IT 4153 Advanced Database

Module 6 Database Environment

|  |
| --- |
| **Introduction and Module Summary** |
| In this module, you will learn several essential DBMS concepts: how to improve database and application performance and security |
| **Objectives and Outcomes** |
| This module directly supports **highlighted** course outcome(s)  Students who complete this course successfully will be able to   1. Describe current and emerging database models and technologies; 2. Develop functions and procedures for data manipulation and database access auditing; 3. **Describe database monitoring and performance tuning;** 4. Describe database security and administration issues, including backup and recovery; 5. Explain the concepts of data warehousing and data mining   **Module outcomes and activities:**   |  |  |  |  | | --- | --- | --- | --- | | After completing this module, students will be able: | to identify sources of metadata in Oracle | to perform denormalization | to query system catalog to extract metadata | | Readings | Reinforced | introduced | introduced | | Practice exercises | Reinforced | Reinforced | Reinforced | | Lab | mastered |  | mastered | |
| **Assigned Reading** |
| 1. Oracle Data structure denormalization <http://www.dba-oracle.com/art_9i_denormal.htm> 2. Isolation <http://en.wikipedia.org/wiki/Isolation_%28database_systems%29> 3. ACID Properties <http://docs.oracle.com/cd/E35855_01/tuxedo/docs12c/ads/adtrn.html> 4. Naming Convention <http://ss64.com/ora/syntax-naming.html> 5. Data Dictionary <http://docs.oracle.com/database/121/CNCPT/datadict.htm> |
| **Optional Reading** |
| 1. Virtual columns <https://logicalread.com/performance-costs-oracle-denormalization-mc08/#.WnNwjKinGUl> 2. Oracle Catalog Views <http://www.oracle.com/pls/db92/db92.catalog_views?remark=homepage> |
| **Assessments and Assignments** |
| 1. Lab (10 points) |
| **Topics** |
| Open the navigation pane |
| **Module Checklist** |
| This is the suggested order of the completion of this module.  Save a copy of this file on your computer and make notes in this document while you are completing your assignments. Use the table below to keep track of your progress.   |  |  | | --- | --- | | **Activity** | **Completion** | | Read this module and assigned materials (3 hours) | NO | | Complete all exercises from the module (1 hour) | NO | | Complete the lab (1 hour) | NO | | Complete a feedback section at the end of the module | NO | | Read feedback provided for your discussion and lab. | NO | |
|  |

# DBMS Environment

## DBMS Architectures

Type of Architecture

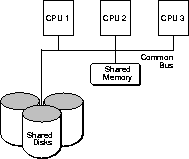
* centralized
* distributed

Distributed databases can be described as a collection of multiple, logically interrelated databases distributed over a computer network. This distribution is transparent to the user. A big advantage of distributed DBMSs is scalability.

Distributed DBMS can have three common types of sharing that can be combined:

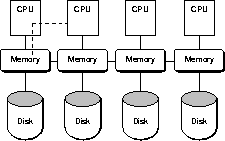
Shared Memory

* Communication between nodes occurs via shared memory.
* Performance is limited by the bandwidth of the memory bus.



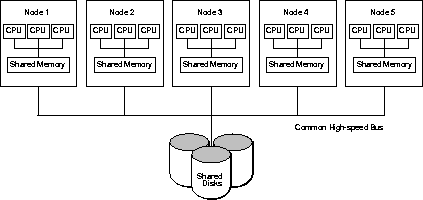
Shared-nothing

* Can exploit simpler, cheaper hardware
* Almost unlimited scalability
* Works well in a high-volume, read-write environment
* Data is partitioned across the cluster



Shared disk

* Quick adaptability to changing workloads
* High availability
* Performs best in a heavy read environment
* Data need not be partitioned



## ACID (Atomicity, Consistency, Isolation, Durability)

A notion of transaction is very important for proper database application functioning. It allows to create distributed applications easily and let commit changes as a single operation. A simple example of a transaction is a withdrawal from a bank account. It consists of three operations:

1. Verify that a withdrawal can be made

2. Withdraw a specified amount from the account

3. Update the balance of the account

When a transaction such as a bank withdrawal is performed, it is imperative that all its constituent operations either succeed or fail together. ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably.

