***ॐ***

***"Nothing is impossible.***

***Impossible just takes a little longer"***

***"Time is the important resource"***

***Right the First Time:-***

***Proof of Concept. (NASA):-*** *A proof of concept (POC) is a demonstration, the purpose of which is to verify that certain concepts or theories have the potential for real-world application.*

***Data on the hard disk is always clean.***

***Client will always get only UI. (Application Layer)***

***Customer is always the market.***

***Investment. (Car Cost)***

***Cost of ownership. (Car maintance)***

***Assembly Line. (Sewing machine)***

***A truth table***

***Final Project***

1. *HR System*
2. *Maintenance Management System*
3. *Electronic Medical Records*
4. *Airline Management System*
5. *Library Management System*

***Sub system***

*Access Control System (Logins)*

*Localization of the screens (Forms)*

***Documentation***

*Features*

*Workflow diagram*

*DFD*

*Commands (BoM) – Bill of Material*

*ERD*

*Database Structure (BoM) – Bill of Material*

*Business object and ORM*

*UI (BoM) – Bill of Material*

***TODO***

[***http://fornera.com/helpdesk/oracleGettingStartedWithJavaStoredProcedures.html***](http://fornera.com/helpdesk/oracleGettingStartedWithJavaStoredProcedures.html)

***dbms relational algebra***

***Functional dependency (FD)***

***https://lagunita.stanford.edu/c4x/Engineering/CS145/asset/opt-rel-algebra.html***

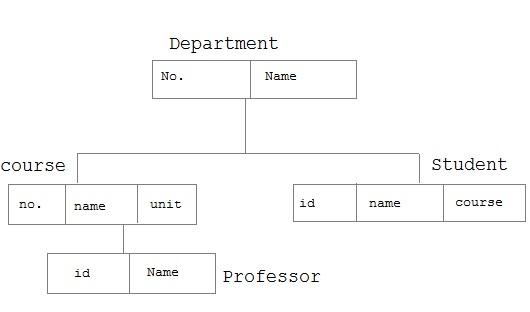
[***http://www.jkinfoline.com/***](http://www.jkinfoline.com/)

***min max cardinational***

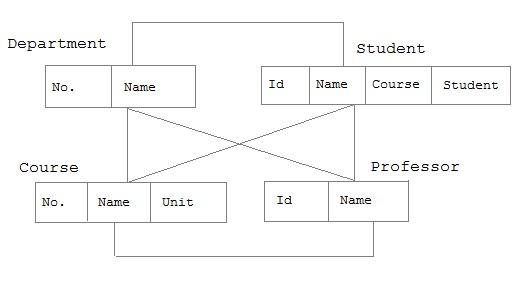
***Database Model***

* *Hierarchical Model*
* *Model*
* *Relational Model*

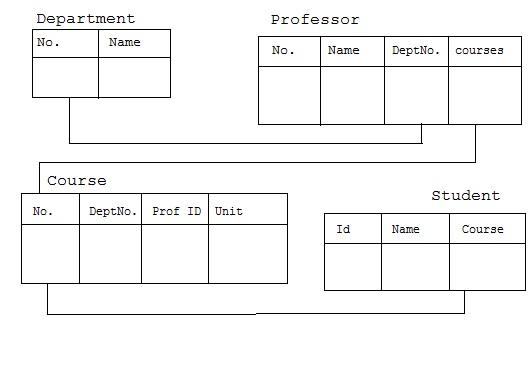
#### Hierarchical Model



#### Network Model



#### Relational Model



***Data*** *are certain* ***facts and statistics*** *that can be* ***recorded*** *and that can be* ***processed*** *by a computer. (Text / Multimedia Data)*

***Information*** *is processed data. Information is the connection of data* ***plus*** *the* ***meaning*** *which becomes meaning full*

*Each* ***database*** *is a collection of tables, which are called* ***relations****, hence the name* ***"relational database"****.*

***Entity***

*An entity can be a* ***real-world object****, either animate* ***(living)*** *or inanimate* ***(non-living)*** *e.* ***For example****, in a school database, students, teachers, classes, and courses offered can be considered as entities.* ***All these entities have some attributes or properties that give them their identity.***

***Every entity has its own characteristics.***

***Entity Type***

*A set of entities that have the* ***same attributes*** *is called an* ***entity type.*** *Each entity type in the database is described by a name and a list of attributes.* ***For example*** *an entity employee is an entity type that has Name, Age and Address attributes.*

***Eg.***

***Entity TYPE Entity***

*Person (Age, Name, Address) - 16 , Sharmin, …*

*The database schema changes very infrequently. The database state changes every time the database is updated. Schema is also called intension, whereas state is called extension.*

***Attribute domain***

*The attribute domain is the set of values allowed in an attribute. Each value in the tuple must be of some basic type, like a* ***STRING*** *or an****INTEGER***

***Attribute***

***In general, an attribute is a characteristic.***

***In Entity Relationship(ER) Model attributes can be classified into the following types.***

1. ***Simple* / *Atomic Attribute --VS-- Composite Attribute***

*(Can’t be divided further) (Can be divided further)*

1. ***Single Value Attribute --VS-- Multivalued Attribute***

*(Only One value) (Multiple values)*

1. ***Stored Attribute --VS-- Derived Attribute***

*( ) (Virtual columns)*

1. ***Complex Attribute*** *(Composite & Multivalued)*

***Single Value Attribute***

***Composite Attribute***



***Attribute*** 

*Age*

***Person***

***Name******First Name, Last Name, Middle Name***

*Address*

*Phone*

***Simple Attribute***

***ID***

***ID Key Attribute***

***Multivalued***

***Attribute***

*Salary HRA* ***10%*** *DA* ***20% Phone1, Phone2***

***Of Salary of Salary*** 



***Derived***

***Attribute***

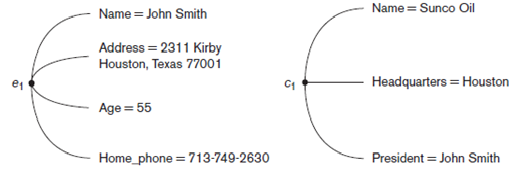
***Stored***

***Attribute***

* **Prime attribute** − An attribute, which is a part of the prime-key, is known as a prime attribute.
* **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

***Entity and Attributes***

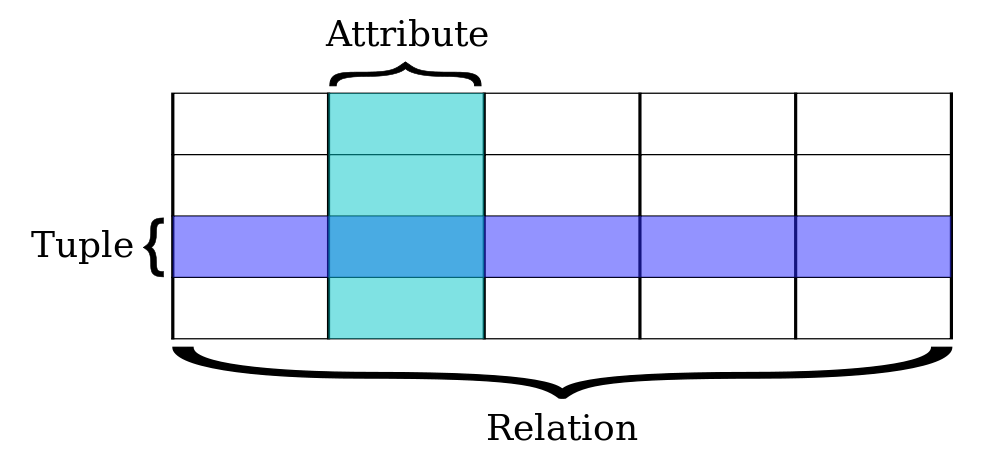
*Two Entities, Employee* ***e1*** *and Company* ***c1****, and* ***their Attributes****.*

******

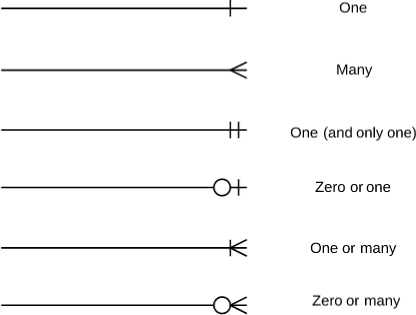
***Relation***

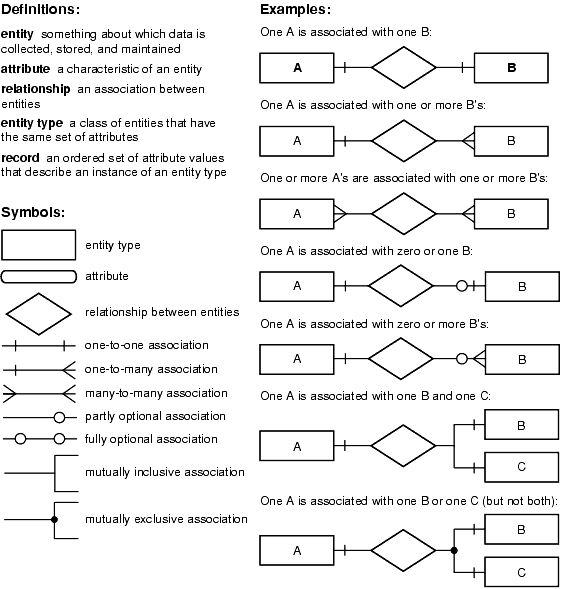
*A relation is usually described as a* ***table****, which is organized into* ***rows*** *and* ***columns****.*

|  |  |  |
| --- | --- | --- |
| **SQL term** | **Relational database term** | **Description** |
| ***Row*** | ***Tuple*** or ***record*** | A data set representing a single item |
| ***Column*** | ***Attribute*** or ***field*** | A labeled element of a tuple, e.g. "Address" or "Date of birth" |
| ***Table*** | ***Relation*** or ***Base relvar*** | A set of tuples sharing the same attributes; a set of columns and rows |
| ***View*** or ***result set*** | ***Derived relvar*** | Any set of tuples; a data report from the RDBMS in response to a query |



***Relationships***

****



*Degree of a Relation -* ***Number of attributes in a relation.***

*Cardinality of a Relation* ***- Number of tuples in a relation.***

***There are three types of relationship that exist between Entities.***

1. Unary Relationship
2. Binary Relationship
3. Ternary Relationship

Unary Relationship Image

***Difference Attribute and Column.***

*In the (logical) Entity Relationship Model (ERM),* ***an "Entity" has "Attributes".***

*When we implement the model into a relational database,* ***an "attribute" for an "entity" gets stored as a "column" in a "table".***

***Data Domain***

*Use* ***E-R model*** *to get a high-level* ***graphical view*** *to describe/* *represent the* ***"ENTITIES"*** *and their* ***"RELATIONSHIP"***

***Entity Relationship Symbols***

***Example for a strong & weak entity:*** *In a* ***parent/child relationship****, a* ***parent*** *is considered as a* ***strong entity*** *and the* ***child*** *is a* ***weak entity.***

*A member of a* ***strong entity set*** *is called* ***DOMINANT ENTITY*** *and member of* ***weak entity******set*** *is called as* ***SUBORDINATE ENTITY****.*

*The relationship between* ***weak entity and strong entity set*** *is called as* ***IDENTIFYING RELATIONSHIP****.*

***Strong Entity***

***Weak Entity***

***Attribute***

***Key Attribute***

***Derived Attribute***

***Multivalued Attribute***

***Composite Attribute***

***Relationship***

***Weak Relationship***

**Total Participation**

**Partial participation**

***Data modeling*** *is a process used to define and analyze data requirements.*

***Data modeling****involves a progression from conceptual model to logical model to physical schema.*

***The three levels of data modeling,***

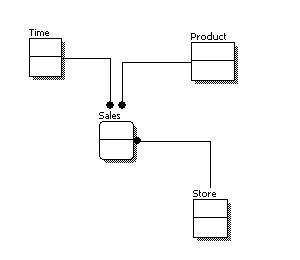
1. *Conceptual data model*
2. *Logical data model*
3. *Physical data model*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Feature*** | ***Conceptual*** | ***Logical*** | ***Physical*** |
| *Entity Names* | ✓ | ✓ |  |
| *Entity Relationships* | ✓ | ✓ |  |
| *Attributes* |  | ✓ |  |
| *Primary Keys* |  | ✓ | ✓ |
| *Foreign Keys* |  | ✓ | ✓ |
| *Table Names* |  |  | ✓ |
| *Column Names* |  |  | ✓ |
| *Column Data Types* |  |  | ✓ |

### *Conceptual Model Design*

*Features of conceptual data model include:*

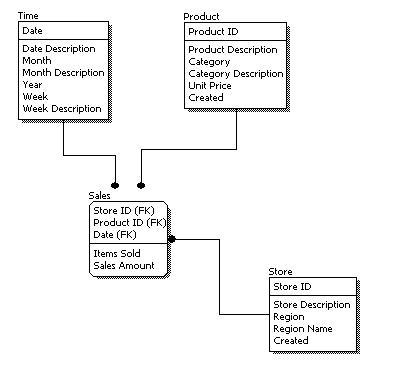
1. *Includes the important entities and the relationships among them.*
2. *No attribute is specified.*
3. *No primary key is specified.*



### *Logical Model Design*

*Features of a logical data model include:*

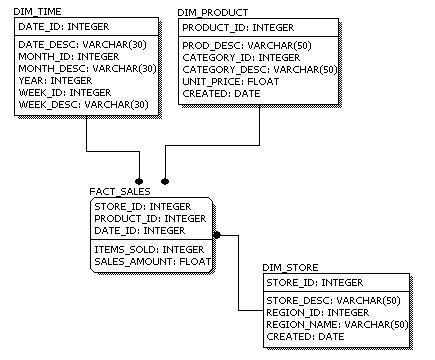
1. *Includes all entities and relationships among them.*
2. *All attributes for each entity are specified.*
3. *The primary key for each entity is specified.*
4. *Foreign keys (keys identifying the relationship between different entities) are specified.*



### *Physical Model Design*

*Features of a physical data model include:*

1. *Convert entities into tables.*
2. *Convert relationships into foreign keys.*
3. *Convert attributes into columns.*



***Relational Algebra Symbols***

***Unary Operators***

*Selection* **σ** ***(Sigma)***

*Projection* **π** ***(Pi)***

*Renaming* ρ ***(rho)***

***Binary Operators***

## *Union* ∪

## *Intersection* ∩

*Difference* **—**

*Cartesian* *product* **×**

*Join* **⋈**

*Left Outer Join* 

*Right Outer Join*  

*Full Outer Join* 

*Assignment* 

***Logic Symbols***

*Logical* *AND* **∧**

*Logical* *OR* **∨**

*Logical* *NOT* **¬**

***Sets***

*x* **∈** *A x* ***belongs to*** *A;* **∈**

*x is an element of the set A*

*x* **∉** *A x* ***does not belong*** *to A;* **∉**

*x is not an element of the set A*

*Such that* ***(so that)*****|**

*Therefore* **∴**

*Because* **∵**

*There exists* **∃**

*There does not exists* **∄**

*For all* **∀**

Selection (**σ**) Operators

σ<Selection condition>(R)

* *The* **σ** *would represent the SELECT command*
* *The* ***<selection condition>*** *would represent the condition for selection.*
* *The* ***(R)*** *would represent the Relation or the Table from which we are making a selection of the tuples.*

***Eg.***

**σempno = 7**(EMPLOYEE)

**σDOB <** **'01-Jan-1980'**(EMPLOYEE)

**σ((DOB < 1935) ∧ (state = 'Arizona'))**(EMPLOYEE)

Projection (**π)** Operators

**π**<attribute list>(R)

* **π** *would represent the ROJECT.*
* ***<attribute list>*** *would represent the attributes (columns) we want from a relational.*
* ***(R)*** *Would represent the relation or table we want to choose the attributes from.*

