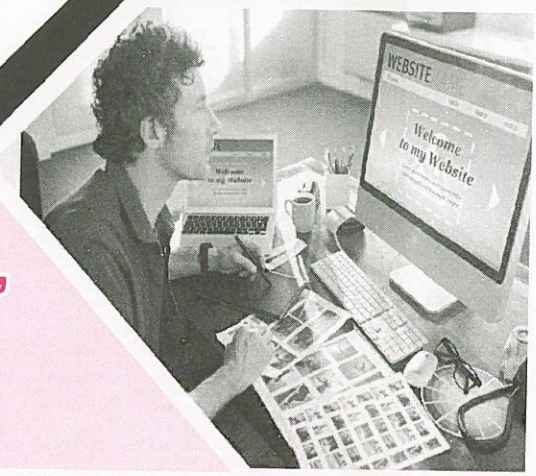
CHAPTER 29

DATABASE OBJECTS, DCL AND TCL STATEMENTS



CHAPTER OBJECTIVES

In this chapter you will learn:

- **Need and usage of View database object**
- **Need and usage of Sequence database object**
- **Need and usage of Index database object**
- **Usage of Transactions Control Language Statements**
- **Usage of Data Control Language Statements**

29.1 CREATION OF VIEW

Views are tables whose contents are taken or derived from other tables. Views themselves do not contain data. They just act as a window through which certain (not all) contents of a table can be viewed. Also, one can manipulate the contents of the original table through these views. The concept of view is depicted in figure 29.1.

To the user, the view appears like a table with columns and rows. But in reality, the view doesn't exists in the database as a stored set of values. The rows and columns that we find inside a view are actually the results generated by a query that defines the view. View is like a window through which we see a limited region of the actual table. The table from where the actual data is obtained is called the source table

For example, suppose we want that the clerk in your office should not have access to the details of the salaries of the other employees, but he needs such information as Emp_Name, Desig and Dept_Name. Then we can create a view for the clerk. This view will contain only the required information and nothing more than that. Similarly we can create a view for the Manager,

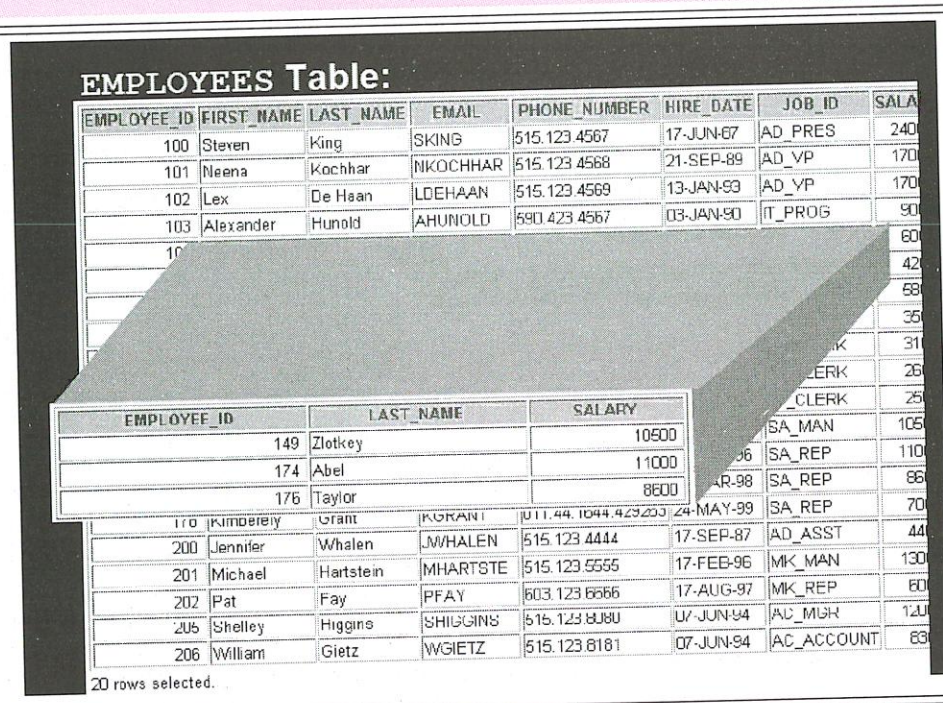


Figure 29.1: Concept of view

containing the information, which he needs. Hence, depending upon the requirement of different users we can create different views, all based upon one or more than one base tables.

Views are created using the CREATE command.

Syntax

CREATE OR REPLACE VIEW <View Name> As <SELECT Statement>;

Example

- To create a view for the clerk, we can use the following statement:

```
SQL>CREATE OR REPLACE VIEW clerk AS
SELECT empno, ename, job, dname from emp,dept
WHERE emp.deptno=dept.deptno;
```

Once a view is created it can be treated just as you treat any other table. You can retrieve the records in a view using the SELECT command, DELETE the records using the DELETE command and UPDATE them using the UPDATE command. Views can also be joined to other views or table.

- To view the records in the clerk view, the clerk user can use the following SELECT statement:

```
SQL>SELECT * FROM clerk;
```

Note that Employee table is an independent table and the Department table is also an independent table. Thus, View can be created to combine two separate tables into a virtual table.

- To create a view for the employees of department 10, the query will be as follows:

```
SQL>CREATE OR REPLACE VIEW dept10 As
SELECT * FROM emp WHERE deptno= 10;
```

We can also change the column names of a view.

Example

- In the above query we can also specify the names of columns which will be viewed by the users of this view as given below.

  SQL>Create Or Replace View dept10 (Code, Name, Desig, Date_of_Joining, Sal) As Select empno, ename, job, doj,sal From emp Where deptno = 10;

Uses of View

Following are the reasons why a good database should contain views:

- Views restrict access to the database. If you want users to see some but not all data of a table, you can create a view showing only certain fields.
- Critical data in the base table is safe guarded as access to such data can be controlled using views.
- Views allow users to make simple queries to retrieve the required results. For example, views allow users to retrieve information from multiple tables without knowing how to perform a join.
- Views allow the same data to be seen by different users in different ways.

29.2 TYPES OF VIEWS

Following are the types of views.

- Horizontal views
- Row/column subset views
- Joined views
- Vertical views
- Grouped views

The description of these types of views is given below.

29.2.1 Horizontal Views

Here a horizontal subset of the source table is taken to create the view. It is very useful when data belonging to different categories are present in the table. The concept of horizontal view is illustrated in figure 29.2.

