Chapter 12

INDEXING, CLUSTERING, SEQUENCE AND PSEUDO COLUMNS

- What is an index
- Why to use an index
- Creating an index
- When Oracle does not use index
- Clustering
- Sequence
- Pseudo columns

What is an Index?

I believe, the best way of understanding an index in Oracle (or any database system) is by comparing it with the index that we find at the end of a textbook. For instance, if you want to read about *indexing* in an Oracle textbook, you will use index of the book to locate the topic *indexing*. Once the topic is found in index then you take the page number of that topic from index. Then you go to the page with that page number and start reading about indexing.

The concept of indexing in Oracle is same as the concept of book index. First let us look at the features of an index of a textbook.

- It is at the end of the textbox so that you need not search for the index in the first place.
- It contains all topics of the book in the ascending (alphabetical) order of the topics.
- After the topic the page number(s) where the topic is found in the book is listed.
- Index does increase the size of the book by a few pages.
- We use index only when we need it. In other words, we use index when we feel it is going to help locating the topic quickly.

All the characteristics of an index in a textbook will be found in an index of Oracle. The following are the characteristics of an index in Oracle.

- Index is used to search for required rows quickly.
- Index occupies extra space. Index is stored separately from table.

- Index contains the values of key – column on which index is created – in the ascending order.
- Just like the page number in book index, Oracle index stores ROWID – a unique value to internally identify each row of the table. For each entry in the index a key and corresponding ROWID are stored.
- Oracle uses index only when it feels the index is going to improve performance of the query.

Note: *ROWID is a unique value assigned to each row created in a table. Once ROWID is assigned to a row it doesn't change during the lifetime of the row. Oracle access rows internally using ROWID.*

An index in Oracle is internally stored as *Self-balancing binary tree.* A data structure that makes searching for a particular value faster.

Why To Use An INDEX

An index in Oracle is used for two purposes.

- To speed up searching for a particular value thereby improving performance of query.
- To enforce uniqueness

Using index to improving performance

Just like how you can quickly locate a particular topic in the book by using index at the end of the book, Oracle uses index to quickly locate the row with the given value in the indexed column. Indexed column is the one on which index is created.

For example if you want to search for a particular student by name then Oracle does the following without and with index.

- Without index, Oracle will start looking for the given name at the first row of the table and continues until it finds the name or until end of the table is reached. This could be very time consuming process especially for tables with many rows.
- With index, Oracle will use index to search for the given name. Since index is stored in the form of binary tree, locating the name in the index is going to be very fast. Then by using the ROWID obtained from index entry, Oracle will take the row from the table.

Tables with more number of rows will greatly benefit from indexes. Having an index and not having an index could make a lot of difference for large tables with thousands of rows.

Enforcing uniqueness with index

An index may also be used to enforce uniqueness in the column(s) given in the index. Once a UNIQUE index is created, Oracle makes sure values in the indexed column(s) are unique.

***Note**: A UNIQUE index is automatically created on the columns with PRIMARY KEY and UNIQUE constraints.*

Creating an Index

DDL command CREATE INDEX is used to create an index. The following is the syntax of this command.

```
CREATE [UNIQUE] INDEX index_name
     ON table (column-1 [, column-2]...);
```

UNIQUE keyword is used to create a unique index. Unique index makes sure that the indexed column(s) is always unique.

To create an index on NAME column of STUDENTS table, enter:

```
create index students_name_idx
 on  students (name);
```

If you want to create a unique index on NAME column of STUDENTS table, enter:

```
create unique index students_name_idx
       on students(name);
```

It is also possible to create an index on more than one column. The following command creates an index on CCODE and FCODE columns of BATCHES table.

```
create index  batches_ccode_fcode_idx
on  batches ( ccode,fcode);
```

***Note:** When two or more columns are used in an index then give the column in the order of frequency of usage. That means the most frequently used column should be given first.*

Just like an index of the book, which is not used every time you read the book, even Oracle index is not used every time by Oracle. In the next section we will see when Oracle uses index and when not.

When Oracle does not use index?