**Atomicity** A transaction is a discrete unit of work: all constituent operations must either succeed or fail. These operations may include queuing messages, updating databases, and displaying the results of a transaction on a screen.

**Consistency** A transaction must either (a) leave the system in a correct state or (b) abort. If a transaction cannot achieve a stable state, it must return to its initial state.

**Isolation** The behavior of a transaction is not affected by other transactions being executed simultaneously. A transaction must serialize all access to shared resources and guarantee that concurrent programs do not corrupt each other's operations.

**Durability** The effects of a committed transaction are permanent. Even if the system fails, the changes resulting from a transaction are permanent and durable.

(oracle.com)

## DBMS Installation

When you made a decision on what DBMS should be used for your application, the next step is DBMS Installation. The following steps will help you to avoid mistakes:

1. Read installation guide
2. Choose the correct DBMS for your need and match your hardware to the requirements of the DBMS. Pay attention to 32 vs 64 bit applications.
3. Do not underestimate storage requirements. Except your database object DBMS needs to store:
   1. System catalog or data dictionary
   2. Log files
   3. Startup or control files
   4. Work files
   5. Default databases
   6. Temporary database structures
   7. System dump and error processing files
   8. DBA databases used for administration, monitoring, and tuning
4. Estimate memory requirements for data processing and program cache.
5. Install DBMS and all service packs and patches.
6. Search DBMS vendor site for white paper dedicated to configuration best practices. Generally it is not recommended to choose default installation and configuration. Sample code has to be deleted from production servers.
7. Verify DBMS connection to the supporting infrastructure (networks, transaction processing monitors, etc)
8. Follow manufacture installation verification instructions.
9. Create at least two DBMS instances: production and test.

## DBMS Upgrade

When new version of DBMS is release consider upgrade.

Upgrade benefits

* new features
* enhanced performance
* phased out support for older version

Upgrade drawbacks

* requires a lot of resources to migrate and test new environment
* might require changing SQL and application code
* possible down time

To reduce number of problems create upgrade plan and perform upgrade on a test environment. Upgrade plan should include step by step instructions for upgrade, migration and verification. Upgrade can require replacement for software that does not support newer DBMS version.

## Database Standards and Procedures

**Standards** are common practices that ensure the consistency and effectiveness of the database environment, such as database naming conventions.

**Procedures** are scripts that direct the processes required for handling specific events, such as a disaster recovery plan.

Failure to implement database standards and procedures will result in a database environment that is confusing and difficult to manage.

Standards and Procedures should address the following:

* Use naming convention. It helps to read code and easily differentiate between function, views, tables etc
* Data administration standards
  + Rules for data creation, data ownership, and data stewardship
  + Metadata management policy
  + Organizational data sharing policies
* Database administration standards
  + How request is made to make a new database or make changes
  + Backup and recovery procedures
* System administration standards
  + Installation and testing procedures
  + Maintenance practices
* Database application development standards
* Database security standards
* Application migration and turnover procedures
* Design review guidelines
* Operational support standards

## Physical Database Design

In order to create physical database from logical model

1. Transform entities to table
2. Transform attributes to columns
3. Transform domains to data types
4. Define primary keys
5. Understand that order of columns might impact performance
6. Build referential constraint for all relationships
7. Build physical data structures
   * Map each table to a tablespace
   * Plan for storage and space usage. Row size is established by adding up the maximum size of all columns
   * Estimate size of table and index structures
   * Keep free space for reorganizing and restructuring
   * Wisely use compression
   * Do not forget Security
8. Design indexes to improve performance

## Locking

The DBMS uses a locking mechanism to enable multiple concurrent users to access and modify data in the database. Locks are used to ensure the integrity of data. Data may be locked at different levels within the database:

* Column
* Row
* Page (or block)
* Table
* Tablespace
* Database

The level of locking is known as lock granularity.

Lock timeout might be set at the DBMS level, the process level, or the connection level. Lock duration is the length of time that a lock is held by the DBMS.

Application should be designed to prevent deadlocks. Deadlocks happen when concurrent processes are competing for locks.

Isolation level specifies the locking behavior for a transaction or statement

## Isolation Level

<http://en.wikipedia.org/wiki/Isolation_%28database_systems%29>

UNCOMMITTED READ read-through lock aka "dirty read" during transaction is allowed.