Horizontal view restricts a user's access to only selected rows of a table.

```
CREATE VIEW view_cust AS
    SELECT *
        FROM Customer_Details
            WHERE Cust_ID in (101,102,103);
```

Figure 29.2: Concept of horizontal view

It is a private (virtual) table for each category to give access to the concerned persons.

```
CREATE VIEW view_cust AS
    SELECT Cust_ID, Account_No, Account_Type
        FROM Customer_Details;
```

Figure 29.3: Concept of Vertical View

29.2.2 Vertical Views

It selects only few columns of a table. It restricts a user's access to only certain columns of a table. The concept of vertical view is illustrated in figure 29.3.

29.2.3 Row/column Subset Views

It is a combination of horizontal and vertical view where a view selects a subset of rows and columns. The example of **row/column subset views is depicted in figure 29.4.**

```
CREATE VIEW View_Cust_VertHor
 AS  SELECT  Cust_Id,Account_No,Account_Type
        FROM Customer_Details
            WHERE CUST_ID IN (101,102,103);
```

Figure 29.4: Example of row/column subset view

29.2.4 Grouped Views

Grouped views are created based on a grouped query. They group rows of data and produce one row in the result corresponding to each group. So, the rows in the view don't have a one to one correspondence with the rows of the source table. For this reason, grouped views cannot be updated. The example of **Grouped views is depicted in figure 29.5.**

```
CREATE VIEW  View_GroupBY(Dept,NoofEmp)
AS SELECT Department, count(Employee_ID)
        FROM Employee_Manager
            GROUP BY Department;
```

Figure 29.5: Example of Grouped view

29.2.5 Joined Views

It is created by specifying more than one table in the query or in simple words; it contains SELECT with joining of tables. The rows in this view don't have a one to one

```
Create view View_Cust_Join as
 Select  a.Cust_Id,b.Cust_First_Name,b.Cust_Last_Name,Amount_in_dollars
        from Customer_loan a, customer_details b
            where a.cust_id = b.cust_id;
```

Figure 29.6: Example of Joined view

correspondence with the rows of the source tables. For this reason, joined views cannot be updated. The example of Joined **views is depicted in figure 29.6.**

29.3. DATA MANIPULATION IN VIEW

As discussed earlier, once a view is created it can be treated just like any other table and user can use Insert, Update and Delete statement over it as shown in figure 29.7.

```
CREATE VIEW View_Cust
        AS SELECT *
                FROM Customer_Details
                        WHERE CUST_ID IN (101,102,103);

--Insert Statement
insert into view_cust values(103,'Langer','G.','Justin',3421,'Savings',' Global
Commerce Bank','Langer_Justin@Yahoo.com');

--Delete Statement
delete view_cust  where cust_id=103;

--Update Statament
Update view_cust
        set Cust_last_name='Smyth'
                where cust_id=101;
```

Figure 29.7: Data Manipulation over a view

29.3.1 Updating a View

Depending on the commercial implementation being used, views may or may not be updateable. A view is updatable if :

- DISTINCT is not specified in the query used to create the view.
- The FROM clause specifies only one source table.
- The select list doesn't contain expressions/calculated columns.
- The WHERE clause doesn't include a subquery.
- The query doesn't include a GROUP BY or HAVING.

29.4 DROPPING A VIEW

Views are dropped in a similar way as tables are dropped. User must own the view in order to drop it.

DROP View <view_name> statement is used to drop a view. If some other views depend on this view, then if you say DROP VIEW <view_name> CASCADE then this view plus all other views that are based on this view are deleted. If you specify DROP VIEW <view_name> RESTRICT, then this view cannot be deleted if other views depend on it.

29.5 VIEWS WITH CHECK OPTION

To understand the importance of Views with CHECK Option, let us consider the scenario depicted in figure 29.8.

```
CREATE VIEW view_customer AS
    SELECT Cust_ID, Cust_Last_Name, Account_No, Account_Type, Bank_Branch
        FROM Customer_Details
            WHERE Bank_Branch = 'Downtown' ;
```

```
INSERT INTO view_customer
VALUES (115, 'Costner', 107, 'Savings', 'Bridgewater');
```

Will it prevent insertion into Customer_details ?

Figure 29.8.

By default, it will allow the insertion of other branches in the specified view. In order to restrict it, there is an option to create view with check option.

The WITH CHECK OPTION clause can be given for an updatable view to prevent inserts to rows for which the WHERE clause in the select_statement is not true. It also prevents updates to rows for which the WHERE clause is true but the update would cause it to be not true (in other words, it prevents visible rows from being updated to non visible rows).

```
CREATE VIEW view_customer AS
    SELECT Cust_ID, Cust_Last_Name, Account_No, Account_Type, Bank_Branch
        FROM Customer_Details
            WHERE Bank_Branch = 'Downtown'
                With CHECK OPTION;
```

29.6 ADVANTAGES OF VIEWS

A view has following advantages.

Security: It allows only a limited set of rows/ columns are viewable by certain users.

Query simplicity: A view can derive data from many tables. So, subsequently we can use queries on the view as single table queries rather than writing queries against the source tables as multi-table queries

Structural simplicity: Views can show only a portion of the table which is relevant to the user there by keeping it simple

29.7 DISADVANTAGES OF VIEWS

Following are the issues with usages of views.

Performance: Views based on joins are merely virtual tables. Every time a query is placed against the view, the query representing creation of the view has to be executed. So, complex joins may have to be performed every time a query is placed against the view.

Restrictions: Not all views are updateable

29.8 SEQUENCES: DATABASE OBJECT

A sequence is a database object used to generate sequence numbers for rows in the tables. It can be used for producing unique primary key values. A sequence is created using the CREATE SEQUENCE command.

Syntax:

```
CREATE SEQUENCE <Sequence_name>
[INCREMENT BY <integervalue>]
[START WITH <integervalue>]
[MINVALUE <integervalue>/NOMINVALUE]
[MAXVALUE <integervalue>/NOMAXVALUE];
[CYCLE/ NOCYCLE];
```

In the Syntax:

INCREMENT BY	Specifies the interval between sequence numbers. I may be positive or negative number but not zero. Its default value is 1.
START WITH	Specifies the first sequence number to be generated. The default for ascending sequence is 1 and for descending is –1.
MINVALUE	Specifies the sequence minimum value.
MAXVALUE	Specifies the maximum value that a sequence can generate.
CYCLE	Specifies that the sequence continues to generate repeat values after reaching either its maximum value.
NOCYCLE	Specifies that a sequence cannot generate more values after reaching the maximum value.