The best part of Oracle index is; it is completely automatic. That means, you never have to explicitly refer to an index. Oracle decides whether to use an index or not depending upon the query.

Oracle can understand whether using an index will improve the performance in the given query. If Oracle thinks using an index will improve performance, it will use the index otherwise it will ignore the index.

Let us assume we have an index on NAME column of STUDETNS table. Then the following query will use the index as we are trying to get information about a student based on the name.

```
select  * from students
where  name = 'Richard Marx';
```

But in the following case Oracle does NOT use index, as index's usage doesn't improve performance of the query.

SELECT doesn't contain WHERE clause

If SELECT command doesn't contain WHERE clause then Oracle doesn't use any index. Since all rows of the table are to be retrieved, Oracle will directly access the table without referring to any index.

In the following example no index of STUDENTS table will be used.

```
select  * from students;
```

SELECT contains WHERE clause, but WHERE clause doesn't refer to indexed column.

In this case SELECT contains WHERE clause but the WHERE clause is not referring to any columns on which an index is created. For example, STUDETNS table contains two indexes – one on ROLLNO, as it is primary key and another on NAME (created earlier in this chapter).

The following query uses WHERE clause but condition is based on DJ column for which there is no index.

```
select * from students
where dj > '16-jan-2001';
```

However, note that if there is any index on DJ then Oracle will use that index.

WHERE clause uses indexed columns but indexed column is modified in the WHERE clause.

In this case WHERE clause refers to indexed columns but doesn't refer to indexed column as it is.

In the query below, NAME column is used but as it is used with SUBSTR function that modifies the column in the condition, index created on NAME column will not be used.

```
select * from students
where   substr(name,4,3) = 'Joh';
```

Note: *Data dictionary view USER_INDEXES provides information about indexes.*

Creating function-based index

Prior to Oracle8i, it is not possible to create an index with an expression as the index column. Index column must be column of the table. If any function is used with indexed column in the query then Oracle does not use index. For example, if we created an index on NAME column of STUDENTS table as follows:

```
create index students_name_idx
 on students (name);
```

Then the above index cannot be used for the following query as indexed column is used with UPPER function.

```
select * from students
 where  upper(name) = 'RICHARD MARX';
```

It is also not possible to create any index based on an expression.

Oracle8i allows indexes to be created on the result of the expression. For example, the following command creates an index on the result of UPPER function.

```
create index students_name_idx
 on  students ( UPPER(name));
```

As the result if we issue the following command Oracle can use index as the expression used in the query and expression used to crate index are same.

```
select * from students
where upper(name) = 'RICHARD MARX';
```

Note: *To create a function-based index, user must have QUERY REWRITE system privilege.*

The following is another example of expression-based index. Here index is created on duration of the batch.

```
create index batches_duration_idx
   on batches ( enddate  - stdate);
```

The following query will make use of the above index.

```
select * from batches
where  enddate - stdate > 30;
```

Online indexing

Prior to Oracle8i, Oracle used to lock the table on which index is being created throughout creation process. This makes table unavailable for data manipulation during the creation of index. But Oracle8i introduced online indexing, where Oracle doesn't lock the table on which index is being built.

Online indexing is provided through the keyword ONLINE.

The following command creates index on NAME column of STUDENTS table without locking the table.

```
create index  students_name_idx
  on students (name)  online;
```

Note: *Online indexing increases availability of the table. This is a very important facility especially for Internet databases where availability matters a lot.*

Drawbacks of indexes

Though indexes increase performance of query, they can also decrease performance of data manipulation.

Many indexes on a table can slow down INSERTS and DELETES drastically. For example, if we take 5 indexes on a table then every insertion of a new row will update all 5 indexes of the

table. That means, one insertion will actually result in six insertions. Similarly every change to an indexed column will need a change to index.

Having more indexes is advantageous in case of tables that are primarily used only for querying and not updated frequently.

Though the extra space occupied by indexes is also a consideration, it may not matter much since the cost of data storage has declined substantially.

Dropping an index

You can drop an index using DROP INDEX command. It removes the named index. Removing an index will effect the performance of existing applications but not the functionality in any way.

