COMP 3311 Database Management Systems Spring 2015

Lab 6. Oracle indexing, Oracle clustering and more about PL/SQL.

Objectives of the Lab

- After this lab you should be able to
  - know how to create an index for a table in Oracle,
  - know how to create a cluster for tables in Oracle,
  - know more about PL/SQL.

Getting the lab SQL script file

- Download the lab6.sql file as follows
  - 1. login to an arbitrary machine csl2wkxx.cse.ust.hk where xx=01-40
  - 2. at the command prompt type

    csl2wk01:lamngok:235> cd ~

    csl2wk01:lamngok:236> wget \
    ?http://course.cs.ust.hk/comp3311/labs/lab6.sql

- Log into Oracle database server using SQL*Plus with your password.
- Execute the script lab6.sql at the prompt

  SQL> @lab6.sql

Oracle Indexing 1

- An index can be created in the Oracle database system to speed up record retrievals.
- An index is defined on one or more columns.
- An index occupies extra space and is stored separately from the table.
- Oracle uses the index only when the index is estimated to improve the performance.

Oracle Indexing 2

- In Oracle, a unique column known as "ROWID" is used to identify records internally.
- The "key" of the index corresponds to the values of the columns on which the index is created.
- When an index is created, the index entries will hold the values of the "key" and the "ROWID" of the records.

Oracle Indexing 3

- Whenever appropriate, the Oracle system uses the index to search for records.
- Since the index is stored in a balanced tree, data retrieval from the index is faster.
- The ROWID information obtained from the index will be used by the Oracle system to directly locate the record in the file system.
- Large tables with large number of records will benefit greatly from the indexes.

Oracle Indexing 4

- The syntax for creating an index

```
CREATE [UNIQUE] INDEX index_name ON table (column1, column2, …);
```

- The UNIQUE keyword specifies that the column(s) must have unique values.

Oracle Indexing 5

- An example

```
CREATE UNIQUE INDEX facility_idx
ON facility (department_id, name);
```

- The above example creates an index on the department_id and name columns of the table facility. The tuples (department_id,name) must be unique.

Oracle indexing 6

- Oracle does not always use the index. An index will not be used in the following scenarios:
  - The SELECT statement does not contain the WHERE clause:

    ```
    SELECT * from facility;
    ```
  - The SELECT statement contains the WHERE clause, but the WHERE clause does not refer to the indexed column(s):

    ```
    SELECT * FROM facility where no_of_computers=60;
    ```
  - The indexed column(s) is/are modified by some function(s) in the WHERE clause:

    ```
    SELECT * FROM facility where
    substr(name,1,8)='Computer';
    ```

Oracle indexing 7

- A function-based index can be created as follows:

  ```
  CREATE INDEX function_idx ON facility (substr(name,1,8));
  ```

- We can check the names of all the indexes created:

  ```
  SELECT index_name FROM user_indexes;
  ```

- We can also drop an index:

  ```
  DROP INDEX index_name;
  ```

- To create the function-based index, the user must have the QUERY REWRITE system privilege.

Oracle indexing 8

- When the Oracle database system is creating an index for a table, the table is locked, and will not be available for data manipulations.
- This could be undesirable in the real-world working environment.
- Starting from version 8i, Oracle supports online indexing by using the keyword ONLINE:

```
CREATE UNIQUE INDEX facility_idx ON facility (department_id,name) ONLINE;
```

Oracle indexing 9

- Creating an index could slow down the insertion and deletion operations.
- If the table is to be updated frequently with the insertion and deletion statements, creating and maintaining the index could be a big overhead to the system.
- Index is good for tables that are primarily used for querying and tables that do not require to be updated frequently.

Oracle clustering 1

- Oracle can store tables that are often used together (i.e. in JOIN operations) in the same data blocks. This is known as "clustering".
- Clustering reduces unnecessary I/O accesses, thus improves the performance.
- To place tables in a cluster, the tables need to have a common column.
- The user needs the "CREATE CLUSTER" system privilege in order to create a cluster.

[+] *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 2

- To create a cluster[+] in Oracle, you need to refer to the following steps:
  - 1. Create a cluster using the CREATE CLUSTER command.
  - 2. Create an index on the cluster using the CREATE INDEX command. THIS MUST BE DONE BEFORE ANY RECORDS ARE INSERTED INTO THE TABLES.
  - 3. Place the tables into the cluster with the CLUSTER option in the CREATE TABLE statement.

[+] *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 3

- Step 1: creating a cluster
- The syntax for creating a cluster:

  CREATE CLUSTER clustername (column1 datatype, column2 datatype,…)
- An example

  CREATE CLUSTER department_facility

  (department_id varchar2(4));
- The above example create a cluster named department_facility, department_id is known as the cluster key.

