members

membership

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto	Books Issued
805	Chandra Bhushan	Singh	Scientist C	A	31/12/2005	4
810	Mukesh	Pund	Scientist B	A	31/12/2003	6
667	K.S.R	Nair	Stenographer	C	30/04/2004	1
696	Vinod Kumar	Gupta	Scientist E11	A	31/12/2002	3
787	Salim	Ansari	JSA	B	31/12/2002	6
507	Narendra	Kumar	Scientist E1	A	31//03/2005	2
730	Kamla	Kujur	JSA	C	31/12/2006	8
540	RajeSingh	Chuahan	UDC	C	31/03/2004	4
623	C. R. R	Pillai	KPO	D	31/12/2005	2

The *WHERE* clause is used to specify that only certain rows of the table are displayed, based on the criteria described in that *WHERE* clause. For example if you wanted to see the Member's IDNos, who have issued 5 or more books, use the following:

SELECT **MemberIDNo**

FROM membership

WHERE booksissued>=5;

Notice that the >= (greater than or equal to) sign is used, as we wanted to see those who issued 5 or more books. This displays:

```
Member ID No
787
730
```

Relational Operators

There are six Relational Operators in SQL, they are

=	**Equal**
<> or != (see manual)	**Not Equal**
<	**Less Than**
>	**Greater Than**
<=	**Less Than or Equal To**
>=	**Greater Than or Equal To**

The *Where* description, booksissued>=5, is known as a *condition* (an operation which evaluates to True or False). The same can be done for text columns. The text values should be enclosed in qoates.

SELECT MemberIDNo
FROM membership
WHERE MembershipType="A";

This displays the ID Numbers of Category 'A' members. As given below

```
MemberIDNo
805
810
696
507
```

The AND operator joins two or more conditions, and displays a row only if that row's data satisfies ALL conditions listed (i.e. all conditions hold true). Both sides of the AND condition must be true in order for the condition to be met and for those rows to be displayed. For example, to display all type 'A' members, who issued 2 or more books, use:

```
SELECT MemberIDNo
FROM membership
WHERE NoofbooksIssued>=2 AND Member Type='A';
```

Result:

MemberIDNo
805
810
696

The OR operator joins two or more conditions, but returns a row if ANY of the conditions listed hold true. However, either side of the OR operator can be true and the condition will be met, hence, the rows will be displayed. With the OR operator, either side can be true or both sides can be true. To see all who issued less than 5 books or category ‘C’ members, use the following query:

```
SELECT MemberIDNo,
FROM Membership
WHERE BooksIssued>2 OR Member Type=‘C’;
```

Result:

MemberIDNo
805
667
810
696
730
540

You can combine AND & OR for example:

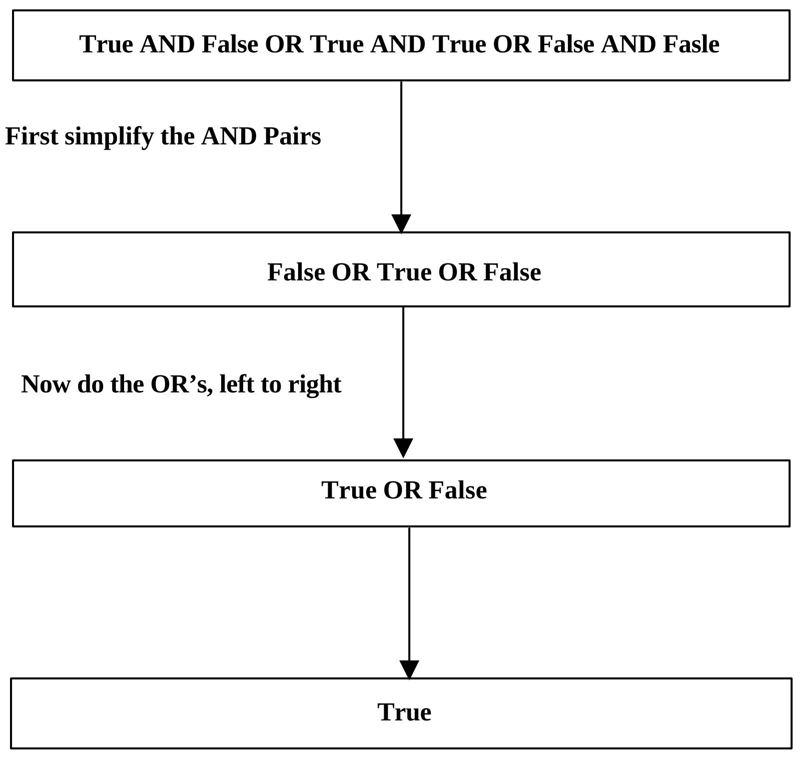
```
SELECT MemberIDNo,
FROM membership
WHERE BooksIssued<5 AND validUpto = 31/12/2002 OR Member Type
= ‘C’
```

