COMP 3311
Database Management Systems

Lab 6

Oracle Indexing
and PL/SQL Exceptions

Lab Objectives

- After this lab you should
  - know how to create an index for a table in Oracle.
  - know more about PL/SQL.
- Specific information about Oracle indexing and PL/SQL can be found by following the links given at the top of some of the slides.

Oracle Indexing

http://docs.oracle.com/cd/B28359_01/server.111/b28310/indexes003.htm#ADMIN11722

- An index can be created in Oracle on one or more attributes to speed up retrievals.
- An index is stored in a balanced tree separately from the table.
- Oracle uses the index only when the index is estimated to improve query performance.
- Creating an index could slow down insertion and deletion operations.
- Thus, an index is good for tables that are primarily used for querying and for tables that do not need to be updated frequently.

Oracle Indexing (cont'd)

- Oracle uses a unique attribute known as ROWID to identify records internally for each table.
- The key of the index corresponds to the values of the attributes 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 containing the values of key.
- The ROWID information obtained from the index is used by Oracle to directly locate the record in the file system.

Oracle Indexing: Creating an Index

- The syntax for creating an index is:

  create [**unique**] **index** *index_name* **on** *table_name* (*attribute_name1*, *attribute_name2*, ...);

  Example: Create an index on the combination of departmentId and name of the Facility table.

  create unique index FacilityIndex **on** Facility (departmentId, name);

- The **unique** keyword specifies that the attribute or combination of attributes, such as (departmentId, name) in the example, must have unique values.

Oracle Indexing: When Not Used

- Oracle does not use an index when processing a query in the following scenarios:
  - The SELECT statement does not contain a WHERE clause:

    Example: select * from Facility;
  - The SELECT statement contains a WHERE clause, but the WHERE clause does not refer to the indexed attribute(s):

    Example: select * from Facility where numberComputers=60;
  - The indexed attribute(s) is/are modified by some function(s) in the WHERE clause:

    Example: select * from Facility where substr(name, 1, 8)='computer';

Oracle Indexing: Administration

- The names of all the indexes created can be checked as follows.

  ```
  select index_name from user_indexes;
  ```

- An index can be dropped as follows.

  ```
  drop index index_name;
  ```

PL/SQL: Basic Structure

- **RECALL:** Basic structure of PL/SQL.

declare	**Declarative section:** declaration of variables, types, and local subprograms go here.
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;	

PL/SQL: Exceptions

http://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm

- Predefined exceptions
  - Listed to the right.
  - Refer to the link at the top of this page for a detailed explanation.
- User-defined exceptions
  - Defined by the users.
  - Raised explicitly by users using the **raise** command.

raise *exception_name*;

ACCESS_INTO_NULL	ORA-06530
CASE_NOT_FOUND	ORA-06592
COLLECTION_IS_NULL	ORA-06531
CURSOR_ALREADY_OPEN	ORA-06511
DUP_VAL_ON_INDEX	ORA-00001
INVALID_CURSOR	ORA-01001
INVALID_NUMBER	ORA-01722
LOGIN_DENIED	ORA-01017
NO_DATA_FOUND	ORA-01403
NOT_LOGGED_ON	ORA-01012
PROGRAM_ERROR	ORA-06501
ROWTYPE_MISMATCH	ORA-06504
SELF_IS_NULL	ORA-30625
STORAGE_ERROR	ORA-06500
SUBSCRIPT_BEYOND_COUNT	ORA-06533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532
SYS_INVALID_ROWID	ORA-01410
TIMEOUT_ON_RESOURCE	ORA-00051
TOO_MANY_ROWS	ORA-01422
VALUE_ERROR	ORA-06502
ZERO_DIVIDE	ORA-01476

PL/SQL: Using User-defined Exceptions

- Declare the exception in the declare section.

```
my_exception exception;
```

- Raise the exception in a begin section.

```
if condition then
    raise my_exception;
end if;
```

- Define the exception-handling code in the exception section.

```
exception
    when my_exception then
        ⋮
```

PL/SQL: Continuing After An Exception

- Execution of the block in which an exception is raised terminates after the exception is handled.
- To continue execution after an exception is raised, the statement that can cause the exception must be placed within its own sub-block (i.e., inside its own **begin** ... **end** block).
- Execution then resumes after the sub-block in which the exception is raised.

PL/SQL: Continuing After An Exception Example

```
declare
    peRatio number(3,1);
begin
    delete from stats where symbol = 'xyz';
    begin -- sub-block begins
        -- The select statement will throw an exception if nvl(earnings, 0) is zero
        select price / nvl(earnings, 0) into peRatio from stocks where symbol = 'xyz';
    exception
        when zero_divide then
            peRatio := 0;
    end; -- sub-block ends
    insert into stats (symbol, ratio) values ('xyz', peRatio);
exception
    when others then
    ...
end;
```

After the zero_divide exception is handled, execution continues with the insert statement, which is outside the inner begin ... end block.

Note: The others keyword is used to handle any exceptions that are not explicitly named.

Summary

- This lab covered the following topics:
  - Creating an index for a table in Oracle.
  - Exceptions in PL/SQL.

Lab Exercise

- You must complete the lab exercise and upload the result to Canvas by **11:59 p.m. this Friday**.

Ask for help if you need it!

IMPORTANT NOTE

If you want to save your modified SQL script files, copy them to the M drive or to a USB drive as any personal files on the lab computers will be automatically deleted periodically.