***Eg.***

**πdob, empno** (EMPLOYEE)

Selection and Projection

***Eg.***

**π** ename, sal (**σ** ((DOB < 1935) **∧** (state = **'**Arizona**'**)) (EMPLOYEE))

Renaming (ρ) Operators

ρ s (B1, B2, B3,….Bn)(R)

* **ρ** *is the RENAME operation.*
* **S** *is the new relation name.*
* **B1, B2, B3, …Bn** *are the new renamed attributes (columns).*
* **R** *is the relation or table from which the attributes are chosen.*
* *The* **arrow symbol ←** *means that we first get the PROJECT operation results on the right side of the arrow then apply the RENAME operation on the results on the left side of the arrow.*

***Eg***

**ρ s (Birth\_Date, Employee\_Number)** (EMPLOYEE) **← πdob, empno** (EMPLOYEE)

***IMP NOTES***

* *When operations of equal precedence are to be performed,* ***they are executed from left to right.***
* *AND and OR also obey precedence rules,* ***with AND having the highest precedence****.*

***Cannot perform a DML operation inside a query. That is if the FUNCTION is having INSERT, UPDATE, DELETE SQL statement then the function cannot be called using SELECT command on SQL prompt.***

***Display all employee earning same salaries.***

*SQL> SELECT a.\* from EMP a where 1 < (SELECT count (\*) from EMP b where b.sal = a.sal);*

*SQL> SELECT c1, c2, c3,* ***abs (c3-&no1) c4****, &no1 from NEAREST where c2= '&c2' order by c4;*

*with*

*a as (SELECT sysdate, sysdate+462 newdate from DUAL),*

*b as (SELECT trunk (months\_between (trunk (sysdate+462), trunk (sysdate))/12) days from DUAL),*

*c as (SELECT trunk (mod (months\_between (trunk (sysdate+462), trunk (sysdate)),12)) months from DUAL),*

*d as (SELECT trunk (trunk (sysdate) + 462 - add\_months (sysdate ,trunc (months\_between (trunc(sysdate+462), trunc (sysdate))))) days from DUAL)*

*SELECT \* from a, b, c, d*

*/*

*SELECT case*

*when (sysdate + 7) - sysdate between 1 and 7 then (sysdate + 7) - sysdate || ' days ago'*

*when (sysdate + 7) - sysdate between 8 and 7 then (sysdate + 7) - sysdate || ' days ago'*

*end case from DUAL*

***Find last 2 records***

*SQL> SELECT \* from (SELECT rownum r1, LOGIN.\* from LOGIN) where r1 > (SELECT count (rownum) - 2 from LOGIN);*

***Oracle SQL***



* ***container database (CDB)***
* ***pluggable databases (PDB)***

*SQL>* ***show con\_name***

*SQL>* ***SELECT NAME, CON\_ID, DBID, CON\_UID, GUID FROM V$CONTAINERS ORDER BY CON\_ID;***

*SQL>* ***SELECT PDB\_ID, PDB\_NAME, STATUS FROM DBA\_PDBS ORDER BY PDB\_ID;***

*Note: - When we ...................*

*SQL>* ***ALTER system flush SHARED\_POOL;***

*SQL>* ***SELECT HASH\_VALUE, OBJECT\_OWNER, OBJECT\_NAME, OBJECT\_TYPE from V$SQL\_PLAN where object\_owner = 'C##SALEEL';***

*SQL>* ***SELECT SQL\_ID, HASH\_VALUE, OBJECT\_OWNER, OBJECT\_NAME, OBJECT\_TYPE from V$SQL\_PLAN where object\_owner='C##SALEEL';***

***O/P***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***SQL\_ID*** | ***HASH\_VALUE*** | ***OBJECT\_OWNER*** | ***OBJECT\_NAME*** | ***OBJECT\_TYPE*** |
| ***dmf6j59yz06mm*** | *2112887411* | *C##SALEEL* | *EMP* | *TABLE* |
| ***dt908h3g30k84*** | *3727706372* | *C##SALEEL* | *EMP* | *TABLE* |
| ***3d3qdug8d0q9z*** | *3503315263* | *C##SALEEL* | *EMP* | *TABLE* |
| ***50us5bnb1p58k*** | *370840850* | *C##SALEEL* | *EMP* | *TABLE* |
| ***a2dk8bdn0ujx7*** | *1745700775* | *C##SALEEL* | *EMP* | *TABLE* |

*SQL>* ***SELECT sql\_text from V$SQLTEXT where SQL\_ID='a2dk8bdn0ujx7';***

***O/P***

*SQL\_TEXT*

*----------------------------------------------------------------*

***SELECT \* FROM EMP***

*SQL>* ***SELECT sql\_text from V$SQLTEXT, V$SQL\_PLAN where OBJECT\_OWNER = 'C##SALEEL' and V$SQLTEXT.SQL\_ID= V$SQL\_PLAN.SQL\_ID;***

***O/P***

*SQL\_TEXT*

*----------------------------------------------------------------*

***SELECT \* from DEPT***

***SELECT \* from EMP***

***SELECT \* from EMP fetch first 7 row only***

***SELECT \* from EMP fetch first 5 rows only***

***Type of optimizer***

***RBO --- Ruled Based optimizer***

***CBO --- Cost Based optimizer***

***Types of Parsing***

*All statements, DDL or DML, are parsed whenever they are executed. The only key fact is that whether it was a* ***Soft parse*** *(statement is already parsed and available in memory) or a* ***Hard parse*** *(all parsing steps to be carried out).*

***Parsing process***

*Oracle internally does the following to arrive at the output of an SQL statement.*

1. ***Syntactical check****. The query fired is checked for its syntax.*
2. ***Semantic check****. Checks on the validity of the objects being referred in the statement and the privileges available to the user firing the statement. This is a data dictionary check.*
3. ***Allocation of private SQL area*** *in the memory for the statement.*
4. *Generating a parsed representation of the statement and allocating Shared SQL area. This involves finding an optimal execution path for the statement.*

***Identical statements***

*Oracle does the following to find identical statements to decide on a* ***soft parse or a hard parse.***

1. *When a new statement is fired, a hash value is generated for the text string. Oracle checks if this new hash value matches with any existing hash value in the shared pool.*
2. *Next, the text string of the new statement is compared with the hash value matching statements. This includes comparison of case, blanks and comments present in the statements.*
3. *If a match is found, the objects referred in the new statement are compared with the matching statement objects. Tables of the same name belonging to different a schema will not account for a match.*
4. *The bind variable types of the new statement should be of same type as the identified matching statement.*
5. *If all of the above is satisfied, Oracle re-uses the existing parse (soft). If a match is not found, Oracle goes through the process of parsing the statement and putting it in the shared pool (hard).*

***ROWID is the physical address for each record in the database and it is a fixed-length binary data.***

***BODMAS Rules***

***B - Brackets first***

***O - Orders (ie Powers and Square Roots, etc.)***

***DM - Division and Multiplication (left-to-right)***

***AS - Addition and Subtraction (left-to-right)***

***Online Transaction Processing (OLTP)***

***Online Analytical Processing (OLAP)***

* **Run sqlplusw (Window base)**
* **Cmd sqlplus (DOS Base)**
* **http://infoserver/isqlplus (9i)**
* **http://laba135:5560/isqlplus/ (10g)**
* [**http://localhost:5560/isqlplus/**](http://localhost:5560/isqlplus/) **(10g)**
* [**https://192.168.100.58:1158/em/console**](https://192.168.100.58:1158/em/console/logon/logon) **(11g)**
* **D:\app\infostaff\product\11.1.0\db\_1\sqldeveloper\sqldeveloper\bin\sqldeveloperw.exe**

Search in Google: - SQLPLUS\_FONT

Search in Google: - SQLPLUS\_FONT\_SIZE

***SQL was developed by IBM in the 1970s***

***Oracle Developed in C, C++ language.***

***Most of the compilers, JVMs, and Kernels used today are written in C/C++.***

HKEY\_LOCAL\_MACHINE/SYSTEM/CurrentControlSet/Services

**SYS user: change\_on\_install**

**If you fail to remember Oracle System password then**

Go to oracle server and give the following

Enter user-name: system as sysdba ⮠

Enter password: **⮠**

***User Roles***

*SQL> SELECT \* from SESSION\_ROLES;*

***Oracle parameters***

***SQL> show parameter***

***SQL> show parameter name***

***SQL> show parameter type***

***SQL> show parameter db\_name***

***SQL> show con\_name***

***SQL> show release***

*SQL> SELECT name from V$ACTIVE\_SERVICES;*

*SQL> set errorlogging on/off*

Error information is written in sperrorlog table.

*SQL> SELECT \* from SPERRORLOG;*

**Different generations of languages: -** *There are currently five generations of computer programming languages. In each generation, the languages syntax has become easier to understand and more human-readable.*

* ***First generation languages (1GL):-*** *Represent the very early, primitive computer languages that consisted entirely of 1's and 0's the actual language that the computer understands (machine language)*
* ***Second generation languages (2GL):-*** *Represent a step up from from the first generation languages. Allow for the use of symbolic names instead of just numbers. Second generation languages are known as assembly languages. Code written in an assembly language is converted into machine language (1GL).*

|  |
| --- |
| **E.G.**  *Add 12 , 8* |

* ***Third generation languages (3GL):-*** *With the languages introduced by the third generation of computer programming, words and commands (instead of just symbols and numbers) were being used. These languages therefore, had syntax that was much easier to understand. Third generation languages are known as "high level languages" and include C, C++, Java, and JavaScript, among others.*

|  |
| --- |
| **E.G.**  *public boolean handleEvent (Event evt) {*  *switch (evt.id) {*  *case Event.ACTION\_EVENT: {*  *if ("Try me" .equald(evt.arg)) {* |

* ***Fourth generation languages (4GL):-*** *The syntax used in 4GL is very close to human language, an improvement from the pervious generation of languages. 4GL languages are typically used to access databases and include SQL and ColdFusion, among others.*

|  |
| --- |
| **E.G.**  *SELECT ALL CUSTOMERS WHERE "PREVIOUS PURCHASES" TOTAL MORE THAN $1000* |

* ***Fifth generation languages (5GL):-*** *Fifth generation languages are currently being used for neural networks. A neural network is a form of artificial intelligence that attempts to imitate how the human mind works.*

***Difference between .DLL and .EXE***

1. DLL - Dynamic Link Library

***.DLL*** *runs is an in process server running in the same memory space as the client process.*

1. EXE – Executable File

***.EXE*** *is an out of process server which runs in its own separate memory space*.

***Escape sequences in ‘C’***

* ***\’***  *Single quotation mark used for character literals*
* ***\”***  *Double quotation mark used for string literals*
* ***\\***  *Backslash used for file path*
* ***\?*** *Question mark*
* ***\a***  *Alert*
* ***\b***  *Backspace*
* ***\f***  *Form feed*
* ***\n***  *New line*
* ***\r***  *Carriage return*
* ***\t***  *Horizontal tab*
* ***\v***  *Vertical tab*
* ***\0***  *Null*

***File access mode***

|  |  |
| --- | --- |
| **mode** | **Description** |
| "r" | Opens a file for reading. The file must exist. |
| "w" | Creates an empty file for writing. If a file with the same name already exists, its content is erased and the file is considered as a new empty file. |
| "a" | Appends to a file. Writing operations, append data at the end of the file. The file is created if it does not exist. |
| "r+" | Opens a file to update both reading and writing. The file must exist. |
| "w+" | Creates an empty file for both reading and writing. |
| "a+" | Opens a file for reading and appending. |

***Difference between Primary storage and secondary storage***

* *Primary memory storages are temporary; whereas the secondary storage is permanent.*
* *Primary memory is expensive and smaller, whereas secondary memory is cheaper and larger*
* *Primary memory storages are faster, whereas secondary storages are slower.*
* *Primary memory storages are connected through data buses to CPU, whereas the secondary storages are connect through data cables to CPU*

***Types of software***

* *System software*
* *Programming software*
* *Application software*

***Server***

A **computer** or **device** on a **network** that manages **network** resources.

***Types of Networks***

* Personal area network, or PAN
* Local area network, or LAN
* Metropolitan area network, or MAN
* Wide area network, or WAN

***Platform Independent***

*Software that can run on any hardware platform (PC, Mac, SunSparc, etc.) or software platform (Linux, MacOS, UNIX, Windows, etc.). In general, programs written in Java language can be executed on practically every platform.*

***Flat-file***

*A collection of data or information that has a name, called the filename. Almost all information stored in a computer must be in a file. There are many different types of files: data files, text files, program files, directory files, and so on.*

***Distributed database***

*A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed, different users can access it without interfering with one another. However, the database must be synchronize to make sure that they all have consistent data.*

***Cloud database***

*A cloud database is a database that typically runs on a cloud computing platform. There are two common deployment models: users can run databases on the cloud independently, using a virtual machine image, or they can purchase access to a database service, maintained by a cloud database provider.*

***Difference between DBMS and RDBMS***

***DBMS:***

1. *In dbms no relationship concept*
2. *It supports Single User only*
3. *It treats Data as Files internally*
4. *It supports 3 rules of E.F.CODD out off 12 rules*
5. *It requires low Software and Hardware Requirements.*
6. *FoxPro, IMS are some examples*

***RDBMS:***

1. *It is used to establish the relationship concept between two database objects, i.e, tables*
2. *It supports multiple users*
3. *It treats data as Tables internally*
4. *It supports minimum 6 rules of E.F.CODD*
5. *It requires High software and Hardware Requirements.*
6. *SQL-Server, Oracle are some examples*

***RDBMS supports***

1. ***C/S Technology***
2. ***Highly Secured***
3. ***Relationship (PK/FK)***

***What is Business Intelligence?***

*Business Intelligence is the process of transforming data into information.*

***ACID properties of transactions***

***Atomicity***

*All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are.*

*For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.*

***Consistency***

*Data is in a consistent state when a transaction starts and when it ends.*

*For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.*

***Isolation***

*The intermediate state of a transaction is invisible to other transactions. Data modifications made by one transaction must be isolated from the data modifications made by all other transactions.*

*For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.*

***Durability***

*After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.*

*For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed.*

*Advantages of RDBMS: - Large data storage, Security, Client/Server Technology.*

***Rownum: -*** *For each row returned by a query, the ROWNUM pseudo column returns a number indicating the order in which Oracle selects the row from a table or set of joined rows.*

***Cardinality: -*** *Total row’s present in Table.*

***Degree: -*** *Total column’s present in Table.*

***What is degree of a Relation?*** *It is the number of attribute of its relation schema.*

***What is Oracle?*** *Oracle is a company; it is also a database server, which manages the data in as structured manner.*

***What is a data?*** *Data is any facts, number or text that can be processed by a computer.*

***What is a database?*** *A database is a system to organize, store and retrieve large amounts of data easily which is stored in one or more data files by one or more users.*

***SQL Structured Query Language*** *is a database language designed and developed for managing data in relational database management systems (RDBMS).* ***SQL is common language for all Relational Databases.***

***Edgar F. Codd Database Rules***

***Rule (0):*** *The system must qualify as relational, as a database, and as a management system.*

***Rule (1): The information rule:*** *All information in the database is to be represented in only one way, namely by values in column positions within rows of tables.*

***Rule (2): The guaranteed access rule:*** *All data must be accessible.*

***Rule (3): Systematic treatment of null values:*** *The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values.*

***Rule (4): Active online catalog based on the relational model:*** *The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.*

***Rule (5): The comprehensive data sublanguage rule:*** *The system must support at least one relational language that*

*1. Has a linear syntax*

*2. Can be used both interactively and within application programs,*

*3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).*

***Rule (6): The view updating rule:*** *All views that are theoretically updatable must be updatable by the system.*

***Rule (7): High-level insert, update, and delete:*** *The system must support insert, update, and delete operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.*

***Rule (8): Physical data independence:*** *Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.*

***Rule (9): Logical data independence:*** *Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.*

***Rule (10): Integrity independence:*** *Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.*

***Rule (11): Distribution independence:*** *The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:*

*1. When a distributed version of the DBMS is first introduced; and*

*2. When existing distributed data are redistributed around the system.*

***Rule (12): The nonsubversion rule:*** *If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.*

***Compile-time error***

*Compile time error is any type of error that prevent a java program compile like a syntax error, a class not found, a bad file name for the defined class*

***Runtime errors***

*Run time errors are those that passed compiler’s checking, but fails when the code gets executed. There are a lot of causes may result in runtime errors, such as incompatible type-casting, referencing an invalid index in an array, using an null-object, trying to invoke a method on an uninitialized variable, trying to open a file that doesn't exist. etc*

***Oracle Client configuration parameters***

* *Service Name (orcl)*
* *Host Name (Server Name / IP address)*
* *Protocol (TCP/IP)* ***(Transmission Control Protocol / Internet Protocol)***
* *Port (1521, 1810, 7778, 2481)* ***(Port number is the gateway from where the data is send or received.)***

**Note: - The term port can refer to either physical or virtual connection points.**

* + ***Physical ports*** *allow connecting cables to computers, routers, modems and other peripheral devices.*
  + ***Virtual ports*** *are part of TCP/IP networking. These ports allow software applications to share hardware resources.*

**What is SID in oracle?** *SID stands for System Identifier and it is a unique name or number to assign to the system on which the oracle has been installed*.