Examples

- Create a sequence 'empnumber' starting with value 100 and incremented by 2.

```
SQL>CREATE SEQUENCE Empcode Increment By 2 Start With 100;
```

INCREMENT BY clause is optional. The default value for increment is 1. A pseudo-column NEXTVAL is used to extract the next number in line from a specified sequence. Another pseudo-column CURRVAL is used to extract the current sequence number.

- To view the next number in sequence, we have to use the following query:

```
SQL> SELECT EmpCode.Nextval From Dual;
```

The output is:

```
NEXTVAL
---------
      100
```

Whenever the NEXTVAL pseudo-column is referenced, the next sequence number is generated. For example, if you repeat the above statement the output will be 102 and not 100. It will be incremented by 2 in each reference to NEXTVAL.

The CURRVAL column is used to know the value last returned by NEXTVAL. When the NEXTVAL is referenced for a given sequence, the current sequence number is placed in CURRVAL.

- The following statement will insert 102 into EMP_Code.

```
SQL>Insert into emp (empno, ename, job, doj, sal, deptno)
Values (Empcode.Currval, 'Mini', 'Manager', '12-MAR-O4',7000,10) ;
```

29.9 INDEX: DATABASE OBJECT

Index is a way to store and search records in the table. Indexes are used to improve the speed with which the records can be located and retrieved from the table. Oracle retrieves rows in a table in one of the two ways:

- By ROWID
- By full-table scan

The address field of an index is called ROWID, which are internally generated and maintained in binary values, which identify records. The ROWID for a table is an address for the row on disk. With the ROWID, Oracle can search for the data on disk rapidly.

The ROWID format used by Oracle is as follows:

BBBBBBB.RRRR.FFFF

Where,

BBBBBBB: It is the block number on which the record is stored.

RRRR: It is a unique record number in each data block.

FFFF: It is a unique number given by oracle engine to each data file.

29.9.1 Uses of Indexing

Indexes have multiple uses on the Oracle database. These are:

- Indexes can be used to ensure uniqueness on a database.
- Indexes also boost performance on searching for records in a table.

29.9.2 Types of Indexes

Oracle allows two types of indexes:

- Duplicate Indexes
- Unique Indexes

Duplicate Indexes

It allows duplicate values for the indexed columns.

Unique Indexes

It does not allow duplicate values for the indexed columns.

Index can further be classified as Simple and Composite depending on the number of columns involved in the indexes.

29.9.3 Creation of Simple Duplicate Index

It is the index created on a single column but it can have duplicate values.

Syntax:

```
CREATE INDEX index_name
ON table_name(column_name);
```

Example:

```
SQL> CREATE INDEX ename_index
ON emp(ename);
```

29.9.4 Creation of Composite Duplicate Index

It is the index created on a multiple columns but can have duplicate values.

Syntax:

```
CREATE INDEX index_name
ON table_name(column_name1,column_name2,……);
```

Example:

```
SQL> CREATE INDEX ename_sal_index
ON emp(ename,sal);
```

NOTE: Upto 16 columns in a table can be included in a single index on that table.

29.9.5 Creation of Simple Unique Index

It is the unique index created on a single column but with no duplicate values.

Syntax:

```
CREATE UNIQUEINDEX index_name
ON table_name(column_name);
```

Example:

```
SQL> CREATE UNIQUE INDEX ename_index
ON emp(ename);
```

In this case we cannot insert duplicate values in ename column of emp table.

29.9.6 Creation of Composite Unique Index

It is the unique index created on a multiple columns but with no duplicate values.

Syntax:

```
CREATE UNIQUE INDEX index_name
ON table_name(column_name1,column_name2,……);
```

Example:

```
SQL> CREATE UNIQUE INDEX ename_sal_index
ON emp(ename,sal);
```

In this case combination of ename and sal values cannot be repeated in emp table.

29.9.7 Dropping of existing Indexes

When an index is no longer needed in the database, the developer can remove it with the use of the DROP INDEX command. If the index is used in relation to a primary-key or unique constraint, then the index will no longer continue to enforce that uniqueness constraint. The syntax for the DROP INDEX statement is the same, regardless of the type of index being dropped. If you wish to rework the index in any way, you must first drop the old index and then create the new one.

Syntax:

```
DROP INDEX index_name;
```

Example:

```
SQL> DROP INDEX ename_index;
```

29.9.8 Problem Caused by too many Indexes

If there are too many indexes in a table then Oracle engine will take longer time to insert, update or delete operations, because index processing must be done for every record.

Thus, we can conclude that the indexes improve the performance of data retrieval but too many indexes can degrade the performance by the data changes performed by DML statements.

29.10 INDEX GUIDELINES

Here are some guidelines from creating efficient indexes:

- Index if you need to access no more than 10-15% of the data in the table. A full table scan (read all the table) is better if you intend to retrieve a high percentage of the table data, this is because a index read requires two reads
- Do not use indexes on small tables, a full table scan would be fine.
- Create primary keys for all tables as a index will be created by default.
- Index the columns that are involved in multi-table join operations
- Index columns that are used frequently in where clauses.
- Index columns that are involved in order by, group by, union and distinct operations.
- Columns that have long character strings are bad for indexing
- Columns that are frequently update are bad for indexing
- Choose tables where few rows have similar values

- Keep the number of indexes small, to many will cause performance problems on inserting data.

29.9.11 Oracle Index Schemes

In Oracle Index are implemented by using two data structures. These are follows:

- **B*Tree Indexes:** These are the common indexes in Oracle. They are similar construct to a binary tree, they provide fast access by key, to an individual row or range of rows, normally requiring very few reads to find the correct row.
- **Bitmap Indexes:** With a bitmap index, a single index entry uses a bitmap to point to many rows simultaneously, they are used with low data that is mostly read-only. They should not be used with OLTP systems.

B*Tree Indexes

In general B*Tree index would be placed on columns that were frequently used in the predicate of a query and expect some small fraction of the data from the table to be returned. It's purely a function on how large of a percentage of the table you will need to access via the index and how the data happens to be laid out. If you can use the index to answer the question accessing a large percentage of the rows makes sense, since you are avoiding the extra scattered I/O to read the table. If you use the index to access the table you will need to ensure you are processing a small percentage of the total table.