COMMITTED READ aka cursor stability; data can't be used until the transaction completes or rolls back.

REPEATABLE READ retrieved data can't be changed.

SERIALIZABLE isolation provides the greatest integrity; all transactions occur in a completely isolated fashion.

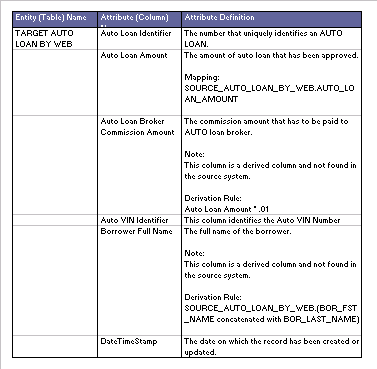
Lock Escalation is a process of converting many fine-grain locks into fewer coarse-grain locks, when the number of locks held by the transaction exceeds its threshold, reducing system overhead

**Programming Technique to Minimize locking problems**

Application developers must understand the impact of locking on the performance and availability of their application. Save all data modification until the end of the transaction.

# Data Dictionaries and Metadata

Metadata is the data that provides information about tables, views, constraints, stored procedures, indexes, etc. stored within the database. Metadata can be a technology metadata or a business metadata. Technology metadata provides technical information about a database object, such as, field structures of a table, the owner of a table, the tablespace, and constraints.



**Data Dictionary**

Most database management systems actively record metadata in system catalog. Database administrators can obtain valuable metadata from system catalogs. The Oracle system catalog is comprised of tables and views which allow the user to obtain information about the database, such as what other tables and views are available, their attributes, constraints, etc. Oracle uses the term "Data Dictionary" for its system catalogs. Oracle has hundreds of available system catalog relations, only some of which are available to the average user (as many are intended for use only by qualified database administrators).

Generally, the data dictionary consists of base tables and user-accessible views. The base tables contain all database information that is dynamically updated by Oracle RDBMS. The user-accessible views summarize and display the information stored in the base tables; they display the information from the base tables in readable and/or simplified form using joins, column aliases, and so on. The SYS user is the owner of all data dictionary tables and views

The tables and views provided by the data dictionary contain information about

* + users and their privileges,
  + tables, table columns and their data types, integrity constraints, indexes,
  + statistics about tables and indexes used by the optimizer,
  + privileges granted on database objects,
  + storage structures of the database.

Many views provided by the data dictionary are divided into three groups: USER, ALL, and DBA. The group name builds the prefix for each view name. For some views, there are associated synonyms as given in brackets below.

**USER\_** : Tuples in the USER views contain information about objects owned by the account performing the SQL query (current user)

USER TABLES all tables with their name, number of columns, storage

information, statistical information etc. (TABS)

USER\_CATALOG tables, views, and synonyms (CAT)

USER\_COL\_COMMENTS comments on columns

USER\_CONSTRAINTS constraint definitions for tables

USER\_INDEXES all information about indexes created for tables (IND)

USER\_OBJECTS all database objects owned by the user (OBJ)

USER\_TAB\_COLUMNS columns of the tables and views owned by the user

(COLS)

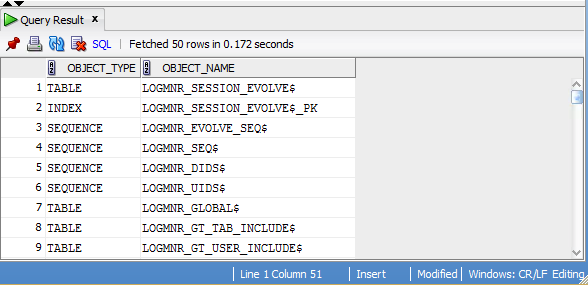
USER\_TAB\_COMMENTS comments on tables and views

USER\_TRIGGERS triggers defined by the user

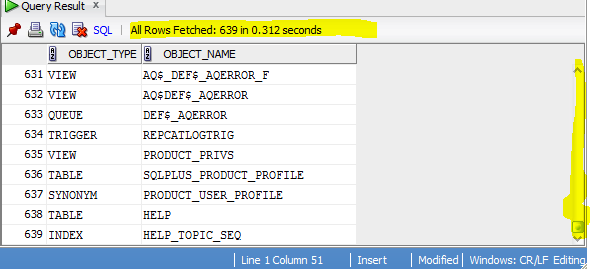
USER\_USERS information about the current user

For example, the query below returns information about all objects the current user has created including procedures, functions, and triggers as well as tables, indexes, constraints, etc.

select object\_type, object\_name from user\_objects;



Please note that not all rows are fetched, you have to scroll to see the last row.



