

Creating Indexes

To create an index in your own schema, at least one of the following conditions must be true:

- The table or cluster to be indexed is in your own schema.
- You have INDEX privilege on the table to be indexed.
- You have CREATE ANY INDEX system privilege.

```
CREATE INDEX emp_ename ON emp(ename);
```

```
CREATE INDEX emp_ename ON emp(ename) TABLESPACE users STORAGE  
(INITIAL 20K NEXT 2010;
```

TOPIC 99

Data Dictionary Views for Tables Three Types of Views

There are three types of Views to manage Tables related information:

DBA_

ALL_

USER_

Data Dictionary Views for Tables

Displaying Views for Tables

SQL statement to display Tables related Views:

```
Select Table_name From Dict
```

```
Where Table_name like '%TAB%';
```

You can use following choices for more views:

```
Table_name like '%COL%' Table_name like '%EXTERNAL%' Table_name like  
'%CONSTRAINT%'
```

Size of a Table

To find size of table in MB

```
select sum(bytes)/1024/1024 SizeMB from dba_segments
where segment_name='EMP';
```

Data Dictionary Views for Tables

Displaying Table's Constraints

```
select u.constraint_name, u.CONSTRAINT_TYPE, substr(c.column_name,1,30) col_name
from user_constraints u, USER_CONS_COLUMNS c
where u.table_name='EMP'
and u.CONSTRAINT_TYPE = 'P'
and (u.table_name=c.table_name
and c.constraint_name=u.constraint_name);
```

CONSTRAINT_NAME	C COL_NAME
PK_EMP	P EMPNO

Data Dictionary Views for Tables

Displaying Table's Columns Details

Column information, such as name, data type, length, precision, scale, and default data values can be listed using one of the views ending with the _COLUMNS suffix. For example, the following query lists all of the default column values for the emp and dept tables:

```
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH,
LAST_ANALYZED FROM DBA_TAB_COLUMNS
```

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	LAST_ANALYZED
COUNTRIES	COUNTRY_ID	CHAR	2	05-FEB-03
COUNTRIES	COUNTRY_NAME	VARCHAR2	40	05-FEB-03
COUNTRIES	REGION_ID	NUMBER	22	05-FEB-03
DEPARTMENTS	DEPARTMENT_ID	NUMBER	22	05-FEB-03

WHERE OWNER = 'HR' ORDER BY TABLE_NAME;

Introduction to Database Administration

TOPIC 100

Creating & Altering an Index

Creating & Altering an Index

Creating & Altering Index

Altering Indexes

Creating Indexes

Creating & Alter Indexes

Rebuild Index

Making Index Visible/ Invisible

Making Index Useable

Creating & Altering an Index

Altering Indexes

To alter an index, your schema must contain the index or you must have the ALTER ANY INDEX system privilege. With the ALTER INDEX statement, you can:

Rebuild or coalesce an existing index

Deallocate unused space or allocate a new extent Specify parallel execution (or not) and alter the degree of parallelism

Alter storage parameters or physical attributes Specify LOGGING or NOLOGGING

Creating & Altering an Index

Creating Indexes

Enable or disable key compression

Mark the index unusable
Make the index invisible
Rename the index
Start or stop the monitoring of index usage
Creating & Altering an Index

Creating Indexes

```
CREATE INDEX emp_ename ON emp(ename)
TABLESPACE users
STORAGE (INITIAL 20K NEXT 20k);
```

```
CREATE UNIQUE INDEX
dept_unique_index ON dept (dname) TABLESPACE indx;
```

Creating & Altering an Index

Creating & Alter Indexes

```
CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY, age  INTEGER)
ENABLE PRIMARY KEY
USING INDEX TABLESPACE Work_Index;
```

```
ALTER INDEX emp_ename STORAGE (NEXT 40);
```

Creating & Altering an Index

Rebuild Index

You have the option of rebuilding the index online. Rebuilding online enables you to update base tables at the same time that you are rebuilding.
The following statement rebuilds the emp_name index online:

```
ALTER INDEX emp_name REBUILD ONLINE;
```

Creating & Altering an Index

Making Index Visible/ Invisible

An invisible index is ignored by the optimizer unless you explicitly set the OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter to TRUE at the session or system level. Making an index invisible is an alternative to making it unusable or dropping it.

```
ALTER INDEX index INVISIBLE; ALTER INDEX index VISIBLE;
```

Creating & Altering an Index

Making Index Visible/ Invisible

An invisible index is ignored by the optimizer unless you explicitly set the `OPTIMIZER_USE_INVISIBLE_INDEXES` initialization parameter to `TRUE` at the session or system level. Making an index invisible is an alternative to making it unusable or dropping it.

`ALTER INDEX index INVISIBLE;` `ALTER INDEX index VISIBLE;`

Creating & Altering an Index

Making Index Useable

Query the data dictionary to determine whether an existing index or index partition is usable or unusable.

```
SELECT INDEX_NAME AS "INDEX OR PART NAME", STATUS, SEGMENT_CREATED
FROM USER_INDEXES
```

INDEX OR PART NAME	STATUS	SEG
I_EMP_ENAME	N/A	N/A
JHIST_EMP_ID_ST_DATE_PK	VALID	YES
JHIST_JOB_IX	VALID	YES
JHIST_EMPLOYEE_IX	VALID	YES
JHIST_DEPARTMENT_IX	VALID	YES
EMP_EMAIL_UK	VALID	NO

UNION ALL

```
SELECT PARTITION_NAME AS "INDEX OR PART NAME",
STATUS, SEGMENT_CREATED FROM USER_IND_PARTITIONS;
```

Introduction to Database Administration

TOPIC 101

Recycle Bin

Recycle Bin

Schema Recycle Bin

What is the Recycle Bin?

What is the Recycle Bin?

Object Naming the Recycle Bin

Recycle Bin Tables

Enable or Disable Recycle Bin
Purging Objects in the Recycle Bin
Restoring Objects From Recycle Bin
Enable or Disable Recycle Bin
Recycle Bin

What is the Recycle Bin?

The recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space.

Recycle Bin

What is the Recycle Bin?

They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints.

Each user can be thought of as having his own recycle bin, because, unless a user has the SYSDBA privilege.

```
SELECT * FROM RECYCLEBIN;
```

Recycle Bin

Object Naming the Recycle Bin

When a dropped table is moved to the recycle bin, the table and its associated objects are given system-generated names. This is necessary to avoid name conflicts that may arise if multiple tables have the same name. circumstances are:

A user drops a table, re-creates it with the same name, then drops it again.

Two users have tables with the same name, and both users drop their tables.

`BIN$unique_id$version`
Recycle Bin

Recycle Bin Tables

```
SQL> select * from tab;
```

TNAME	TABTYPE	CLUSTERID
BIN\$hqQXpmBGShVHay6wPyUlQ==\$0	TABLE	
BIN\$n4XwANZSS3mT8l+SHf6gGA==\$0	TABLE	
BONUS	TABLE	
DEPT	TABLE	
EMP	TABLE	
SALGRADE	TABLE	

6 rows selected.

Recycle Bin

Enable or Disable Recycle Bin

To disable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = OFF;
```

```
ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

To enable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = ON;
```

```
ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

Ref. DBA E25494 Oracle11g.pdf pp. 20-58

Recycle Bin

Purging Objects in the Recycle Bin

If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space. You need the same privileges as if you were dropping the item.

PURGE TABLE "BIN\$jsleilx392mk2=293\$0"; PURGE TABLESPACE example;
PURGE TABLESPACE example USER oe;
Recycle Bin

Restoring Objects From Recycle Bin

The following example restores int_admin_emp table and assigns to it a new name:

FLASHBACK TABLE int_admin_emp TO BEFORE DROP
RENAME TO
int2_admin_emp;
Recycle Bin
Enable or Disable Recycle Bin

```
SELECT object_name, original_name FROM dba_recyclebin  
WHERE owner = 'HR';
```

OBJECT_NAME	ORIGINAL_NAME

BIN\$yrMKlZaLMhfgNAgAIMenRA==\$0	EMPLOYEES

You can also view the contents of the recycle bin using the SQL*Plus command SHOW RECYCLEBIN.

```
SQL> show recyclebin
```

ORIGINAL NAME	RECYCLEBIN NAME	OBJECT TYPE	DROP TIME

EMPLOYEES	BIN\$yrMKlZaVMhfgNAgAIMenRA==\$0	TABLE	2003-10-27:14:00:19

Ref. DBA E25494 Oracle11g.pdf pp. 20-59
Introduction to Database Administration

TOPIC 102

Dropping Index

Dropping Index

Dropping Index

Dropping Index

Reasons for Dropping Index

Extents for Index

SQL for Dropping Index

When not to Drop Index

Dropping Index

Dropping Index

To drop an index, the index must be contained in your schema, or you must have the DROP ANY INDEX system privilege.

Dropping Index

Reasons for Dropping Index

Some reasons for dropping an index include:

The index is no longer required.

The index is not providing anticipated performance improvements for queries issued against the associated table. For example, the table might be very small, or there might be many rows in the table but very few index entries.

Dropping Index

Reasons for Dropping Index

Applications do not use the index to query the data.

The index has become invalid and must be dropped before being rebuilt.

The index has become too fragmented and must be dropped before being rebuilt.

Dropping Index

Extents for Index

When you drop an index, all extents of the index segment are returned to the containing tablespace and become available for other objects in the tablespace.

Dropping Index

SQL for Dropping Index

The following statement drops the emp_ename index:

DROP INDEX emp_ename;

Dropping Index

When not to Drop Index

You cannot drop only the index associated with an enabled UNIQUE key or PRIMARY KEY constraint. To drop a constraints associated index, you must disable or drop the constraint itself.

End

Introduction to Database Administration

TOPIC 103

Data Dictionary Views for Index

Data Dictionary Views for Index

Data Dictionary Views

Three Types of Views

Displaying Views for Indexes

Display Index Name

Displaying Index on Table's Column

Displaying Index's Columns Details

Data Dictionary Views for Index

Three Types of Views

There are three types of Views to manage Index related information:

DBA_ ALL_ USER_

Data Dictionary Views for Index

Displaying Views for Indexes

SQL statement to display Indexes related Views:

Select Table_name
From Dict
Where Table_name like '%IND%';

You can use following views as well:
INDEX_STATS INDEX_HISTOGRAM V\$OBJECT_USAGE
Data Dictionary Views for Index

Display Index Name

To find table which contains Index

```
SQL> set linesize100
SQL> select index_name, index_type, Table_name
  2  from user_indexes
  3  where Table_name = 'EMP';
```

INDEX_NAME	INDEX_TYPE	TABLE_NAME
PK_EMP	NORMAL	EMP

Data Dictionary Views for Index

Displaying Index on Table's Column

```
SQL> select i.index_name, i.index_type, substr(c.column_name, 1, 20) col_name
  2  from user_indexes i, user_ind_columns c
  3  where i.Table_name = 'EMP'
  4  and i.table_name=c.table_name;
```

INDEX_NAME	INDEX_TYPE	COL_NAME
PK_EMP	NORMAL	EMPNO

Data Dictionary Views for Index

Displaying Index's Columns Details

create index ind_name on EMP(ename);

```
select i.index_name, i.index_type, substr(c.column_name, 1, 20) col_name
```

from user_indexes i, user_ind_columns c where i.Table_name = 'EMP'
and i.table_name=c.table_name

INDEX_NAME	INDEX_TYPE	COL_NAME
IND_NAME	NORMAL	ENAME
PK_EMP	NORMAL	EMPNO

and i.index_name=c.index_name;

Introduction to Database Administration

TOPIC 104

Defining Clusters

Defining Clusters

Clusters

What is a Cluster?

Cluster Key

Choosing Appropriate Tables for Cluster

Appropriate Column for Cluster

Defining Clusters

What is a Cluster?

A cluster provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together.

For example, the

emp and dept table share the deptno column. When you cluster the emp and dept tables, Oracle stores all rows for each department from both the emp and dept tables in the same data blocks.

Defining Clusters

What is a Cluster?

A cluster provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together.

For example, the

emp and dept table share the deptno column. When you cluster the emp and dept tables, Oracle stores all rows for each department from both the emp and dept tables in the same data blocks.