**Extension and Intension**

***Extension*** - *It is the number of tuples present in a table at any instance.*

***Intension*** - *It is a constant value that gives the name, structure of table and the constraints laid on it.*

***Oracle database architecture consists of a combination of three main components.***

1. **Oracle Memory Components**
2. **Oracle Server Process**
3. **Oracle Physical Files.**
4. **Oracle Memory Components & Oracle Server Process**: - *Oracle Memory Components and Oracle Server Process runs in the main memory of the Windows NT on which Oracle engine is installed and run.*
5. **Oracle Physical Files**: - *Oracle Physical files are all on the Windows NT server's hard disk drive from which their instance is loaded in main memory.*

Oracle Physical file Structure

a. **Data Files:** - *These are the files that actually store the physical data is been stored.*

b. **Redo Log Files:** - *Redo Log Files or Redo Files as they are known to store copies of every transaction.*

c. **Control Files:** - *Oracle control files store the structure, location and size of all files within a database.*

d. **Parameter File:** - *It is known as init.ora file. This file stores all initialization parameters of an Oracle database. This tells the Oracle engine how to configure its memory and background process. The init.ora file is read every time an Oracle database instance is started.*

***Oracle Background Process***

***SMON (System Monitor):*** *- SMON checks, if changes recorded in the Redo Log File are written to the database or not. If these changes are not written to the data files on the hard disk, it only means that the database was not shutdown normally.*

***PMON (Process Monitor)****: - The Process Monitor performs a memory clean up after any user process fails or crash. It includes*

*(A) Freeing up database locks.*

*(B) Cleaning up Caches*

*(C) Removing the process ID of the failed process from list of active processes.*

***DBWR (Database Writer):*** *- Database Writer is responsible for writing of data from Redo Log file to the data files on the hard disk.*

***LGWR (Log Writer):*** *- Log Writer is responsible for physically writing data from the on-line Redo Log Buffer in memory to the Redo Log Files on the hard disk.*

***CKPT (Checkpoint):*** *- Checkpoint is an event that occurs when the DBWR process writes all modified Database Buffer contents.*

***Normalization***

***First Normal Form (1NF)***

1. Eliminate duplicative columns from the same table.
2. Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).

* Manager
* Subordinate1
* Subordinate2
* Subordinate3
* Subordinate4

**[Manager Table]**

ManagerID

SubordinateID

***Second Normal Form (2NF)***

1. Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
2. Create relationships between these new tables and their predecessors through the use of foreign keys.

* SubordinateID
* FirstName
* LastName
* Address
* City
* State
* ZIP
* Working Hrs
* Rate / Hrs

**[ZIP Table]**

* ZIP (PK)
* City
* State

**[Employee Details]**

* SubordinateID (PK)
* FirstName
* LastName
* Address
* ZIP (FK)
* Working Hrs
* Rate / Hrs

***Third Normal Form (3NF)***

1. Meet all the requirements of the second normal form.
2. Remove columns that are not dependent upon the primary key.

**[Hour Table]**

* WorkingID (PK)
* Working Hrs
* Rate / Hrs

**[Employee WorkingHrs]**

* SubordinateID (PK)
* WorkingID (FK)

***Fourth Normal Form (4NF)***

* Meet all the requirements of the third normal form.
* A relation is in 4NF if it has no multi-valued dependencies.

*http://databases.about.com/od/specificproducts/a/normalization.htm*

***Tablespaces and Data Files*** *The usable data of an ORACLE database is logically stored in the tablespaces and physically stored in the data files associated with the corresponding tablespace.*

***SGA*** *The System Global Area (SGA) is a memory area that contains data shared between all database users such as buffer cache and a shared pool of SQL statements. The SGA is allocated in memory when an Oracle database instance is started.*

***Oracle Instance*** *Every running Oracle database is associated with an Oracle instance. When a database is started on a database server (regardless of the type of computer), Oracle allocates a memory area called the System Global Area (SGA) and starts one or more Oracle processes. This combination of the SGA and the Oracle processes is called an Oracle instance. The memory and processes of an instance manage the associated database's data efficiently and serve the one or multiple users of the database.*

***REDO Buffer File*** *The Redo Log file is a memory component within the SGA, which records all the changes made to the Database. Such as INSERT, UPDATE, DELETE*

***Transport Network Substrate (TNS)?*** *TNS, Transport Network Substrate, is a foundation technology, built into the Oracle Net foundation layer that works with any standard network transport protocol.*

***Library cache:*** *Stores the text, parsed format, and execution plan of SQL statements that have been submitted to the RDBMS, as well as the headers of PL/SQL packages and procedures that have been executed. For each SQL statement the server first checks the library cache to see if an identical statement has already been submitted and cached. If it has, then the server uses the stored parse tree and execution path for the statement, rather than building these structures from scratch.*

***User Scott / tiger*** *Bruce Scott (one of the original Oracle employees) and the password is the name of his daughter's cat.*

***Physical Backups and Logical Backups***

**Physical backups** *are backups of the physical files used in storing and recovering your database, such as datafiles, control files, and archived redo logs. Ultimately, every physical backup is a copy of files storing database information to some other location, whether on disk or some offline storage such as tape.*

**Logical backups** *contain logical data (for example, tables or stored procedures) exported from a database with an Oracle export utility and stored in a binary file, for later re-importing into a database using the corresponding Oracle import utility.*

***Types of Commands in Oracle***

* Set System Variable Commands (Sets a system variable to alter the SQL\*Plus environment settings for your current session.)
* SQL+ Commands
* SQL Commands
* PL/SQL Commands

***SQL> Select all \* from EMP;***

***Expressions in where clause process in the following order:***

|  |  |
| --- | --- |
| **Number** | **Expression** |
| **1** | Arithmetic Operators |
| **2** | Concatenation Operator |
| **3** | Comparison Condition |
| **4** | IS [NOT] NULL, LIKE, [NOT] IN |
| **5** | [NOT] BETWEEN |
| **6** | Not Equal To |
| **7** | NOT Logical Condition |
| **8** | AND Logical Condition |
| **9** | OR Logical Condition |

***Types of Objects (12)***

1. ***Table Object***
2. ***View Object***
3. ***Synonym Object***
4. ***Sequence Object***
5. ***Index Object***
6. ***Type Object***
7. ***User Object***
8. ***Java Object***
9. ***Procedure Object***
10. ***Function Object***
11. ***Trigger Object***
12. ***Package Object***

***CREATE table as statement***

*SQL> CREATE table TEMP as SELECT \* from EMP;*

***Rollback*** *will not work because CREATE is DDL which is auto committed.*

***There are five types of keys in database***

<https://www.youtube.com/watch?v=EavNLiTk-eo>

* *Candidate key*
* *Primary Key*
* *Foreign Key*
* *Alternate Key*
* *Composite Key*

*'ll take example of an Employee table:*

*Employee (  
Employee ID,   
FullName,   
SSN,   
DeptID)****1. Candidate Key:****are individual columns in a table that qualifies for uniqueness of all the rows. Here in Employee table EmployeeID & SSN are Candidate keys.****2. Primary Key:****is the columns you choose to maintain uniqueness in a table. Here in Employee table you can choose either EmployeeID or SSN columns, EmployeeID is preferable choice, as SSN is a secure value.****3. Alternate Key:****Candidate column other the Primary column, like if EmployeeID is PK then SSN would be the Alternate key.****4. Super Key:****If you add any other column/attribute to a Primary Key then it became a super key, like EmployeeID + FullName is a Super Key.****5. Composite Key:****If a table do have a single columns that qualifies for a Candidate key, then you have to select 2 or more columns to make a row unique. Like if there is no EmployeeID or SSN columns, then you can make FullName + DateOfBirth as Composite primary Key. But still there can be a narrow chance of duplicate row.*

***Types of Constraints***

1. *Primary Key (32 cols)*
2. *Foreign Key (32 cols)* ***(On Delete Cascade / On Delete Set Null)***
3. *Unique (32 cols)*
4. *Check (****ENUM*** *datatype in MySQL)*
5. *Not Null*
6. *Default*

|  |  |
| --- | --- |
| ***Constraint Type*** | ***Character*** |
| *PRIMARY KEY* | ***P*** |
| *UNIQUE KEY* | ***U*** |
| *FOREIGN KEY* | ***R*** |
| *CHECK, NOT NULL* | ***C*** |

***A composite unique key cannot have more than 32 columns.***

***A composite primary key cannot have more than 32 columns***

***A composite foreign key cannot have more than 32 columns***

***NOT NULL Constraint***

***Syntax:***

1. *COLUMN [data type] [NOT NULL]*
2. *COLUMN [data type] [CONSTRAINT constraint\_name ] constraint\_type*

***UNIQUE constraint***

***Syntax:***

1. *COLUMN [data type] [CONSTRAINT <name>] [UNIQUE]*

***Primary Key***

*If the primary key was created using an existing index,* ***then the index is not dropped.***

*If the primary key was created using a system-generated index,* ***then the index is dropped.***

***Syntax:***

***Column level:***

1. *COLUMN [data type] [CONSTRAINT <constraint name> PRIMARY KEY]*

***Table level:***

1. *CONSTRAINT [constraint name] PRIMARY KEY [column (s)]*

*SQL> CREATE table DEMO (C1 integer* ***primary key [deferrable] initially immediate****);*

***// constraint is evaluated immediately i.e. once the INSERT is performed.***

*SQL> CREATE table DEMO (C1 integer* ***primary key [deferrable] initially deferred****);*

***// constraint is evaluated when you perform a COMMIT.***

***Foreign Key***

***Syntax:***

***Column Level:***

1. *COLUMN [data type] [CONSTRAINT] [constraint name] [REFERENCES] [table name (column name)]*

***Table level:***

1. *CONSTRAINT [constraint name] [FOREIGN KEY (foreign key column name) REFERENCES] [referenced table name (referenced column name)]*

***Check constraint***

***Syntax:***

***Column level:***

1. *COLUMN [data type] CONSTRAINT [name] [CHECK (condition)]*

***Table level:***

1. *CONSTRAINT [name] CHECK (condition)*

*SQL> ALTER table DEMO* ***modify (c1 primary key);***

*SQL> ALTER table DEMO* ***drop primary key;***

*SQL> ALTER table DEMO drop* ***unique (c2);***

*SQL> INSERT into EMP values (1,* ***default****)*

*SQL> CREATE table Table2 (ID number, SAL number constraint chk\_sal check (sal>2000) constraint NN\_sal not null);*

*SQL> CREATE table Table2 (ID number, ENAME varchar2 (10) check (regexp\_like (ENAME, '^S')));*

*SQL> CREATE table Table2 (ID number, ENAME varchar2 (10) check (regexp\_like (ENAME,'[[:alpha:]]' )));*

***Foreign Key (On Delete Cascade / On Delete Set Null)***

*SQL> CREATE table Table2 (c1 number references t1* ***on delete cascade****, c2 varchar2 (10));*

*SQL> CREATE table Table2 (c1 number references t1* ***on delete set null****, c2 varchar2 (10));*

*SQL> CREATE table Table2 (c1 number constraint fk\_c1 references Table1, c2 varchar2 (10));*

*SQL> CREATE table Table2 (c1 number constraint fk\_id references Table1* ***on delete cascade,*** *c2 varchar2 (10));*

***A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table with automatically be deleted. This is called a cascade delete in Oracle.***

***Primary Key / Foreign Key***

***Steps***

*SQL> CREATE table TEMP (****no number constraint pk\_no primary key****, c1 number);*

*SQL> CREATE table TEMP (****no number constraint fk\_no references t****, c1 number);*

*SQL> SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, STATUS from user\_constraints where TABLE\_NAME=' TEMP';*

*SQL> SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, STATUS from user\_constraints where TABLE\_NAME='TEMP'*

*SQL> ALTER table TEMP* ***disable constraint pk\_no cascade;***

*SQL> SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, STATUS from user\_constraints where TABLE\_NAME='TEMP';*

*SQL> SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, STATUS from user\_constraints where TABLE\_NAME='TEMP'*

***Note: -*** *It (constraint) would remain disabled and has to be enabled manually using the ALTER TABLE command.*

***Self Reference FOREIGN KEY***

*SQL> CREATE table TEMP (c1 number primary key, c2 number* ***references TEMP);***

*SQL> CREATE table TEMP (c1 number primary key, c2 number* ***constraint foreignkey\_c2 references TEMP);***

*SQL> CREATE table TEMP (c1 number primary key, c2 number* ***constraint foreignkey\_c2 references TEMP (c1));***

***Default variable in PL/SQL***

1. *varSalary number not null := 10;* ***in PL/SQL***
2. *varSalary number default 10;* ***in PL/SQL***

***declare***

***-- Global variables***

num1 number := 95;

num2 number := 85;

begin

dbms\_output.put\_line ('Outer Variable num1: ' || num1);

dbms\_output.put\_line ('Outer Variable num2: ' || num2);

***declare***

***-- Local variables***

num1 number := 195;

num2 number := 185;

begin

dbms\_output.put\_line ('Inner Variable num1: ' || num1);

dbms\_output.put\_line ('Inner Variable num2: ' || num2);

end;

end;

/

***Rollback***

***\* ROLL*** */ ROLLBACK;*

***Types of Subqueries***

***In terms of the placement of the subquery, there are three types:***

***1. Nested Subquery:*** *The subquery appears in the WHERE clause of the SQL. You can nest as many as 255 levels of subqueries in the WHERE clause.*

***2. Inline View:*** *The subquery appears in the FROM clause of the SQL.*

***3. Scalar Subquery:*** *The subquery appears in the SELECT clause of the SQL.*

Find out employees who are not working as MANAGER

*SQL> SELECT \* from EMP where empno not in (SELECT m.empno from EMP E, EMP M WHERE E.MGR = M.EMPNO);*

***Types of Tables (9)***

1. ***Relational tables***
2. ***Object-relational tables***
3. ***Index tables***
4. ***View tables***
5. ***Partitioned tables***
6. ***Temporary tables***
7. ***Dropped tables***
8. ***Clustered tables***
9. ***External tables***

***System Date***

1. *sysdate*
2. *current\_date*
3. *systimestamp*
4. *current\_timestamp*

***CURRENT\_DATE:-*** *Represents the date time from the computer. CURRENT\_DATE returns the current date in the session time zone.*

***SYSDATE: -*** *Represents the date time from your database server. This could be set different from your computer physically hosting the db server.*