**ALL\_** : Rows in the ALL views include rows of the USER views and all information about objects that are accessible to the current user. The structure of these views is analogous to the structure of the USER views.

ALL\_CATALOG owner, name and type of all accessible tables, views, and

synonyms

ALL\_TABLES owner and name of all accessible tables

ALL\_OBJECTS owner, type, and name of accessible database objects

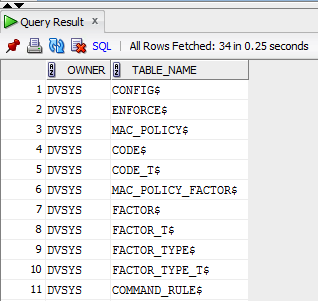
ALL\_TRIGGERS

ALL\_USERS lists all users of the database visible to the current user.

ALL\_VIEWS

For example, to find out all tables owned by current user, the following query can be issued:

select owner, table\_name from all\_tables where owner = 'DVSYS';



<https://docs.oracle.com/database/121/DVADM/db_objects.htm#DVADM70326>

The DVSYS schema contains Oracle Database Vault database objects, which store Oracle Database Vault configuration information and support the administration and run-time processing of Oracle Database Vault. In a default installation, the DVSYS schema is locked. The DVSYS schema also owns the AUDIT\_TRAIL$ table.

**DBA**\_ : The DBA views encompass information about all database objects, regardless of the owner. Only users with DBA privileges can access these views.

DBA\_TABLES tables of all users in the database

DBA\_CATALOG tables, views, and synonyms defined in the database

DBA\_OBJECTS object of all users

DBA\_ DATA\_FILES information about data files

DBA\_USERS information about all users known in the database

DBA\_ROLES all roles which exist in the database

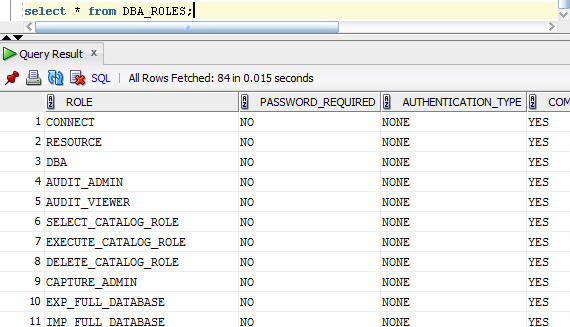
DBA\_ROLES\_PRIVS roles granted to users and roles

DBA\_TAB\_PRIVS All grants on objects in the database

DBA\_TS\_QUOTAS Tablespace quotas for all users

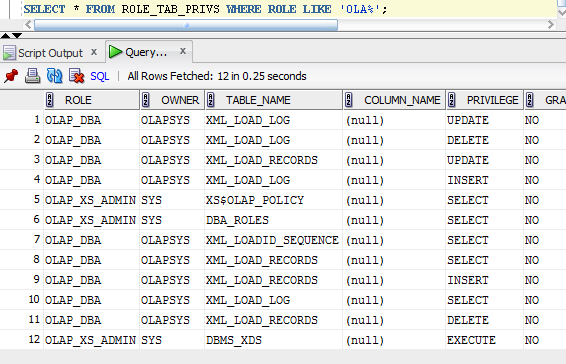
ROLE\_TAB\_PRIVS Table privileges granted to roles

SELECT \* FROM DBA\_ROLES;

 For example, ROLE\_TAB\_PRIVS describes table privileges granted to roles. Information is provided only about roles to which the user has access.