Defining Clusters

Cluster Key

The cluster key is the column, or group of columns, that the clustered tables have in common. You specify the columns of the cluster key when creating the cluster.

You subsequently specify the same columns when creating every table added to the cluster. Each cluster key value is stored only once each in the cluster and the cluster index, no matter how many rows of different tables contain the value.

Defining Clusters

Cluster Key

After creating a cluster, you can create tables in the cluster. However, before any rows can be inserted into the clustered tables, a cluster index must be created.

Using clusters does not affect the creation of additional indexes on the clustered tables; they can be created and dropped as usual.

Defining Clusters

Choosing Appropriate Tables for Cluster

Use clusters for tables for which the following conditions are true:

The tables are primarily queried--that is, tables that are *not* predominantly inserted into or updated.

Records from the tables are frequently queried together or joined.

Defining Clusters

Appropriate Column for Cluster

Choose cluster key columns carefully. If multiple columns are used in queries that join the tables, make the cluster key a composite key.

In general, the characteristics that indicate a good cluster index are the same as those for any index.

End

Introduction to Database Administration

TOPIC 105

Create & Alter Clusters

Create & Alter Clusters

Create & Alter Clusters

Creating a Cluster

SQL DDL Command for Cluster

Creating a Table in a Cluster

Creating Cluster using SQL

Altering Clusters

Dropping Clusters

Create & Alter Clusters

Creating a Cluster

To create a cluster in your schema, you must have the CREATE CLUSTER system privilege and a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.

To create a cluster in another user's schema you must have the CREATE ANY CLUSTER system privilege, and the owner must have a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.

Create & Alter Clusters

SQL DDL Command for Cluster

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
```

```
SIZE 600
```

```
TABLESPACE users
```

```
STORAGE (INITIAL 200K NEXT 300K MINEXTENTS 2 PCTINCREASE 33);
```

Create & Alter Clusters

Creating a Table in a Cluster

To create a table in a cluster, you must have either the CREATE TABLE or CREATE ANY TABLE system privilege. You do not need a tablespace quota or the UNLIMITED TABLESPACE system privilege to create a table in a cluster.

Create & Alter Clusters

Creating Cluster using SQL

```
CREATE TABLE emp (  
empno NUMBER(5) PRIMARY KEY,  
ename VARCHAR2(15) NOT NULL,  
...  
deptno NUMBER(3) REFERENCES dept) CLUSTER emp_dept (deptno);
```

```
CREATE TABLE dept (  
deptno NUMBER(3) PRIMARY KEY, ... )  
CLUSTER emp_dept (deptno);
```

Create & Alter Clusters

Altering Clusters

To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
Physical attributes (INITTRANS and storage characteristics)

The average amount of space required to store all the rows for a cluster key value (SIZE)

The default degree of parallelism

Create & Alter Clusters

Altering Clusters

To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
Physical attributes (INITTRANS and storage characteristics)

The average amount of space required to store all the rows for a cluster key value (SIZE)

The default degree of parallelism

Create & Alter Clusters

Dropping Clusters

To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
Physical attributes (INITTRANS and

storage characteristics)

The average amount of space required to store all the rows for a cluster key value (SIZE)

The default degree of parallelism

Create & Alter Clusters

Dropping Clusters

DROP CLUSTER emp_dept INCLUDING TABLES;

DROP CLUSTER emp_dept INCLUDING
TABLES CASCADE CONSTRAINTS;

End

Introduction to Database Administration

TOPIC 106

Data Dictionary Views for Clusters

Data Dictionary Views for Clusters

Data Dictionary Views

Three Types of Views

Displaying Views for Clusters

Display Cluster's Details

Displaying Cluster on Table's Column

Data Dictionary Views for Clusters

Three Types of Views

There are three types of Views to manage Cluster related information:

DBA_ ALL_ USER_

Data Dictionary Views for Clusters

Displaying Views for Clusters

SQL statement to display Indexes related Views:

```
Select Table_name
From Dict
Where Table_name like '%CLU%';
```

You can use following views as well: DBA_CLU_COLUMNS USER_CLU_COLUMNS
Data Dictionary Views for Clusters

Display Cluster's Details

To find cluster's information

```
SQL> conn sys / as sysdba
Enter password: _
Connected.
SQL> select cluster_name, Tablespace_name, cache
       2  from user_clusters;
```

CLUSTER_NAME	TABLESPACE_NAME	CACHE
C_USER#	SYSTEM	N
C_TS#	SYSTEM	N
C_TOID_VERSION#	SYSTEM	N
C_RG#	SYSTEM	N
C_OBJ#_INTCOL#	SYSTEM	N
C_OBJ#	SYSTEM	N
C_MLOG#	SYSTEM	N
C_FILE#_BLOCK#	SYSTEM	N
C_COBJ#	SYSTEM	N
SMON_SCN_TO_TIME_AUX	SYSAUX	N

10 rows selected.

Data Dictionary Views for Clusters

Displaying Cluster on Table's Column

```
SQL> select c.cluster_name, c.Tablespace_name, c.cluster_name, cc.clu_column_name
       2  from user_clusters c, user_clu_columns cc
       3  where c.cluster_name=cc.cluster_name;
```

CLUSTER_NAME	TABLESPACE_NAME	CLUSTER_NAME	CLU_COLUMN_NAME
C_COBJ#	SYSTEM	C_COBJ#	OBJ#
C_COBJ#	SYSTEM	C_COBJ#	OBJ#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGFILE#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGBLOCK#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	TS#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGFILE#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGBLOCK#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	TS#

TOPIC 107

Defining Hash Clusters

Defining Hash Clusters

Hash Clusters

What is a Hash Cluster?

Finding Rows in Index or Cluster?

Situation where useful Hashing

Creating Hash Clusters

Querying Hash Clusters

Defining Hash Clusters

What is a Hash Cluster?

Storing a table in a hash cluster is an optional way to improve the performance of data retrieval. A hash cluster provides an alternative to a non-clustered table with an index or an index cluster. With an indexed table or index cluster, Oracle Database locates the rows in a table using key values that the database stores in a separate index.

Defining Hash Clusters

What is a Hash Cluster?

To use hashing, you create a hash cluster and load tables into it. The database physically stores the rows of a table in a hash cluster and retrieves them according to the results of a hash function.

Oracle Database uses a hash function to generate a distribution of numeric values, called hash values, that are based on specific cluster key values.

Defining Hash Clusters

What is a Hash Cluster?

The key of a hash cluster, like the key of an index cluster, can be a single column or composite key (multiple column key).

To find or store a row in a hash cluster, the database applies the hash function to the cluster key value of the row.

The resulting hash value corresponds to a data block in the cluster, which the database then reads or writes on behalf of the issued statement.

Defining Hash Clusters

Finding Rows in Index or Cluster?

One or more I/Os to find or store the key value in the index

Another I/O to read or write the row in the table or cluster

Defining Hash Clusters

Situation where useful Hashing

Most queries are equality queries on the cluster key:

```
SELECT ... WHERE cluster_key = ...;
```

In such cases, the cluster key in the equality condition is hashed, and the corresponding hash key is usually found with a single read. In comparison, for an indexed table the key value must first be found in the index, and then the row is read from the table (another read).

Defining Hash Clusters

Situation where useful Hashing

The tables in the hash cluster are primarily static in size so that you can determine the number of rows and amount of space required for the tables in the cluster. If tables in a hash cluster require more space than the initial allocation for the cluster, performance degradation can be substantial because overflow blocks are required.

Defining Hash Clusters

Creating Hash Clusters

```
CREATE CLUSTER trial_cluster ( trialno NUMBER(5,0) )  
TABLESPACE users STORAGE ( INITIAL 250K  
NEXT 50K MINEXTENTS 1  
MAXEXTENTS 3  
PCTINCREASE 0 )  
HASH IS trialno HASHKEYS 150;
```

Defining Hash Clusters

Creating Hash Clusters

The following statement creates the trial table in the trial_cluster hash cluster:

```
CREATE TABLE trial (
trialno NUMBER(5,0) PRIMARY KEY,
...
)
CLUSTER trial_cluster (trialno);
```

Defining Hash Clusters

Querying Hash Clusters

TELEPHONE_NUMBER	CALL_TIMESTAMP	CALL_DURATION	OTHER_INFO
6505551212	0	9	misc info
6505551212	1	17	misc info
6505551212	2	5	misc info
6505551212	3	90	misc info

Execution Plan

Plan hash value: 2118876266

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		1	56	0 (0)
* 1	TABLE ACCESS HASH	CALL_DETAIL	1	56	

Introduction to Database Administration

TOPIC 108

Creating & Altering Hash Clusters
Creating & Altering Hash Clusters

Creating & Altering Hash Clusters

Creating Hash Clusters
Querying Hash Clusters

Estimating Hash Clusters Size
Altering Hash Clusters
Dropping Hash Clusters
Creating & Altering Hash Clusters

Creating Hash Clusters

```
CREATE CLUSTER trial_cluster ( trialno NUMBER(5,0) )  
TABLESPACE users STORAGE ( INITIAL 250K  
NEXT 50K MINEXTENTS 1  
MAXEXTENTS 3  
PCTINCREASE 0 )  
HASH IS trialno HASHKEYS 150;
```

Creating & Altering Hash Clusters

Creating Hash Clusters

The following statement creates the trial table in the trial_cluster hash cluster:

```
CREATE TABLE trial (  
trialno NUMBER(5,0) PRIMARY KEY,  
...  
)  
CLUSTER trial_cluster (trialno);
```

Creating & Altering Hash Clusters

Querying Hash Clusters

TELEPHONE_NUMBER	CALL_TIMESTAMP	CALL_DURATION	OTHER_INFO
------------------	----------------	---------------	------------

6505551212	0	9	misc info
6505551212	1	17	misc info
6505551212	2	5	misc info
6505551212	3	90	misc info

Execution Plan

Plan hash value: 2118876266

Id	Operation	Name	Rows	Bytes	Cost (%CPU)
0	SELECT STATEMENT		1	56	0 (0)
* 1	TABLE ACCESS HASH	CALL_DETAIL	1	56	

Creating & Altering Hash Clusters

Estimating Hash Clusters Size

Example:

STORAGE (INITIAL 100K
NEXT 150K MINEXTENTS 1
PCTINCREASE 0)
SIZE 1500
HASHKEYS 100

In this example, only one hash key can be assigned for each data block. Therefore, the initial space required for the hash cluster is at least **100*2K or 200K**.

Creating & Altering Hash Clusters

Altering Hash Clusters

You can alter a hash cluster with the ALTER CLUSTER statement:

ALTER CLUSTER emp_dept . . . ;

The implications for altering a hash cluster are identical to those for altering an index cluster, described in. However, the SIZE, HASHKEYS, **"Altering Clusters" on page 22- 6** and HASH IS parameters cannot be specified in an ALTER CLUSTER statement. To change these parameters, you must re- create the cluster, then copy the data from the original cluster.

Creating & Altering Hash Clusters

Dropping Hash Clusters

You can drop a hash cluster using the DROP CLUSTER statement:

```
DROP CLUSTER emp_dept;
```

A table in a hash cluster is dropped using the DROP TABLE statement. The implications of dropping hash clusters and tables in hash clusters are the same as those for dropping index clusters.

Introduction to Database Administration

TOPIC 109

Data Dictionary Views for Hash Clusters

Data Dictionary Views for Hash Clusters

Data Dictionary Views

Three Types of Views

Displaying Views for Hash Clusters

Display Hash or just Cluster's Details

Displaying Cluster on Table's Column

Data Dictionary Views for Hash Clusters

Three Types of Views

There are three types of Views to manage Hash Cluster related information:

DBA_ ALL_ USER_

Data Dictionary Views for Hash Clusters

Displaying Views for Hash Clusters

SQL statement to display Indexes related Views:

```
Select Table_name From Dict
```

```
Where Table_name like '%SYN%';
```

You can use following views as well: DBA_UPDATABLE_COLUMNS
ALL_UPDATABLE_COLUMNS USER_UPDATABLE_COLUMNS
Data Dictionary Views for Hash Clusters

Display Hash or just Cluster's Details

To find cluster's information