Result:

MemberIDNO
667
696
730
540

First, SQL finds the rows where the Books issued is less than 5 and valid upto 31/12/2002 then taking this new list of rows, SQL then sees if any of these rows satisfies the previous AND condition or the condition that the member type equal to C. Subsequently, SQL only displays this second new list of rows, keeping in mind that anyone with member type ‘C’ will be included as the OR operator includes a row if either resulting condition is True.

To generalize this process, SQL performs the AND operations to determine the rows where the AND operations hold true then these results are used to compare with the OR conditions. SQL only displays those remaining rows where any of the conditions joined by the OR operator hold true. Mathematically, SQL evaluates all of the conditions, then evaluates the AND “pairs”, and then evaluates the OR’s. Both the operators are evaluated from left to right. To have an example, for a given row for which the DBMS is evaluating the SQL statement WHERE clause to determine whether to include the row in the query result (the whole WHERE clause evaluates to True), the DBMS has evaluated all of the conditions, and is ready to do the logical comparisons on this result:


True AND False OR True AND True OR False AND Fasle
First simplify the AND Pairs
False OR True OR False
Now do the OR's, left to right
True OR False
True

The result is True, and the row passes the query conditions. Be sure to see the next section on NOT's, and the order of logical operations.

If you want to perform OR's before AND's, then parentheses are to be used.

NOT is a unary operator that evaluates one condition, reversing its value, whereas, AND's & OR's evaluate two conditions. All NOT's are performed before any AND's or OR's. SQL order of Logical Operations (each operates from left to right) is:

1) NOT 2) AND 3) OR

IN & BETWEEN

An easier method of using compound conditions uses IN or BETWEEN. The IN conditional operator is really a set membership test operator. That is, it is used to test whether or not a value (stated before the keyword IN) is "in" the list of values provided after the keyword IN.

```
SELECT column-list
FROM "list-of-tables"
WHERE column IN (list-of-values);
```

For example, if you wanted to list all Stenographer and UDC (refer to membership table):

```
SELECT Firstname, LastName
FROM Membership
WHERE Designation IN ('Stenographer', 'UDC');
```

This gives the same result of the compound condition statement using OR:

```
SELECT Firstname, LastName
FROM Membership
WHERE Designation = 'stenographer' OR Designation= 'UDC'
```

Result:

FirstName	LastName
K.S.R	Nair
RajeSingh	Chuahan

As you can see, the IN operator is much shorter and easier to read when you are testing for more than two or three values. You can also use **NOT IN** to exclude the rows in your list.

The **BETWEEN** conditional operator is used to test to see whether or not a value (stated before the keyword **BETWEEN**) is "between" the two values stated after the keyword **BETWEEN**

```
SELECT column-list
FROM "list-of-tables"
WHERE column BETWEEN value1 AND value2
```

For example to list those who issued 3 to 10 books, use:

```
SELECT FirstName, LastName
FROM Membership
WHERE BooksIssued BETWEEN 3 AND 10;
```

Result:

FirstName	LastName
Chandra Bhushan	Singh
Mukesh	Pund
Vinod Kumar	Gupta
Salim	Ansari
Kamla	Kujur
RajeSingh	Chuahan

This will give the same result as the following statement without BETWEEN

```
SELECT First Name, Last Name
FROM Membership
WHERE BooksIssued >=3 AND BooksIssued <=10;
```

LIKE Clause

From the Membership table, to list out all people whose designation started with "Sci"; use

```
SELECT FirstName, LastName
FROM Membership
WHERE Designation LIKE 'Sci%';
```

Result:

First Name	Last Name
Chandra Bhushan	Singh
Mukesh	Pund
Vinod Kumar	Gupta
Narendra	Kumar

The percent sign (%) is used to represent any possible character (number, letter, or punctuation) or set of characters that might appear after the "S". To find those people with Last Name ending in "S", use '%S', or if you wanted the "S" in the middle of the word, try '%S%'. The '%' can be used for any characters in the same position relative to the given characters. NOT LIKE displays rows not fitting the given description.

The DISTINCT clause is used to select the "distinct" or unique records in the query results. If you would like to retrieve just the unique records in specified columns, you can use the "DISTINCT" keyword. DISTINCT will discard the duplicate records for the columns you specified after the "SELECT" statement: For example:

```
SELECT DISTINCT MemberType
FROM Membership;
```

This statement will return all of the unique Membership Types in the table.