Bitmap Indexes

Bitmap indexes are structures that store pointers to many rows with a single index key entry. In a bitmap index there will be a very small number of index entries, each of which point to many rows. Bitmap indexes are best used on low cardinality data, this is were the number of distinct items in the set of rows divided by the number of rows is a small number for example a gender column may have MF, F and NULL. If you had a table with 20000 rows you would find that 3/20000 = 0.00015, this would be an ideal candidate for a bitmap index.

Remember also that a single bitmap entry key points to many rows. If a session modifies the index then all of the rows that the index points to are effectively locked. Oracle cannot lock an individual bit in a bitmap index entry; it locks the entire bitmap, this will seriously inhibit concurrency as each update will appear to lock potentially hundreds of rows preventing their bitmap columns from being concurrently updated.

An Oracle bitmap index would look like

Value/Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Analyst	0	0	0	0	0	0	0	1	1	1	0	0	1	0
Clerk	1	0	0	0	0	0	0	0	0	0	1	1	0	1
Manager	0	0	0	1	0	1	1	0	0	0	0	0	0	0
President	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Saleman	0	1	1	0	1	0	0	0	0	0	0	0	0	0

Using the above table you can see that rows 1, 4, 6, 7, 11, 12 and 14 would represent a manager and clerk.

B-Tree Indexes vs Bitmap Indexes

The guidelines for usage of B-Tree Indexes and Bitmap Indexes are given below.

B-tree Index	Bitmap Index
Good for high-cardinality data	Good for low-cardinality data
Good for OLTP databases (lots of updating)	Good for data warehousing applications
Use a large amount of space	Use relatively little space
Easy to update	Difficult to update

29.10 TRANSACTION CONTROL LANGUAGE STATEMENTS

SQL provides a construct for specifying the set of actions that will form a transaction. Note that, SQL transactions end with either of the following commands:

- COMMIT
- ROLLBACK

COMMIT command is used to make changes to the database permanent. To commit the work you have done till now you just write COMMIT;

ROLLBACK is used to reject parts or all the work done in the current transaction. To undo the changes you made till now, write ROLLBACK;

29.11 DATA CONTROL LANGUAGE STATEMENTS

A privilege can either be granted to a User with the help of GRANT statement. The privileges assigned can be SELECT, ALTER, DELETE, EXECUTE, INSERT, INDEX etc. In addition to granting of privileges, you can also revoke it by using REVOKE command.

29.11.1 GRANT Command

SQL is normally used in multiuser environments. The GRANT command is used to permit users access to the database.

The syntax for GRANT command is:

```
GRANT <privilege_name> | ALL
On <object> To <user|PUBLIC>
[WITH GRANT OPTION];
```

<privilege_name> : specifies the actions such as Insert, Delete, Update, Select, etc. the user can perform on the <object>. ALL indicates all privileges.

<user>: specifies the name of the user to whom the privileges has been granted. To grant the privileges to all the users PUBLIC option can be used.

[WITH GRANT OPTION] is optional and a user having this privileges can pass on his privileges to another user by using the Grant command. The recepient user can then grant the same privileges to some other user. For example, when an owner of a table grants privilege on the table to another user B, the privilege can be given with or without GRANT

OPTION. If the [WITH GRANT OPTION] is given, this means that B can also grant that privilege to other users, else B can not grant privileges to other users.

The syntax of GRANT is illustrated below.

```
GRANT {
        [ALTER[, ]]
        [DELETE[, ]]
        [INDEX[, ]]
        [INSERT[, ]]
        [SELECT[, ]]
        [UPDATE [(column-name[,...])][, ]]
        | ALL [PRIVILEGES]]      }
ON [TABLE] {table-name[,...] | view-name[,...]}
TO [AuthID][,...]
[WITH GRANT OPTION]
```

Examples

- To provide SELECT permission on the Department table to user 'mini' and allow her to give further grants will be as follows:

  SQL> GRANT SELECT On dept to mini with GRANT OPTION;

  We can also grant privileges on individual columns in a table.

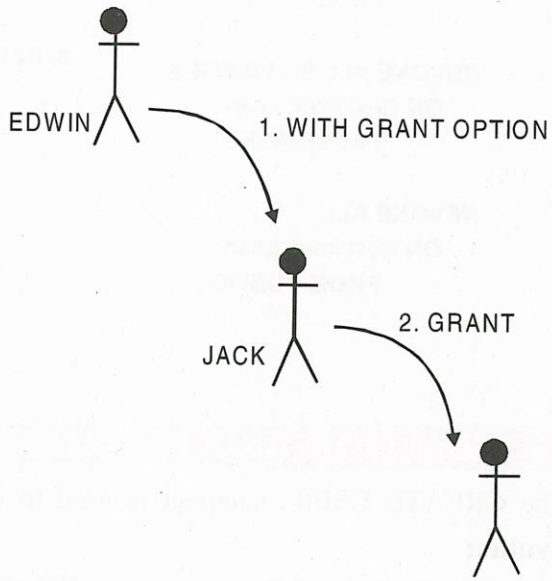
- To give user 'mini' the privilege to update the column Emp_Name, we will require the following query:

  SQL>GRANT UPDATE (ename) On emp To mini;

The usage of GRANT is illustrated below.

With Grant option

```
GRANT SELECT
   ON Customer_Loan
      TO EDWIN
       With GRANT OPTION;
```



29.11.2 REVOKE Command

The Revoke command is used to cancel database privileges from user(s).

The syntax for revoke command is:

REVOKE <privilege_name> | All

On <Object> From <user|PUBLIC>;

The syntax of REVOKE is illustrated below.