```
SQL> conn sys / as sysdba
Enter password: _
Connected.
SQL> select cluster_name, Tablespace_name, cache
2  from user_clusters;
```

CLUSTER_NAME	TABLESPACE_NAME	CACHE
C_USER#	SYSTEM	N
C_TS#	SYSTEM	N
C_TOID_VERSION#	SYSTEM	N
C_RG#	SYSTEM	N
C_OBJ#_INTCOL#	SYSTEM	N
C_OBJ#	SYSTEM	N
C_MLOG#	SYSTEM	N
C_FILE#_BLOCK#	SYSTEM	N
C_COBJ#	SYSTEM	N
SMON_SCN_TO_TIME_AUX	SYSAUX	N

10 rows selected.

Data Dictionary Views for Hash Clusters

Displaying Cluster on Table's Column

```
SQL> select c.cluster_name, c.Tablespace_name, c.cluster_name, cc.clu_column_name
2  from user_clusters c, user_clu_columns cc
3  where c.cluster_name=cc.cluster_name;
```

CLUSTER_NAME	TABLESPACE_NAME	CLUSTER_NAME	CLU_COLUMN_NAME
C_COBJ#	SYSTEM	C_COBJ#	OBJ#
C_COBJ#	SYSTEM	C_COBJ#	OBJ#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGFILE#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGBLOCK#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	TS#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGFILE#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	SEGBLOCK#
C_FILE#_BLOCK#	SYSTEM	C_FILE#_BLOCK#	TS#

TOPIC 110

Defining & Managing Views

Defining & Managing Views

Defining & Managing Views

How a Schema is Created?

How a Schema is Implemented?

Simple & Complex Views

SQL Practices on Views

Defining & Managing Views

How a Schema is Created?

We have studied that DBA has to create default Tablespace and assign temporary Tablespace with storage in terms of data files.

Create first user (owner of schema) with grants such as rights and privileges.

We will cover schema objects Tables and Views.

Ref. DBA-73

Defining & Managing Views

How a Schema is Implemented?

A view is a logical representation of a table or combination of tables.

In essence, a view is a stored query.

A view derives its data from the tables on which it is based.

These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.

All operations performed on a view actually affect the base table of the view.

Defining & Managing Views

How a Schema is Implemented?

You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.

Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.

Views can be Simple or Complex

Views can be used for security purposes

View does not contain its own data

Defining & Managing Views

Simple & Complex Views

```
CREATE VIEW sales_staff AS SELECT empno, ename, deptno FROM emp
WHERE deptno = 10;
```

```
CREATE VIEW sales_staff AS
SELECT job, count(*) FROM emp
WHERE deptno = 10 Group by job;
```

Defining & Managing Views

SQL Practices on Views

We can apply Insert, delete and update operations on simple views. for base tables

End

Introduction to Database Administration

TOPIC 111

Updating a Join View

Updating a Join View

Updating a Join View

Introduction

Rules for Updating Join Views

Example-1 of a Join View

Example-2 of a Join View

Data Dictionary View for Join Views

Updating a Join View

Introduction

An updatable join view (also referred to as a modifiable join view) is a view that contains multiple tables in the top-level FROM clause of the SELECT statement, and is not restricted by the WITH READ ONLY clause.

Updating a Join View

Rules for Updating Join Views

The rules for updatable join views are shown in the following table. Views

Rule	Description
General Rule	Any INSERT, UPDATE, or DELETE operation on a join view can modify only one underlying base table at a time.
UPDATE Rule	All updatable columns of a join view must map to columns of a key-preserved table . See "Key-Preserved Tables" on page 24-7 for a discussion of key-preserved tables. If the view is defined with the WITH CHECK OPTION clause, then all join columns and all columns of repeated tables are not updatable.
DELETE Rule	Rows from a join view can be deleted as long as there is exactly one key-preserved table in the join. The key preserved table can be repeated in the FROM clause. If the view is defined with the WITH CHECK OPTION clause and the key preserved table is repeated, then the rows cannot be deleted from the view.
INSERT Rule	An INSERT statement must not explicitly or implicitly refer to the columns of a non-key-preserved table . If the join view is defined with the WITH CHECK OPTION clause, INSERT statements are not permitted.

that meet these criteria are said to be inherently updatable.

DBA E25494 Oracle11g.pdf pp. 24-6

Updating a Join View

Example-1 of a Join View

```
CREATE VIEW emp_dept AS
SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc FROM emp,
dept
WHERE emp.deptno = dept.deptno
AND
```

dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');

EMPNO	ENAME	DEPTNO	DNAME	LOC
7782	CLARK	10	ACCOUNTING	NEW YORK
7839	KING	10	ACCOUNTING	NEW YORK
7934	MILLER	10	ACCOUNTING	NEW YORK
7369	SMITH	20	RESEARCH	DALLAS
7876	ADAMS	20	RESEARCH	DALLAS

Updating a Join View

Example-1 of a Join View

```
CREATE VIEW emp_dept AS
SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc FROM emp,
dept
WHERE emp.deptno = dept.deptno
AND
```

```
UPDATE emp_dept
SET sal = sal * 1.10
WHERE deptno = 10;
```

dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');

```
UPDATE emp_dept
SET deptno = 10
WHERE ename = 'SMITH';
ORA-01779 cannot modify a column
```

```
UPDATE emp_dept
SET loc = 'BOSTON'
WHERE ename = 'SMITH';
ORA-01779 cannot modify a column which maps to a non key-preserved table)
```

Updating a Join View

Example-2 of a Join View

```
CREATE VIEW emp_mgr AS
```

```
SELECT e1.ename, e2.ename mname FROM emp e1, emp e2
WHERE e1.mgr = e2.empno
WITH CHECK OPTION;
```

Following statement works successfully:

```
INSERT INTO emp_dept (ename, empno, deptno)
VALUES ('KURODA', 9010, 40);
```

Updating a Join View

Example-2 of a Join View

The following INSERT statements would fail with an error (ORA-01776 cannot modify more than one base table through a join view):

```
INSERT INTO emp_dept (ename, empno, deptno)
VALUES ('KURODA', 9010, 77); --FK value not found in department table
```

```
INSERT INTO emp_dept (empno, ename, loc)
VALUES (9010, 'KURODA', 'BOSTON'); --cannot insert data in two tables
```

Updating a Join View

Data Dictionary View for Join Views

If you are in doubt whether a view is modifiable, then you can select from the USER_UPDATABLE_COLUMNS view to see if it is. For example:

```
SELECT owner, table_name, column_name, updatable FROM USER_UPDATABLES
WHERE TABLE_NAME = 'EMP_DEPT_VIEW';
```

This returns output similar to the following:

OWNER	TABLE_NAME	COLUMN_NAME	UPD
SCOTT	EMP_DEPT_V	EMPNO	NO
SCOTT	EMP_DEPT_V	ENAME	NO
SCOTT	EMP_DEPT_V	DEPTNO	NO
SCOTT	EMP_DEPT_V	DNAME	NO
SCOTT	EMP_DEPT_V	LOC	NO

5 rows selected.

Introduction to Database Administration

TOPIC 112

Create Sequences

Create Sequences

Create Sequences

An Overview

Sequence Characteristics

Sequence Example-1

Concurrent Access to Sequence

Sequence Syntax

Sequence Example-2

Data Dictionary Views for Sequence

Create Sequences

An Overview

A sequence (Autonumber) is a schema object from which multiple users can generate unique integers. A sequence generator provides a highly scalable and well-performing method to generate surrogate keys for a number data type.

Create Sequences

Sequence Characteristics

A sequence definition indicates general information, such as the following:

The name of the sequence

Whether the sequence ascends or descends The interval between numbers

Whether the database should cache sets of generated sequence numbers in memory

Whether the sequence should cycle

when a limit is reached

Create Sequences

Sequence Example-1

```
CREATE SEQUENCE customers_seq START WITH 1000  
INCREMENT BY 1 NOCACHE NOCYCLE;
```

Create Sequences

Concurrent Access to Sequence

The same sequence generator can generate numbers for multiple tables. In this way, the database can generate primary keys automatically and coordinate keys across multiple rows or tables.

For example, a sequence can generate primary keys for an orders table and a customers table.

Create Sequences

Sequence Syntax

```
CREATE SEQUENCE user.sequence_name [INCREMENT BY interval]  
[START WITH first_number]  
[MAXVALUE max_value | NOMAXVALUE] [MINVALUE min_value | NOMINVALUE]  
[CYCLE | NOCYCLE]  
[CACHE cache_size | NOCACHE] [ORDER | NOORDER];
```

Create Sequences

Sequence Example-2

```
CREATE SEQUENCE id_seq INCREMENT BY 10  
START WITH 10  
MINVALUE 10
```

MAXVALUE 100 CYCLE

CACHE 2;

To get the next value of the sequence, you use the NEXTVAL pseudo-column:

SELECT id_seq.NEXTVAL FROM dual;

Create Sequences

Sequence Example-2

INSERT INTO tasks(id, title) VALUES(seq.NEXTVAL, 'Create Sequence in Oracle');

SELECT id_seq.NEXTVAL FROM dual;

Create Sequences

Data Dictionary Views for Sequence

USER_SEQUENCES ALL_SEQUENCES SEQ

End

Introduction to Database Administration

TOPIC 113

Alter & Drop Sequences

Alter & Drop Sequences

Alter & Drop Sequences

An Overview

NEXTVAL & CURRVAL

Caching Sequence Number

Dropping Sequences

Alter & Drop Sequences

An Overview

To alter a sequence, your schema must contain the sequence, you must have the ALTER object privilege on the sequence, or you must have the ALTER ANY SEQUENCE system privilege.

You can alter a sequence to change any of the parameters that define how it generates sequence numbers except the sequence starting number. To change the starting point of a sequence, drop the sequence and then re-create it.

Alter & Drop Sequences

An Overview

Alter a sequence using the ALTER SEQUENCE statement. For example, the following statement alters the emp_seq:

```
ALTER SEQUENCE emp_seq  
INCREMENT BY 10  
MAXVALUE 10000 CYCLE  
CACHE 20;
```

Alter & Drop Sequences

NEXTVAL & CURRVAL

CURRVAL and NEXTVAL cannot be used in the following places:

A subquery

A view query or materialized view query

A SELECT statement with the DISTINCT operator

A SELECT statement with a GROUP BY or ORDER BY clause

Alter & Drop Sequences

NEXTVAL & CURRVAL

A SELECT statement that is combined with another SELECT statement with the UNION, INTERSECT, or MINUS set operator

The WHERE clause of a SELECT statement

The condition of a CHECK constraint

Alter & Drop Sequences

Caching Sequence Number

Follow these guidelines for fast access to all sequence numbers:

Be sure the sequence cache can hold all the sequences used concurrently by your applications.

Increase the number of values for each sequence held in the sequence cache.

Alter & Drop Sequences

Dropping Sequences

You can drop any sequence in your schema. To drop a sequence in another schema, you must have the DROP ANY SEQUENCE system privilege. If a sequence is no longer required, you can drop the sequence using the DROP SEQUENCE statement.

Alter & Drop Sequences

Dropping Sequences

For example, the following statement

drops the order_seq sequence: **DROP SEQUENCE order_seq;**

When a sequence is dropped, its definition is removed from the data dictionary. Any synonyms for the sequence remain, but return an error when referenced.

Introduction to Database Administration

TOPIC 114

Defining Synonyms

Defining Synonyms

Defining Sequences

An Overview

Public & Private Synonyms

Securable Synonyms

DDL Create Synonyms

Select script for all Tables

Defining Synonyms

An Overview

A synonym is an alias for a schema object.

Synonyms can provide a level of security by masking the name and owner of an object and by providing location transparency for remote objects of a distributed database.

Also, they are convenient to use and reduce the complexity of SQL statements for database users.

Defining Synonyms

An Overview

Synonyms allow underlying objects to be renamed or moved, where only the synonym must be redefined and applications based on the synonym continue to function without modification.

User u1 can access table of user scott as `Select * from scott.emp;`

But we want u1 uses statement as `Select * from emp;`

User scott should create emp synonym with the same name

Defining Synonyms

Public & Private Synonyms

You can create both public and private synonyms.

A public synonym is owned by the special user group named PUBLIC and is accessible to every user in a database.

A private synonym is contained in the schema of a specific user and available only to the user and to grantees for the underlying object.

Defining Synonyms

Securable Synonyms

Synonyms themselves are not securable. When you grant object privileges on a synonym, you are really granting privileges on the underlying object, and the synonym is acting only as an alias for the object in the GRANT statement.

Defining Synonyms

Securable Synonyms

To create a private synonym in your own schema, you must have the **CREATE SYNONYM** privilege.

To create a private synonym in another user's schema, you must have the **CREATE ANY SYNONYM** privilege.

To create a public synonym, you must have the **CREATE PUBLIC SYNONYM** system privilege.

Defining Synonyms

DDL Create Synonyms

Following sample DDL command is required to create public synonym:

```
CREATE PUBLIC SYNONYM public_emp  
FOR ejaz.emp
```

Creating synonym inside same user's schema

create synonym **syn_emp** for scott.emp;

Defining Synonyms

Select Script for all Tables

```
select 'create synonym '||tname||' for u1.'||tname||';'
from tab;
```

```
'CREATESYNONYM' || TNAME || 'FORU1.' || TNAME || ';'
-----
create synonym BONUS for u1.BONUS;
create synonym DEPT for u1.DEPT;
create synonym DUMMY for u1.DUMMY;
create synonym EMP for u1.EMP;
create synonym PROJASSIGN for u1.PROJASSIGN;
create synonym PROJECT for u1.PROJECT;
create synonym SALGRADE for u1.SALGRADE;

7 rows selected.