*SYSDATE, SYSTIMESTAMP returns the* ***Database's date*** *and timestamp, whereas CURRENT\_DATE, CURRENT\_TIMESTAMP returns the date and timestamp of* ***the location from where you work.***

*SQL> ALTER SESSION SET TIME\_ZONE = '+05:30';*

***Quite simply the number is the precision of the timestamp, the fraction of a second held in the column***

*SQL>* ***CREATE table T1 (ts0 timestamp (0), ts3 timestamp (3), ts6 timestamp (6));***

***Output***

TS0

---------------------------------------------------------------------------

TS3

---------------------------------------------------------------------------

TS6

---------------------------------------------------------------------------

24-JAN-12 05.57.12 AM

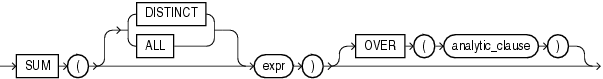
24-JAN-12 05.57.12.003 AM

24-JAN-12 05.57.12.002648 AM

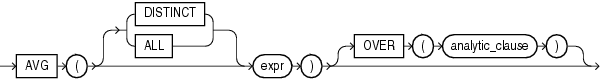
***Aggregate functions***

***Note: If we use aggregate function on table having no records, then it will return NULL***

1. ***SUM( [ALL | DISTINCT/UNIQUE] expression )***



1. ***AVG( [ALL | DISTINCT/UNIQUE] expression )***

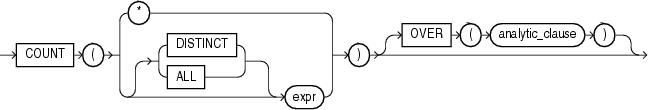


1. ***COUNT( [ALL | DISTINCT/UNIQUE] expression ) // Counts non-null data in***

***// the columns.***

1. ***COUNT(\*) // Counts total number of rows in the table, it includes blank***

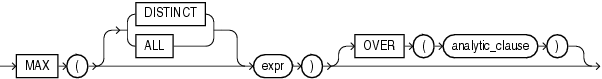
***// (NULL) rows also.***



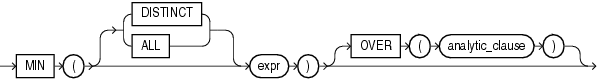
1. ***APPROX\_COUNT\_DISTINCT(expression)***

Description of approx_count_distinct.gif follows

1. ***MAX(expression)***



1. ***MIN(expression)***



***Group by***

***Rules***

* ***GROUP BY clause does not support the use of column alias, but the actual names.***
* ***Aggregate functions cannot be used in a GROUP BY clause.***

*SQL> SELECT \* from DEPT group by deptno, dname, loc, walletid;*

***Window Functions***

1. ***rank() over([partition by cols] order by cols)***
2. ***dense\_rank() over([partition by cols] order by cols)***
3. ***row\_number() over([partition by cols] order by cols)***

*SQL> SELECT \* from (SELECT row\_number () over (partition by job order by job) as "R1", emp.\* from EMP) where to\_number ("R1") = 2;*

*SQL> SELECT row\_number () over (order by null), ename, job, sal from EMP;*

*SQL> SELECT job, sum (sal),* ***dense\_rank() over(order by sum(Sal)) r1*** *from EMP group by job;*

***Obtain a cumulative salary total, row by row, by department***

*SQL> SELECT ename, job, sal, deptno, sum (sal) over (partition by deptno order by job, ename) as "R1" from EMP;*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ENAME** | **JOB** | **SAL** | **DEPTNO** | ***R2*** |
| MILLER | CLERK | 1300 | 10 | ***1300*** |
| CLARK | MANAGER | 2450 | 10 | ***3750*** |
| KING | PRESIDENT | 5000 | 10 | ***8750*** |
| FORD | ANALYST | 3000 | 20 | ***3000*** |
| SCOTT | ANALYST | 3000 | 20 | ***6000*** |
| ADAMS | CLERK | 1100 | 20 | ***7100*** |
| SMITH | CLERK | 800 | 20 | ***7900*** |
| JONES | MANAGER | 2975 | 20 | ***10875*** |
| JAMES | CLERK | 950 | 30 | ***950*** |
| BLAKE | MANAGER | 2850 | 30 | ***3800*** |
| ALLEN | SALESMAN | 1600 | 30 | ***5400*** |
| MARTIN | SALESMAN | 1250 | 30 | ***6650*** |
| TURNER | SALESMAN | 1500 | 30 | ***8150*** |
| WARD | SALESMAN | 1250 | 30 | ***9400*** |

***Cursor in Select command***

*SQL> SELECT DEPT.\*,* ***cursor (SELECT \* from EMP where emp.deptno = dept.deptno)*** *from DEPT;*

*SQL> SELECT location.\*,* ***cursor (SELECT dept1.\*, cursor (SELECT \* from EMP1 where emp1.deptno = dept1.deptno) from DEPT1 where dept1.locationid = location.locationid)*** *from LOCATION;*

*SQL> SELECT count (\*), DEPT.\*, cursor (SELECT EMP3.\*, count (\*) from EMP3 where EMP3.DEPTNO = DEPT.DEPTNO group by EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO, WALLETID, TOTAL) from DEPT group by DEPTNO, DNAME, LOC, WALLETID;*

***Count Records in table***

*SQL> SELECT count (1) from EMP;*

***Type’s of Lock***

**There Are 2 types of Lock**

* SHARE
* EXCLUSIVE

**SHARE: -** SHARE permits concurrent queries but prohibits updates to the locked table.

**\* Eg.**

**Lock table EMP in share mode;**

**EXCLUSIVE**: - EXCLUSIVE permits queries on the locked table but prohibits any other activity on it.

**\* Eg.**

**Lock table EMP in exclusive mode;**

Note: - If an arithmetic expression contains more than one operator, **multiplication and division are evaluated first**, if operators in as expression are of the same priority, then **evaluation is done from left to right**

**1521, 1810, 2481, 7778.** These are just a few of the port numbers Oracle uses in networking.

* ***D:\app\infostaff\product\11.1.0\db\_1\sqlplus\admin\glogin.sql*** (Login.sql file)

***SQL> SVRMGRL***

***D:\> LSNRCTL***

***Demobld7.sql for EMP, DEPT etc. tables***

*Note: - Total number of Primary key given in table is on* ***32 columns***

*Note: - Nested group function without* ***GROUP BY*** *will not work.*

**\* Eg.**

*SELECT* ***max (sum (sal))*** *from EMP;* ***(error)***

***Order by clause.***

*SQL> SELECT* ***distinct (job)*** *from EMP order by sal;* ***(error)***

*SQL> SELECT ename from EMP* ***union*** *SELECT dname from DEPT* ***order by dname;******(error)***

***Rules of Precedence:***

* *Multiplication and division occurs before addition and subtraction*
* *Operators of the same priority are evaluated from left to right*
* *Parentheses are used to override the default precedence or to clarity the statement*

**\* Eg.**

*SELECT* ***12 \* 5 + 100*** *from DUAL;*

**O/P: - 160**

**\* Eg.**

*SELECT* ***12 \* (5 + 100)*** *from DUAL;*

**O/P: - 1260**

***Data Type***

* *A* ***char*** *value can contain up to* ***2000 bytes*** *of* [*data*](http://www.orafaq.com/glossary/faqglosd.htm#Data)
* *A* ***varchar2*** *value can contain up to* ***4000 bytes*** *of* [*data*](http://www.orafaq.com/glossary/faqglosd.htm#Data)*,* ***and in PL/SQL 32767***
* *Date data type used to store* ***date and time*** *values in a* ***7-byte*** *structure.*

***SQL Statements***

|  |  |
| --- | --- |
| *SELECT (pick out, choose, take)* | *Data query language (DQL)* |
| *INSERT (to put, to place, Include)*  *UPDATE (adding new information or making corrections)*  *DELETE (to strike out, remove, cancel, erase)*  *MERGE* | *Data manipulation language (DML)* |
| *CREATE (to develop, to build)*  *ALTER (to change, to renovate, to recast, to revise)*  *DROP*  *RENAME*  *TRUNCATE*  *COMMENT* | *Data definition language (DDL)* |
| *COMMIT (to act, to carry out, to complete)*  *ROLLBACK (to undo, to reverse an action)*  *SAVEPOINT*  *SET TRANSACTION* | *Transaction control language (DTL / TCL)* |
| *GRANT*  *REVOKE* | *Data control language (DCL)* |

***Capabilities of SQL SELECT Statements***

***Selection: -*** *You can use the selection capability in SQL to choose the rows in a table that you want to return by a query. You can use various criteria to restrict the row that you see.*

|  |  |  |  |
| --- | --- | --- | --- |
| ***EMPNO*** | ***ENAME*** | ***JOB*** | ***HIREDATE*** |
| *1* | *Saleel* | *Manager* | *01-Jan-95* |
| ***2*** | ***Arati*** | ***Sales*** | ***20-Dec-94*** |
| *3* | *Snehal* | *Manager* | *21-May-97* |
| ***4*** | ***Rahul*** | ***Account*** | ***30-Jul-97*** |
| *5* | *Ketan* | *Sales* | *01-Jan-94* |

***Projection: -*** *You can use the projection capability in SQL to choose the columns in a table that you want to return by your query. You can choose as few or as many columns of the table as you required.*

|  |  |  |  |
| --- | --- | --- | --- |
| ***EMPNO*** | ***ENAME*** | ***JOB*** | ***HIREDATE*** |
| ***1*** | *Saleel* | ***Manager*** | *01-Jan-95* |
| ***2*** | *Arati* | ***Sales*** | *20-Dec-94* |
| ***3*** | *Snehal* | ***Manager*** | *21-May-97* |
| ***4*** | *Rahul* | ***Account*** | *30-Jul-97* |
| ***5*** | *Ketan* | ***Sales*** | *01-Jan-94* |

***Joining: -*** *You can use the join capability in SQL to bring together data that is stored in different tables by creating a link between them.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***EMPNO*** | ***ENAME*** | ***JOB*** | ***HIREDATE*** | ***DEPTNO*** |
| *1* | *Saleel* | *Manager* | *01-Jan-95* | ***10*** |
| *2* | *Arati* | *Sales* | *20-Dec-94* | ***20*** |
| *3* | *Snehal* | *Manager* | *21-May-97* | ***10*** |
| *4* | *Rahul* | *Accountant* | *30-Jul-97* | ***30*** |
| *5* | *Ketan* | *Sales* | *01-Jan-94* | ***20*** |

|  |  |
| --- | --- |
| ***DEPTNO*** | ***DNAME*** |
| ***10*** | *Purchase* |
| ***20*** | *Sales* |
| ***30*** | *Account* |
| ***40*** | *HRD* |

**Parent Child Table**

|  |  |
| --- | --- |
| ***Parent ID*** | ***Child Id*** |
| ***1*** | *2* |
| ***1*** | *3* |
| ***2*** | *4* |
| ***2*** | *5* |
| ***3*** | *6* |
| ***3*** | *7* |
| ***3*** | *8* |
| ***8*** | *9* |

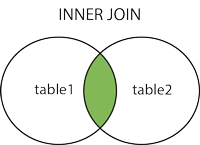
**Arithmetic Operators**

|  |  |  |
| --- | --- | --- |
| ***Precedence Level*** | ***Operator Symbol*** | ***Operation*** |
| *Highest* | *( )* | *Brackets or parentheses* |
| *Medium* | */* | *Division* |
| *Medium* | *\** | *Multiplication* |
| *Lowest* | *-* | *Subtraction* |
| *Lowest* | *+* | *Addition* |

***Join Types***

***A join is a query that combines rows from two or more tables or views.***

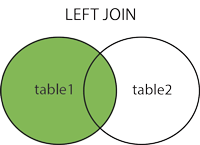
* ***Cartesian product Join / Cross Join*** *(Table with less record in the table will be compared with the records with more record in the table.* ***To prove this use autotrace on and rowid (block) of the rows in the table.)***
* ***Equiv Join / Inner Join (Simple Join)***



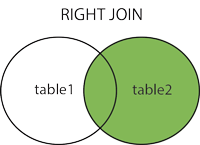
* ***Outer Join (Left, Full, Right)***

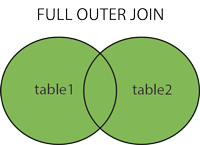
*Note :- outer join operator (+) not allowed for* ***[ OR or IN ]*** *operators.*

***LEFT OUTER JOIN:*** *(To write a query that performs an outer join of tables Table1 and Table1 and returns all rows from Table1 (a left outer join), use the LEFT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of Table2 in the join condition in the WHERE clause.)*

**

***RIGHT OUTER JOIN:*** *(To write a query that performs an outer join of tables Table1 and Table2 and returns all rows from Table2 (a right outer join), use the RIGHT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of Table1 in the join condition in the WHERE clause.)*

**

**

***Eg.***

*SQL> SELECT col1, col2, col3 c3, col4 c4,* ***nvl2 (to\_char (col3),'True','False') col3, nvl2 (to\_char (col4),'True', 'False') col4*** *from* ***TABLEA LEFT JOIN TABLEB*** *on col2=col4 order by col3*

* ***Self Joins*** *(A self join is a join of a table to itself. This table appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. To perform a self join, Oracle Database combines and returns rows of the table that satisfy the join condition.)*
* ***Natural Join*** *(A natural join offers a further specialization of equi-joins. The join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables. The resulting joined table contains only one column for each pair of equally-named columns.)*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Table A*** | | ***Table B*** | |
| ***Col1*** | ***Col2*** | ***Col3*** | ***Col4*** |
| *1* | *2* | *5* | *7* |
| *3* | *4* | *3* | *8* |
| *5* | *6* | *1* | *4* |
| *7* | *8* | *8* | *7* |
| *9* | *10* |  | *10* |
| *11* | *12* |  |  |

***What will be the output of the following?***

* *SELECT Col1, Col4 from A, B where Col1= Col3;*
* *SELECT Col3, Col4 from A, B;*
* *SELECT Col1, Col4 from B, A;*
* *SELECT \* from A union SELECT \* from B;*
* *SELECT Col1, Col4 from A, B where Col2 = Col4;*

***The UNION [ALL], INTERSECT, MINUS Operators***

*If a SQL statement contains multiple set operators, then Oracle Database evaluates them from the* ***left to right unless parentheses*** *explicitly specify another order.*

*The output in sub sets will be in* ***ascending order always****.*

*SQL> SELECT ename from EMP* ***union*** *SELECT dname from dept* ***order by dname;******(error)***

**Eg.**

*SQL> SELECT 2 col1, 'y' col2 FROM DUAL* ***UNION*** *SELECT 1, 'x' FROM DUAL* ***UNION*** *SELECT 3, NULL from DUAL ORDER BY 2*;

COL1 C

---------- -

1 x

2 y

3

**Eg.**

*SQL> SELECT COL4 from TableB* ***intersect*** *SELECT COL2 from TableA* ***minus*** *SELECT COL1 from TableA*

COL4

--------

4

8

10

* ***Outer Join (Left, Full, Right)***

*SQL> SELECT \* from EMP* ***left join*** *DEPT on EMP.deptno = DEPT.deptno (+);*

*SQL> SELECT \* from EMP* ***right join*** *DEPT on EMP.deptno (+) = DEPT.deptno;*

*SQL> SELECT \* from DEPT d* ***left outer join*** *(SELECT deptno from EMP where deptno=10) s on d.deptno = s.deptno;*

*SQL> SELECT \* from DEPT d* ***left outer join*** *(SELECT deptno from EMP where deptno=10) s using (deptno)*

