CREATING OTHER SCHEMA OBJECTS

Apart from tables, other essential schema objects are view, sequences,indexes and synonyms.A view is a logical or virtual table. Synonyms are simply alias names for database objects.Synonyms also simplify query writing and provide an element of system security by disguising the actual name of a database object.Sequences are special database objects that support the automatic generation of integer values,and are often used to generate primary key values for tables.Indexes are created on table columns to facilitate the rapid retrieval of information from tables.

Views

A database view is a logical or virtual table based on a query.Views are queried just like tables.This means that from your perspective as a developer or from a database system user's perspective, a view looks like a table.The definition of a view as an object is stored within a database's data dictionary; however,a view stores no data itself.A database also stores the execution plan for creating a view-this means that data can be retrieved rapidly through use of a view even though the actual data presented by a SELECT query of a view is not stored as part of a view.Rather,the data is "gathered together" each time that a view is queried from the database tables for which a view is defined-these are termed base tables.

The general syntax is given below.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW [ViewName]
[(Column Alias Name...)]
AS [Query]
[WITH [CHECK OPTION] [READ ONLY] [CONSTRAINT]];
```

From the syntax,

The FORCE option allows a view to be created even if a base table that the view references does not already exist.This option is used to create a view prior to the actual creation of the base tables and accompanying data.

The NOFORCE option is the opposite of FORCE and allows a system user to create a view if they have the required privileges to create a view, and if the tables from which the view is created already exist. This is the default option.

The WITH READ ONLY option allows creation of a view that is read-only.You cannot use the DELETE,INSERT,or UPDATE commands to modify data for a read-only view.

The WITH CHECK OPTION clause allows the update of rows that can be selected through the view.It also enables you to specify constraints on values.The CONSTRAINT clause works in conjunction with the WITH CHECK OPTION clause to enable a database administrator to assign a unique name to the CHECK OPTION.If a database administrator omits the CONSTRAINT clause,Oracle will automatically assign the constraint a system-generated name that will not be very meaningful.

Types of Views

A Simple view is created on top of one table only.It is a simple SELECT query with no functions or group clause,but just selection of columns from the table without any transformation.If a DML is performed on the view, it is straightaway reflected in the base table.

A Complex view is created on multiple tables using joins.It can contain SQL functions,Group by functions.But since the view is on multiple data and selection of columns is also not simple, it does not allow DML operation on it.

Illustration

Simple View: The below simple view select employee name, department id and salary for the employees with JOB ID as DEV.

```
CREATE OR REPLACE VIEW v_emp_dev
```

```
AS
SELECT first_name, department_id, salary
FROM employees
WHERE job_id = 'DEV';
```

Complex view: The below example shows the department name, average salary drawn in the department and the count of employees working in it.

```
CREATE OR REPLACE VIEW EMP_VU
AS
SELECT department_name, AVG (salary) avg_sal, COUNT (first_name) count
FROM employees E, departments D
WHERE E.department_id = D.department_id
GROUP BY department_name;
```

DESCRIBE [view name] describes the view structure. Columns are listed in the same sequence as in the view definition.

DML operations on a View

DML operations can be easily exercised on simple views.As stated earlier, the insert, update and delete operations actually happen on the base table.

When you execute an UPDATE, DELETE, or INSERT DML statement on a view, you are actually manipulating the data rows for the base table or tables on which the view is defined.There are restrictions on the use of UPDATE, DELETE, and INSERT statements with views.First, to use the UPDATE, DELETE, or INSERT statement with a view, the view must be updateable.A view is updateable if the SELECT clause does not specify any aggregate function in the SELECT listing.Additionally, the view could not have been created through use of a GROUP BY, DISTINCT, or UNION clause or clauses.It is permissible for aggregate functions to be used in a SELECT subquery in a FROM clause. Also, the view cannot have any derived columns in the SELECT list. Next, if a view is created as the result of a JOIN operation *ajoinview*, the UPDATE and INSERT statements can only modify or insert rows into one of the base tables at a time. You cannot modify rows from two or more tables with a single data manipulation language *DML* statement.Finally, a DELETE statement can only execute against a view if a table is referenced in a FROM clause. This simply means that you cannot delete rows from a table that has not been specified.

WITH CHECK OPTION clause

WITH CHECK OPTION is an optional clause that specifies the level of checking to be done when inserting or updating data through a view.If a view is created using WITH CHECK OPTION clause, every row which gets inserted or updated in the base table through the view must comply with the view definition. Note that the option cannot be specified if the view is created as read-only.

For example, a view V_EMP_DEV is created for employees who are developers $JOB_ID = DEV$.