Select \* from role\_tab\_privs where role like 'ROLENAME%';

Select \* from role\_tab\_privs where role like 'OLA%';



| **Column** | **Datatype** | **NULL** | **Description** |
| --- | --- | --- | --- |
| ROLE | VARCHAR2(30) | NOT NULL | Name of the role |
| OWNER | VARCHAR2(30) | NOT NULL | Owner of the object |
| TABLE\_NAME | VARCHAR2(30) | NOT NULL | Name of the object |
| COLUMN\_NAME | VARCHAR2(30) |  | Name of the column, if applicable |
| PRIVILEGE | VARCHAR2(40) | NOT NULL | Object privilege granted to the role |
| GRANTABLE | VARCHAR2(3) |  | YES if the role was granted with ADMIN OPTION; otherwise NO |

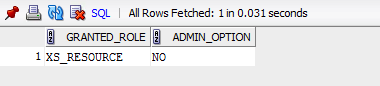
ROLE\_SYS\_PRIVS describes system privileges granted to roles. Information is provided only about roles to which the user has access.

| **Column** | **Datatype** | **NULL** | **Description** |
| --- | --- | --- | --- |
| ROLE | VARCHAR2(30) | NOT NULL | Name of the role |
| PRIVILEGE | VARCHAR2(40) | NOT NULL | System privilege granted to the role |
| ADMIN\_OPTION | VARCHAR2(3) |  | Indicates whether the grant was with the ADMIN option (YES) or not (NO) |

ROLE\_ROLE\_PRIVS describes the roles granted to other roles. Information is provided only about roles to which the user has access.

Select granted\_role, admin\_option from role\_role\_privs where role like 'ROLENAME%';

Select granted\_role, admin\_option from role\_role\_privs where role like 'OLA%';

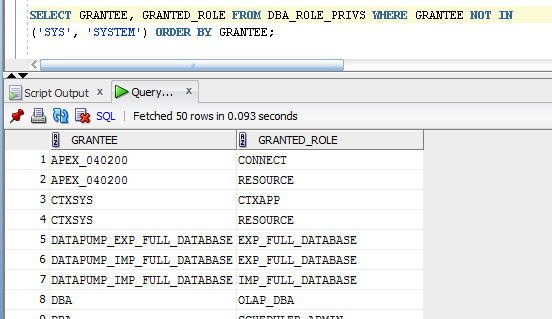


| **Column** | **Datatype** | **NULL** | **Description** |
| --- | --- | --- | --- |
| ROLE | VARCHAR2(30) | NOT NULL | Name of the role |
| GRANTED\_ROLE | VARCHAR2(30) | NOT NULL | Role that was granted |
| ADMIN\_OPTION | VARCHAR2(3) |  | Signifies that the role was granted with ADMIN option |

The following query can be used to find what users (grantees) have been granted what roles.

select grantee, granted\_role from dba\_role\_privs where grantee not in

('SYS', 'SYSTEM') order by grantee;



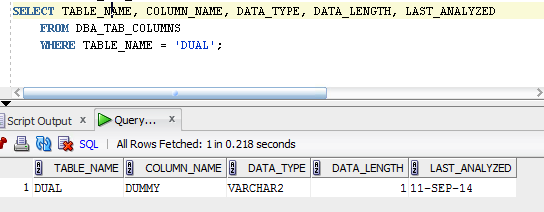
Column information, such as name, datatype, length, precision, scale, and default data values can be listed using one of the views ending with the \_COLUMNS suffix. For example, the following query lists all of the default column values for the emp and dept tables:

SELECT TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE, DATA\_LENGTH, LAST\_ANALYZED

FROM DBA\_TAB\_COLUMNS

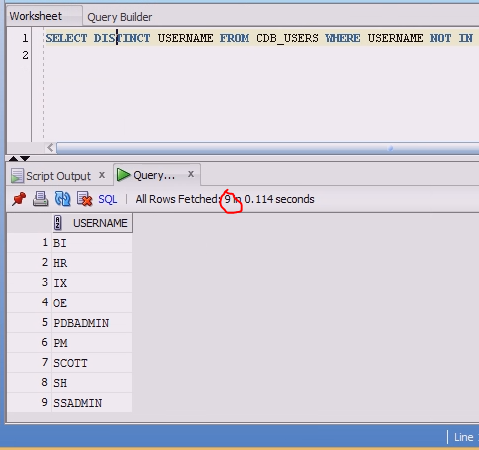
WHERE TABLE\_NAME = 'DUAL';

The following is the output from the query:



CDB\_ ... everything in every container only if the container is opened.