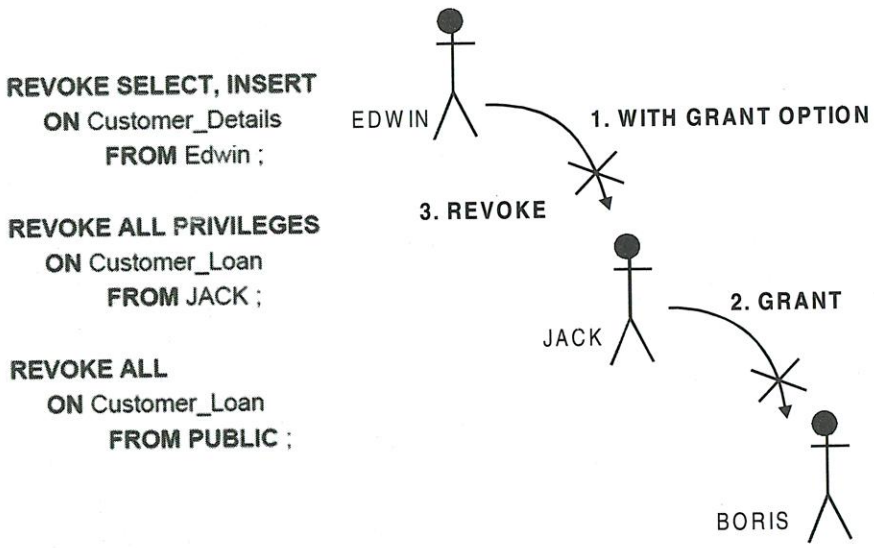
```
REVOKE{
        [ALTER[, ]]
        [DELETE[, ]]
        [INDEX[, ]]
        [INSERT[, ]]
        [SELECT[, ]]
        [UPDATE [(column-name[,...])][, ]]
        | ALL [PRIVILEGES] }
ON [TABLE] {table-name[,...] | view-name [,...]}
FROM AuthID[,...]
```

Example

To revoke the privilege we gave to user 'mini' will be as follows:

SQL>REVOKE SELECT On dept FROM mini;

The usage of REVOKE is illustrated below.



29.12 CREATION AND MANAGEMENT OF USER ACCOUNTS

The CREATE USER statement is used to create a new user.

Syntax:

CREATE USER <username> IDENTIFIED BY <password>;

For Example:

SQL>CREATE USER rahat IDENTIFIED BY Thapar;

29.12.1 Steps to Create User

- Connect to the database by username/password = Sys/change_on_install or by system/manager (these are two prefabricated user Schemas which are having all the privileges of DBA and to create user)
- Now type the above stated create user command at SQL prompt as

  SQL> Connect System/manager ;

  SQL> CREATE USER rahat IDENTIFIED BY Thapar;

  The above statement will create user with username "rahat" having password "Thapar".
- To grant all the privileges to the new created user following command is used:

  SQL> GRANT ALL PRIVILEGES TO rahat;

29.12.2 Altering User Account

Sometimes we do need to change some of the parameters after the creation of user account such as changing of password etc; for this ALTER USER statement is used. For ALTER statement the user must be already created. Following is the syntax for altering an already existing user for changing its password.

SQL>ALTER USER rahat IDENTIFIED BY Patiala;

The above given statement will change the password of user "rahat" from "Thapar" to "Patiala".

29.12.3 Deleting User Account

User accounts created by the DBA can also be dropped out permanently. DROP USER statement is used to delete the user. But before dropping the user one has to drop all the objects of that user first. But dropping of objects of the user and the user itself can be done with a single clause "CASCADE". With this statement Oracle also removes all the referential integrity associated to the objects of the removed user.

SQL > CONNECT system/manager;

SQL > DROP USER rahat;

SQL> DROP USER rahat CASCADE;

29.13 ORACLE TREE WALKING

A handy feature in Oracle is the tree-walking feature. It allows you to recursively join a table back on to itself to retrieve all records in a hierarchy.

For example you can retrieve all employees who report to a given person and all the employees who report to those people and so on until you have all the people.

To achieve this you need to make use of the CONNECT BY command. In the following example we select the employee name and then specify the value in the tree where we want to begin. To do this we use the START WITH command to identify where to begin, then we issue the CONNECT BY PRIOR which recursively joins all employee records where the emp_reports_to_emp_id is equal to the previous emp_id.

```
SQL>SELECT emp_name
FROM employee
START WITH emp_id = 270
CONNECT BY PRIOR emp_id = emp_reports_to_emp_id;
```

This is a great feature and can be a real timesaver.

FLASH BACK

Some important objects in Oracle are like views, sequences and indexes. Views are tables whose contents are taken or derived from one or more tables, they themselves do not contain data. The major advantage of views is that they restrict the access to the database and provides a safeguard to critical data.

A sequence is a database object used to generate sequence numbers for rows in the tables. it can be used to produce unique primary key. The index database object is used to store and search records in the table. Indexes improve the speed with which the records are located and retrieved from the table.

REVIEW QUESTIONS

1. What do you mean by views? Explain.
2. What are the advantages of creating the views?
3. What is the purpose of creating Sequences?
4. How can you distinguish between indexes created manually and those created automatically?
5. What are the reasons for using indexes?
6. Must all the data in an index be unique? Explain.
7. What method is used to create a unique index and non-unique index?
8. How is a unique index dropped?
9. What are the effects of dropping an index?

BRAIN STORMING SESSION

1. Consider the employee database shown below, where the primary keys are underlined.
   employee (employee_name,street,city)
   works (employee_name, company_name,salary)
   company (company_name,city)
   manages (employee_name, manager_name)
2. Give an expression in SQL for each of the following queries.
   (a) Find the names of all employees who work for Punjab National Bank.
   (b) Find the names and cities of residence of all employees who work for Punjab National Bank.

(c) Find the names, street address, and cities of residence of all employees who w for State Bank of Patiala and earn more than Rs 20,000 per annum.

(d) Give all employees of Punjab Sind Bank a 10 percent salary raise.

(e) Find the names of all employees in this database who live in the same city as the company for which they work.

(f) Assume the companies may be in several cities. Find all companies located in every city in which Punjab National Bank is located.

(g) Find all employees in the database who live in the same cities and on the same streets as do their managers.

(h) Find the employees who earn more than the average salary of all employees of their company

(i) Delete all tuples in the *works* relation for employees of Small Bank Corporation.

3. Let *R = (A, B, C),* and let and both be relations on schema *R.* Give an expression in SQL that is equivalent to each of the following queries

(a) r_1? r_2 (b) r_1? r_2 (c) $r_1 - r_2$

4. Show that, in SQL, < > all is identical to not in.

5. By considering the employee database of Question No. 1, define a view consisting of *manager_name* and the average salary of all employees who work for that manager. Explain why the database system should not allow updates to be expressed in terms of this view.

6. Consider the insurance database shown below, where the primary keys are underlined. Construct the following SQL queries for this relational database

person (driver_id, name, address)

car (licence, model, year)

accident (report_number, date, location)

owns (driver_id, license)

(a) The total number of people who owned cars that were involved in accidents in 1989.

(b) Add a new accident to the database; assume any values for required attributes.

(c) Delete the Mazda belonging to "John Smith".

(d) Find the number of accidents in which the cars belonging to "John Smith" were involved.

(e) Update the damage amount for the car with license number "AABB2000" in the accident with report number "AR2197" to $3000.