```
CREATE OR REPLACE VIEW v_emp_dev
AS
SELECT first_name, department_id, salary,
FROM employees
WHERE job_id = 'DEV'
WITH CHECK OPTION empvu_dev;
```

A user attempts to update salary of an HR employee through the view but encounters an exception. Its because the view was created WITH CHECK OPTION.

```
UPDATE v_emp_dev
SET salary = salary+500
WHERE JOB_ID = 'HR';
ORA-01402: view WITH CHECK OPTION where-clause violation
```

If it would have been a simple view, the UPDATE statement would not have raised any exception.

Dropping the view

A database administrator *DBA* or view owner can drop a view with the DROP VIEW statement.If a view has defined constraints, then you need to specify the CASCADE CONSTRAINTS clause when dropping a view; otherwise, the DROP VIEW statement fails to process.If another view or other database object such as a synonym or materialized view *bothoftheseobjectsarediscussedlaterinthischapter* references a dropped view,Oracle does not drop these database objects; rather, Oracle marks them as invalid.You can drop these invalid objects or redefine them in order to make them valid again.

The below DROP VIEW command drops the view EMP_VU from the database.

```
DROP VIEW EMP_VU;
```

Sequences

Oracle provides the capability to generate sequences of unique numbers for this type of use, and they are called sequences.Generally,sequences are used to generate unique,sequential integer values that are used as primary key values in database tables.A sequence of numbers can be generated in either ascending or descending order.Note that a number once generated by sequence cannot be rolled back.

Syntax

```
CREATE SEQUENCE <sequence name>
[INCREMENT BY < number >]
[START WITH < start value number>]
[MAXVALUE < MAXIMUM VLAUE NUMBER>]
[NOMAXVALUE]
[MINVALUE < minimum value number>]
[CYCLE | NOCYCLE]
[CACHE < number of sequence value to cache> | NOCACHE]
[ORDER | NOORDER];
```

From the syntax,

The CREATE SEQUENCE statement must specify a unique sequence name. This is the only required clause in the statement. If you do not specify any of the other clauses,all sequence numbers generated will follow the Oracle default settings.

The INCREMENT BY clause determines how a sequence increments as each number is generated. The default increment is one; however,if you have a good reason for a sequence to skip numbers, you can specify a different increment.A positive numeric increment generates ascending sequence numbers with an interval equal to the interval you select.A negative numeric increment generates descending sequence numbers.

The START WITH clause specifies the starting numeric value for the sequence-the default starting number is one.Additionally,you must specify a start value if you already have some rows with data in the column that will now store sequence values.

The MAXVALUE clause specifies the maximum value to which a sequence can be incremented. In the absence of a MAXVALUE, the maximum allowable value that can be generated for a sequence is quite large, 10 to the 27th power - 1. The default is NOMAXVALUE.

The MINVALUE clause specifies the minimum value of a sequence for a decrementing sequence *onethatgeneratesnumbersindescendingorder*. The default is NOMINVALUE.

The CYCLE clause specifies that sequence values can be reused if the sequence reaches the specified MAXVALUE. If the sequence cycles, numbers are generated starting again at the START WITH value.

The CACHE clause can improve system performance by enabling Oracle to generate a specified batch of sequenced numbers to be stored in cache memory.

If you specify CACHE without specifying a number,the default cache size is 20 sequence numbers.Optionally,you can specify NOCACHE to prevent the cache of sequence numbers.

The ORDER clause specifies that sequence numbers are allocated in the exact chronological order in which they are requested.

NEXTVAL and CURRVAL

Sequence values are generated through the use of two pseudo columns named currval and nextval.A pseudo column behaves like a table column, but pseudo columns are not actually stored in a table.The first time you select the nextval pseudo column, the initial value in the sequence is returned.Subsequent selections of the nextval pseudo column cause the sequence to increment as specified in the INCREMENT BY clause and return the newly generated sequence value.The currval pseudo column returns the current value of the sequence, which is the value returned by the last reference to nextval.

In a session,NEXTVAL, and not the CURRVAL must be the first action on the sequence. This is because in a session, when NEXTVAL generates the first number of the session from the sequence, Oracle keeps the current value in the CURRVAL.

Syntax:

```
Sequence.NEXTVAL
Sequence.CURRVAL
```

Points to be noted -

- CURRVAL and NEXTVAL can only be used in the Outer SQL of a select statement.
- CURRVAL and NEXTVAL can be used in INSERT statement to substitute a column primary key.It can be used both as a subquery clause and also in VALUES clause.
- CURRVAL and NEXTVAL can be used to update values in the tables.
- CURRVAL and NEXTVAL cannot be in VIEW select list,with DISTINCT keyword,with GROUP BY,HAVING,or ORDER BY clauses,and DEFAULT expression in a CREATE TABLE or ALTER TABLE statement.

Modifying the sequence

Sequence owner can modify a sequence to alter the attributes like INCREMENT BY value, MINVALUE, MAXVALUE, CYCLE or CACHE clauses only. Note that the changes done would be reflected in the upcoming numbers.

Syntax:

```
ALTER SEQUENCE [sequence name]
INCREMENT BY n
MAXVALUE n
NOCACHE
NOCYCLE
```

Dropping the sequence

The DROP SEQUENCE command drops sequences that need to be recreated or are no longer needed.