SELECT DISTINCT USERNAME FROM CDB\_USERS WHERE USERNAME NOT IN (SELECT USERNAME FROM DBA\_USERS) ORDER BY USERNAME;



**Data Dictionary and Database Performance**

In Oracle, Much of the data dictionary information is kept in the SGA in the **dictionary cache,** because Oracle constantly accesses the data dictionary during database operation to validate user access and to verify the state of schema objects. All information is stored in memory using the least recently used (LRU) algorithm. With the increase in the number of database objects, the size of the data dictionary grows considerably. Any information that is not cached in the dictionary cache must be obtained from the database block every time it is required, that involves I/O and potentially lots of it. Performance bottlenecks in the data dictionary affect all Oracle users.

**Data Repository**

A **repository** is used to store information about an organization’s data assets. Typical a repository can be used to

* Store information about your data, processes, and environment
* Store in-depth documentation as well as produce detail and management reports from that documentation.
* Support data model creation and administration

The repository should be capable of directly reading the system catalog or views on the system catalog for each DBMS you use. This ensures that the repository will have current information on database objects. Repositories are usually created and maintained using tools. Metadata repositories are essential components of Business Intelligence and Data Warehousing systems.

# Denormalization

Demoralization is a process of deliberately introducing redundancy to your data in order to speed up the data retrieval process.

When to denormalize:

* When system cannot achieve acceptable performance without denormalization.
* Only if required performance will be achieved after a denormalization.
* Only if reliability of the system will not be compromised.

When a column is replicated in many different tables, always use triggers to update it everywhere simultaneously.

Cases when denormalization usually helps:

1. When several joins are used to produce frequently retrieved report.
2. When in order to successfully run a frequent query many computation need to be performed one or many columns.

**Types of Denormalization**

**Prejoined Tables**: When two or more tables need to be frequently joint and they have relatively static data.

**Report tables**: Used for carrying the result of multiple joins, correlated subqueries or other complex SQL statement that require extensive calculations.

**Split Tables**: Table can be use split in one of two ways:

* vertical (groups of columns are placed in separate tables, primary key in each table)
* horizontal via key ranges

**Denormalization Example**

1. Split table

Original table

TABLE ITEM

(ItemNum integer

ItemSize string

ItemColor string

ItemDescr very long string)

ItemDescr column is not used very often in queries, the table can be denormalized to keep full description in a separate table.

TABLE ITEM

(ItemNum integer

ItemSize string

ItemColor string

ItemDescr 10 first characters of the very long string)

TABLE ITEM\_DESC

(ItemNum integer

ItemDesc full description – very long string)

To ensure that all data remains valid and accurate after denormalization, two triggers on table ITEM have to be implemented:

1. INSERT trigger. The trigger should insert first 10 characters of the full description in table ITEM and also insert full description in table ITEM\_DESC.
2. DELETE trigger. The trigger should delete item from both tables ITEM and ITEM\_DESC.

2. Redundant Data

When one or more columns from one table are accessed almost every time, the columns can be appended to the table as redundant data. Putting zip code, state and the city in the customer address table makes sense, while it is true that the city and state can be determined directly from the zip code.

3. Derivable Data

Consider storing derived data only when

* The source data used for the derivation calculation is relatively static
* The cost of performing the derivation calculation is quite high
* The usage pattern of the source tables is such that recalculation can be performed quickly when the source data chances.

Consider a table that has 48 columns for temperature reading that are updated every 30 minutes. The most common query is selecting an average temperature. If this query is executed more often than update, a new column *average\_temperature* should be added along with on update trigger that will recalculate its value.

# Views

A view is a way of turning a SELECT statement into a table. A view logically represents subsets of data from one or more tables.

Using views allows

* provide row and column level security,
* ensure efficient access paths,
* mask complexity from the users,
* rename tables or columns

Simple Views

Simple views created on one table, does not use functions and does not have GROUP BY clause. Simple views are always updatable.