Using and not using an index is transparent to users. Oracle decides whether to use or not on its own. However, it is possible for users to control the usage of index to certain extent using **hints,** which are directive to Oracle regarding how to execute a command. But hints are too heavy in a book meant for beginners.

```
DROP INDEX  indexname;
```

The following example drops the index created on NAME of STUDENTS table.

```
drop index student_name_idx;
```

Clustering

Clustering is a method of storing tables that are often used together (in joining) in one area of the disk. As tables that are related are stored in one area of the disk, performance of joining will improve.

In order to place tables in a cluster, tables have to have a common column. For example the following steps will create a cluster and place COURSES and BATCHES tables in the cluster.

A cluster is created with a name and a key. The key is the column, which must be present in each table that is placed in the cluster. The key is also called as *cluster key*.

Steps
The following are the required steps to create a cluster and place tables into it.

1. Create a cluster with CREATE CLUSTER command.
2. Create an index on the cluster using CREATE INDEX command. This must be done before you insert rows into clustered tables.
3. Place the tables into cluster using CLUSTER option of CREATE TABLE command.

Creating a cluster
A cluster is created using CREATE CLUSTER command. At the time of creating a cluster the name of the cluster and data type and name of the key must be given.

The following is the syntax of CREATE CLUSTER command.

```
CREATE CLUSTER clustername
    (column  datatype [, column datatype] ... );
```

The following command creates a cluster to store BATCHES and COURSES table. As CCODE is the common column of these two tables, it is to be the cluster key. We specify the data type and name of the cluster key. However, name may be anything and it is never used after cluster is created.

```
create cluster course_batches
      (ccode  varchar2(5));
```

Creating cluster index
After the cluster is created and before any rows can be inserted into tables in the cluster, an index on the cluster must be created.

CREATE INDEX command is used to create an index on the cluster. Unless an index is created on the cluster no data manipulation can be done on the tables that are placed in the cluster.

```
create index cl_ccode_index
           on cluster course_batches;
```

Placing tables into a cluster

Once a cluster is created, it may be used to store tables that are related. Each table loaded into cluster, must have a column that matches with cluster key.

A table must be placed in to cluster at the time of creation. CLUSTER option of CREATE TABLE command specifies the cluster in to which the table must be placed. It also specifies the name of the column in the table that matches the cluster key of the cluster.

The following commands create COURSES and BATCHES tables and place them into cluster.

```
create table courses
(
  ccode varchar2(5),
  name  varchar2(30),
  . . .
)
cluster course_batches(ccode);


create table batches
(
   bcode   varchar2(5),
   ccode   varchar2(5),
   . . .
)
cluster course_batches(ccode);
```

Note: *Placing a table into cluster is transparent to users. Users and application will use the table in the same manner whether the table is in the cluster or not.*

Storage of clustered tables

When two tables are placed in a cluster, they are stored together on the disk making joining these tables faster. Apart from that storing table in a cluster will also reduce the space requirement. This is because of cluster storing common column of the clustered tables only for once. In the above example, each unique CCODE is stored only for once. That means for course ORA though there are multiple batches, the value ORA is stored only for once in the cluster.

Sequence

Sequence is an object in Oracle database, which is used by multiple users to generate unique numbers. Sequence is typically used to generate primary keys like account number, employee number etc., where uniqueness and sequence matter.

In order to use a sequence, first it is to be created using CREATE SEQUENCE command. Then pseudo columns NEXTVAL and CURRVAL are used to retrieve unique values from sequence.

The following is the syntax of CREATE SEQUENCE command .

```
CREATE SEQUENCE sequencename
       [INCREMENT BY integer]
       [START WITH integer]
       [MAXVALUE integer | NOMAXVALUE]
       [MINVALUE integer | NOMINVALUE]
       [CYCLE | NOCYCLE];
```

The following is the description of available options.

Option	Meaning
START WITH	Specifies the values at which sequence must start. Default is 1.
MAXVALUE	Maximum value the sequence can generate. Default is 10e27-1.
MINVALUE	Minimum value the sequence can generate. Default is 1.
INCREMENT BY	Specifies by how much the value of the sequence is to be incremented. If you want numbers in the descending order give negative value. Default is 1.
CYCLE	Restarts numbers from MINVALUE after reaching MAXVALUE.