```
DROP SEQUENCE [sequence name]
```

Indexes

Indexes are the database objects that are used to tune the performance of the SELECT query.There are different types of indexes including those used to enforce primary key constraints,unique indexes,non-unique indexes,and concatenated indexes,among others.Without indexes,queries would require Oracle to scan all rows in a table in order to return the required rows for the result table.An index is created on table columns,which then stores all the values of the column under index segment.Unlike sequence,indexes are table specific.They are automatically dropped once the table has been dropped.

Indexes can be created automatically or manually.When you specify a PRIMARY KEY constraint or UNIQUE constraint,Oracle will automatically create a unique index to support rapid data retrieval for the specified table.

Alternatively,user can create indexes manually to optimize the query performance.Manually created indexes can be unique or non unique.Non-unique indexes can be B-Tree,Bitmap or Function based index.By default,Oracle creates B-Tree indexes on columns.Here is the syntax

Syntax

```
CREATE [UNIQUE][BITMAP]INDEX index
ON table (column [, column]...);
```

Note that UNIQUE and BITMAP must be specified only for unique and bitmap indexes.By default, Oracle creates B-Tree indexes for normal indexes.

A composite index *alsocalledaconcatenatedindex* is an index created on multiple columns of a table. Columns in a composite index can appear in any order and need not be adjacent columns in the table. Composite indexes enhance row retrieval speed for queries in which the WHERE clause references all or the leading portion of the columns in the composite index. An index can contain a maximum of 32 columns.

For example, a user creates index IDX_EMP on HIRE_DATE column of EMPLOYEES table.The index usage will reduce the disk I/O by traversing the indexed path scan and finds the data which is filtered on HIRE_DATE column.

```
CREATE INDEX IDX_EMP ON employees(hire_date);
```

Dropping the Index

Indexes cannot be modified but can be altered for analysis,rebuilding or stats computation purposes.If the index definition has to be modified,it has to be dropped and recreated.The syntax of the DROP INDEX command is simple.

```
DROP INDEX index_name;
```

Synonyms

A synonym is an alias,that is,a form of shorthand used to simplify the task of referencing a database object.The concept is analogous to the use of nicknames for friends and acquaintances.Referencing an object owned by another user requires the schema name to be prefixed with it. With the help of a synonym, you reduce the effort of referencing the object along with the schema name.In this way, synonym provides location transparency because the synonym name hides the actual object name and its owner.

There are two categories of synonyms, public and private.A public synonym can be used to allow easy access to an object for all system users. In fact, the individual creating a public synonym does not own the synonym-rather,it will belong to the PUBLIC user group that exists within Oracle.Private synonyms, on the other hand,belong to the system user that creates them and reside in that user's schema.

Syntax

```
CREATE [PUBLIC] SYNONYM [synonym name]
FOR OBJECT;
```

A system user can grant the privilege to use private synonyms that they own to other system users.In order to create synonyms, you need to have the CREATE SYNONYM privilege.Further, you must have the CREATE PUBLIC SYNONYM privilege in order to create public synonyms.If a synonym is declared as public,the synonym name cannot already be in use as a public synonym.Attempting to create a public synonym that already exists will cause the CREATE PUBLIC SYNONYM command to fail, and Oracle will return the ORA-00955: name is already used by an existing object error message.

Illustration

Consider two users U1 and U2.U1 has access to EMPLOYEES table. So to enable the access on EMPLOYEES table to U2 also, a synonym can be created in U2 schema. Access must be granted by U1 to U2.

```
CONN U2/U2
SQL> CREATE SYNONYM EMP_SYN FOR U1.employees;

CONN U1/U1
SQL> GRANT ALL ON EMP_SYN TO U2;

CONN U2/U2
SQL> SELECT * FROM EMP_SYN;
```

Dropping a Synonym

A uer can drop the synonym which it owns. To drop a public synonym, you must have the DROP PUBLIC SYNONYM privilege.

```
DROP SYNONYM EMP_SYN;
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js