* ***Natural Join***

***NATURAL [ { LEFT | RIGHT } [ OUTER ] | INNER ] JOIN { TableViewOrFunctionExpression | ( TableExpression ) }***

*SQL> SELECT \* from DEPT* ***natural left outer join*** *EMP;*

***Data Types***

|  |  |
| --- | --- |
| ***Data Types*** | ***Size*** |
| *char* | *Size optional 2000 char* |
| *varchar2* | *Size compulsory 4000 char*  *32767 char in oracle r12c* |
| *number* | *Size optional (default size 38)* |
| *number****(\*,1)*** | Oracle guarantees portability of numbers with a precision equal to or less than 38 digits. You can specify a scale and no precision: |
| *int / integer* | *Size not to be given (default size 38)* |
| *float* | *Size not to be given (default size 126)* |
| *dec* | *Size optional (38 auto roundup)* |
| *double PRECISION* | *create table t(c1 double PRECISION);* |
| *date* | *Size not to be given 7 bit*  *(century, year, month, day, hours, minutes and seconds)*  *Jan 1, 4712 B.C to Dec 31, 9999 A.D* |
| *timestamp* |  |
| *long* | *2GB* |
| *user* | *Size not to be given* |
| *rowid* | *Size not to be given* |
| *simple\_integer (PL/SQL)* | *Cannot be keep null (Assign value).* |
| *natural (PL/SQL)* | *Nonnegative* |
| *naturaln (PL/SQL)* | *Nonnegative Cannot be keep null (Assign value).* |
| *positive (PL/SQL)* | *Positive* |
| *positiven (PL/SQL)* | *Positive Cannot be keep null (Assign value).* |
| *pls\_integer (PL/SQL) Range 1..100* |  |
| *SUBTYPE x is varchar2 (100);* | *y x;*  *x datatype of assigned to variable y* |
| *Constant variable* | *X CONSTANT NUMBER (3) := 2;*  *x CONSTANT NUMBER (8,7) DEFAULT 3.14;* |

***Tablespace***

* *OnLine*
* *OffLine*
* *Read Only*
* *Read Write*

***OffLine: -*** *Making a tablespace offline prevents (SELECT, INSERT, UPDATE, DELETE, ALTER (DROP, MODIFY),* ***ALTER (ADD, RENAME) will work****.*

***Read Only: -*** *Making a tablespace read-only prevents updates on all tables in the tablespace, regardless of a user's update privilege level.*

a) *Create tablespace InfoDataFile*

*Datafile 'InfoDataFile.dat' size 1m* ***AUTOEXTEND ON***;

Note: - The integer can be followed by **B, K, M, or G (indicating bytes, kilobytes, megabytes, and gigabytes respectively).** Byte is the default.

b) Select tablespace\_name from DBA\_TABLESPACES / USER\_TABLESPACES;

c) Alter tablespace InfoDataFile offline / online (Lock Tablespace) / (Unlock Tablespace)

d) Select table\_name, tablespace\_name from user\_tables; (see, Where Table Is Stored)

e) Alter tablespace InfoDataFile ADD datafile ‘InfoDataFile2’; (Add’s new datafile to existing tablespace)

f) ALTER TABLESPACE my\_space RENAME TO your\_space;

**\*** Create user sharmin identified by sharmin default tablespace InfoDataFile;

**\*** Alter user saleel default tablespace InfoDataFile (To attach tablespace to the user)

**\*** Drop tablespace info2; (If, ‘USER’ is dropped [using cascade option] the tablespace is not removed)

**\*** Drop tablespace info2 including contents;

**\*** Drop tablespace info2 including contents cascade constraints;

**\*** Drop tablespace info2 including contents and datafiles;

**\*** ALTER TABLESPACE InfoDataFile READ ONLY; (To make the tablespace Readonly.)

**\*** ALTER TABLESPACE InfoDataFile READ WRITE; (To make the tablespace Readonly.)

**\*** DBA\_TABLESPACES / USER\_TABLESPACES (Oracle Data Dictionary)

***Exporting files (CommandLine)***

**\*** Exp saleel/sharmin@orcl file=file1.dmp tables = (saleel.emp, saleel.dept)

**\*** Exp saleel/sony@orcl file=d:\file1.dmp tables=saleel.emp query=\ "WHERE job='MANAGER' \"

***Exporting oracle data into Excel files***

SQL> Spool emp.csv;

SQL> Select EMPNO || ‘,’ || ENAME || ‘,’ || JOB from EMP;

SQL> Open emp.csv in Excel.exe

***Importing FILE (CommandLine)***

**\*** Imp saleel/sharmin@orcl File=file1.dmp fromuser=saleel touser=saleel rows=y

***Importing FILE from Excel***

**\*** Create an Excel File in “C:\LoadData.csv”

**\*** Create a CONTROL File in “C:\ ControlFile.ctl”

In that

Load data

Infile ‘c:\ LoadData.csv’

Into table <Table NM>

Fields terminated by “,”

(Empno, ename, job,...,)

**Execute .CTL file**

**\*** C:\> sqlldr Username/Password@hoststring control=C:\ControlFile.ctl

***Connect by***

*SELECT* ***sysdate + rownum*** *as "New Date" from* ***DUAL connect by level <=10;***

New Date

------------------------------

29-Mar-2011 15:40:54

30-Mar-2011 15:40:54

31-Mar-2011 15:40:54

01-Apr-2011 15:40:54

02-Apr-2011 15:40:54

03-Apr-2011 15:40:54

04-Apr-2011 15:40:54

05-Apr-2011 15:40:54

06-Apr-2011 15:40:54

07-Apr-2011 15:40:54

10 rows selected.

*SELECT to\_char (sysdate, 'hh: mi: ss’) as "R1",* ***to\_char (sysdate + rownum/24, 'hh: mi: ssam’) as "R2” from*** *DUAL* ***connect by level <= 24;***

R1 R2

------------ ----------------

02:38:32 03:38:32pm

02:38:32 04:38:32pm

02:38:32 05:38:32pm

02:38:32 06:38:32pm

02:38:32 07:38:32pm

02:38:32 08:38:32pm

02:38:32 09:38:32pm

02:38:32 10:38:32pm

02:38:32 11:38:32pm

02:38:32 12:38:32am

02:38:32 01:38:32am

02:38:32 02:38:32am

02:38:32 03:38:32am

02:38:32 04:38:32am

02:38:32 05:38:32am

02:38:32 06:38:32am

02:38:32 07:38:32am

02:38:32 08:38:32am

02:38:32 09:38:32am

02:38:32 10:38:32am

02:38:32 11:38:32am

02:38:32 12:38:32pm

02:38:32 01:38:32pm

02:38:32 02:38:32pm

24 rows selected.

***SEQUENCE***

**\*** *CREATE sequence S1*

*start with 5*

*increment by 1*

*maxvalue 10*

*minvalue 4*

*cycle*

*cache 2;*

***The cache option specifies how many sequence values will be stored in memory for faster access.***

*Note: - START WITH cannot be less than MINVALUE.*

*Note: - If MINVALUE not given then cycle will start from 1.*

*Note: - START WITH only for the first time will start’s then it starts with MINVAUE.*

***NOTE: - You cannot alter starting sequence number***

***EG.***

*SQL>* ***ALTER sequence S1 START WITH 1;*** *//* ***START WITH cannot be altered.***

***Session wise Sequence***

*SQL> CREATE sequence S1* ***SESSION****;*

***EG.***

*SQL> CREATE table TEMP*

*(COL1 NUMBER* ***DEFAULT S1.NEXTVAL,***

*COL2 NUMBER* ***DEFAULT ON NULL S2.NEXTVAL,***

*ENAME VARCHAR2 (10));*

***In PL block***

***EG.***

*X number;*

*begin*

***X := s1.nextval;***

*dbms\_output.put\_line (X);*

*end;*

/

***SYNONYM***

*CREATE SYNONYM statement to create a synonym, which is an alternative name for a* ***table, view, sequence, procedure, stored function, package, materialized view, Java class schema object, user-defined object type, or another synonym.***

***Syntax:***

*CREATE [PUBLIC] SYNONYM [synonym name]*

*FOR OBJECT;*

*SQL>* ***CREATE synonym E for EMP;***

*SQL>* ***CREATE public synonym E;***

***VIEWS***

***Syntax:***

*CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW [ViewName]*

*[(Column Alias Name...)]*

*AS [Query]*

*[WITH [CHECK OPTION] [READ ONLY] [CONSTRAINT]];*

***A VIEW is a logical representation of one or more tables. A view contains no data itself. The tables upon which a view is based are called******base tables.***

***A VIEW is a predefined, named query stored in the database.***

*SQL> CREATE or* ***REPLACE******force*** *view V1 as SELECT \* from EMP;*

*SQL> ALTER view V1* ***compile;***

***user\_updatable\_columns*** ***/\* USER\_UPDATABLE\_COLUMNS describes columns in a join view that can be updated by the current user*** \****/***

*View with (****with check option)***

***Note: - INSERT, UPDATE AND DELETE on view (with check option given on view) will work only when the*** DATA MATCHES IN WHERE CLAUSE.

*SQL>* *CREATE or REPLACE view V1 as SELECT \* from EMP* ***where deptno = 10 with check option;***

*SQL> CREATE or REPLACE view V1 as SELECT \* from EMP* ***where deptno=10 with check option;***

*Insert with Check option*

*SQL> INSERT into V1 (empno, deptno) values (1, 20);*

*INSERT into V1 (empno, deptno) values (1, 20)*

*\**

*ERROR at line 1:*

*ORA-01402: view WITH CHECK OPTION where-clause violation*

*View with (with read only)*

***Note: - Cannot perform a DML operation on a read-only view.***

*SQL> CREATE or REPLACE view V1 as SELECT \* from EMP* ***with read only;***

***If a VIEW is created by giving following statement then***

*SQL> CREATE or REPLACE view V1 as SELECT \* from DEPT* ***where deptno=10;***

*SQL> DELETE from V1* ***where deptno=20;***

***0 rows deleted.***

*SQL> UPDATE V1 set loc='ABC'* ***where deptno=20;***

***0 rows updated.***

***0 rows updated message will come because the condition in where clause of DELETE and UPDATE is not matching with the condition given to create a VIEW***

***Replaceable view***

*SQL> CREATE or REPLACE view V1 as SELECT \* from EMP;*

*SQL> SELECT view\_name, text from USER\_VIEWS;*

*SQL> CREATE or REPLACE view V1 as SELECT \* from DEPT;*

*SQL> SELECT view\_name, text from USER\_VIEWS;*

***INDEXES***

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. **By default, Oracle creates B-tree indexes.**

1. *Indexes are optional structures associated with tables.*
2. *An index is also a schema object.*
3. *We can drop an index without dropping the table it indexes.*
4. *An index can be created explicitly or automatically.*
5. *When we drop a table the corresponding indexes of the table are dropped.*
6. *A unique index gets created when we create a unique key or primary key in a table definition.*
7. *The name of the index is the name of the constraint.*
8. *Indexes can be unique or non-unique. Unique indexes guarantee that no two rows of a table have duplicate values in the columns that define the index. Non-unique indexes do not impose this restriction on the column values.*
9. *The presences of more number of indexes on a table decreases the performance of DML statements, because Oracle must make changes to the indexes associated with the table*

*SQL> SELECT empno from EMP;* ***// will use EMP\_PRIMARY\_KEY index file***

***Index Explicitly***

*SQL>CREATE* ***index*** *emp\_ENAME on EMP (ename)* ***TABLESPACE tSpace1;***

***Unique Index Explicitly***

*SQL> CREATE* ***UNIQUE******index*** *DEPT\_UNIQUE\_INDEX on DEPT (dname)* ***TABLESPACE tSpace1;***

***Invisible Index***

*SQL> CREATE index EMP\_ENAME on EMP (ename) TABLESPACE tSpace1* ***INVISIBLE****;*

Note: - An invisible index is an index that is maintained by the database but ignored by the optimizer unless explicitly specified.

*SQL> ALTER index INDEX\_NAME* ***INVISIBLE;***

*SQL> ALTER index INDEX\_NAME* ***VISIBLE;***

*SQL>* ***ALTER session set optimizer\_use\_invisible\_indexes = true;***

***Function based index***

*SQL> CREATE index EMP\_ENAME on EMP* ***(UPPER (ename));***

***Enable / Disable Function based index***

*SQL> ALTER index EMP\_ENAME* ***Enable / Disable;***

***Listagg***

*SQL> SELECT deptno,* ***listagg (ename, ',') WITHIN GROUP (ORDER BY ename)*** *enames from EMP group by deptno;*

*SQL> SELECT deptno,* ***listagg (ename ||' (' || job || ')', ';') within group (order by job, name) as "R1"*** *from EMP group by deptno*

***With clause***

*SQL>* ***WITH***

***A as*** *(SELECT sum (sal) as "R1" from EMP where deptno=10),*

***B as*** *(SELECT sum (sal) as "R2" from EMP where deptno=20),*

***C as*** *(SELECT "R2" - "R1" from A, B)*

*SELECT \* from* ***C***

*SQL>* ***WITH***

***A as*** *(SELECT max (count (\*)) as "R1" from EMP group by deptno),*

***B as*** *(SELECT deptno, count (deptno) as "R2" from EMP group by deptno)*

*SELECT deptno,"R1","R2" from* ***A, B*** *group by deptno,"R1","R2" having "R2" = "R1"*

*SQL> SELECT (****SELECT max (sal) from EMP****), (SELECT max (sal) from EMP) - sal from EMP where deptno=10*

***WITH clause for PL/SQL***

***EG.***

***WITH***

*function F1 (x number, y number) return number is*

*begin*

*return (x + y);*

*end F1;*

*function F2 (x1 number, y1 number) return number is*

*begin*

*return (F1 (10, 10) + x1 + y1);*

*end F2;*

*SELECT F2 (1, 1) from DUAL*

*/*

***Inline View Table***

*SQL> SELECT "R1" - "R2" from (SELECT sum (sal) as "R1" from EMP where deptno = 10)* ***TableA****, (SELECT sum (sal) as "R2" from EMP where deptno = 20)* ***TableB****;*

*SQL> SELECT rownum,* ***e.\**** *from (SELECT rownum as "R1", emp.\* from EMP order by job) as* ***e;***

*SQL> SELECT rownum, e.\* from (SELECT rownum as "R1”, emp.\* from EMP)* ***e where e.deptno=10;***

***Pivot***

***It is used to display the output in columnar format.***

*SQL****>*** *SELECT \* from (SELECT deptno, sal, job from EMP)* ***pivot*** *(sum (sal) for deptno in (10, 20, 30));*

*SQL****>*** *SELECT \* from (SELECT deptno from EMP)* ***pivot*** *(count (deptno) for deptno in (10, 20, 30));*

*SQL****>*** *SELECT \* from EMP* ***pivot*** *(sum (sal) for deptno in (10, 20, 30));*

***Case***

Note: - IN, LIKE, and BETWEEN SQL operators can be used with cases.