Table 1: Options of CREATE SEQUENCE command.

The following command creates a sequence called ROLLNO to generate roll numbers for students.

```
create sequence rollno
 start with 100
 increment by 1;
```

The above sequence starts generating numbers at 100 and increments the number by 1 every time the number is taken.

Once a sequence is created, it can be used to generate unique numbers in the sequence. Once a value is taken from sequence, Oracle automatically increments the value by one (or whatever value specified using INCREMENT BY option). Oracle guarantees uniqueness. Once it gives the number then it doesn't give that number to anyone else.

The following two pseudo columns are used to access the next and current value of the sequence.

NEXTVAL

This pseudo column will yield the next value from sequence and automatically increments the value of the sequence.

CURRVAL

This returns the value that is taken by most recent NEXTVAL. This cannot be used unless NEXTVAL is first called.
The following examples will demonstrate how to use NEXTVAL and CURRVAL pseudo columns.

```
select  rollno.nextval  from  dual;

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

As the sequence starts with 100, first NEXTVAL returned 100. And it also increments the sequence by 1. See the next example below.

```
select  rollno.nextval  from  dual;

  NEXTVAL
---------
      101
```

CURRVAL pseudo column returns the current value of the sequence, which is the value returned by most recent NEXTVAL. In the following example CURRVAL returns 101 since that is the most recent value returned by NEXTVAL.

```
select rollno.currval from dual;

  CURRVAL
---------
      101
```

CURRVAL is used to reuse the values returned by most recent NEXTVAL.

The real usage of sequence is in inserting rows into table. The following INSERT command will use ROLLNO sequence to get next available roll number for a new student.

```
insert  into  students
       values (rollno.nextval ,   ...);
```

And if you want to insert a row into PAYMENTS table immediately with the same roll number then you can use CURRVAL as follows.

```
insert  into payments
      values ( rollno.currval, ...);
```

Though a sequence can guarantees uniqueness and sequence, its usage may not guarantee consecutiveness. See the following scenario.

```
insert into  students  values ( rollno.nextval, . . . );

commit;

insert into  students  values ( rollno.nextval, . . . );

rollback;

insert into  students  values ( rollno.nextval, . . . );

commit;
```

In the above scenario, if you take the value of ROLLNO.NEXTVAL as 102 for first INSERT then the value in the second INSERT will be 103. But as second INSERT command is rolled back that number is not actually used as roll number. Third INSERT command now takes 104 but not 103 as Oracle returns next available number, which is 104.

As you can see, 103 is not used at all in the roll number. While using sequence one must be aware of this potential gaps.

Altering a sequence

Some attributes of a sequence can be altered using ALTER SEQUENCE command.

```
ALTER SEQUENCE sequencename
         [INCREMENT BY integer]
         [MAXVALUE integer | NOMAXVALUE]
         [MINVALUE integer | NOMINVALUE]
```

For instance, the following command changes MAXVALUE of ROLLNO sequence to 1000.

```
alter sequence rollno maxvalue 1000;
```

Dropping a Sequence

DROP SEQUENCE command is used to drop a sequence.

```
DROP SEQUENCE sequencename;
```

The following command drops ROLLNO sequence.

```
drop sequence rollno;
```

Pseudo Columns

A pseudo column is used like a column in a table but it is not a column of any table. You can use pseudo columns just like table columns but you cannot modify them.

The following is the list of available pseudo columns in Oracle.

Pseudo Column	Meaning
CURRVAL	Returns the current value of a sequence.
NEXTVAL	Returns the next value of a sequence.
NULL	Return a null value.
ROWID	Returns the ROWID of a row. See ROWID section below.
ROWNUM	Returns the number indicating in which order Oracle selects rows. First row selected will be ROWNUM of 1 and second row ROWNUM of 2 and so on.
SYSDATE	Returns current date and time.
USER	Returns the name of the current user.
UID	Returns the unique number assigned to the current user.

Table 2: Pseudo Columns.

The following are a few examples on usage of pseudo columns.

Display the name of the current user and user's id.