7. Consider the following relational database:

employee(employee_name, street, city)

works(employee_name, company_name, salary)

company(company_name, city)

manages(employee_name, manager_name)

Give an SQL DDL definition of this database. Identify referential-integrity constraints that should hold, and include them in the DDL definition.

Write check conditions for the schema to ensure that:

Every employee works for a company located in the same city as the city in which the employee lives.

No employee earns a salary higher than that of his manager.

8. Consider an employee database with two relations

   employee (employee_name, street, city)

   works (employee_name, company_name, salary)

   Where the primary keys are underlined. Write a query to find companies whose employees earn a higher salary, on average, than the average salary at Punjab National Bank.

   (a) Using SQL functions as appropriate.

   (b) Without using SQL functions.

Multiple Choice Questions on SQL

1. SQL statements are divided into following categories
   (a) DDL & DML (b) DCL (c) TCL (d) ALL
2. Embedded SQL statements include
   (a) DEFINE & OPEN (b) DECLARE
   (c) EXECUTE (d) ALL
3. Oracle Data types include
   (a) Varchar2 (b) BLOB (c) Long (d) ALL
4. SELECT statement include
   (a) SELECT from where (b) Views
   (c) Sequences (d) None
5. SQL functions include
   (a) Single-row functions (b) Group functions
   (c) Both a & b (d) None
6. CHR function returns the
   (a) character for the decimal equivalent (b) strips trailing character
   (c) converts string to lowercase (d) converts string to uppercase
7. RTRIM functions returns the
   (a) character for the decimal equivalent (b) strips trailing character
   (c) converts string to lowercase (d) converts string to uppercase
8. LOWER function converts
   (a) character for the decimal equivalent (b) strips trailing character
   (c) converts string to lowercase (d) converts string to uppercase
9. Upper function converts
   (a) character for the decimal equivalent (b) strips trailing character
   (c) converts string to lowercase (d) converts string to uppercase
10. Self join
    (a) join two copies of same table (b) join two copies of different table
    (c) copies the table (d) None
11. The special operators for sub-queries are
    (a) EXISTS (b) ANY
    (c) SOME & ALL (d) ALL

12. What is the best data type definition for Oracle when a field is alphanumeric and has a fixed length?

(a) VARCHAR2 (b) CHAR (c) LONG (d) NUMBER

13. What is the best data type definition for Oracle when a field is alphanumeric and has a length that can vary?

(a) VARCHAR2 (b) CHAR (c) LONG (d) NUMBER

14. Selecting a data type involves which of the following?

(a) Maximize storage space (b) Represent most values

(c) Improve data integrity (d) All of the above.

15. A rule of thumb for choosing indexes for a relational database includes which of the following?

(a) Indexes are more useful on smaller tables.

(b) Indexes are more useful for columns that do not appear frequently in the WHERE clause in queries.

(c) Do not specify a unique index for the primary key of each table.

(d) Be careful of indexing attributes that have null values.

16. Structured data may include which of the following?

(a) Photo image (b) Video clip (c) Dates (d) None of the above.

17. Two tables, PRODUCT and STORAGE_BOX, exist in a database. Individual products are listed in the table by unique ID number, product name, and the box a particular product is stored in. Individual storage boxes (identified by number) listed in the other table can contain many products, but each box can be found in only one location. Which of the following statements will correctly display the product ID, name, and box location of all widgets in this database?

(a) select p.prod_id, p.prod_name, b.box_loc from product p, storage_box b where p.prod_id = b.prod_id and prod_name = ' WIDGET ' ;

(b) select p.prod_id, p.prod_name, b.box_loc from product p, storage_box b where prod_name = 'WIDGET";

(c) select p.prod_id, p.prod_name, b.box_loc from product p, storage_box b where p.stor_box_num = b.stor_box_num and p.prod_name = 'WIDGET' ;

(d) select prod_id, prod_name, box_loc from product, storage_box where stor_box_num = stor_box_num and prod_name = ' WIDGET ' ;

18. You want to join information from three tables as part of developing a report. The tables are EMP, DEPT, and SALGRADE. Only records corresponding to employee, department location, and salary range are required for employees in grades 10 and higher for the organization. How many comparison operations are required for this query?

(a) Two (b) Three (c) Four (d) Five

19. Use the following code block to answer the question:

```
select deptno, job, avg (sal)
from emp
group by deptno, j ob
having avg (sal) >
```

(select sal
from emp
where ename = 'MARTIN');

Which of the following choices identifies the type of subquery used in the preceding statement?

(a) A single-row subquery
(b) A multirow subquery
(c) A from clause subquery
(d) A multicolumn subquery

20. SQL was developed as an integral part of

(a) A hierarchical database
(b) A data warehouse
(c) A relational database
(d) All the above

21. Your company's sales database contains one table, PROFITS, which stores profits listed by product name, sales region, and quarterly time period. If you wanted to obtain a listing of the five best-selling products in company history, which of the following SQL statements would you use?

(a) select p.prod_name, p.profit from (select prod_name, profit from profits order by profit desc) where rownum <=5;

(b) select p.prod_name, p.profit from (select prod_name, sum (profit) from profits group by prod_name order by sum(profit) desc) subq where p.prod_name = subq. prod_name;

(c) select prod_name, profit from (select prod_name, sum(profit) from profits group by prod_name order by sum(profit) desc) where rownum <= 5;

(d) select prod_name, profit from (select prod_name, sum(profit) from profits order by sum(profit) desc) where rownum <=5;

22. The database for an international athletic competition consists of one table, ATHLETES, containing contestant name, age, and represented country. To determine the youngest athlete representing each country, which of the following queries could be used?

(a) select name, country, age from athletes where (country, age) in (select min (age), country from athletes group by country) ;

(b) select name, country, age from athletes where (country, age) in (select country, min(age) from athletes) group by country;

(c) select name, country, age from athletes where age in (select country, min(age) from athletes group by country) ;

(d) select name, country, age from athletes where (country, age) in (select country, min(age) from athletes group by country) ;

23. You are sorting data in a table in your select statement in descending order. The column you are sorting on contains NULL records. Where will the NULL records appear?

(a) At the beginning of the list
(b) At the end of the list
(c) In the middle of the list
(d) The same location they are listed in the unordered table

24. For the relation Supplies(Snumber,Pnumber,Qty), which of the following SQL statements corresponds to a relational algebra project and selection operators?