*SQL> SELECT* ***case when ename like 'A%' then 'data1' end case*** *from EMP;*

*SQL> SELECT deptno,* ***case when deptno in (10, 20) then 'data1' end case*** *from EMP;*

*SQL> SELECT sal,* ***case when sal between 1000 and 3000 then 'data1' end case*** *from EMP;*

*SQL> SELECT job, deptno,* ***case when deptno = 30 then******(case when job ='MANAGER' then 'Good' end)******end*** *from EMP;*

**Case in SELECT where clause**

*SQL> SELECT \* from EMP* ***where (case when :x = 100 and :y=100 then deptno end) = 20;***

**Case in UPDATE**

*SQL> UPDATE EMP*

*set job = case deptno*

*when 10 then 'a'*

*when 20 then 'b'*

*when 30 then 'c'*

*end;*

*SQL> UPDATE EMP*

*set job = case*

*when deptno=10 then 'a'*

*when deptno = 20 then 'b'*

*when deptno = 30 then 'c'*

*end;*

*SQL> UPDATE EMP Set comm = case when deptno=10 then 1000 else comm End;*

*SQL> UPDATE (SELECT \* from EMP2 where empno in (5,10,15,20,25,30,35,40,45,50,55)) set comm=1;*

*SQL> UPDATE (SELECT \* from EMP2 where empno in(5,10,15,20,25,30,35,40,45,50,55)) set comm =*

*case deptno*

*when 10 then 1*

*when 20 then 2*

*when 30 then 3*

*else 4*

*end;*

*SQL> UPDATE EMP set sal = - 1 where (empno, 0) in (SELECT empno, mod (rownum, 2) from EMP); // (UPDATES all even records)*

*SQL> SELECT ename, job, deptno,*

*case deptno*

*when 10 then 'ACCOUNTING'*

*when 20 then 'RESEARCH'*

*when 30 then 'SALES'*

*end*

*, case job*

*when 'MANAGER' then 'M'*

*when 'CLERK' then 'C'*

*when 'SALESMAN' then 'S'*

*when 'PRESIDENT' then 'P'*

*when 'ANALYST' then 'A'*

*end from EMP;*

*SQL> SELECT rownum, case when rownum between 1 and 5 then '1 to 5' when rownum between 6 and 10 then '6 to 10' else 'Remaining' end "R1" from EMP;*

*SQL>**SELECT case when sum (sal) > 5500 then sum (sal) else 100 end “Salary”, job from EMP group by job;*

Note: - The WHEN clause can be repeated for 128 times only and ELSE is optional

Note: - In / like / between predicates can be used in cases.

***boolean datatype***

***Note:-*** *The only value that you can assign to a BOOLEAN variable is a BOOLEAN expression. You cannot pass a BOOLEAN value to the DBMS\_OUTPUT.PUT or DBMS\_OUTPUT.PUTLINE subprogram.* ***To print a BOOLEAN value, use an IF or CASE statement to translate it to a character value.***

***EG.***

*CREATE or REPLACE procedure PL1 (****b BOOLEAN****)*

*AS*

*begin*

***DBMS\_OUTPUT.put\_line (***

***CASE***

***WHEN b IS NULL THEN 'Unknown'***

***WHEN b THEN 'Yes'***

***WHEN NOT b THEN 'No'***

***END***

***);***

*end PL1;*

*/*

**How to Call**

*begin*

***pl1 (TRUE);***

***pl1 (FALSE);***

***pl1 (NULL);***

*end;*

*/*

***Update command***

*SQL> UPDATE EMP* ***set walletid = (SELECT dept.walletid from dept where emp.deptno = dept.deptno);***

**Multiple column update**

*SQL> UPDATE EMP* ***set (sal, comm) = (SELECT -1,-2 from DUAL);***

***EG.***

*declare*

*TYPE DEPTRecord is RECORD (*

*DEPTNO NUMBER (2),*

*DNAME VARCHAR2 (10),*

*LOC VARCHAR2 (10),*

*WALLETID NUMBER (5)*

*);*

*xRecord DEPTRecord;*

*begin*

***xRecord.loc := -1;***

*for i in 1..30*

*loop*

***xRecord.DEPTNO := i;***

***UPDATE DEPT set Row = xRecord where DEPTNO = i;***

*end loop;*

*end;*

*/*

***Kill Session***

*SQL> ALTER system* ***kill session*** *'8, 1245’* ***/\* '<sid>, <serial#>'***

*SQL> ALTER system* ***disconnect session*** *'8, 1245’* ***post\_transaction****;*

***Session***

*SQL>**ALTER SESSION SET SQL\_TRACE = True / False*

*SQL>**ALTER SESSION SET NLS\_DATE\_FORMAT = 'dd-mm-yy’ (National Language Support)*

*SQL>**ALTER SESSION SET NLS\_LANGUAGE= AMERICAN;*

*SQL>**ALTER SESSION SET NLS\_CURRENCY= ‘Rs.’;*

*SQL>**ALTER SESSION SET CURRENT\_SCHEMA = <other schema name>;*

*SQL>**SELECT \* from NLS\_DATABASE\_PARAMETERS;*

*SQL>**SELECT \* from NLS\_SESSION\_PARAMETERS;*

***TOP N Row***

*SQL>**SELECT \* from (SELECT \* from EMP order by sal desc) where rownum < 3;*

***Or***

*SQL>**SELECT ename, job, sal from EMP e where 3 > (SELECT count (\*) from EMP where e.sal < sal)*

***First 3 Highest***

*SQL>**SELECT job, Sal from (****SELECT job, Sal from EMP group by job, Sal order by sal desc****) where rownum <=3;*

*SQL>**SELECT \** ***from (SELECT emp.\*, dense\_rank () over (order by sal desc) as "R1" from EMP) where R1<=3;***

***Only First Highest Salary Row***

*SQL>**SELECT \* from EMP where* ***sal = (SELECT sal from (select rownum as "R1", sal from EMP order by sal desc) where "R1" = 1)***

*SQL>**SELECT \* from EMP where* ***sal = (SELECT max (sal) from EMP);***

*SQL>**SELECT \* from* ***(SELECT row\_number () over (order by sal desc) as "R1", emp.\* from EMP) where "R1" = 1***

***Second Highest Salary onwards***

*SQL>**SELECT \* from (****SELECT row\_number () over (order by sal desc) as "R1", sal from EMP group by sal****) where "R1"=3*

*SQL>**SELECT \* from (****SELECT dense\_rank () over (order by sal desc) as "R1", sal from EMP****) where "R1"=4*

***Grouping Sets***

*SQL>**SELECT job, deptno, sum (Sal), min (sal), max (Sal) from EMP* ***group by grouping sets ((job, deptno), (deptno, mgr));***

***Rollup***

*SQL>**SELECT deptno, job, sum (sal) from EMP* ***group by rollup (deptno, job)***

***Cube***

*SQL>**SELECT deptno, job, sum (sal) from EMP* ***group by cube (deptno, job)***

***Alter Table Rename***

*SQL>**ALTER table EMP* ***RENAME column old\_name to new\_name;***

*SQL>**ALTER table EMP* ***RENAME constraint old\_name to new\_name;***

*SQL>**ALTER table EMP* ***RENAME to NEWEMP;*** (Table rename)

*SQL>**ALTER table EMP* ***RENAME PARTITION cPart2 to NewName;***

***Alter Table Drop***

*SQL>**ALTER table EMP* ***DROP column hiredate;***

*SQL>**ALTER table EMP* ***DROP (mgr, hiredate);*** (Multiple columns)

***Visible / Invisible columns***

***Following operations don’t see invisible columns***

* *SELECT \* FROM statements in SQL*
* *DESCRIBE commands in SQL\*Plus*
* *%ROWTYPE attribute declarations in PL/SQL*
* *Describes in Oracle Call Interface (OCI)*

***Any columns can be made invisible if records are present.***

***SQL> set colinvisible on***

*SQL>**ALTER table EMP* ***modify sal visible;***

*SQL> ALTER table EMP* ***modify sal invisible;***

*SQL> CREATE table NEW (C1 number generated always as identity, C2 number, NEW varchar2 (10), OLD varchar2 (10), C3 number invisible, C4 number invisible encrypt);*

***SHOW INVISIBLE COLUMNS***

*SQL> SELECT COLUMN\_NAME, HIDDEN\_COLUMN, VIRTUAL\_COLUMN, DEFAULT\_ON\_NULL, IDENTITY\_COLUMN from* ***USER\_TAB\_COLS*** *where table\_name='NEW';*

***DBMS\_RANDOM***

***Note: - dbms\_random package can be used in where clause.***

*SQL>**SELECT dbms\_random.NORMAL () from DUAL;* (Will, generate Random No.)

*SQL>**SELECT round (dbms\_random.VALUE (1, 10)) from DUAL;* (Will, generate Random No. between 1 to 10)

*SQL>**SELECT dbms\_random.VALUE () from DUAL;* (Will, generate Random No.)

*SQL>**SELECT dbms\_random.STRING ('1', 20) STR from DUAL;* (Will, generate Random Char.)

***(u -> upper, l ->lower, a ->upper & lower, x ->alpha & number, p ->any printable chars)***

***EXIT***

*SQL> EXIT [success|failure|warning|n|variable| :bindvariable]*

*[COMMIT | ROLLBACK]*

***dbms\_transaction***

***Commit / Rollback are used to remove the transaction handle.***

*SQL > exec dbms\_transaction.COMMIT;*

*SQL > exec dbms\_transaction.ROLLBACK;*

*SQL> exec dbms\_transaction.SAVEPOINT ('S1');*

*SQL> exec dbms\_transaction.ROLLBACK\_SAVEPOINT ('S1');*

*SQL> exec dbms\_transaction.READ\_ONLY;*

*SQL> exec dbms\_transaction.READ\_WRITE;*

***Define Editor***

*DEFINE \_EDITOR = notepad*

***DISABLE '&'***

*Set define on / off / c*

*SQL> SELECT 'You & me' FROM DUAL;*

*SQL> SELECT 'You '|| chr (38) ||' me' FROM DUAL*

***SET Options***

*SQL> Set linesize 100*

*SQL> Set pagesize 40*

*SQL> Set colsep “ “*

*SQL> Set verify off (will not display old line when we user “&” sign)*

*SQL> Set markup html on 🡪 spool on 🡪 spool a.html*

***Set Transaction***

***Commit / Rollback are used to remove the transaction handle.***

*SQL>* ***Set transaction read only;***

*(will not perform insert/delete/update operation inside a READ ONLY transaction)*

*SQL>* ***Set transaction read write;***

*(Will perform insert/delete/update operation inside a READ WRITE transaction)*

***& and &&***

**\* *'&'*** *is used to create a temporary substitution variable and will prompt you for a value every time it is referenced.*

**\* *'&&'*** *is used to create a permanent substitution variable as with the DEFINE command and the* ***OLD\_VALUE*** *or* ***NEW\_VALUE*** *clauses of a COLUMN statement.*

***Copy***

Copy from scott/tiger@local\_db to scott/tiger@remote\_db \_

Create image\_table using select image\_no, image from images;

***NVL2***

*SQL> SELECT* ***nvl2 (comm, 100, 200)*** *from EMP where empno=7654;*

Output

NVL2 (COMM, 100, 200)

--------------------------------

100

**Note: - If COMM IS NULL then will return 200, else if COMM IS NOT NULL then will return 100.**

**Note: - Only the functions CONCAT, DECODE, DUMP, NVL, NVL2 and REPLACE can return non-NULL values when called with a NULL argument.**

***COALESCE***

*SQL> SELECT sal, mgr, comm,* ***coalesce (sal, mgr, comm)*** *from EMP;*

***EG.***

*SQL> SELECT* ***coalesce (address1, address2, address3) as result*** *from EMP;*

The above **coalesce** statement is equivalent to the following **IF-THEN-ELSE** statement:

*IF address1 is not null THEN*

*result := address1;*

*ELSIF address2 is not null THEN*

*result := address2;*

*ELSIF address3 is not null THEN*

*result := address3;*

*ELSE*

*result := null;*

*END IF;*

**Note: - The coalesce function will compare each value, one by one.**

***Decode***

***Syntax: DECODE (value, if1, then1, if2, then2 if3, then3, . . . . . . else)***

*SQL> SELECT count (a1) as "Manger Count”, count (a2) as "Salesman Count" from (SELECT decode (job, 'MANAGER', 1) as a1, decode (job, 'SALESMAN', sal) as a2 from EMP)*

***Nullif***

*SQL> SELECT length (ename), length (job),* ***nullif (length (ename), length (job))*** *from EMP1;*

**Note: - The NULLIF function compares two expression. If they are equal, the function returns null.**

***CACHE Table***

***CACHE*** *For data that is accessed frequently, this clause indicates that the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This attribute is useful for small lookup tables.You cannot specify CACHE for an index-organized table. However, index-organized tables implicitly provide CACHE behavior.*

***NOCACHE*** *For data that is not accessed frequently, this clause indicates that the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed. NOCACHE is the default for LOB storage.*

***EG.***

*SQL> CREATE table TEMP (No VARCHAR2 (20))* ***CACHE****;*

*SQL> ALTER table TEMP* ***nocache;***

***Row Movement for (HWM)***

***EG.***

*CREATE table EMP (no number)* ***enable row movement;***

***Global Temporary Table***

* ***on commit delete rows (default)***
* ***on commit preserve rows***

***Note: - The data in a global temporary table is PRIVATE***

***Note: - Each user sees only his or her rows in the table.***

***EG.***

***CREATE GLOBAL TEMPORARY table TEMP***

***(No number) on commit delete rows*** (Record will be deleted on commit)

***EG.***

***CREATE GLOBAL TEMPORARY table TEMP***

***(No number) on commit preserve rows*** (Record will be delete when the session is closed)

***EG.***

***CREATE GLOBAL TEMPORARY table TEMP on commit preserve rows as SELECT \* from EMP;***

***Steps (Global Temporary Table)***

**SQL> Create global temporary table temp (no number);**

**SQL> INSERT into TEMP values (1);**

**SQL> INSERT into TEMP values (2);**

**SQL> INSERT into TEMP values (3);**

**SQL> INSERT into TEMP values (4);**

**SQL> INSERT into TEMP values (5);**

**SQL> CREATE table TEMP as SELECT \* from EMP;**

***External Table***

**Step 1:**

**Create a folder in 'C:\Temp\_DIR'**

The **dataFile**.**txt** text file contains the following rows of information:

1,Saleel Bagde,1000

2,Vrushali Bagde,2000

3,Sharmin Bagde,3000

**and save in Temp\_DIR Folder.**

**Step 2:**

**SQL> Create directory Temp\_DIR as 'C:\Temp\_DIR ';**

**SQL> Grant read, write on directory Temp\_DIR to saleel;**

**Step 3:**

**Create table indata1**

**(**

**Cust\_id number,**

**Cust\_name varchar2 (20),**

**Credit\_limit number (10)**

**)**

**organization external**

**(**

**Type oracle\_loader**

**Default directory Temp\_DIR**

**Access parameters**

**(**

**Records delimited by newline**

**Fields terminated by ","**

**Missing field values are null**

**)**

**Location ('indata1.txt')**

**)**

***fetch last inserted / updated***

***ORA\_ROWSCN:*** *Is pseudo column & used to find-out* ***system change-number (SCN)*** *of that row or multiple rows committed once in single block.*

*Consider following query in this we create table with* ***ROWDEPENDENCIES****. Using* ***scn\_to\_timestamp*** *function we can find-out last inserted or updated query.*

***Step’s***

*SQL> CREATE table DEMO (n NUMBER (9))* ***ROWDEPENDENCIES;***

*SQL> SELECT n,* ***ora\_rowscn, scn\_to\_timestamp (ora\_rowscn)*** *from DEMO;*

*SQL> COMMIT; (commit the data after INSERT / UPDATE records)*

*With the help of* ***ORA\_ROWSCN column & scn\_to\_timestamp*** *function we can easily find-out recently inserted row from any oracle table.*

***NOTE:***

* *If single row are committed then we can find-out recent inserted or updated row.*
* *If multiple rows are committed then we cannot find-out exact row to be inserted or updated.*

***Virtual columns in table***

* *The values are not physically stored, it is generated when needed.*
* *Case and other functions can be used for virtual columns.*
* *Indexes and partitions can be created on virtual columns.*
* *Primary & Foreign key can be given on virtual columns.*
* *Virtual column cannot be encrypted.*
* *Virtual column cannot be redirected to another table. (CREATE table TEMP as SELECT \* from TEMP2;* ***Note: One of the column in TEMP2 table is a virtual column****)*

*If you wanted to see what columns have a default value assigned; the view DBA\_TAB\_COLUMNS has a* ***DATA\_DEFAULT column*** *which will give you the information.*