SQL> _
```

Introduction to Database Administration

TOPIC 115

Create & Drop Synonyms

Create & Drop Synonyms

Create & Drop Sequences

Specifying Schema

FOR Clause

Create Synonym for Objects

Create Synonym Examples

Accessing Synonym

Drop Synonym

Synonym Syntax

Create & Drop Synonyms

Specifying Schema

Specify the schema to contain the synonym. If you omit schema, then Oracle Database creates the synonym in your own schema. You cannot specify a schema for the synonym if you have specified PUBLIC.

Create & Drop Synonyms

FOR Clause

Specify the object for which the synonym is created. The schema object for which you are creating the synonym can be of the following types:

Table or object table

View or object view

Sequence

Stored procedure, function, or package

Materialized view

Java class schema object

User-defined object type & Synonym

Create & Drop Synonyms

Create Synonym for Objects

Specify the object for which the synonym is created. The schema object for which you are creating the synonym can be of the following types:

Table or object table

View or object view

Sequence

Stored procedure, function, or package

Materialized view

Java class schema object

User-defined object type & Synonym

Create & Drop Synonyms

Create Synonym Examples

To define the synonym offices for the table locations in the schema hr, issue the following statement:

```
CREATE SYNONYM offices FOR hr.locations;
```

Create & Drop Synonyms

Create Synonym Examples

To create a PUBLIC synonym for the emp table in the schema hr on the remote database, you could issue the following statement:

```
CREATE PUBLIC SYNONYM emp_table  
FOR hr.emp@remote.us.oracle.com;
```

Create & Drop Synonyms

Accessing Synonym

If the user hr's schema does not contain an object named customers, and if hr has select permission on oe.customers, then hr can access the customers table in oe's schema by using the public synonym customers:

```
SELECT COUNT(*) FROM customers;
```

Create & Drop Synonyms

Drop Synonym

Following commands can be used to drop synonyms

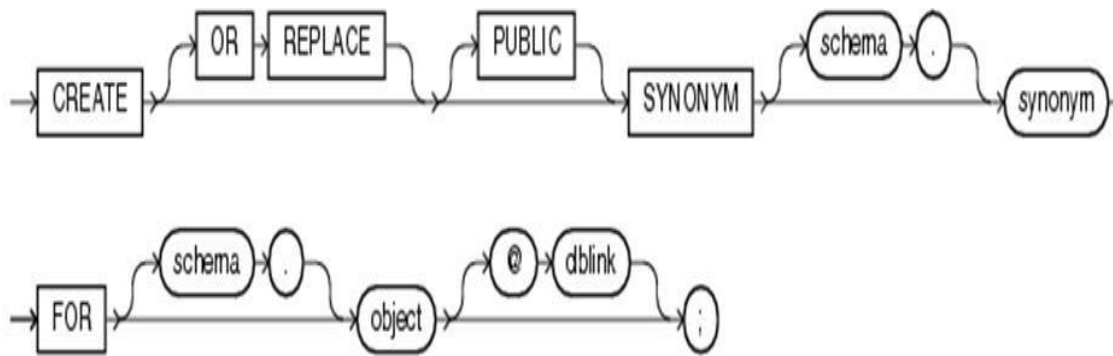
```
drop synonym dept;  
drop synonym emp;
```

```
drop Public synonym customer;
```

Create & Drop Synonyms

Synonym Syntax

Following is a complete syntax of creating Synonym



Ref: docs.oracle.com

Introduction to Database Administration

TOPIC 116

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Synonyms, Views & Sequences

Create & Drop Sequences

Three Types of Views

Displaying Views

Data Dictionary Views for Synonym

Data Dictionary Views for Sequence

Data Dictionary Views for VIEWS

Data Dictionary Views for Synonyms, Views & Sequences

Three Types of Views

There are three types of Views to manage Hash Cluster related information:

DBA_ALL_USER_

Data Dictionary Views for Synonyms, Views & Sequences

Displaying Views

SQL statement to display Indexes related Views:

Select Table_name

From Dict

Where Table_name like '%SYN%';

Or use ('%VIEW%' or '%SEQ%')

You can use following views as well: DBA_UPDATABLE_COLUMNS

ALL_UPDATABLE_COLUMNS USER_UPDATABLE_COLUMNS

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Synonym

Following views can be used to see synonyms

Select object_name, object_type

from user_objects

where object_type='SYNONYM'; Select * from TAB;

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Synonym

Following view can be used to see synonyms

select synonym_name, table_owner,

table_name

from user_synonyms;

```
SQL> select synonym_name, table_owner, table_name  
2 from user_synonyms;
```

SYNONYM_NAME	TABLE_OWNER	TABLE_NAME
SYN_EMP	SCOTT	EMP

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Sequence

Following sequence is created:

CREATE SEQUENCE customers_seq START WITH 1000

INCREMENT BY 1 NOCACHE NOCYCLE;

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Sequence

Following view can be used to see synonyms

```
select object_name, object_type  
from user_objects  
where object_type='SEQUENCE';
```

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for Sequence

Following view can be used to see synonyms

```
select sequence_name, cycle_flag,  
cache_size, last_number from user_sequences;
```

SEQUENCE_NAME	C	CACHE_SIZE	LAST_NUMBER
-----	-----	-----	-----
CUSTOMERS_SEQ	N	0	1000

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for VIEWS

```
grant create view to scott; create view v_emp  
AS
```

```
select job, count(*) Count_EMP from emp  
group by job;
```

Data Dictionary Views for Synonyms, Views & Sequences

Data Dictionary Views for VIEWS

```
SQL> select view_name, text
      2  from user_views;

VIEW_NAME
-----
TEXT
-----
V_EMP
select job, count(*) Count_EMP
from emp
group by job

SQL>
```

Following view can be used to see synonyms

End

Introduction to Database Administration

TOPIC 117

Introduction to Database Server Programming using PL/SQL

Introduction to Database Server Programming using PL/SQL

Introduction to PL/SQL

What is PL/SQL

PL/SQL & Other Languages

Introduction to Database Server Programming using PL/SQL

What is PL/SQL

Embedded structured programming

Performed on Server Machine

Centrally accessible from single database

Executed on server without any network overhead

Introduction to Database Server Programming using PL/SQL

PL/SQL & Other Languages

Embedded SQL (PL/SQL, JAVA/ VB & DB)

Database Server Level Programming(PL/SQL, Transact-SQL, IBM DB2-Cobol, ProC, ProCobol)

(Front end programming can be changed without affecting Server programming)

Database Client Programming (Developer 9i, JDeveloper 9i, Java (J2EE), VB, .Net)



Introduction to Database Administration

TOPIC 118

PL/SQL Constructs

PL/SQL Constructs

PL/SQL Constructs

SQL & PL/SQL

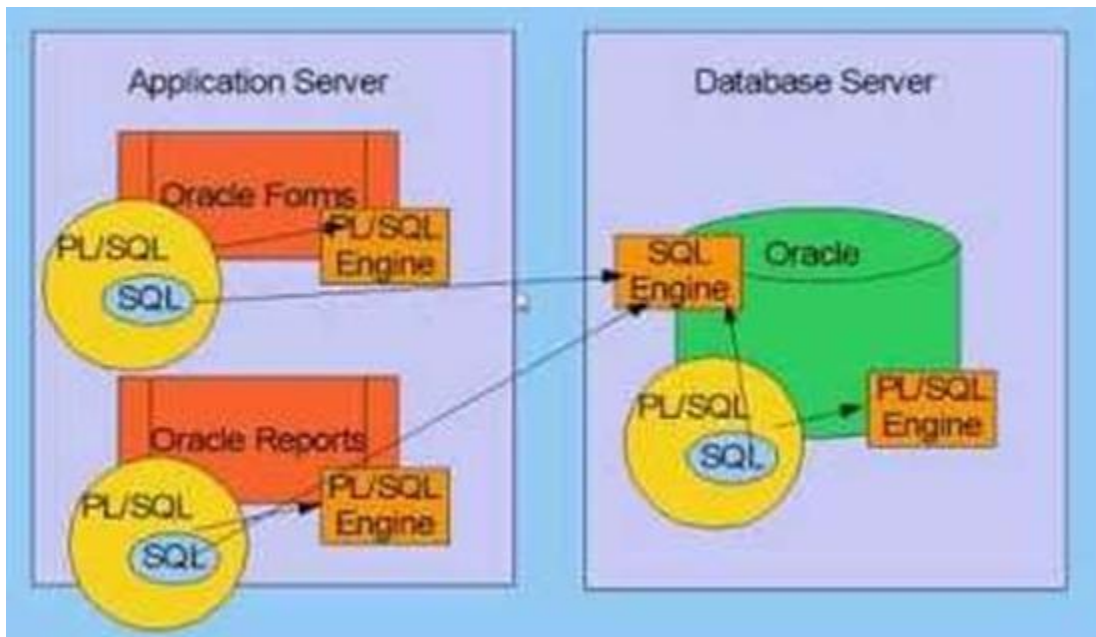
PL/SQL Data Types

PL/SQL Contents

PL/SQL Constructs

SQL & PL/SQL

SQL & PL/SQL on Application & Server Sides

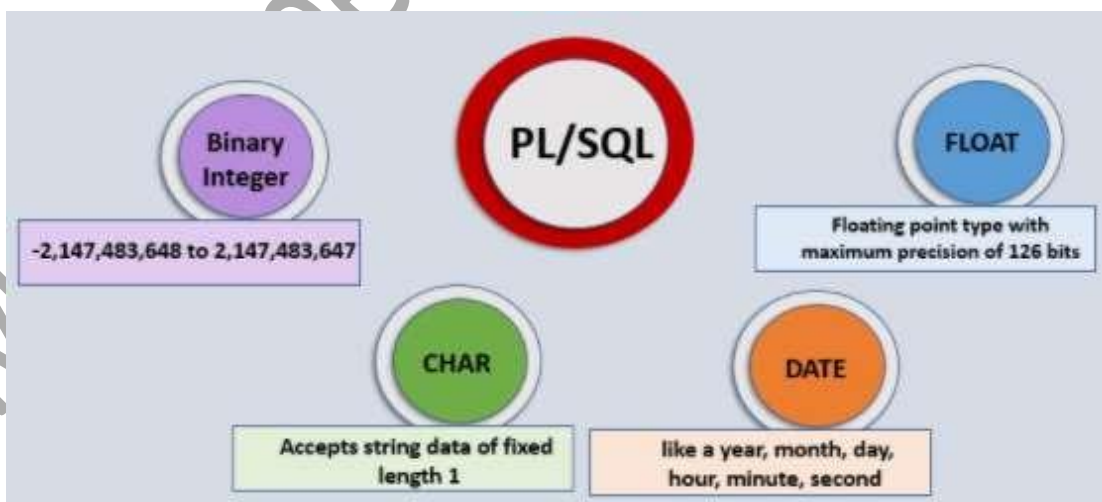


Ref: SQL ZeroToPro.com

PL/SQL Constructs

PL/SQL Data Types

Following main data types are used in PL/SQL



PL/SQL Constructs

PL/SQL Contents

Benefits
Basic Constructs
Anonymous blocks
Procedures
Functions
Packages
Triggers
Cursors
Dynamic SQL

End
Introduction to Database Administration

TOPIC 119

PL/SQL Benefits & Basic Constructs
PL/SQL Benefits & Basic Constructs

PL/SQL Benefits & Basic Constructs

PL/SQL Benefits
PL/SQL Basic Constructs
PL/SQL Benefits & Basic Constructs

PL/SQL Benefits

More powerful than pure SQL because it combines the power of SQL and
Iteration (loops)
Selection (Ifs)
Cursors
Block Structures
Stored Procedures
etc.
PL/SQL Benefits & Basic Constructs

PL/SQL Basic Constructs

Basic Structure Running a program Variables

SELECT INTO

Comments IFs

LOOPS

Output

Introduction to Database Administration

TOPIC 120

Basic Program Block Structure and Display output

Basic Program Block Structure and Display output

Basic Program Structure

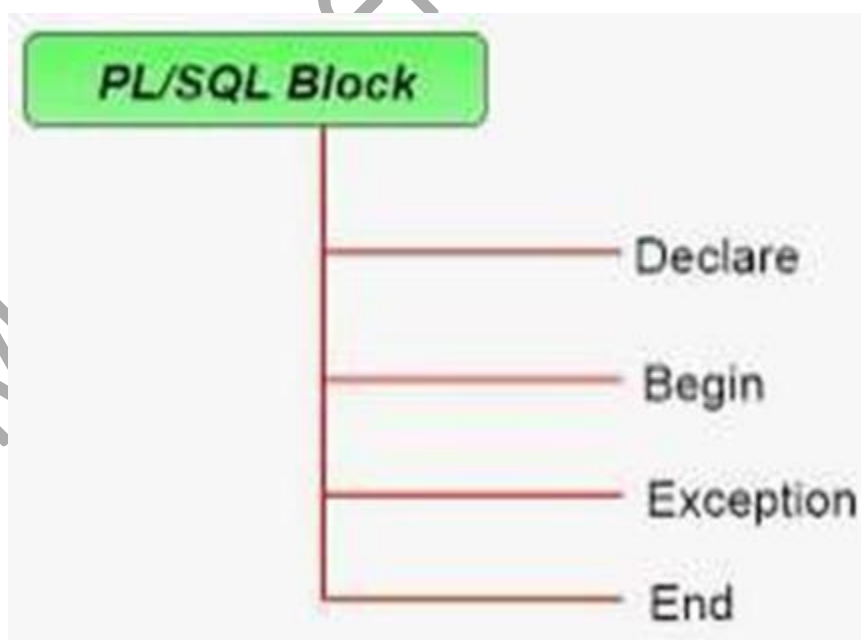
Nested Basic Structures

First PL/SQL Program

First PL/SQL Program with Declaration

Basic Program Block Structure and Display output

Basic Structure



Basic Program Block Structure and Display output

Basic Structure

DECLARE



BEGIN



EXCEPTION

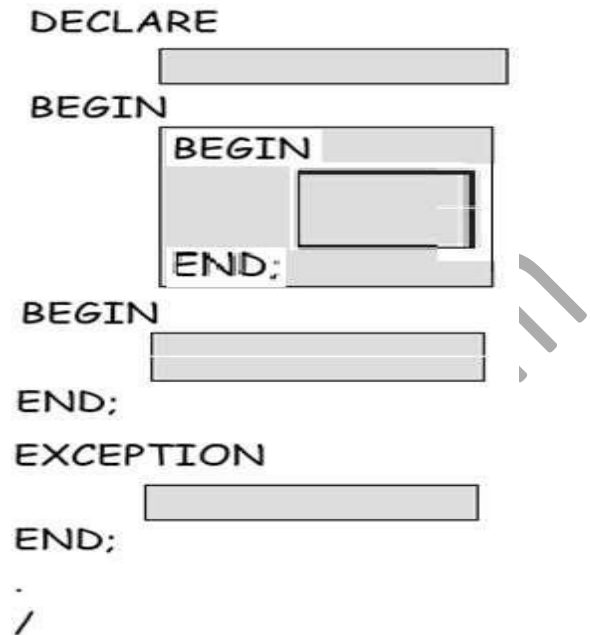
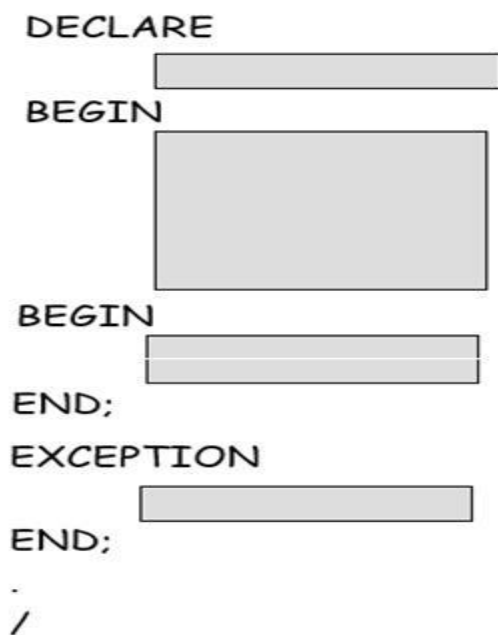


END;

·
/

Basic Program Block Structure and Display output

Nested Basic Structures



Basic Program Block Structure and Display output

First PL/SQL Program