Examples:

1. Horizontal view to restrict access:

CREATE VIEW dept10emp AS SELECT ssn, ename, job FROM emp WHERE deptno = 10;

SELECT \* from dept10emp;

2. Vertical view to restrict access and/or rename columns:

CREATE VIEW directory AS SELECT ename NAME, job POSITION FROM emp;

SELECT \* FROM directory WHERE name LIKE 'Dow%';

Complex Views

Complex views created on one or more tables, may contain functions (e.g. count), GROUP BY clause. Complex views are not always updatable.

Example:

Not updatable view:

CREATE TABLE testable1 (name varchar2(20), rank int);

INSERT INTO testable1 values ('John', 5);

INSERT INTO testable1 values ('John', 7);

INSERT INTO testable1 values ('John', 3);

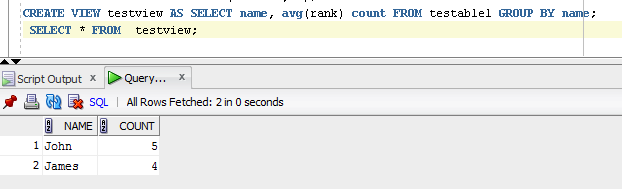
INSERT INTO testable1 values ('James', 3);

INSERT INTO testable1 values ('James', 4);

INSERT INTO testable1 values ('James', 5);

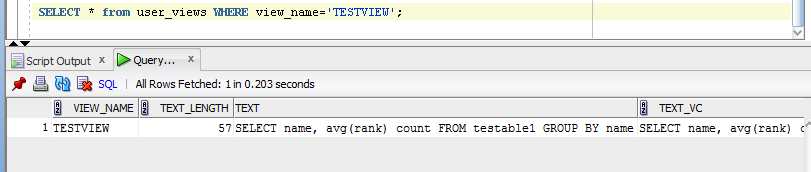
CREATE VIEW testview AS SELECT name, avg(rank) count FROM testable1 GROUP BY name;

SELECT \* FROM testview;



Oracle keeps information about views in data dictionary: dba\_views, all\_views, and user\_views

SELECT \* from user\_views WHERE view\_name='TESTVIEW';



To modify view:

* + - Oracle REPLACE VIEW <http://www.psoug.org/reference/views.html>

Removing views

* + - DROP view\_name;

# Lab

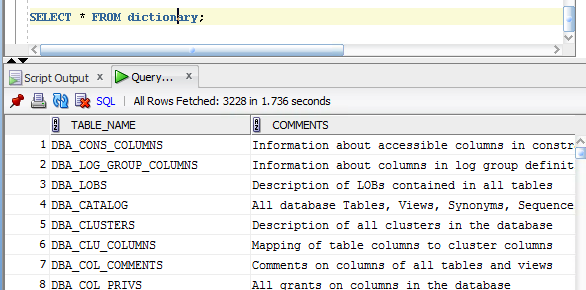
**Objectives**

Write an SQL query to extract the following information from the Oracle data dictionary:

For each question you need to submit a query and screenshot with results.

**Start Oracle SQL Developer**

1. Use SELECT \* FROM dictionary; to find all available dictionary views.



1. Find all tables that keep information about roles.
2. Use found tables to check what privileges does role CONNECT have?
3. Find all dictionary views that contain information about indexes.
4. How many indexes are defined on DVSYS.RULE\_SET\_RULE$ table? (Hint: all views have meaningful name, just search the page)
5. For each index you found in #3 how many distinct keys?
6. Describe the purpose of V$THREAD . Use V$THREAD to find enable time and las redo time.
7. Describe the purpose of V$BACKUP . Use V$BACKUP to find details about backups.
8. Find V$... view that will tell you what size of the following data types: DOUBLE, FLOAT, BOOLEAN
9. Use V$...stat views to find information about "free memory" and "DB time"

Hint: DB time is system parameter  
free memory is SGA (System Global Area) parameter

1. From your standpoint what is the most useful Oracle data dictionary view/table/procedure. Describe it and provide examples.

Include your code and results of the execution

**Feedback:**

Difficulty (-2 - too easy ... 0 - just right ... 2 - too hard)

Interest level (-2 - low interest ... 0 - just right ... 2 - high interest)

Time to complete (min)

Make a suggestion to improve