(a) Select Snumber From Supplies;

(b) Select Distinct S1.Snumber From Supplies S1, Supplies S2 Where S1.Qty > S2.Qty;

(c) Select Distinct Snumber From Supplies Where Qty > 35;
(d) Select * From Supplies;

25. The command to eliminate a table from a database is:
(a) REMOVE TABLE CUSTOMER; (b) DROP TABLE CUSTOMER;
(c) DELETE TABLE CUSTOMER; (d) UPDATE TABLE CUSTOMER;

26. The command to eliminate a row from a table is:
(a) REMOVE FROM CUSTOMER (b) DROP FROM CUSTOMER
(c) DELETE FROM CUSTOMER (d) UPDATE FROM CUSTOMER

27. Which of the following is the correct processing order for SQL SELECT statements?
(a) SELECT, FROM, WHERE (b) FROM, WHERE, SELECT
(c) WHERE, FROM,SELECT (d) SELECT,WHERE,FROM

28. A view is which of the following?
(a) A virtual table that can be accessed via SQL commands
(b) A virtual table that cannot be accessed via SQL commands
(c) A base table that can be accessed via SQL commands
(d) A base table that cannot be accessed via SQL commands

29. Find the SQL statement below that is equal to the following: SELECT NAME FROM CUSTOMER WHERE STATE = 'VA';
(a) SELECT NAME IN CUSTOMER WHERE STATE IN ('VA');
(b) SELECT NAME IN CUSTOMER WHERE STATE = 'VA';
(c) SELECT NAME IN CUSTOMER WHERE STATE = 'V';
(d) SELECT NAME FROM CUSTOMER WHERE STATE IN ('VA');

30. The benefits of a standard relational language include which of the following?
(a) Reduced training costs
(b) Increased dependence on a single vendor
(c) Applications are not needed. (d) All of the above.

31. The HAVING clause does which of the following?
(a) Acts like a WHERE clause but is used for groups rather than rows.
(b) Acts like a WHERE clause but is used for rows rather than columns.
(c) Acts like a WHERE clause but is used for columns rather than groups.
(d) Acts EXACTLY like a WHERE clause.

32. What type of join is needed when you wish to include rows that do not have matching values?
(a) Equi-join (b) Natural join (c) Outer join (d) All of the above.

33. What type of join is needed when you wish to return rows that do have matching values?
(a) Equi-join (b) Natural join (c) Outer join (d) All of the above.

34. The following SQL is which type of join: SELECT CUSTOMER_T. CUSTOMER_ID, ORDER_T. CUSTOMER_ID, NAME, ORDER_ID FROM CUSTOMER_T,ORDER_T WHERE CUSTOMER_T. CUSTOMER_ID = ORDER_T. CUSTOMER_ID
(a) Equi-join (b) Natural join (c) Outer join (d) Cartesian join

Solution Keys

1. (d)	2. (d)	3. (d)	4. (a)	5. (c)	6. (a)	7. (b)	8. (c)	9. (d)
10. (a)	11. (d)	12. (b)	13. (a)	14. (c)	15. (d)	16. (c)	17. (c)	18. (b)
19. (a)	20. (c)	21. (c)	22. (d)	23. (a)	24. (c)	25. (b)	26. (c)	27. (b)
28. (a)	29. (d)	30. (a)	31. (a)	32. (d)	33. (d)	34. (a)		

Lab Assignment-8

Solve the following on the given database:

S(Sno, Sname, City, Status)

P(Pno, Pname, Color, Weight)

SP(Sno, Pno, Qty)

6. Create the above tables by identifying appropriate Primary keys and foreign keys.
7. Apply Column level constraints on City such that it may be only "Amritsar", "Delhi", "Batala" and "Qadian".
8. Apply Column level constraints such that Qty should be between 100 to1000, Weight is NOT NULL, Sname is NOT NULL and Pname is UNIQUE.
9. Insert five records in each table.
10. Get supplier name where city is "Amritsar".
11. Get quantity supplied by "Ajay".
12. Get the supplier name and city that supply part P1.
13. Get color of parts supplied by S1.
14. Get color of parts supplied by Ajay.
15. Get supplier name who supply at least one red part.
16. Get part name where weight>100 and Sno="S1".
17. Get supplier name who supply part P1 with quantity>100.
18. Get supplier name who supply pencils with quantity>100.
19. Get supplier number who supplies maximum quantity.
20. Get supplier number who supply quantity greater than average quantity.
21. Get supplier name that supply maximum quantity.
22. Get total number of suppliers.
23. Get total number of parts supplied by supplier S1.
24. Count the parts having red color.
25. Count red parts supplied by Ajay.
26. Get the total quantity supplied by S1.
27. Get maximum quantity supplied by S1.
28. Get the maximum quantity supplied by Rakesh.
29. Get total quantity supplied for each supplier.
30. Get total number of records supplied for each part.
31. Increase the Quantity of part P1 by 10%.

32. Change the color of Red parts supplied by Ajay to Green.
33. Delete all the red parts supplied by supplier S1.
34. Delete all the entries of part P1.
35. Create view on table S that contains Sno and Sname of suppliers Amritsar.
36. Create a sequence that has minimum value 100 and maximum 1000.
37. Use the above sequence to store quantity information.
38. Alter the structure of table S to change the width of column Sname.
39. Alter the structure of table P to apply constraint NOT NULL on column Color.
40. Drop view created above.

Solution of the Selected Problems

6. Creation of Tables

```
CREATE TABLE S(SNO VARCHAR(4) PRIMARY KEY, SNAME CHAR(15), CITY CHAR(15), STATUS NUMBER(5));
CREATE TABLE P(PNO VARCHAR(4) PRIMARY KEY, PNAME CHAR(15), COLOR CHAR(15), WEIGHT NUMBER(5));
CREATE TABLE SP(SNO VARCHAR(4) REFERENCES S(SNO), PNO VARCHAR (4) REFERENCES P(PNO), QTY NUMBER(5), PRIMARY KEY(SNO,PNO);
```

7. Creation of S Table with Constraints

```
CREATE TABLE S(SNO VARCHAR(4) PRIMARY KEY, SNAME CHAR(15), CITY CHAR(15) CHECK(CITY IN('Amritsar','Delhi','Batala')), STATUS NUMBER(5));
```

8. Creation of SP Table with Constraints

```
CREATE TABLE SP(SNO VARCHAR(4) REFERENCES S(SNO), PNO VARCHAR (4) REFERENCES P(PNO), QTY NUMBER(5) CHECK (QTY BETWEEN 100 AND 1000), PRIMARY KEY(SNO,PNO));
```