12.4.2 Aggregate Functions

There are five important *aggregate functions:* SUM, AVG, MAX, MIN, and COUNT. They are called aggregate functions because they summarize the results of a query, rather than listing all of the rows.

SUM ()	gives the total of all the rows, satisfying any conditions, of the given column, where the given column is numeric.
AVG ()	gives the average of the given column.
MAX ()	gives the largest figure in the given column.
MIN ()	gives the smallest figure in the given column.
COUNT(*)	gives the number of rows satisfying the conditions.

Looking at the tables at the top of the document, let's look at three examples:

```
SELECET SUM (BooksIssued), AVG (BooksIssued)
FROM Membership;
```

This query shows the total of the Issued Books in the table, and the average books Issued by a member in the table.

Result:

SUM(BooksIssued)	AVG(BooksIssued)
36	4

```
SELECT MIN (BooksIssued)
FROM Membership
WHERE MemberType='C';
```

Result:

MIN(BooksIssued)
1

This query gives the Lowest number of the Books Issued by a Category "C" member.

```
SELECT COUNT (*)
FROM Membership
WHERE MemberType='A';
```

Result:

COUNT(*)
3

This query tells you how many Type 'A' members are there in the table.

GROUP BY

The GROUP BY clause will gather all of the rows together that contain data in the specified column(s). The syntax is:

SELECT *column list*
FROM *list-of tables*
GROUP BY *column list*

This can best be explained by an example:

Let's say you would like to retrieve a list of the maximum number of books issued in each member type:

```
SELECT MAX (BooksIssued), MemberType
FROM Membership
GROUP BY MemberType;
```

This statement will select the maximum books issued for the members in each unique Member type.

Result:

MAX(BooksIssued)	**MemberType**
6	A
6	B
8	C
2	D

HAVING

The HAVING clause allows you to specify conditions on the rows for each group - in other words, which rows should be selected will be based on the specified conditions. The HAVING clause should follow the GROUPBY clause. The syntax is:

```
SELECT column list
FROM list-of tables
GROUP BY column list
HAVING Condition
```

Example:

```
SELECT MemberType, AVG(BooksIssued)
FROM Membership
GROUP BY Member Type
HAVING AVG(BooksIssued) > 4;
```

Result:

MemberType	**AVG(Books Issued)**
B	6
C	4.33

Self Check Exercise

1) Write SQL statements to

   a) list all items with reorder level less than 2 Item Details table

   b) list all items in alphabetical order from Item Details table

   c) to find out which item has the maximum available quantity from Item Details table

   d) list the journals with CDs accompanied with them from IssueReceipt table

   e) list the unique journals in different language group from SerialsDetails table

   f) list the serials in different subject group from SerialsDetails table

   g) On which subjects the serials are available from SerialsDetails table

   h) How many Library science serials are available which costs less than 50$ from SerialsDetails table.

2) What are the possible value constraints of an attribute?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

....

....

....

....

....

12.4 JOINS

Good database design suggests that each table lists the data about a single entity only, and detailed information can be obtained in a relational database, by using additional tables, and by using a join. A join is used when a SQL query requires data from more than one table on a database. Rows in one table may be joined with rows in another according to common values existing in corresponding columns. There are two types of Joins condition: equi-join and non_equi_join.

When two tables are joined, they must share a common key or join key, which defines how rows in the table corresponds to each other.

First, take a look at these example tables:

SerialsDetails

ISSN	Title	Subject	Type
0001253X	ASLIB Proceedings	Library Science	P
0002-7969	American Libraries	Library Science	P
1041-7915	Computer Libraries	Science	S
0340-0352	IFLA Journal	Library Science	P
1057-2317	The International information & library review	Information science	P

Publisher

PublisherID	Name	Address	Country
001	Portland Press Ltd.	Whitehall Industrial Estate, Colchester	United Kingdom
002	SOE Human Street.	Chicago, IL 60611	United States of America
003	Information Today Inc.	143 old Marton pipe, Medford	New Jersi
004	KG Saur Verlag	Ortester 8,munchem	France
005	Academic Press	32 James Town Road, London	United Kingdom
006	At.Wiley Interscience	1320 fenwick lane, silverspring	United Kingdom
007	Kluwer Academic Publisher	3300 AA dorrecht	NetherLand
008	Lucknow University	C-239, Indira Nagar, Lucknow	India
009	ISALIC	Kankurgati, Calcutta	India
010	ILA	Flat 201, Ansal Building, Mukherjee Nagar, Delhi	India

Acquisition

serialID	VendorID	Copies
0001253X	001	1
1041-7915	003	3
1057-2317	005	1
0002-7969	010	12
0340-0352	004	6
002-7969	002	2

Keys

A *primary key* is a column or set of columns that uniquely identifies the rest of the data in any given row. For example, in the Serials Detail table, the ISSN column uniquely identifies that row. This means two things: no two rows can have the same ISSN, and, even if two serials have the same Title, subject and type, the ISSN column ensures that the two serials are different.