```

SET SERVEROUTPUT ON; BEGIN
DBMS_OUTPUT.PUT_LINE('This is my first program'); END;
/

```

Before executing code that contains DBMS_OUTPUT.PUT_LINE, must run (in a current session) at SQL prompt: **set serveroutput on.**

System Package DBMS_OUTPUT exists in Oracle Dictionary. It accepts and return SINGLE argument.

Basic Program Block Structure and Display output

First PL/SQL Program with Declaration

```

SET SERVEROUTPUT ON; --if not runing already DECLARE
msg varchar2(40) := 'I love Oracle Programing' ; BEGIN

```

```
SQL> DECLARE
  2      msg varchar2(40) := 'I love Oracle Programing' ;
  3  BEGIN
  4      dbms_output.put_line(msg);
  5  END;
  6  /
I love Oracle Programing

PL/SQL procedure successfully completed.

SQL>
```

dbms_output.put_line(msg); END;

/

Introduction to Database Administration

TOPIC 121

Basic Programs Block Structure with more examples
Basic Program Block Structures with More Examples

Basic Program Structure

Program Execution with an error
Program Fixing of an error
Program Compilation
Basic Program Block Structures with more Examples

Program Execution with an Error

Following program declarations, assigned values and display result:

```

SQL> Declare
  2  -- declare variable x, y and z of datatype number
  3  x number(5);
  4  y number(5);
  5  z number(7);
  6  Begin
  7  -- Here we Assigning 10 into x
  8  x:=10;
  9  -- Assigning 20 into x
 10  y:=20;
 11  -- Assigning sum of x and y into z
 12  z:=x+y;
 13  --Print the Result
 14  dbms_output.put_line('Sum is '||z);
 15  End;
 16  /
Sum is 30

```

Basic Program Block Structures with more Examples

Program Fixing of an Error

Following program terminates:

```

SQL> Declare
  2  -- declare variable x, y and z of datatype number
  3  x varchar2(5);
  4  y number(5);
  5  z number(5);
  6  Begin
  7  -- Here we Assigning 10 into x
  8  x:=10;
  9  -- Assigning 20 into x
 10  y:=2V0;
 11  -- Assigning sum of x and y into z
 12  z:=x+y;
 13  --Print the Result
 14  dbms_output.put_line('Sum is '||z);
 15  End;
 16  /
y:=2V0;
*
ERROR at line 10:
ORA-06550: line 10, column 5:
PLS-00103: Encountered the symbol "V0" when expecting one of the following:
* & = - + ; < / > at in is mod remainder not rem
<an exponent (**)> <> or != or ~= >= <= <> and or like like2
like4 likec between || multiset member submultiset
The symbol "*" was substituted for "V0" to continue.

```

Basic Program Block Structures with more Examples

Program Compilation

Following program (Anonymous Block) executes with error due to compile &

```
SQL> Declare
 2  -- declare variable x, y and z of datatype number
 3  x varchar2(5);
 4  y number(5);
 5  z number(5);
 6  Begin
 7  -- Here we Assigning 10 into x
 8  x:=10;
 9  -- Assigning 20 into x
10  y:=2V0;
11  -- Assigning sum of x and y into z
12  z:=x+y;
13  --Print the Result
14  dbms_output.put_line('Sum is '||z);
15  Exception
16      WHEN OTHERS THEN
17  dbms_output.put_line('Error in program');
18  End;
19  /
y:=2V0;
*
ERROR at line 10:
ORA-06550: line 10, column 5:
PLS-00103: Encountered the symbol "V0" when expecting one of th
* & = + * < / > at in is mod remainder not rem
```

execute together

Introduction to Database Administration

TOPIC 122

Basic Programs Block Structure with more examples

Basic Program Block Structures with More Examples

Basic Program Structure

Program execution with an error

Program fixing of an error

Program with SELECT INTO

Basic Program Block Structures with more Examples

Program execution with an error

Following program executes with an error during compilation/ execution:

```
SQL> DECLARE
  2      -- taking input for variable a
  3      a number := &a;
  4      -- taking input for variable b
  5      b varchar2(30) := &b;
  6  BEGIN
  7      c := a + b ;
  8      dbms_output.put_line('Sum of '||a||' and '||b||' is = '||c);
  9  END;
10  /
Enter value for a: 4
old 3:      a number := &a;
new 3:      a number := 4;
Enter value for b: 9
old 5:      b varchar2(30) := &b;
new 5:      b varchar2(30) := 9;
      c := a + b ;
      *
ERROR at line 7:
ORA-06550: line 7, column 7:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 7, column 7:
PL/SQL: Statement ignored
```

Basic Program Block Structures with more Examples

Program fixing of an error

Following program executes successfully:

```

SQL> DECLARE
  2      -- taking input for variable a
  3      a number := &a;
  4      -- taking input for variable b
  5      b varchar2(30) := &b;
  6      c integer;
  7  BEGIN
  8      c := a + b ;
  9      dbms_output.put_line('Sum of '||a||' and '||b||' is = '||c);
 10  END;
 11  /
Enter value for a: 4
old 3:      a number := &a;
new 3:      a number := 4;
Enter value for b: 9
old 5:      b varchar2(30) := &b;
new 5:      b varchar2(30) := 9;
Sum of 4 and 9 is = 13

PL/SQL procedure successfully completed.

```

Basic Program Block Structures with more Examples

Program with SELECT INTO

Following program executes successfully with the retrieval of single record

```

SQL> DECLARE
  2      v_id      INTEGER;
  3      v_empno   NUMBER;
  4  BEGIN
  5      v_id := 7369;
  6      SELECT EMPNO
  7      INTO V_EMPNO
  8      FROM EMP
  9      WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  EXCEPTION
 12      WHEN NO_DATA_FOUND THEN
 13          DBMS_OUTPUT.PUT_LINE('No record exists');
 14  END;
 15  /
Value is 7369

PL/SQL procedure successfully completed.

```

Basic Program Block Structures with more Examples

Program with SELECT INTO

Following program is terminated with the retrieval of single record

```

SQL> DECLARE
  2      v_id      INTEGER;
  3      v_empno   NUMBER;
  4  BEGIN
  5      v_id := 7777; --employee does not exists
  6      SELECT EMPNO
  7          INTO V_EMPNO
  8          FROM EMP
  9          WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  END;
 12  /
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 6

```

Basic Program Block Structures with more Examples

Program with SELECT INTO

Following program is successfully executed with the retrieval of single record and

```

SQL> DECLARE
  2      v_id      INTEGER;
  3      v_empno   NUMBER;
  4  BEGIN
  5      v_id := 7777; --employee does not exists
  6      SELECT EMPNO
  7          INTO V_EMPNO
  8          FROM EMP
  9          WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  EXCEPTION
 12      WHEN NO_DATA_FOUND THEN
 13          DBMS_OUTPUT.PUT_LINE('No record exists');
 14  END;
 15  /
No record exists