9. Insertion of records

```
INSERT INTO S VALUES('S1','CHINTAN','AHMEDABAD',50);
INSERT INTO P VALUES('P1','HEADPHONE','BLACK',30);
INSERT INTO SPM VALUES('S1','P1',100);
```

10. `SELECT SNAME FROM SPL WHERE CITY='Amritsar';`
11. `SELECT QTY FROM SP WHERE SNO IN(SELECT SNO FROM S WHERE SNAME='Ajay');`
12. `SELECT SNAME,CITY FROM S WHERE SNO IN(SELECT SNO FROM SP WHERE PNO='P1');`
13. `SELECT COLOR FROM P WHERE PNO IN(SELECT PNO FROM SP WHERE SNO='S1');`
14. `SELECT COLOR FROM P WHERE PNO IN(SELECT PNO FROM SP WHERE SNO IN(SELECT SNO FROM S WHERE SNAME='Ajay'));`
15. `SELECT SNAME FROM S WHERE SNO IN(SELECT SNO FROM SP WHERE PNO IN(SELECT PNO FROM P WHERE COLOR='Red'));`
16. `SELECT PNAME FROM P WHERE WEIGHT>25 AND PNO IN(SELECT PNO FROM SP WHERE SNO='S1');`
17. `SELECT SNAME FROM S WHERE SNO IN(SELECT SNO FROM SP WHERE PNO='P1' AND QTY>100);`

18. SELECT SNAME FROM S WHERE SNO IN(SELECT SNO FROM SP WHERE QTY>100 AND PNO IN(SELECT PNO FROM P WHERE PNAME='Pencils'));
19. SELECT SNO FROM SP WHERE QTY =(SELECT MAX(QTY) FROM SP);
20. SELECT SNO FROM SP WHERE QTY>(SELECT AVG(QTY) FROM SP);
21. SELECT SNAME FROM S WHERE SNO IN(SELECT SNO FROM SP WHERE QTY =(SELECT MAX(QTY) FROM SP));
22. SELECT COUNT(SNO) FROM SP;
23. SELECT COUNT(PNO) FROM SP WHERE SNO='S1';
24. SELECT COUNT(PNO) FROM P WHERE COLOR='RED';
25. SELECT COUNT(PNO) FROM SP WHERE SNO IN(SELECT SNO FROM SP WHERE SNAME='Ajay') AND PNO IN(SELECT PNO FROM P WHERE COLOR='Red');
26. SELECT SUM(QTY) FROM SP WHERE SNO='S1';
27. SELECT MAX(QTY) FROM SP WHERE SNO='S1';
28. SELECT MAX(QTY) FROM SP WHERE SNO IN(SELECT SNO FROM S WHERE SNAME='Rakesh');
29. SELECT SNO, SUM(QTY) FROM SP GROUP BY SNO;
30. SELECT PNO, COUNT(PNO) FROM SP GROUP BY PNO;
31. UPDATE SP SET QTY=(QTY+QTY*0.1) WHERE PNO='P1';
32. UPDATE P SET COLOR='Green' WHERE COLOR='Red' AND PNO IN(SELECT PNO FROM SP WHERE SNO IN(SELECT SNO FROM S WHERE SNAME='Ajay'));
33. DELETE FROM P WHERE COLOR='Red' AND PNO IN(SELECT PNO FROM SP WHERE SNO='S1');

    For this, first delete the corresponding records in the child table SP as given below:

    DELETE FROM SP WHERE PNO IN (SELECT PNO FROM P WHERE COLOR='Red') AND SNO='S1';
34. DELETE FROM SP WHERE PNO='P1';

    After deleting the records from child table Sp, delete the corresponding records in parent P table.

    DELETE FROM P WHERE PNO='P1';
35. CREATE VIEW S_AMRITSAR AS SELECT SNO,SNAME FROM S WHERE CITY=' Amritsar';
36. CREATE SEQUENCE QUANTITY MINVALUE 100 MAXVALUE 1000;
37. INSERT INTO SP(SNO,PNO,QTY) VALUES('S1','P1',QUANTITY.NEXTVAL);
38. ALTER TABLE S MODIFY SNAME VARCHAR2(25);
39. ALTER TABLE P MODIFY COLOR CHAR(15) NOT NULL;
40. DROP VIEW S_AMRITSAR;

HANDS ON SESSION

1. Which of the following integrity constraints automatically create an index when defined?

   (a) Foreign keys
   (b) Unique constraints
   (c) NOT NULL constraints
   (d) Primary keys

2. To add the number of columns selected by a view
   (a) Add more columns to the underlying table
   (b) Issue the alter view statement
   (c) Use a correlated subquery in conjunction with the view
   (d) Drop and re-create the view with references to select more columns
3. The following statement is issued against the Oracle database. Which line will produce an error?
   (a) create view EMP_VIEW_01
   (b) as select E.EMPID, E.LASTNAME, E.FIRSTNAME, A.ADDRESS
   (c) from EMPLOYEE E, EMPL_ADDRESS A
   (d) where E.EMPID = A.EMPID
   (e) with check option;
   (f) This statement contains no errors.
4. Dropping a table has which of the following effects on a nonunique index created for the table?
   (a) No effect.
   (b) The index will be dropped.
   (c) The index will be rendered invalid.
5. Which of the following statements about indexes is true?
   (a) Columns with low cardinality are handled well by B-tree indexes.
   (b) Columns with low cardinality are handled poorly by bitmap indexes.
   (c) Columns with high cardinality are handled well by B-tree indexes
6. After referencing NEXTVAL, the value in CURRVAL
   (a) Is incremented by one
   (b) Is now in PREVVAL
   (c) Is equal to NEXTVAL
   (d) Is unchanged

Solution Keys

1. **(b, d)** Every constraint that enforces uniqueness creates an index to assist in the process.
2. **(d)** The solution is to change the existing view definition by dropping and re-creating the view.
3. **(f)** Even though the reference to with check option is inappropriate, considering that inserts into complex views are not possible, the statement will not actually produce an error when compiled.
4. **(b)** Indexes created manually on a table will be dropped if the table is dropped.
5. **(c)** Different index types are designed to handle performance improvements where column data is both relatively unique and relatively non-unique.
6. **(c)** Once NEXTVAL is referenced, the sequence increments the integer and changes the value of CURRVAL to be equal to NEXTVAL.

○○○○○