A *foreign key* is a column in a table where that column is a primary key of another table, which means that any data in a foreign key column must have corresponding data in the other table where that column is the primary key. This correspondence is known as *referential integrity* in DBMS. For example, in the acquisition table, both the ISSN and VenderID are foreign keys to Serials Detail and Publisher tables respectively. This referential integrity helps in deletion and up-dation as a row from the primary key column is deleted or updated, the value of the foreign key in those other tables will also be deleted or updated.

The purpose of these *keys* is that data can be related across tables, without having to repeat data in every table; this is the power of relational databases. The term "relationships" (often termed "relation") usually refers to the relationships among primary and foreign keys between tables. This concept is important because when the tables of a relational database are designed, these relationships must be defined because they determine which columns are or are not primary or foreign keys.

A *One-to-one relationship* means that you have a primary key column that is related to a foreign key column, and that for every primary key value, there is one foreign key value.

One-to-many relationship ("1-M") means that for every column value in one table, there is **one or more** related values in another table. Key constraints may be added to the design, or possibly just the use of some sort of identifier column may be used to establish the relationship.

Many-to-many relationship ("M-M") does not involve keys generally, and usually involves identifying columns. The unusual occurrence of a "M-M" means that one column in one table is related to another column in another table, and for every value of one of these two columns, there are one or more related values in the corresponding column in the other table (and vice-versa), or a more common possibility, two tables have a 1-M relationship to each other (two relationships, one 1-M going each way). **(Refer Block 3)**

Performing a Join

In order to select from multiple tables, we need to "join" them. The join appears in the WHERE clause. It defines the relationship between the tables. If we don't join the tables, the query results will not be as we expect. The join should be done on the key of the table, which may involve more than one field.

Joins in the WHERE clause are most often 'equi-joins', joining on equality.

```
SELECT Title, Subject, copies
FROM serialsdetails, acquisition
WHERE    ISSN=serialID
```

When joining two tables, one join condition is to be specified in order to avoid cartion product. When joining three tables, we must specify two join conditions. As a general rule, when joining n tables, we must specify n-1 join conditions.

12.5 SUB QUERIES

A sub query is a query (select) which is used as part of another SQL statement. It is used in the WHERE clause to include or exclude rows for output.

Consider two tables Member and Circulation

Member

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
507	Narendra	Kumar	Scientist E1	A	31//03/2005
540	RajeSingh	Chuahan	UDC	C	31/03/2004
623	C. R. R	Pillai	KPO	D	31/12/2005
667	K.S.R	Nair	Stenographer	C	30/04/2004
696	Vinod Kumar	Gupta	Scientist E11	A	31/12/2002
730	Kamla	Kujur	JSA	C	31/12/2006
787	Salim	Ansari	JSA	B	31/12/2002
805	Chandra Bhushan	Singh	Scientist C	A	31/12/2005
810	Mukesh	Pund	Scientist B	A	31/12/2003

Circulation

IDNo	Books Issued
805	4
810	6
667	1
696	3
787	6
507	2
730	8
540	4
623	2

You can use a subquery to get a single value for WHERE clause criteria.

Example:

```
SELECT * FROM member
WHERE BooksIssued =
(SELECT MAX(BooksIssued) FROM Circulation)
```

This query will return all of the Membership details information for all the members who issued the maximum number of books. This example uses equality (=) as the sub query join, but you can use any valid single-value logical operator on a sub query.

Result :

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
730	Kamla	Kujur	JSA	C	31/12/2006

You can also use a sub query to get a list of values for WHERE clause criteria. It is done in the same manner as shown above, but uses a many-value logical operator.

```
SELECT * FROM member
WHERE MemberID IN
(SELECT MemberID FROM Circulation WHERE BooksIssued>5)
```

Result :

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
730	Kamla	Kujur	JSA	C	31/12/2006
787	Salim	Ansari	JSA	B	31/12/2002
810	Mukesh	Pund	Scientist B	A	31/12/2003

This example uses IN as the sub query join, but you can use any valid many-value logical operator on a sub query.