PL/SQL procedure successfully completed.

```

exception raised.

Introduction to Database Administration

TOPIC 123

Compile & Run Program

Compile & Run Program

Compile & Run Program

Program Compilation & Execution

Program execution with an error

Run Program using SQL Developer

Compile Program using SQL Developer

Compile & Run Program

Program Compilation & Execution

DECLARE

BEGIN

EXCEPTION

END;



Following program compile and executes on SQL prompt:

Compile & Run Program

Program Compilation & Execution

DECLARE

BEGIN

EXCEPTION END;

Following program run without compiling and execution on SQL prompt:
Compile & Run Program

Program execution with an error

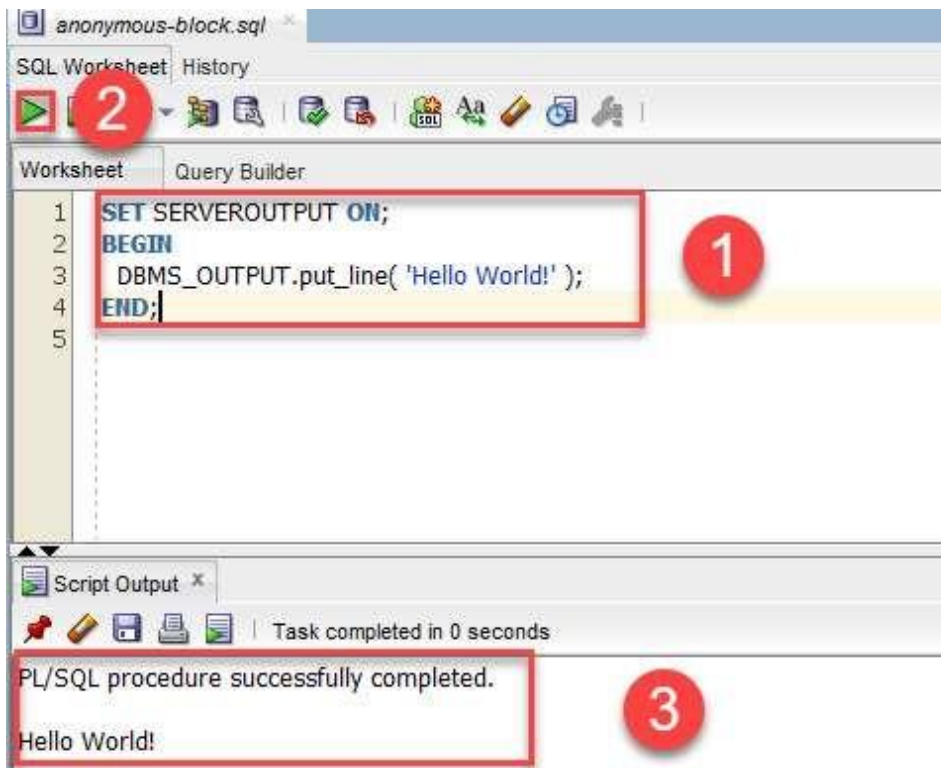
Following program executes with an error during compilation/ execution:

```
SQL> DECLARE
2      -- taking input for variable a
3      a number := &a;
4      -- taking input for variable b
5      b varchar2(30) := &b;
6  BEGIN
7      c := a + b ;
8      dbms_output.put_line('Sum of '||a||' and '||b||' is = '||c);
9  END;
10 /
Enter value for a: 4
old 3:      a number := &a;
new 3:      a number := 4;
Enter value for b: 9
old 5:      b varchar2(30) := &b;
new 5:      b varchar2(30) := 9;
      c := a + b ;
      *
ERROR at line 7:
ORA-06550: line 7, column 7:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 7, column 7:
PL/SQL: Statement ignored
```

Compile & Run Program

Run Program using SQL Developer

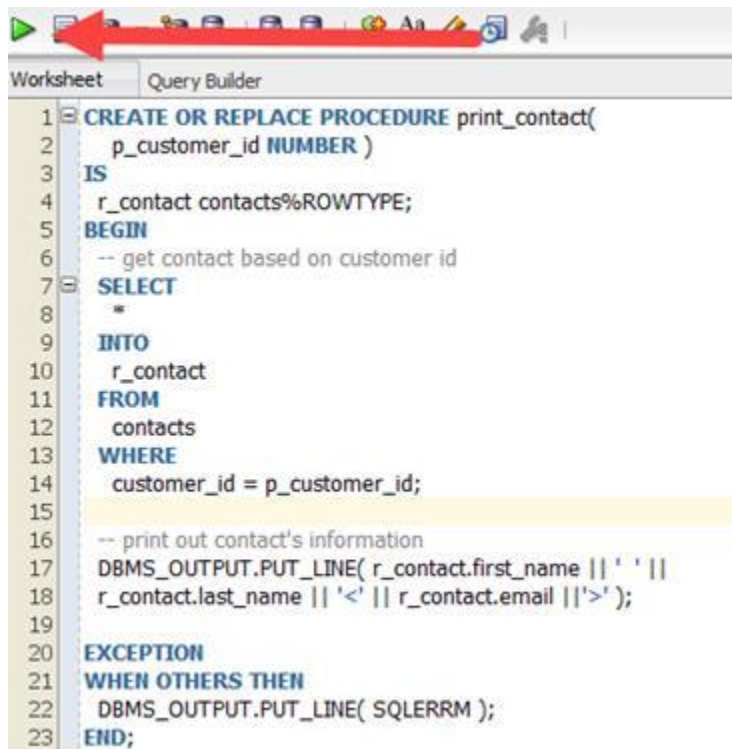
To compile the procedure, you click the Run Statement button:



Compile & Run Program

Run Program using SQL Developer

To compile the procedure, you click the Run Statement button:

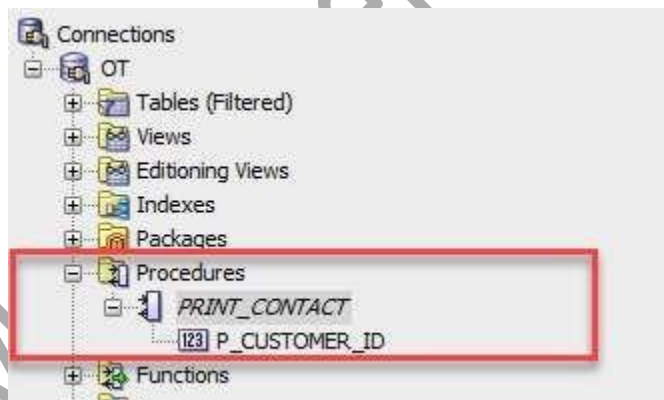


```
1 CREATE OR REPLACE PROCEDURE print_contact(  
2   p_customer_id NUMBER )  
3 IS  
4   r_contact contacts%ROWTYPE;  
5 BEGIN  
6   -- get contact based on customer id  
7   SELECT  
8     *  
9   INTO  
10    r_contact  
11  FROM  
12    contacts  
13  WHERE  
14    customer_id = p_customer_id;  
15  
16   -- print out contact's information  
17   DBMS_OUTPUT.PUT_LINE( r_contact.first_name || ' ' ||  
18     r_contact.last_name || '<' || r_contact.email || '>' );  
19  
20 EXCEPTION  
21 WHEN OTHERS THEN  
22   DBMS_OUTPUT.PUT_LINE( SQLERRM );  
23 END;
```

Compile & Run Program

Run Program using SQL Developer

If the procedure is compiled successfully, you will find the new procedure under the Procedures node as shown below:



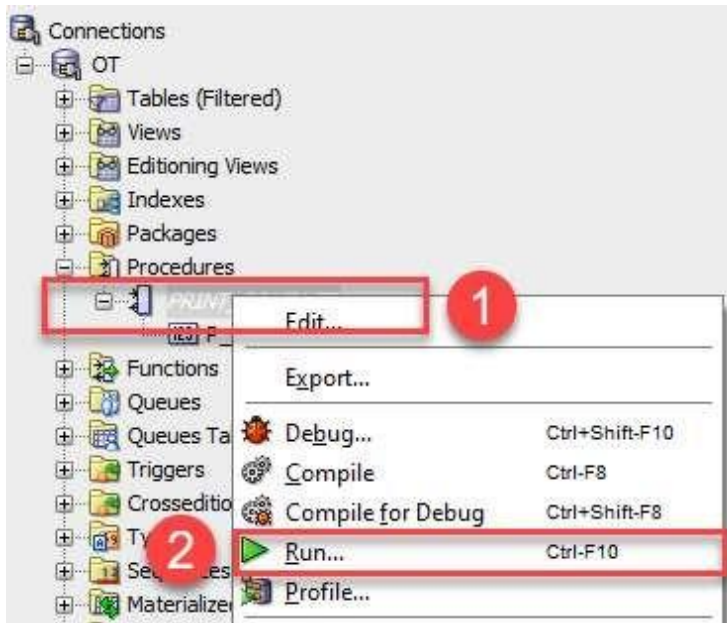
EXEC procedure_name(arguments);

This link or path is not available for blocks codes without name

Compile & Run Program

Run Program using SQL Developer

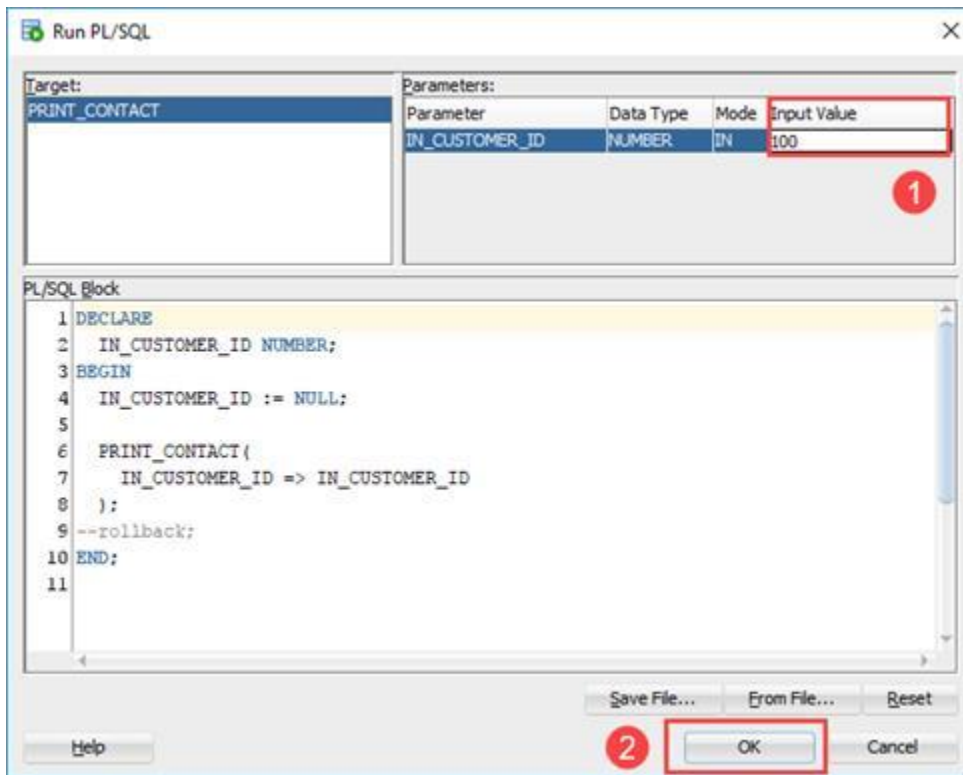
Right-click the procedure name and choose Run ... menu item:



Compile & Run Program

Run Program using SQL Developer

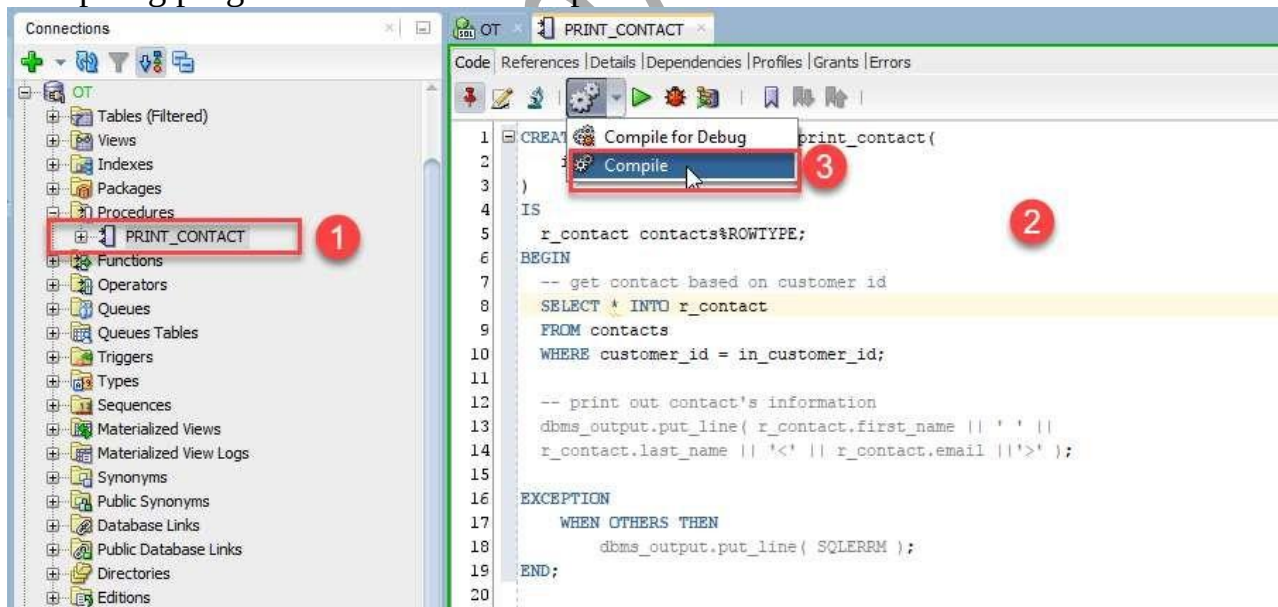
Enter a value for the in_customer_id parameter and click OK button:



Compile & Run Program

Compile Program using SQL Developer

Compiling program with selection steps as shown:



<https://www.oracletutorial.com/plsql-tutorial/plsql-procedure/>

Introduction to Database Administration

Made by Mahjabeen

mahjabeen97869@gmail.com

contact # 0321 2711298

TOPIC 124

Data Types & Select INTO Clause

Data Types & Select INTO Clause

Data Types & Select INTO Clause

Introduction

Scalar Data Types Families

Numeric Data Types

Boolean Data Types

Character or String Data Types

Date/Time Data Types

Composite Data Types

Data Types Synonyms

Data Types & Select INTO Clause

Introduction

Each value in PL/SQL such as a constant, variable and parameter has a data type that determines the storage format, valid values, and allowed operations.

PL/SQL has two kinds of data types:

scalar and composite.

Data Types & Select INTO Clause

Scalar Data Types Families

Number

Boolean

Character

Datetime

A scalar data type may have **subtypes**. A subtype is a data type that is a subset of another data type, which is its base type. A subtype further defines a base type by restricting the value or size of the base data type.

Data Types & Select INTO Clause

Numeric Data Types

The numeric data types represent real numbers, integers, and floating-point numbers. They are stored as NUMBER, IEEE floating-point storage types (BINARY_FLOAT and BINARY_DOUBLE), and PLS_INTEGER.

The data types **NUMBER**, BINARY_FLOAT, and BINARY_DOUBLE are SQL data types. The PLS_INTEGER datatype is specific to PL/SQL. It represents signed 32 bits int.

Data Types & Select INTO Clause

Boolean Data Types

The BOOLEAN datatype has three data values: TRUE, FALSE, and NULL. Boolean values are typically used in control flow structure such as IF-THEN, CASE, and loop statements like LOOP, FOR LOOP, and WHILE LOOP.

SQL does not have the BOOLEAN data type, you can assign data type NUMBER(1) or CHAR(1)

Data Types & Select INTO Clause

Character or String Data Types

The character data types represent alphanumeric text. PL/SQL uses the SQL character data types such as

CHAR, VARCHAR2, LONG, RAW, LONG

RAW, ROWID, and UROWID

CHAR is used for fixed length columns

such primary keys or boolean

Data Types & Select INTO Clause

DateTime Data Types

The datetime data types represent dates, timestamp with or without time zone and intervals.

PL/SQL datetime data types are **DATE**, TIMESTAMP, TIMESTAMP WITH TIME ZONE,

TIMESTAMP WITH LOCAL TIME ZONE, INTERVAL YEAR TO MONTH, and

INTERVAL DAY TO SECOND

Data Types & Select INTO Clause

Composite Data Types

TableName.Column%TYPE is a composite data type. For example, **EMP.sal%TYPE**

It will follow same data type and size of

column of a table.

In case if size or type of column is changed then there is no need to re- compile the program

Data Types & Select INTO Clause

Data Types Synonyms

Data Type	Synonyms
NUMBER	DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, INT, NUMERIC, REAL, SMALLINT
CHAR	CHARACTER, STRING
VARCHAR2	VARCHAR

Data Types & Select INTO Clause

SELECT INTO Clause

We cannot use simple SQL Select clause statement, rather we use Select INTO by storing values in variables with condition of choosing single record