```
SQL> select user, uid from dual;

USER                                UID
------------------------------ ---------
BOOK                                38
```

ROWID

Pseudo column ROWID returns the address of each row in the table. Oracle assigns a ROWID to each row. Oracle uses ROWID internally to access rows. For instance, Oracle stores ROWID in index and uses it to access the row in the table.

You can display ROWID of rows using SELECT command as follows:

```
select rowid, ccode from courses;

ROWID              CCODE
------------------ -----
AAADC5AABAAAKaSAAA ora
AAADC5AABAAAKaSAAB vbnet
AAADC5AABAAAKaSAAC c
AAADC5AABAAAKaSAAD asp
AAADC5AABAAAKaSAAE java
AAADC5AABAAAKaSAAF xml
AAADC5AABAAAKaSAAG cs
```

Oracle provides a package called DBMS_ROWID to decode ROWID. The following SELECT command displays only the row number from ROWID.

```
select  dbms_rowid.rowid_row_number(rowid) as rownumber ,ccode
from
courses

ROWNUMBER CCODE
--------- -----
        0 ora
        1 vbnet
        2 c
        3 asp
        4 java
        5 xml
        6 cs
```

Note: *Once a row is assigned a ROWID Oracle does not change ROWID during the lifetime of the row. As the result you may see some row numbers missing in the output of above command. It means those rows were deleted.*

For more information about ROWID and DBMS_ROWID package please see online documentation.

ROWNUM

This pseudo column yields the order in which Oracle has selected rows.

The following command displays row number of each row.

```
select rownum, ccode from courses;

   ROWNUM CCODE
--------- -----
        1 ora
        2 vbnet
        3 c
        4 asp
        5 java
        6 xml
        7 cs
```

ROWNUM can also be used in conditions but only two operators; **<** and <=. make sense with ROWNUM.

The following query displays first five courses.

```
select rownum, ccode from courses
where  rownum <= 5

   ROWNUM CCODE
--------- -----
        1 ora
        2 vbnet
        3 c
        4 asp
        5 java
```

However, the following is not meaningful usage of ROWNUM. It is because of the fact that Oracle assigns number to row after row is selected.

```
select rownum, ccode from courses
where  rownum = 5;

no rows selected
```

Also remember using Oracle assigns numbers to rows before rows are ordered using ORDER BY clause as demonstrated by the following query.

```
select rownum,ccode,duration,fee from courses
order by fee;

   ROWNUM CCODE  DURATION       FEE
--------- ----- --------- ---------
        3 c            20      3500
        6 xml          15      4000
        1 ora          25      4500
        5 java         25      4500
        4 asp          25      5000
        2 vbnet        30      5500
        7 cs           30      7000
```

Summary

Index and cluster are used to improve performance. The concept of index in Oracle is similar to index found at the end of textbook. Oracle doesn't always use index instead Oracle uses index only when it feels, usage of index improves performance. Cluster is another method of improving performance by storing related tables together on the disk.

Sequence is an object used to generate unique number in a sequence. Pseudo column yield values and used as columns of the table though they are not columns of any table.

Exercises

1. Which constraints automatically create index?
2. What does ONLINE option in CREATE TABLE command do?
3. How do you create an index on FCODE and CCODE of BATCHES table.
4. _____ option in CREATE SEQUENCE is used to generate numbers in reverse order.
5. _____ is the pseudo column used to get the next available number from a sequence.
6. _____ option is used in CREATE TABLE command to place the table in a cluster.
7. Which option of CREATE INDEX is used to create an index on the cluster.
8. Create a sequence called REVERSE to generate numbers in the descending order from 10000 to 1000 with a decrement of 5.
9. Create a cluster to store STUDENTS and PAYMENTS tables and place tables into it.
10. Change the decrement value of sequence REVERSE (created earlier) to 2.

11. Is it possible to place an existing table into cluster? If yes how? If no, then what is the remedy you suggest?
12. What is the purpose of ROWID?