The sub query either must have only one column, or it must compare its selected columns to multiple columns in parentheses in the main query. The sub query must be enclosed in parentheses. Sub queries that produce only one row can be used with either single- or many-value operators. Sub queries that produce more than one row can be used only with many-value operators. BETWEEN cannot be used with a sub query. All other many-value operators will work.

12.6 UNION

To combine the result of multiple queries together, UNION can be used. UNION is equivalent to OR. It will find every record that matches either of the sub queries to which it applies.

For Example: To merge the output of the following two queries:

```
SELECT * FROM member
WHERE MemberID IN
(SELECT MemberID FROM Circulation WHERE BooksIssued>5)

UNION
SELECT * FROM member
WHERE Membertype = “D”
```

Result:

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
730	Kamla	Kujur	JSA	C	31/12/2006
787	Salim	Ansari	JSA	B	31/12/2002
810	Mukesh	Pund	Scientist B	A	31/12/2003
623	C. R. R	Pillai	KPO	D	31/12/2005

12.7 VIEWS

View allows assigning the results of a query to a virtual table that can be used in other queries, where this new table is given the view name in FROM clause. A view includes only certain columns and rows from one or many tables. When you access a view, the query that is defined in your view creation statement is performed (generally), and the results of that query look just like another table in the query that you wrote invoking the view.

Views can be used to restrict database access, as well as to simplify a complex query. A view that is restricted to certain rows is called *Horizontal* view and a *Vertical* views restricted to certain columns. The general structure of VIEW statement is:

CREATE VIEW *View Name AS query expression*

For example, to create a view:

CREATE VIEW Science AS SELECT issn FROM serialDetails WHERE Subject=‘Science’;

Science

ISSN
1041-7915
1057-2317
1532-2882

This view can be used to write another query.

```
SELECT VendorID
FROM aqucisition, Science
WHERE serialID = science.issn;
```

This query shows all vendors of Science serials.

SerialID	VendorID	Copies
1041-7915	003	3
1057-2317	005	1

12.8 NON-STANDARD SQL

Some non-standard SQL features available with various DMBS are discussed here.

Index

A database index is like an index to a book. In a database index , for every item in the index field, an index file holds a key to its location in the table or database file. The key is like the page number in a printed book index, it leads directly to the proper place. Without an index, to search a particular term the computer has to search the entire file item by item.

Index allows a DBMS to access data quicker particularly when the database is large. The system creates this internal data structure (the index) which causes selection of rows, when the selection is based on indexed columns, to occur faster. Given an indexed-column value, this index tells the DBMS where a certain row is in the table, much like a book index tells you what page a given word appears.

Example:

1) CREATE INDEX member_idx ON
   Member(IDNo);
   SELECT * FROM PMember

Result:

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
507	Narendra	Kumar	Scientist E1	A	31//03/2005
540	RajeSingh	Chuahan	UDC	C	31/03/2004
623	C. R. R	Pillai	KPO	D	31/12/2005
667	K.S.R	Nair	Stenographer	C	30/04/2004
696	Vinod Kumar	Gupta	Scientist E11	A	31/12/2002
730	Kamla	Kujur	JSA	C	31/12/2006
787	Salim	Ansari	JSA	B	31/12/2002
805	Chandra Bhushan	Singh	Scientist C	A	31/12/2005
810	Mukesh	Pund	Scientist B	A	31/12/2003

2) CREATE INDEX Membername_idx ON Member (First,last);

   SELECT * FROM PMember

Result:

IDNo	FirstName	LastName	Designation	Member Type	ValidUpto
623	C. R. R	Pillai	KPO	D	31/12/2005
805	Chandra Bhushan	Singh	Scientist C	A	31/12/2005
667	K.S.R	Nair	Stenographer	C	30/04/2004
730	Kamla	Kujur	JSA	C	31/12/2006
810	Mukesh	Pund	Scientist B	A	31/12/2003
507	Narendra	Kumar	Scientist E1	A	31//03/2005
540	RajeSingh	Chuahan	UDC	C	31/03/2004
787	Salim	Ansari	JSA	B	31/12/2002
696	Vinod Kumar	Gupta	Scientist E11	A	31/12/2002

To get rid of an index, drop it:

DROP INDEX Vendor_idx;

INTERSECT

INTERSECT and MINUS are like the UNION statement, except that INTERSECT produces rows that appear in both queries, and MINUS produces rows that result from the first query, but not the second.