```
DECLARE
    v_salary EMP.sal%TYPE;
    v_deptno NUMBER(3);
BEGIN
    SELECT sal, deptno
    INTO v_salary, v_deptno
    FROM EMP
    WHERE empno = 7369;

    ...
END;
/
```

Introduction to Database Administration

TOPIC 125

Program Examples
Program Examples

Program Examples with Select INTO Clause

Program-1 without Exception
Program-2 using Exception
Program-3 using Constant identifier with Exception
Program Examples
Program-1 without Exception

```

declare
    v_ename          emp.ename%TYPE;
    v_sal            emp.sal%TYPE;
begin
    Select ename, sal
    INTO v_ename, v_sal
    from emp
    where empno = '7844'; --This id exists

    DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has '||v_sal);

end;
/

```

Before executing code that contains DBMS_OUTPUT.PUT_LINE,
must run at SQL prompt: set serveroutput on

Program Examples

```

declare
    v_ename          emp.ename%TYPE;
    v_sal            emp.sal%TYPE;
begin
    Select ename, sal
    INTO v_ename, v_sal
    from emp
    where empno = '7844'; --This id exists

    DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has '||v_sal);

exception
    When others then
        DBMS_OUTPUT.PUT_LINE('no record found');
end;
/

```

Program-2 using Exception

Program Examples

Program-3 using Constant identifier with Exception

```
declare
  v_ename          emp.ename%TYPE;
  v_sal            emp.sal%TYPE;
  v_empno          emp.empno%TYPE := '7844'; --Constant identifier
begin
  Select ename, sal
  INTO v_ename, v_sal
  from emp
  where empno = v_empno; --This id exists

  DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has salary '||v_sal);

exception
  When others then
    DBMS_OUTPUT.PUT_LINE('no record found');
end;
/
```

Introduction to Database Administration

TOPIC 126

Control IF Conditions

Control IF Conditions

PL/SQL Control IF-THEN-ELSE Conditions

IF THEN Condition

Structure of IF THEN Condition

Example of IF THEN Condition

Structure of IF THEN ELSE Condition
Example of IF THEN ELSE Condition
Structure of IF THEN ELSIF Condition
Example of IF THEN ELSIF Condition
Control IF Conditions

IF THEN Condition

There are three ways to write IF condition:

IF – END IF
IF – ELSE – END IF
IF – ELSIF – ELSE – END IF
Control IF Conditions

Structure of IF THEN Condition

There are three ways to write IF condition:

IF condition THEN
statements;
END IF;

For example:

IF v_sal > 2000 Then total_sal := v_sal + 5000;
END IF;
Control IF Conditions

Example of IF THEN Condition

```

DECLARE
    v_sal EMP.SAL%TYPE;
BEGIN
    Select sal INTO v_sal
    From emp
    Where ename = 'SMITH';

    IF v_sal < 850 THEN
        DBMS_OUTPUT.PUT_LINE('Low paid salary person');
    END IF;
END;
/

```

Control IF Conditions

Structure of IF THEN ELSE Condition

There are three ways to write IF condition:

```

IF condition THEN
statements;
ELSE
statements;
END IF;
Control IF Conditions

```

Example of IF THEN ELSE Condition

```

DECLARE
    v_sal EMP.SAL%TYPE;
BEGIN
    Select sal INTO v_sal
    From emp
    Where ename = 'SMITH';

    IF v_sal < 850 THEN
        DBMS_OUTPUT.PUT_LINE('Low paid salary person');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Highly paid salary person');
    END IF;
END;
/

```

Control IF Conditions

Structure of IF THEN ELSIF Condition

There are three ways to write IF condition:

```
IF condition THEN
statements;
ELSIF
statements;
ELSE
statements;
END IF;
Control IF Conditions
```

Example of IF THEN ELSIF Condition

```
DECLARE
    v_sal EMP.SAL%TYPE;
BEGIN
    Select sal INTO v_sal
    From emp
    Where ename = 'SMITH';

    IF v_sal < 850 THEN
        DBMS_OUTPUT.PUT_LINE('Low paid salary person');
    ELSIF (v_sal >= 850 AND v_sal <=1000) THEN
        DBMS_OUTPUT.PUT_LINE('Average paid salary person');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Higly paid salary person');
    END IF;
END;
```

Introduction to Database Administration

TOPIC 127

FOR & WHILE Loops

FOR & WHILE Loops

LOOPS, FOR & WHILE Loops

LOOPS

Loop with EXIT

FOR Loop

FOR Loop Syntax

FOR Loop - Example

FOR Loop - Bulk Data Insertion

WHILE Loop

WHILE Loop - Example

WHILE Loop Syntax

FOR & WHILE LOOPS

LOOPS

LOOP – EXIT WHEN – END LOOP

FOR – LOOP – END LOOP

WHILE – LOOP – END LOOP

FOR & WHILE LOOPS

Loop with EXIT

Following is its example (Repeat until LOOP):

```

DECLARE
    ...
    v_deptno    dept.deptno%TYPE    := 50;
    v_counter    integer := 1;
    ...
BEGIN
    ...
    LOOP
        INSERT INTO dept(deptno)
            VALUES(v_deptno);
        v_counter := v_counter + 1;
        v_deptno := v_deptno + 10;
        EXIT WHEN v_counter > 5;
    END LOOP;
    ...
END;
/

```

FOR & WHILE LOOPS

FOR Loop

Following is its Syntax:

```

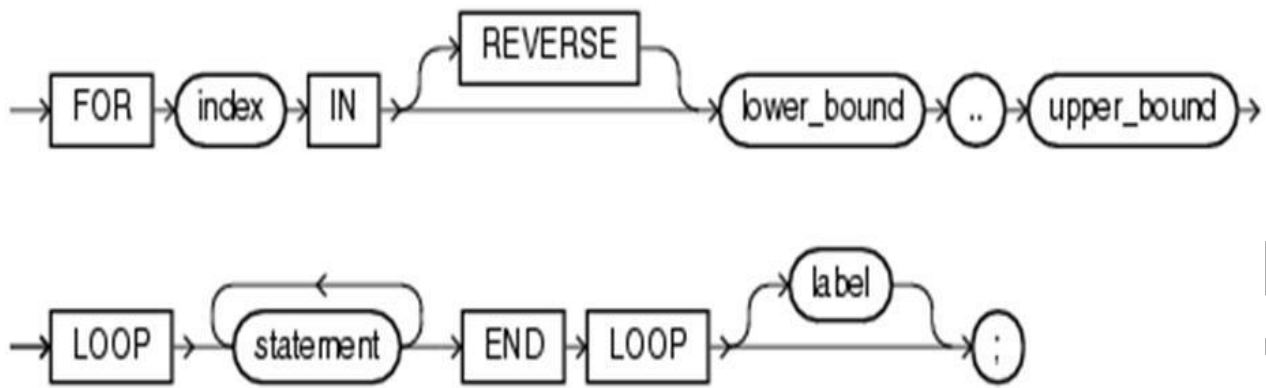
FOR index IN lower_bound .. upper_bound
LOOP
statements;
END LOOP;

```

FOR & WHILE LOOPS

FOR Loop Syntax

Following is its generic syntax:



FOR & WHILE LOOPS

FOR Loop - Example

Following example displays serial numbers:

```

BEGIN
  FOR l_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE( l_counter );
  END LOOP;
END;
/

```

FOR & WHILE LOOPS

FOR Loop - Example

Following example displays serial numbers in reverse:

```

BEGIN
  FOR l_counter IN REVERSE 1..3
  LOOP
    DBMS_OUTPUT.PUT_LINE( l_counter );
  END LOOP;
END;
/

```

FOR & WHILE LOOPS

FOR Loop - Bulk Data Insertion

Following example displays serial numbers in reverse:

```

BEGIN
  FOR i IN 55..70
  LOOP
    insert into dept(deptno, dname, loc)
    values (i, 'Dummy Dept-'||i, 'No Loc');
    DBMS_OUTPUT.PUT_LINE( i );
  END LOOP;
  Commit;
END;
/

```

DEPTNO	DNAME	LOC
55	Dummy Dept-55	No Loc
56	Dummy Dept-56	No Loc
57	Dummy Dept-57	No Loc
58	Dummy Dept-58	No Loc
59	Dummy Dept-59	No Loc
60	Dummy Dept-60	No Loc
10	ACCOUNTING	NEW YORK

FOR & WHILE LOOPS

WHILE Loop

Following is its Syntax:

WHILE condition
LOOP
statements;
END LOOP;
FOR & WHILE LOOPS

WHILE Loop - Example

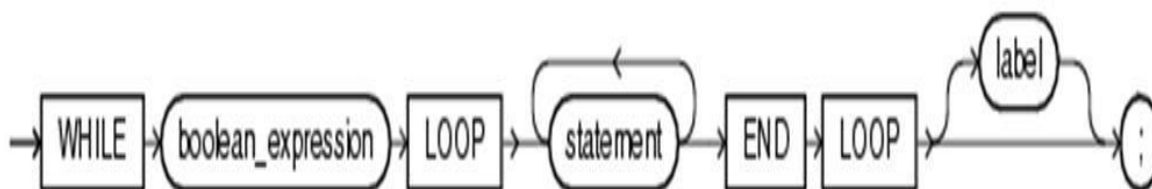
Following is its example:

```
DECLARE
  n_counter NUMBER := 1;
BEGIN
  WHILE n_counter <= 5
  LOOP
    DBMS_OUTPUT.PUT_LINE( 'Counter : ' || n_counter );
    n_counter := n_counter + 1
  END LOOP;
END;
```

FOR & WHILE LOOPS

WHILE Loop Syntax

Following is its generic syntax:



Introduction to Database Administration

TOPIC 128

Anonymous Blocks with Examples

Anonymous Blocks with Examples

Anonymous Blocks with Examples

Nesting Anonymous Blocks

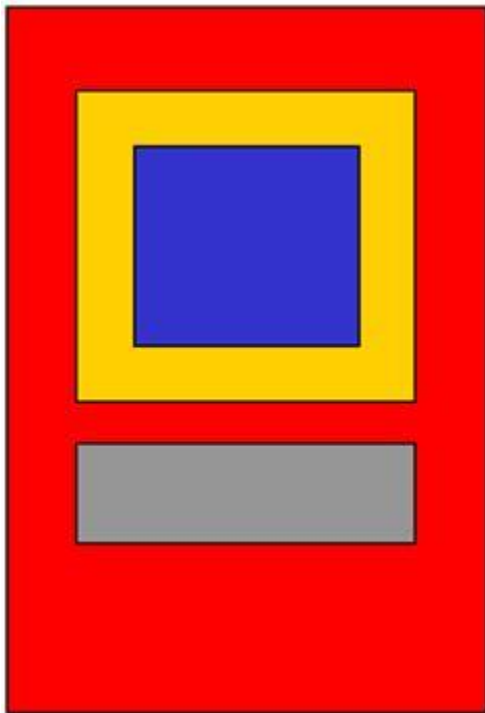
Nesting Anonymous Blocks – Example

Repeat Until LOOP - Example

Anonymous Blocks with Examples

Nesting Anonymous Blocks

Blocks within blocks give the concept of Nested blocks



Anonymous Blocks with Examples

Repeat Until LOOP - Example

This loop executes once first time. It is same as DO-WHILE Loop.

```

LOOP
    monthly_value := daily_value * 31;
    EXIT WHEN monthly_value > 4000;
END LOOP;

```

Anonymous Blocks with Examples

Nesting Anonymous Blocks - Example

```

DECLARE
    n_counter NUMBER := 1;
BEGIN
    DBMS_OUTPUT.PUT_LINE('*****----Outer Block----*****');
    FOR i IN 55..70
    LOOP
        DBMS_OUTPUT.PUT_LINE('Outer Block Counter : ' || i );
    END LOOP;
    BEGIN
        WHILE n_counter <= 5
        LOOP
            DBMS_OUTPUT.PUT_LINE('Inner Block Counter : ' || n_counter );
            n_counter := n_counter + 1;
        END LOOP;
    END; --inner block end
END; --outer block end
/

```

Introduction to Database Administration

TOPIC 129

Exception Handlings-1

Exception Handlings-1

PL/SQL Exception Handlings

PL/SQL Exception

PL/SQL Exception Usage

PL/SQL Exception - Example-1

PL/SQL Exception - Example-2

Exception Handlings-1

PL/SQL Exception

PL/SQL treats all errors that occur in an [anonymous block](#), [procedure](#), or [function](#) as exceptions.

The exceptions can have different causes such as coding mistakes, bugs, even hardware failures. It is not possible to anticipate all potential exceptions, however, you can write code to handle exceptions to enable the program to continue running as normal.

Exception Handlings-1

PL/SQL Exception

The code that you write to handle exceptions is called an exception handler.

A [PL/SQL block](#) can have an exception- handling section, which can have one or more exception handlers.

Exception Handlings-1

PL/SQL Exception Usage

BEGIN

-- executable section ...

-- exception-handling section EXCEPTION

WHEN **e1** THEN

-- exception_handler1

WHEN **e2** THEN

-- exception_handler1

WHEN **OTHERS** THEN

-- other_exception_handler

END;

/

Exception Handlings-1

PL/SQL Exception – Example-1

Following program will terminate without successfully execution when employee no does not exist in table EMP.

```

DECLARE
    v_ename EMP.ename%TYPE;
    v_empno EMP.empno%TYPE := &empno;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE empno = v_empno;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
END;
/

```

Exception Handlings-1

PL/SQL Exception – Example-1

Following program will execute successfully when employee no does not exist in table EMP.

```

DECLARE
    v_ename EMP.ename%TYPE;
    v_empno EMP.empno%TYPE := &empno;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE empno = v_empno;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Employee ID|' || v_empno ||
END;
/

```

Exception Handlings-1

PL/SQL Exception – Example-2

```

DECLARE
    v_ename EMP.ename%TYPE;
    v_job EMP.job%TYPE := &job;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE job = v_job;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Employee ID' || v_ename || ' does not exist');
    WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('Database returns many employees');
END;
/

```

```

Enter value for job: 'CLERK'
old 3: v_job EMP.job%TYPE := &job;
new 3: v_job EMP.job%TYPE := 'CLERK';
Database returns many employees

PL/SQL procedure successfully completed.

```

Following program will execute successfully using multiple Exceptions.
Introduction to Database Administration

TOPIC 130

Exception Handlings-2

PL/SQL Exception Handlings

Raise Exception
 User-Defined Exception
 Affecting PL/SQL Exception
 Other Exceptions
 Exception Handlings-2

Raise Exception

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command RAISE. Following is the simple syntax for raising an exception:

```

DECLARE
    exception_name EXCEPTION;
BEGIN
    IF condition THEN
        RAISE exception_name;
    END IF;
EXCEPTION
    WHEN exception_name THEN
        statement;
END;
/

```

Exception Handlings-2

User-Defined Exception

```

DECLARE
    v_empno EMP.empno%type := &cv_empno;
    v_name EMP.ename%type;
    v_job EMP.job%type;
    -- user defined exception
    ex_invalid_empno EXCEPTION;
BEGIN
    IF v_empno <= 0 THEN
        RAISE ex_invalid_empno;
    ELSE
        SELECT ename, job INTO v_name, v_job
        FROM EMP
        WHERE empno = v_empno;
        DBMS_OUTPUT.PUT_LINE ('Name: ' || v_name);
        DBMS_OUTPUT.PUT_LINE ('job: ' || v_job);
    END IF;

EXCEPTION |
    WHEN ex_invalid_empno THEN
        dbms_output.put_line('empno must be greater than zero!');
    WHEN no_data_found THEN
        dbms_output.put_line('No such employee!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
/

```

Exception Handlings-2

Affecting PL/SQL Exception

```

DECLARE
v_tname varchar2(100);
BEGIN
  BEGIN
    select tname into v_tname
    from tab
    where tname='CMG_EFFORTT';
    DBMS_OUTPUT.PUT_LINE('table name: '||v_tname);
  exception
    when no_data_found then
      DBMS_OUTPUT.PUT_LINE('No data');
  END;
  select tname into v_tname
  from tab
  where tname='CMG_EFFORT';
  DBMS_OUTPUT.PUT_LINE('table name: '||v_tname);
  exception
  when no_data_found then
    DBMS_OUTPUT.PUT_LINE('No data second exception');
  --go to 1;
END;
/

```

One exception does not affect the rest of program
Exception Handlings-2

Other Exceptions

Examples are
NO_DATA_FOUND, ZERO_DIVIDE, OTHERS

Raise some exception

WHEN NO_DATA_FOUND THEN

raise_application_error(-20011,'Invalid FK value'); Raise My_Excep;

To display details of oracle standard error message EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Error detail is: '||SQLERRM)

Introduction to Database Administration

TOPIC 131

PL/SQL Procedure

PL/SQL Procedure

PL/SQL Procedure

What is a Procedure in PL/SQL?

Procedure - Example

Compiling & Executing Procedure

Invoking a Procedure in a Block

Executing a Procedure & Show Errors

Data Dictionary for Procedures

Managing Procedure using SQL Developer

PL/SQL Procedure

What is a Procedure in PL/SQL?

Is a block with a name

The DECLARE key word is not used

Parameters can be IN

OUT

IN OUT

Is stored (**USER_SOURCE**)

PL/SQL Procedure

What is Procedure in PL/SQL?

Writing PL/SQL procedure

```

CREATE OR REPLACE PROCEDURE pname( ) IS
BEGIN
EXCEPTION
END;
/

```

Procedure – Example

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE

proc_test(p_empno IN VARCHAR2) IS

v_job EMP.job PE:

v_sal EMP.sal PE:

BEGIN

EXCEPTION

SELECT job, sal INTO v_job*v_sal FROM emp

WHERE empno = p_empno;

DBMS_OUTPUT.PUT_LINE('Job is '||v_job);

END;

EXCEPTION WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('ERROR .

.');

PL/SQL Procedure

Compiling & Executing Procedure

```
SQL> Show errors
```

```
SQL> execute proc_test(5893);
```

PL/SQL Procedure

Invoking a Procedure in a Block

```
DECLARE
  ...
BEGIN
  ...
  proc_test('23');
  ...
END;
/
```

Or

```
SQL> exec proc_test('1123')
```

PL/SQL Procedure

Executing a Procedure & Show Errors

```
1 create or replace procedure test_proc is
2     v_id  INTEGER;
3     v_empno emp.empno%TYPE;
4 BEGIN
5     v_id := 1234567;
6     select empno into v_empno
7     FROM EMP
8     WHERE empno = v_id;
9 EXCEPTION
10    WHEN NO_DATA_FOUND THEN
11        DBMS_OUTPUT.PUT_LINE('No record exists');
12* END;
SQL> /
```

Procedure created.

```
SQL> exec test_proc
No record exists
```

PL/SQL procedure successfully completed.

```
SQL> show errors          (to see errors for procedures, functions)
```

PL/SQL Procedure

Data Dictionary for Procedures

```

SQL> select Text from user_source
      2  where name='PROC_TEST';

TEXT
-----
PROCEDURE
    proc_test(p_empno IN VARCHAR2) IS
        v_job  EMP.job%TYPE;
        v_sal  EMP.sal%TYPE;
BEGIN
    SELECT job, sal
    INTO  v_job,v_sal
    FROM  emp
        WHERE empno = p_empno;
    DBMS_OUTPUT.PUT_LINE('job is '||v_job);
EXCEPTION

TEXT
-----

    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('ERROR...');
END;

```

PL/SQL Procedure

Managing Procedure using SQL Developer