***EG.***

*CREATE table EMP*

*(*

*Id number,*

*Salary number (9, 2),*

*Comm1 number (3),*

*Comm2 number (3),*

***Salary1 AS (ROUND (salary\*(1+comm1/100), 2)),***

***Salary2 number GENERATED ALWAYS AS (ROUND (salary\*(1+comm2/100), 2)) VIRTUAL***

*)*

*SQL> INSERT into EMP (id, salary, comm1, comm2) values (1, 5000, 100, 200);*

*SQL> INSERT into EMP values (1, 5000, 100, 200****, default, default);***

***EG.***

CREATE table VEMP

(

*Empno number,*

*Ename varchar2 (10),*

*Hra number,*

*Sal number,*

***Total as (hra + sal)***

) tablespace iway;

SQL> *INSERT into VEMP (Empno, Ename, Hra, Sal) values (1, 'Saleel', 1000, 9100);*

SQL> *INSERT into VEMP values (1, 'Saleel ', 1000, 9100,* ***default);***

***EG. with case***

*CREATE table EMP*

*(*

*Empno number,*

*Ename varchar2 (10),*

*ZoneID number (1),*

***Zone varchar2 (10) as***

***(Case ZoneID***

***When 1 then 'NORTH'***

***When 2 then 'SOUTH'***

***When 3 then 'EAST'***

***When 4 then 'WEST' END) VIRTUAL***

*)*

***EG.***

*CREATE table EMP*

*(*

*Ename varchar2 (10),*

*Ename1 varchar2 (10)* ***as (TRANSLATE (Ename, 'abcdefghijklmnopqrstuvwxyz', 'defghijklmnopqrstuvwxyzabc'))***

*)*

***EG.1***

*CREATE or REPLACE FUNCTION F1 (x NUMBER, y NUMBER) RETURN NUMBER*

***DETERMINISTIC***

*is*

*begin*

*RETURN x + y;*

*end F1;*

*/*

*CREATE table EMPLOYEE*

(*Empl\_id number,*

*Empl\_nm varchar2 (50),*

*Monthly\_sal number (10, 2),*

*Bonus number (10, 2),*

*Total\_sal number (10, 2)* ***GENERATED ALWAYS AS (F1 (monthly\_sal, bonus)) VIRTUAL***

);

*SQL> CREATE table TEMP (c1 number, c2 number, c3 number* ***invisible generated always as (c1+c2)****);*

***EG.2***

***Step1***

***F1.JAVA File***

*public class F1 {*

*public static java.lang.Integer* ***F1****(java.lang.Integer x, java.lang.Integer y) {*

*return (x+y);*

*}*

*}*

***Step2***

***C:\> loadjava -u c##saleel/saleel F1.class -v***

***Step3***

*CREATE or REPLACE FUNCTION* ***F2****(x number, y number) return number* ***DETERMINISTIC*** *is* ***language java name ‘F1.F1 (java.lang.Integer, java.lang.Integer) return java.lang.Integer';***

*/*

***Step4***

*SQL> (c1 number, c2 number, c3 number* ***generated always as (F2 (c1, c2)) virtual****);*

***Types of Partition***

* *Partition by Hash*
* *Partition by Range*
* *Partition by List*

Note: - Tables with Long columns cannot be partitioned.

Note: - If, we ***DROP PARTITION*** then it will also delete the records.

Note: - After giving **UPDATE** statement on the partitioned table, and if the record is been moved from one partition to another partition then will **show error ORA-14402: updating partition key column would cause a partition change.**

**Then give *Alter Table emp Enable Row Movement.***

**Note:- Alter *Table emp Enable Row Movement* is used for *UPDATE* command.**

**Partitioned Table**

***Partitioning is a way to make the tables and indexes more manageable by breaking them down into smaller pieces.***

*(Data Dictionary user\_tab\_partitions, user\_part\_tables)*

## *Partition by Interval*

***EG.***

*CREATE table EMP*

*(Empl\_id NUMBER,*

*Empl\_nm VARCHAR2 (50),*

*Sal NUMBER)*

***Partition by RANGE (Sal) interval (1000)***

***(Partition p1 values less than (1000))***

**Note: - The default partition name will start with *SYS\_P*.**

***EG.***

*CREATE table EMP*

*(Empl\_id NUMBER,*

*Empl\_nm VARCHAR2 (50),*

*Sal NUMBER)*

***Partition by RANGE (Sal) interval (1000)***

***(Partition p1 values less than (1000),***

***Partition p2 values less than (2000),***

***Partition p3 values less than (3000))***

*SQL> ALTER table EMP* ***SET INTERVAL ();*** *(The table is converted back to a range partitioned table and the boundaries for the interval partitions are set to the boundaries for the range partitions.)*

*SQL> ALTER table EMP* ***SET INTERVAL (1000);*** *(back to 1000 interval)*

## *Partition by System*

***EG.***

*CREATE table EMPLOYEE*

*(Empl\_id NUMBER,*

*Empl\_nm VARCHAR2 (50),*

*Sal NUMBER)*

***Partition by SYSTEM***

***(Partition PART1,***

***Partition PART2,***

***Partition PART3)***

**\*** INSERT into EMPLOYEE ***partition (part1) values (100, 'Saleel', 5000);***

## *Partition by Range*

***EG.* *(on Number)***

*CREATE table PART*

*(Partno number (5),*

*PartName varchar2 (20))*

***Partition by RANGE (partno)***

***(Partition PART1 values less than (10),***

***Partition PART2 values less than (20)***

***Partition PART3 values less than (MAXVALUE))***

*SQL>* ***SELECT \* from PART partition (PART1)***

***EG. (on Virtual Columns)***

*CREATE table EMPLOYEE*

*(Empl\_id NUMBER,*

*Empl\_nm VARCHAR2 (50),*

*Monthly\_sal NUMBER (10, 2),*

*Bonus NUMBER (10, 2),*

*Total\_sal NUMBER (10, 2)* ***AS (monthly\_sal\*12 + bonus))***

***PARTITION BY RANGE (total\_sal)***

*(PARTITION sal\_200000 VALUES LESS THAN (200000),*

*PARTITION sal\_400000 VALUES LESS THAN (400000),*

*PARTITION sal\_600000 VALUES LESS THAN (600000),*

*PARTITION sal\_800000 VALUES LESS THAN (800000),*

*PARTITION sal\_default VALUES LESS THAN* ***(MAXVALUE));***

***EG.(on Character)***

*CREATE table PART*

*(Empno number,*

*Ename varchar2 (20))*

***Partition by RANGE (ename)***

*(Partition PART1 values less than ('H'),*

*Partition PART2 values less than ('N'),*

*Partition PART3 values less than (MAXVALUE))*

## *Partition by List*

***EG.***

*CREATE table CITY*

*(Empl\_id NUMBER,*

*City varchar2 (20))*

***Partition by LIST (city)***

***(Partition PART1 values ('PUNE','MUMBAI'),***

***Partition PART2 values ('BARODA','SURAT'))***

***EG.***

*CREATE table CITY*

*(Empl\_id NUMBER,*

*City varchar2 (10))*

***Partition by LIST (city)***

***(Partition PART1 values ('PUNE','MUMBAI'),***

***Partition PART2 values (NULL),***

***Partition PART3 values (DEFAULT))***

***Note: - The DEFAULT works like MAXVALUE of Range partition.***

*SQL> ALTER TABLE "byRange"* ***ADD*** *partition part4 values less than (6000);*

*SQL> Alter table city* ***ADD*** *partition PART3 values ('SURAT');*

*SQL> ALTER TABLE city MODIFY PARTITION cPart2* ***ADD*** *VALUES (‘ANAND’*); (add new value to existing list)

*SQL> ALTER TABLE city* ***MODIFY*** *PARTITION cPart2* ***DROP*** *VALUES (' ANAND’);* (deleting the value from existing list)

*SQL> ALTER TABLE city* ***RENAME*** *partition cPart2 to NewName;*

*SQL> ALTER TABLE city* ***DROP*** *partition cPart2;*

***Note: - We cannot drop the partition VALUE if partition contains rows corresponding to values.***

*SQL> ALTER table city DROP PARTITION cPart2;* (dropping the entire partition with records)

*SQL> INSERT into part partition (part1) values (1, 1);*

*SQL> UPDATE part partition (part1) set sal = 3001;*

*SQL> DELETE from part partition (part1);*

***Type Table***

**\*** **CREATE TYPE Type1 AS Table OF varchar2 (4000);**

**\* Select deptno, cast (collect (ename) as type1) from EMP group by deptno;**

***Type Varray***

**\*** **CREATE TYPE VLOCATION AS VARRAY (3) OF NUMBER (3);**

CREATE table DEPT3 (

DEPTNO NUMBER (2) NOT NULL,

DNAME VARCHAR2 (20),

**LOC VLOCATION,**

CONSTRAINT DEPT3\_PRIMARY\_KEY3 PRIMARY KEY (DEPTNO));

**SQL>** desc **VLOCATION**

***Alter Varray***

**SQL> Alter type VLOCATION modify limit 4 cascade;**

Note: The limit of a VARRAY can only be increased and to a maximum 2147483647

**SQL> Alter type VLOCATION modify element type NUMBER (6) cascade;**

***SELECTING from Varray***

**SQL>** Select dname, **n.\* from DEPT3, TABLE (DEPT3.LOC) n;**

**SQL>** Select dname, **n.column\_value from DEPT3, TABLE (DEPT3.LOC) n;**

**SQL>** Select dname, **n.column\_value from DEPT3, TABLE (DEPT3.LOC) n, LOCATION where n.column\_value=location.locationid and locationname='DALLAS';**

**SQL>** Select locationname, dname, **n.column\_value from DEPT3, TABLE (DEPT3.LOC) n, LOCATION where n.column\_value=1 and n.column\_value=location.locationid;**

***INSERTING into Varray***

**SQL>** Insert into DEPT3 values(10,'ACCOUNTING', **VLOCATION (1, 2, 3));**

***UPDATING into Varray***

**SQL>** Update **DEPT3 set LOC = VLOCATION (5, 6, 7);**

***Nested Table (Type)***

**First create the Type**  
  
Create or replace type **bankAddress** as object

(Add1 varchar2 (10),

City varchar2 (10));

**SQL>** desc bankAddress

**Then create the table**  
  
Create table **BANK**

(Bankid number,

**Address bankaddress**);

**SQL> set desc depth 3**

**SQL>** desc bank

***Alter type***

**SQL> Alter type bank** **Add attribute (Clo1 number, Col2 number) cascade;**

**SQL> Alter type bank Modify attribute (Col1 number (4)) cascade; (**You can increase the length of a VARCHAR2 attribute, or you can increase the precision or scale of a numeric attribute.**)**

**SQL> Alter type bank** **Drop attribute (Col1, Col2) cascade;**

**SQL> Alter type bank compile;**

Note: The user needs to exit application and modify application to accommodate the type change.

Note: **Cascade** is used if records are present in the table.

***INSERTING into Nested Table***

**SQL>** insert into bank values (100, **bankaddress ('paud road', 'pune'));**

**SQL>** insert into bank values (&no, **bankaddress ('&add’,’ City'));**

**SQL>** **insert all**

Into bank values (300, **bankaddress ('1','1'))**

Into bank values (300, **bankaddress ('2','2'))**

Into bank values (300, **bankaddress ('3','3'))**

Select \* from tab;

***UPDATING into Nested Table***

**SQL> Update** bank **b set b.address.city = '555' where b.address.city = '1';**

***SELECTING from Nested Table***

**SQL> Select** bankid, **n.address.city from bank n;**

**SQL> Select** \* from bank n where **n.address.city = ‘555’;**

***Drop type force***

**SQL>** Droptype **bankAddress force;**

***Combination of Varray & Nested Table***

**\*** **Create or replace type CITY as varray (3) of varchar2 (10);**

**\* Create or replace type BANK as object**

**(Bankname varchar2 (10),**

**Area CITY);**

**\* Create table EMPL**

**(Empno number,**

**Ename varchar2 (10),**

**Bankname BANK)**

**\*** Insert into empl values (1,'saleel ', **BANK** ('HDFC', **CITY** ('Pune','Mumbai','Surat')));

Insert into empl values (2,'sharmin', **BANK** ('ICIC', **CITY** ('Baroda', 'Anand', 'Surat')));

Insert into empl values (3,'vrushali', **BANK** ('BOI', **CITY** ('Pune', 'Nagar', 'Nasik')));

**\*** Select empno, ename, **n.bankname.bankname**, **nn.column\_value** from **empl n**, **table (n.bankname.area) nn;**

***Create Table***

*SQL> CREATE table "Emp"*

*("Srno" number,*

*Ename varchar2 (20),*

*Job varchar2 (20 char),*

*Job1 varchar2 (20 byte),*

*Job2 char (10),*

*Job3 char (10 char),*

*Job4 char (10 byte),*

*UserName user*

*XRow rowid)* ***tablespace tspace1;***

*SQL> desc "Emp"*

*SQL> ALTER table "Emp" move tablespace tspace2;*

***Table***

*CREATE table V$DATETIME*

*(Date1 date,*

*Date2* ***timestamp****,*

*Date3* ***timestamp with time zone****);*

*SQL> INSERT into V$DATETIME values* ***(to\_date ('01-Jan-08 18:35:22', 'dd-Mon-yy hh24: mi: ss’), '01-Jan-08 05:23:25pm','01-Jan-08 05:23:25am'****);*

*SQL> CREATE table NEWTABLE*

*(Empno number (2),*

*Ename varchar2 (20))* ***tablespace tspace1;***

*SQL> CREATE table EMP1* ***tablespace tspace1*** *as SELECT \* from EMP;*

*SQL> CREATE table EMP1* ***(no, no1)*** *as SELECT empno, sal from EMP;*

*SQL> CREATE table EMP1 as SELECT* ***empno as no, sal as no1*** *from EMP;*

*The keyword* ***SNAPSHOT*** *is supported in place of* ***MATERIALIZED VIEW*** *for backward compatibility.*

***Snapshot***

***A SNAPSHOT is a recent copy of a table from db or in some cases, a subset of rows/cols of a table. They are used to dynamically replicate the data between distributed databases.***