+ *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 4

- Step 2: creating an index
- Oracle requires the cluster to have an index (otherwise record insertion will not be allowed).
- But Oracle will not build an index automatically for the cluster.
- The following statement creates an index on the department_facility cluster:

```
CREATE INDEX
department_facility_idx
ON CLUSTER department_facility;
```

+ *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 5

- Step 3: placing tables into a cluster
- The tables being put into the cluster must have a column that matches with the cluster key.
- The following statement creates the departments table and put it into the cluster:

```
CREATE TABLE departments
( department_id       varchar2(4) not null,
    name varchar2(40),
    room_number       number(4)  )
 CLUSTER department_facility(department_id);
```

+ *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 6

- The following statement creates the facility table and put it into the cluster:

```
CREATE TABLE facility
( department_id    varchar2(4) not null,
  name       varchar2(40),
 no_of_projectors number(4),
  no_of_computers  number(5))
CLUSTER department_facility(department_id);
```

- In Oracle you can not put existing tables into a cluster.
- The tables are required to be put into the cluster when they are created.
- To see the names of all of the clusters created we can use the following command:

```
SELECT cluster_name FROM user_clusters;
```

+ *Don't worry if you do not have the privilege to create a cluster.*

Oracle clustering 7

- The following statement drop the cluster we created:

```
DROP CLUSTER department_facility
[INCLUDING TABLES
[CASCADE CONSTRAINTS]];
```

- INCLUDING TABLES: drops the cluster as well as tables in that cluster
- CASCADE CONSTRAINTS: drop any referential integrity constraints that refer to primary or unique keys in tables.
- You cannot uncluster an individual table. Instead you must perform these steps:
  - Create a new table with the same structure and contents as the old one, but with no CLUSTER clause.
  - Drop the old table.
  - Use the RENAME statement to give the new table the name of the old one.

More about PL/SQL 1

- Just a reminder, the basic structure of PL/SQL is as follows:

```
DECLARE
 /* Declarative section: variables, types, and local subprograms. */

BEGIN
 /* Executable section: procedural and SQL statements go here. */
/* This is the only section of the block that is required. */

EXCEPTION
/* Exception handling section: error handling statements go here. */

END;
```

More about PL/SQL 2 (additional flow control statements)

- In addition to the control statements we discussed in the last lab, there are some more control statements in PL/SQL. The following is a list of them (covered statements are in red):

  Conditional control statements:
  - IF … THEN … ELSIF … ELSE … END IF;
  - CASE … WHEN … THEN … ELSE … END CASE;

  Iterative statements:
  - LOOP … END LOOP;
  - WHILE … LOOP … END LOOP;
  - FOR … IN … LOOP … END LOOP;

More about PL/SQL 3 (exceptions)

- Handling exceptions in PL/SQL
  - Predefined exceptions
    - NO_DATA_FOUND, TOO_MANY_ROWS, etc
    - For the complete listing of predefined exceptions, refer to: http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96624/07_errs.htm
  - User-defined exceptions
    - Defined by the users
    - Raised explicitly by users using the RAISE command:

      RAISE <exception name>;

More about PL/SQL 4 (exceptions)

- To use a user-defined exception, we need to:
  - Declare the exception under the "DECLARE" section,
  - Raise it (whenever applicable) under the "BEGIN" section,
  - Define the codes under the "EXCEPTION" section.

More about PL/SQL 5 (exceptions)

- We will show the use of a user-defined exception cga_too_low in a piece of PL/SQL code on the next slide.
- The code checks a student with the email address 'lamngok'.
- If the cga value of that student is lower than 10 then a user-defined exception will be raised:
  - A message 'LAM IS LAZY' will be displayed to the screen.
  - The student's last_name will be updated to 'LAZY'

More about PL/SQL 6 (exceptions)

```
DECLARE
      cga_too_low EXCEPTION;
      cga_of_lam  NUMBER(2);
BEGIN
      SELECT cga INTO cga_of_lam
      FROM students WHERE email='lamngok';
      IF (cga_of_lam<10) THEN
            RAISE cga_too_low;
      END IF;

EXCEPTION
      WHEN cga_too_low THEN
            DBMS_OUTPUT.PUT_LINE('LAM IS LAZY!');
            UPDATE students SET last_name='LAZY'
            WHERE email='lamngok';
END;
/
```

+ If SQLPlus does not display the sentence "LAM IS LAZY!", please type "set serveroutput on;" at the SQLPlus prompt and run the PL/SQL block again

Conclusions

- We covered the following topics in this lab:
  - creating an index for an table in Oracle,
  - creating a cluster for tables in Oracle,
  - further PL/SQL knowledge.