DECODE in Oracle

DECODE allows conditional output based on the value of a column or function. DECODES can be deeply nested and can have an unlimited number of "if then" values. The syntax looks like this

SELECT ...DECODE (Value, If1, Then1, [If 2, Then 2, ...,] Else) ...From ...;

This is evaluated as: Look at the value of 'field' in this record. If this is equal to if1, then return then1, if the value of field is equal to if2 then return then2. If the field doesn't match any value specified the else value is returned

Example:

Select Designation,

DECODE (Designation, 'Scientist', 'A', 'JSA', 'B', 'UDC', 'C', 'LDC', 'C', 'Stenographer', 'C','D') AS MemberType

FROM MemberDetails;

Result:

Designation	MemberType
Scientist	A
Scientist	A
Stenographer	C
KPO	D

Scientist	A
JSA	B
Scientist	A
JSA	B

Report Generation Features

The COMPUTE clause is placed at the end of a query to place the result of an aggregate function at the end of a listing, like COMPUTE SUM **(PRICE);**

Additional Functions

In addition to the above listed aggregate functions, some DBMSs allow more functions to be used in Select lists, except that these functions (some character functions allow multiple-row results) are to be used with an individual value (not groups), on single-row queries. The functions are to be used only on appropriate data types, also. Here are some

Mathematical Functions:

ABS (X)	Absolute value - converts negative numbers to positive, or leaves positive numbers alone
CEIL (X)	X is a decimal value that will be rounded up.
FLOOR (X)	X is a decimal value that will be rounded down.
GREATEST(X,Y)	Returns the largest of the two values.
LEAST (X,Y)	Returns the smallest of the two values.
MOD(X,Y)	Returns the remainder of X / Y.
POWER(X,Y)	Returns X to the power of Y.
ROUND(X,Y)	Rounds X to Y decimal places. If Y is omitted, X is rounded to the nearest integer.
SIGN(X)	Returns a minus if X < 0, else a plus.
SQRT(X)	Returns the square root of X.

Character Functions

LEFT (<string>,X)	Returns the leftmost X characters of the string.
RIGHT (<string>,X)	Returns the rightmost X characters of the string.
UPPER (<string>)	Converts the string to all uppercase letters.
LOWER (<string>)	Converts the string to all lowercase letters.
IINITCAP (<string>)	Converts the string to initial caps.
LENGTH (<string>)	Returns the number of characters in the string.

<string>||<string>	Combines the two strings of text into one, concatenated string, where the first string is immediately followed by the second.
LPAD (<string>, X,'*')	Pads the string on the left with the * (or whatever character is inside the quotes), to make the string X characters long.
RPAD (<string>,X,'*')	Pads the string on the right with the * (or whatever character is inside the quotes), to make the string X characters long.
SUBSTR (<string>,X,Y)	Extracts Y letters from the string beginning at position X.
NVL(<column>,<value>)	The Null value function will substitute <value> for any NULLs for in the <column>. If the current value of <column> is not NULL, NVL has no effect.

Self Check Exercise

1) Can foreign key contain NULL?
2) Can a table have two Primary Keys?
3) In what order does SQL process nested queries?
4) Using the tables in the in the example, write a SQL statement to retrieve:
   a) all publishers from 'India'
   b) all publishers from Calcutta city
   c) all foreign publishers
   d) who supplied more than five copies of their journal
   e) alphabetical list of members
   f) list of members in the order of membership type.
   g) list of members in the order of membership type and then by designation.
   h) List the publishers, who publishes the journals on library Science
5) What is the distinction of the command SELECT DISTINCT?
6) What is the difference between an equi-join and non-equi-join?
7) UNION is equivalent to which comparison operator?
8) By connecting two SQL statements with INTERSECT which records can be retrieved?
9) What is a computed or calculated column ?
10) What is the difference between ROUND( ) and TRUNC( ) ?
11) What are the main reasons to use views?

12) When a table is updated, what happens to any views based on that table?

13) Can a table be updated by updating a view derived from that table?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of this unit.

....

....

....

....

....

....

12.9 SUMMARY

SQL is a standard query language used in most commercial DBMSs. Other query languages for different models are also evolved. The important one of them for the object model system is AQL.

In the era of Internet, as the Web contains an abundance of useful semi-structured information that needs to be mined, various query languages have emerged for the web. Squeal, which facilitates structure-based queries, is an important one among them. The Squeal user can query the Web as if it were in a standard relational database. An example query of Sequel will explain its resemblance with SQL.

```
SELECT url
FROM page p, tag t
WHERE p.contents LIKE "%hypertext%"
AND t.url = p.url
AND t.name = "IMG";
```

XML is a new standard that supports data exchange on the World-Wide Web. It is likely to become as important and as widely used as HTML. W3C proposes a query language for XML, called XML-QL for extracting data from large XML documents, for translating XML data between different ontologies (DTD's), for integrating XML data from multiple XML sources, and for transporting large amounts of XML data to clients or for sending queries to XML sources. The language has a SELECT-WHERE construct, like SQL.

Example:

```
WHERE <pub> &p </> in "www.a.b.c/bib.xml",
<title> $t </> in $p,
<year> $y </> in $p
<month> $z </> in $p
ORDER-By $y,$z
CONSTRUCT $t
```

XML-QL and sequel provides support for querying the semi-structured data. A lot more of query languages are emerging for querying the large amount of semi-structured data available on the Web. Almost all of these languages are taking ideas from SQL for clauses and keywords like SELECT, WHERE and ORDER BY etc. to provide a pattern for restructuring data.

12.10 ANSWERS TO SELF CHECK EXERCISES

Exercise I

1) a) CREATE TABLE Issuereceipt(journalID INTEGER NOT NULL, Volumenumber INTEGER, IssueNumber INTEGER, publication Date DATE, receivingdate DATE, accmaterial CHAR (60))

b) ALTER TABLE Issuereceipt ADD (numberofpages INTEGER, language CHAR (20))

c) ALTER TABLE Issuereceipt DROP (accmaterial CHAR (60))

d) CREATE TABLE Items Details (Itemcode INTEGER NOT NULL, itemname CHAR (30), itemPrice DECIMAL (10,2), quantity integer)

e) ALTER TABLE Items Details ADD (reorderlevel INTEGER)

Exercise II

1) INSERT INTO Publisher (1001,“abc publications”, “newdelhi”, “india”, 6964968)

2) INSERT INTO IssueReciept (201,4,12,“01/01/97”,300,“english”,“cdrom”)

3) INSERT INTO Itemdetail (900,“washing soap”,23.00,102,50)

4) UPDATE TABLE Publisher SET place=“Kolkata” WHERE place=“Calcutta”

5) UPDATE TABLE Publisher SET place=“Chennai” WHERE place=“Madras”

6) UPDATE TABLE Itemdetail SET quantity=10

Exercise III

1) a) SELECT * FROM Itemdetail where reorderlevel<2

b) SELECT * FROM Itemdetails ORDERBY Itemname ASC.

c) SELECT * FROM Itemdetails where quantity=Max (quantity)

d) SELECT * FROM IssueReceipt WHERE accmaterial= “CD”

e) SELECT * FROM Itemdetails where quantity=Max (quantity)

f) SELECT DISTINCT * FROM Serialdetails GROUP BY Language

g) SELECT DISTINCT subjects FROM Serialtable

h) SELECT COUNT (name) HAVING subject=“Library Science”

2) - NULL OR NOT NULL

   - UNIQUE enforces that no two rows will have the same value for this column

   - PRIMARY KEY tells the database that this column is the primary key column

   - CHECK allows a condition to be checked for when data in that column is updated or inserted; for example, CHECK (PRICE 0) causes the system to check that the Price column is greater than zero before accepting the value...sometimes implemented as the CONSTRAINT statement.

   - DEFAULT inserts the default value into the database if a row is inserted without that column's data being inserted; for example, Member Fee INTEGER DEFAULT = 100. This will insert 100 on memberFee column in all rows otherwise not mentioned.

   - FOREIGN KEY works the same as Primary Key, but is followed by: REFERENCES <TABLE NAME> (<COLUMN NAME>), which refers to the referential primary key.

Exercise IV

1) Foreign Key is the primary key of another table. It can have NULL value, which represent that there is no corresponding row in the other table.

2) No. There could be more than one unique attribute for a table, which can act as primary key in a database. But only one should be defined as primary in order to maintain the integrity of the database. The other unique attributes are known as alternate keys.

3) From innermost parentheses, working outward.

4) a) SELECT NAME FROM Publisher WHERE country = "INDIA"

   b) SELECT NAME FROM Publisher WHERE address like "CALCUTTA"

   c) SELECT NAME FROM Publisher WHERE country NOT LIKE "INDIA"

   d) SELECT NAME FROM Publisher, acquisition WHERE publisher. publisherID = Acquisition.vendorID AND HAVING copies>5

   e) country = "INDIA"

   f) SELECT firstname, lastname FROM membership table GROUP BY membertype.

   g) SELECT firstname, lastname FROM membership table GROUP BY designation.

   h) SELECT name FROM Publisher, Acquisition, Serialsdetails WHERE serialsdetail.ISSN = acquisition.serialID AND publisher.publisherID = acquisition.vendorID.

5) It will select only one example of a duplicate value.

6) In an equi-join, there is a direct one-to-one relationship between the records in each file. In a non-equi-join, this need not be the case.

7) UNION is equivalent to OR.

8) This will retrieve those records that satisfy both conditions.

9) A column that does not exist in the database but which can be displayed using calculated values.

10) ROUNS ( ) converts the value to the nearest whole number. TRUNC ( ) simply removes the digits to the right of the decimal point.

11) To protect data by restricting access and to make it easier to execute frequently repeated queries.

12) They are automatically updated too.

13) Yes.

12.11 KEYWORDS

ANSI	**:**	American National Standards Institute, founded on October 18, 1918. ANSI is a private, non-profit organization that administers and coordinates the U.S. voluntary standardization and conformity assessment system. The Institute's mission is to enhance both the global competitiveness of U.S. business and the U.S. quality of life by promoting and facilitating voluntary consensus standards and conformity assessment systems, and safeguarding their integrity.
Cartesian Product	**:**	It is a join without a WHERE clause. It gives you every row in the first table, joined with every row in the second table. The number of rows in the result has the number of rows in the first table times the number of rows in the second table, and is sometimes called a Cross-Join.
Data	**:**	The encoded representation of facts, ideas and instructions such that the representation can be

	processed, communicated, and interpreted by computer.
Database Management System (DBMS) :	A DBMS provide facilities that allow the users to deal with data abstract terms without regard for how the data are actually stored or retrieved. It is software used to manipulate and access data stored in a database. It can be designed in hierarchical, network-relational model or object oriented model.
Data Definition Language (DDL) :	A language used to define or describe the database entries and the relationship among he entities during the setup phase of a database management system.
Data Manipulation Language(DML) :	A language used to query the items in a database management system database.
Data Type :	Variable in a programming language defined by the kind of data that can be associated with the variable. Example of data types are logical, integer, character etc.
RDBMS :	Relational Database Management System. The basic structure of a relational database design is a table, known as relation. Each row of a table is known as a tuple. The term attribute and column are also used interchangeably.
Integrity Constraints :	Functional dependencies in a database design that ensure that the database satisfies the intended semantics. Typically integrity constraints restrict attribute value to some range or express some relationship among certain attributes that must be satisfied and maintained.
Internal View :	A low level representation of a database that includes storage and access concern but not

		physical addressing concerns. Internal view would define the records, the indices, the order the records are stored in, etc
SQL	:	Structured Query Language
XML	:	The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web
Internet	:	The largest world wide electronic network, interconnecting thousands of smaller networks and millions of computer users.
QL	:	Query Language
HTML	:	Hyper Text Markup Language is a system of codes made up of tags and attributes that serve to identify parts and characteristics of HTML documents. Some tags provide document structure, others reference while some others provide files. Hyper text documents are electronic documents containing pointers to a web of interrelated documents. By clicking on these links; the related documents can also be accessed.
W3C	:	WWW (World Wide Web) Consortium is a forum for information, commerce, communication, and collective understanding. The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential.
AQL	:	A query Language
XML Query Language	:	The XSL pattern language provides an extremely understandable way to describe a class of nodes to process. It is

declarative rather than procedural. One simply describes the types of nodes to look for using a simple pattern modeled after directory notation. For example, book/ author means find author elements contained in book elements. XQL (XML Query Language) provides a natural extension to the XSL pattern language. It builds upon the capabilities XSL provides for identifying classes of nodes, by adding Boolean logic, filters, indexing into collections of nodes, and more.XQL is designed specifically for XML documents. It is a general purpose query language, providing a single syntax that can be used for queries, addressing, and patterns. XQL is concise, simple, and powerful.

12.12 REFERENCES AND FURTHER READING

CAROLYN WATTERS, Dictionary of Information Science and Technology, Academic Press, Inc,1250,Sixth Avenue, SanDiego, CA9201,1992

Carter, John, Programming in SQL with Oracle, INGRES and dBASE, Blackwell Scientific Publications, Oxford,1992

Jefferey D Ullman, Principles of Databse Systems and Knowledgebase systems, Computer Science Press, Rockville Maryland,1988

Parag Diwan,R.K Suri, Sanjay Kaushik, IT Encyclopedia.com, fundamentals of Information Technology, 2nd Revised Edition, Pentagon Press, New Delhi,2001

Richard H Baker (1993).Guide to SQL database management for IBM PCs & compatibles.London: Scott Foresman and Company.

Sanin, Leo, client /server programming with access and SQL server, Galgotia Publications, New Delhi,1998

Wesley, Massachusetts, 2000.

Rood, Harold, J, Logic and structured design for computer programmers, P WS Kent Publ House, Boston, 1992

Spanior, Otto, Computer arithmetoic logic and design, John Wiley &Sons, Chichester,1981