*SQL> CREATE snapshot S1* ***as SELECT \* from EMP where deptno=10;***

*SQL> CREATE snapshot S2* ***as SELECT \* from EMP where deptno=20;***

*SQL> INSERT into EMP (empno, ename, deptno) values (1, 1, 10);*

*SQL> INSERT into EMP (empno, ename, deptno) values (2, 2, 20);*

*SQL> desc* ***user\_mviews****;*

*SQL> desc* ***all\_mviews***

*SQL> desc* ***user\_mview\_logs***

*SQL> desc* ***user\_refresh***

***Refresh***

*SQL>* ***exec DBMS\_SNAPSHOT.REFRESH ('s1','cf');***

*SQL>* ***exec DBMS\_MVIEW.REFRESH ('mview');***

1. ***C - Complete***
2. ***F - Full***

*SQL> SELECT \* from S1;*

*SQL> SELECT \* from S2;*

***Automatic Refresh***

***Steps***

*SQL> CREATE MATERIALIZED VIEW LOG ON DEMO;*

*SQL> CREATE SNAPSHOT SNAP* ***REFRESH COMPLETE START WITH SYSDATE NEXT SYSDATE + 5/1440*** *AS SELECT \* FROM DEMO;*

*SQL> INSERT INTO DEMO VALUES (1, 1, 1, 1);*

*SQL> INSERT INTO DEMO VALUES (2, 2, 2, 2);*

*SQL> INSERT INTO DEMO VALUES (3, 3, 3, 3);*

*SQL> COMMIT;*

*SQL> SELECT \* FROM SNAP;*

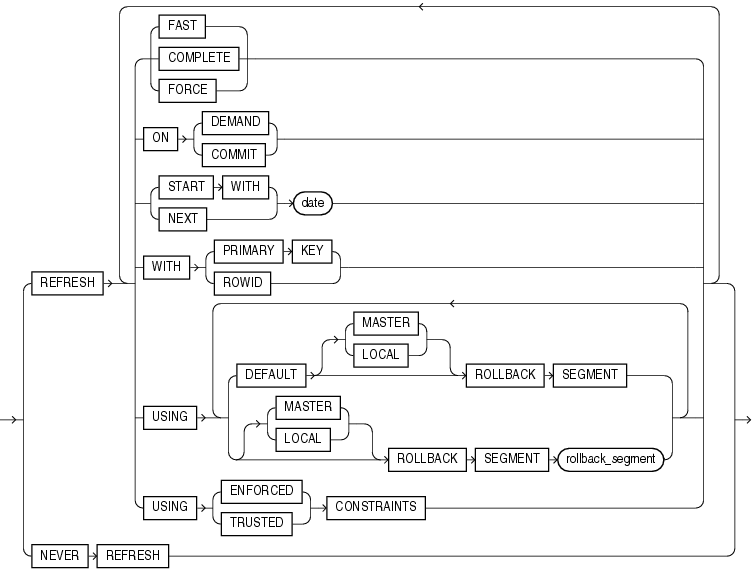
***MATERIALIZED VIEW***

1. ***DML*** *operations are not allowed on materialized view*
2. *IF we* ***DROP*** *the base table, still the records will be present in materialized view*
3. *Both* ***TABLES******(Table & Materialized view table)*** *will have different* ***ROWID****.*

|  |  |  |
| --- | --- | --- |
| *SELECT rowid from DEMO;* | *SELECT rowid from VIEW1;* | *SELECT rowid from MVIEW;* |
| *ROWID*  *------------------*  *AAAWerAAGAAAAHLAAA*  *AAAWerAAGAAAAHLAAB*  *AAAWerAAGAAAAHLAAC*  *AAAWerAAGAAAAHLAAD* | *ROWID*  *------------------*  *AAAWerAAGAAAAHLAAA*  *AAAWerAAGAAAAHLAAB*  *AAAWerAAGAAAAHLAAC*  *AAAWerAAGAAAAHLAAD* | *ROWID*  *------------------*  *AAAWwxAAGAAAAITAAF*  *AAAWwxAAGAAAAITAAG*  *AAAWwxAAGAAAAITAAH*  *AAAWwxAAGAAAAITAAI* |

*Note: ROWID will remain same for the records in TABLE and VIEW, but will change in MATERIALIZED VIEW*

Description of build_clause.gif follows



***Syntax***

***CREATE MATERIALIZED VIEW view-name***

***BUILD [IMMEDIATE | DEFERRED]***

***REFRESH [FAST | COMPLETE | FORCE]***

***ON [COMMIT | DEMAND]***

***[[ENABLE | DISABLE] QUERY REWRITE]***

***[ON PREBUILT TABLE]***

***AS***

***SELECT ...;***

***The BUILD clause options are shown below.***

* ***IMMEDIATE*** *: The materialized view is populated immediately.*
* ***DEFERRED*** *: The materialized view is populated on the first requested refresh.*

***The following refresh types are available.***

* ***FAST*** *: A fast refresh is attempted. If materialized view logs are not present against the source tables in advance, the creation fails.*
* ***COMPLETE*** *: The table segment supporting the materialized view is truncated and repopulated completely using the associated query.*
* ***FORCE*** *: A fast refresh is attempted. If one is not possible a complete refresh is performed.*

***A refresh can be triggered in one of two ways.***

* ***ON COMMIT*** *: The refresh is triggered by a committed data change in one of the dependent tables.*
* ***ON DEMAND*** *: The refresh is initiated by a manual request or a scheduled task.*

***The QUERY REWRITE*** *clause tells the optimizer if the materialized view should be consider for query rewrite operations. An example of the query rewrite functionality is shown below.*

***The ON PREBUILT TABLE*** *clause tells the database to use an existing table segment, which must have the same name as the materialized view and support the same column structure as the query.*

*SQL>* ***CREATE MATERIALIZED VIEW LOG ON DEMO;***

*SQL> CREATE MATERIALIZED VIEW MVIEW* ***BUILD IMMEDIATE REFRESH FAST ON COMMIT*** *AS SELECT \* FROM DEMO;*

*SQL> CREATE MATERIALIZED VIEW MVIEW* ***BUILD IMMEDIATE REFRESH COMPLETE START WITH SYSDATE NEXT SYSDATE + 5/1440*** *AS SELECT \* FROM DEMO;*

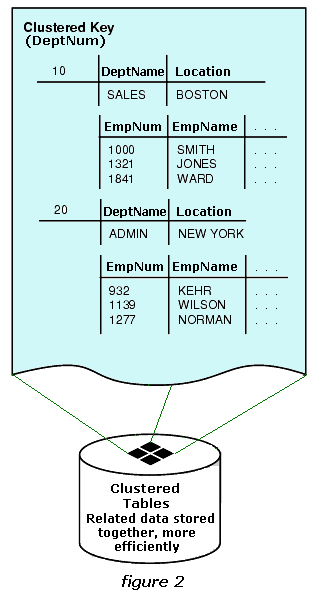
***Drop materialized view log***

*SQL> DROP materialized view log on DEMO;*

***Drop materialized view***

*SQL> DROP materialized view MVIEW;*

***Cluster***



***A cluster is an object that contains data from one or more tables, all of which have one or more columns in common. Oracle Database stores together all the rows from all the tables that share the same cluster key.***

**A cluster is a group of tables that share the same data blocks because they share common columns and are often used together.**

*SQL> CREATE cluster* ***department (deptno number);***

*SQL> CREATE index indCluster* ***on cluster department;***

*SQL> CREATE table ONE*

*(Deptno number,*

*Dname varchar2 (20))* ***cluster department (deptno);***

*SQL> CREATE table TWO*

*(Empno number,*

*Ename varchar2 (20),*

*Deptno number)* ***cluster department (deptno);***

*SQL> CREATE table THREE*

*(Empno number,*

*Designation number,*

*Deptno number)* ***cluster department (deptno);***

*SQL> DROP cluster* ***department including tables cascade constraints;***

***Note: - Specify INCLUDING TABLES to drop all tables that belong to the cluster.***

***Note: - A cluster can contain a maximum of 32 tables.***

***Note: - You can specify up to 16 cluster key columns.***

***You cannot specify a cluster key column of datatype LONG, LONG RAW, REF, nested table, varray, BLOB, CLOB, BFILE, or user-defined object type.***

***Transparent data encryption***

* ***Virtual column cannot be encrypted***
* ***You cannot encrypt a foreign key column.***

There are 4 encryption algorithms available for Transparent Data Encryption (TDE).

* 3DES168
* AES128
* AES192
* AES256

**AES192 is the default.**

***Step 1:*** *Create a folder with the* ***wallet.***

***Step 2:*** *Edit* ***sqlnet.ora*** *file (D:\app\Administrator\product\11.1.0\db\_1\NETWORK\ADMIN)*

*SQLNET.AUTHENTICATION\_SERVICES= (NTS)*

*NAMES.DIRECTORY\_PATH= (TNSNAMES, EZCONNECT)*

*ENCRYPTION\_WALLET\_LOCATION = (SOURCE = (METHOD=FILE) (METHOD\_DATA = (DIRECTORY =* ***C:\Wallet****)))*

***(C:\Wallet) is the folder***

***Step 3:*** *Re-start Oracle Services.*

***Step 4: SQL>*** *ALTER SYSTEM SET ENCRYPTION KEY AUTHENTICATED BY saleel;*

*SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY saleel;*

*SQL> ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY saleel;*

*SQL> ALTER SYSTEM SET WALLET CLOSE IDENTIFIED BY saleel;*

*SQL> ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY saleel;*

***EG.***

*SQL> CREATE table EMP (empno number, ename varchar2 (10)* ***encrypt****)*

*SQL> CREATE table TEMP (empno, ename, job, mgr, hiredate, sal, comm****, deptno encrypt) as SELECT \* from EMP;***

*SQL> ALTER table EMP* ***modify walletid number (5) encrypt;***

*SQL> desc* ***v$encryption\_wallet***

*SQL> SELECT status from* ***v$encryption\_wallet;***

***Define Variables***

***\**** *DEFINE V\_DEPTNO = 20*

***\**** *DEFINE x = ''' SALESMAN'', ''MANAGER'''*

***\**** *DEFINE y = '&x'*

***\**** *UNDEFINED V\_DEPTNO*

***\**** *SELECT last\_name, salary from EMPLOYEE where department\_id = &****V\_DEPTNO;***

***\**** *SELECT* ***&V\_DEPTNO*** *from DUAL;*

***Define x = 1;***

***\**** *SELECT* ***'100 &x.st Street'*** *from DUAL;*

Note: - The live of the defined variable is till the instance is activated **(SQL Plus editor is open).**

***Accept***

***\* ACCEPT variable data type***

***EG.***

*SQL>* ***Accept*** *x number* ***prompt*** *'Enter Employee No. '*

*SQL> Accept y char prompt 'Enter Name ';*

*SQL> Accept z number* ***default 2000*** *prompt 'Enter No ';*

*SQL>* ***Define***

***Bind Variables***

*SQL> var <var NM> <Data Type>*

***EG.***

*SQL> var X varchar2 (10)*

*SQL> var Y number*

*SQL> var Z date*

*SQL> execute :X := 10000;*

***Step***

*SQL>* ***var :c1 refcursor ;***

***EG.***

*Begin*

***Open :c1 for SELECT \* from EMP;***

*End;*

*/*

*SQL> Print****:c1 ;***

***Step***

*SQL> var x number*

***EG.***

*begin*

***:x := 1001;***

*end;*

*/*

*SQL> Print :x*

Note: - This variable is used to store the value of OUT parameter of PL Block.

*SQL> var x number*

*SQL> exec* ***:x :=10***

*SQL> SELECT \* from EMP where* ***deptno = :x***

***Deadlock***

*SQL> SELECT \* from EMP where job=’MANAGER’* ***for update;***

*SQL> SELECT \* from EMP where job=’MANAGER’* ***for update of sal, comm;***

*SQL> SELECT \* from EMP* ***for update skip locked;***

*SQL> INSERT into DEPT values (****(SELECT max (deptno) + 1 from DEPT)****,'a','a',1);*

***// For multiple instances of same user.***

***Insert all***

***INSERT all***

*into DEPT1 values (11, 'IBM', 'PUNE')*

*into DEPT1 values (12, 'Microsoft', 'PUNE')*

*into DEPT1 values (13, 'Google', 'PUNE')*

*SELECT \* from DUAL*

***INSERT all***

***when deptno <= 20 then***

*into T values (empno, ename, job, mgr, hiredate, sal, comm, deptno, grade)*

***when deptno < 30 then***

*into TT values (empno, ename, job, mgr, hiredate, sal, comm, deptno, grade)*

*SELECT \* from EMP1;*

***Insert when***

***INSERT all***

***when deptno = 10 then***

*into D1 values (deptno, dname, loc, walletid)*

***else***

*into D2 values (deptno, dname, loc, walletid)*

*SELECT \* from DEPT*

***Insert first***

***INSERT first***

***when job in ('MANAGER','SALESMAN') then***

*into T1 (empno ,ename, mgr, job, deptno) values(empno, ename, mgr, job, deptno)*

***when job in ('CLERK','SALESMAN') then***

*into T2 (empno, ename, mgr, job, deptno) values(empno, ename, mgr, job, deptno)*

*SELECT \* from EMP*

*/*

***Different SELECT statement***

*SQL>* *SELECT trunc (sal,-3)/1000 as "R1", count (trunc (sal,-3)/1000) as "R2" from emp1 group by trunc (sal,-3)/1000 order by "R2"; (count total no of employees having salary in one, two, three … thousands)*

*SQL>* *SELECT \* from DEPT where not exists (SELECT deptno from EMP where dept.deptno = emp.deptno)*

*SQL>* *SELECT orderid, order, order\_date from ORDERS where to\_number (to\_char (order\_date, 'HH24MI')) between 1200 and 1400 (Will, show result between two Time.)*

*SQL>* *SELECT rank (1300) within group (order by sal desc) from PART;*

*SQL>* *INSERT into EMP (empno, ename) values (555, (SELECT dname from DEPT where deptno=40))*

*SQL>* *INSERT into PART PARTITION (part3) values (9000,'SALEEL','MANAGER', 9000);*

*SQL>* *SELECT Ename “Employee Name”, job JOB from EMP; (The column alias name Job appears in all capital letters because we did not enclose it in double quotation marks.)*

*SQL>* *SELECT job ejob from EMP order by ejob (COLUMN alias name can be given in ORDER BY clause)*

***Comparison operators ANY, SOME and ALL***

***ANY/ SOME***

**Assuming subqueries don't return zero rows, the following statements can be made for both list and subquery versions:**

* *"x = ANY (...)": The value must match one or more values in the list to evaluate to TRUE.*
* *"x != ANY (...)": The value must not match one or more values in the list to evaluate to TRUE.*
* *"x > ANY (...)": The value must be greater than the smallest value in the list to evaluate to TRUE.*
* *"x < ANY (...)": The value must be smaller than the biggest value in the list to evaluate to TRUE.*
* *"x >= ANY (...)": The value must be greater than or equal to the smallest value in the list to evaluate to TRUE.*
* *"x <= ANY (...)": The value must be smaller than or equal to the biggest value in the list to evaluate to TRUE.*

***ALL***

**Assuming subqueries don't return zero rows, the following statements can be made for both list and sub query versions:**

* *"x = ALL (...)": The value must match all the values in the list to evaluate to TRUE.*
* *"x != ALL (...)": The value must not match any values in the list to evaluate to TRUE.*
* *"x > ALL (...)": The value must be greater than the biggest value in the list to evaluate to TRUE.*
* *"x < ALL (...)": The value must be smaller than the smallest value in the list to evaluate to TRUE.*
* *"x >= ALL (...)": The value must be greater than or equal to the biggest value in the list to evaluate to TRUE.*
* *"x <= ALL (...)": The value must be smaller than or equal to the smallest value in the list to evaluate to TRUE.*

*SQL>* *SELECT \* from EMP where sal <=ANY (2000, 3000);*

*SQL> SELECT \* from EMP where job = SOME ('SALESMAN','MANAGER','A');*

*SQL>* *SELECT \* from EMP where sal >=ALL (2000, 3000)*

*SQL>* *SELECT USER from DUAL;*

*SQL>* *SELECT ora\_login\_user from DUAL;*

*SQL>* *SELECT job, comm from EMP order by comm NULLS FIRST;*

*SQL>* *SELECT job, comm from EMP order by comm NULLS LAST;*

*SQL>* *ALTER table EMP RENAME to Employee;*

*SQL>* *ALTER table EMP RENAME COLUMN sal to salary; (Oracle9i Release 2)*

*SQL>* *ALTER table EMP RENAME CONSTRAINT test1\_pk to test\_pk; (Oracle9i Release 2)*

*SQL>* *UPDATE xdate set srno = ROWNUM;*

*SQL>* *SELECT job, sum (sal) from EMP group by rollup (job, sal)*

*SQL>* *SELECT \* from EMP1 where rowid not in (SELECT min (rowid) from EMP1 group by empno, ename, job, mgr, hiredate, sal, comm, deptno) (will, display only the duplicate records.)*

*SQL>* *SELECT \* FROM EMP1 a WHERE rowid > (SELECT min (rowid) FROM EMP1 b WHERE b.empno = a.empno) (will, display only the duplicate records.)*

*SQL>* *DELETE FROM EMP1 a WHERE rowid > (SELECT min (rowid) from EMP1 b where b.empno = a.empno) (will, delete only the duplicate records.)*

*SQL>* *SELECT \* from EMP3 e where rowid = (SELECT max (rowid) from EMP3 ee where ee.empno=e.empno)*

*SQL>* *SELECT emp1.ename, rowid from EMP1 where rowid in (SELECT rowid from EMP1 where rownum <= 20 MINUS SELECT rowid from EMP1 where rownum < 7) (will, display range of record’s)*

*SQL>* *SELECT \* from (SELECT rownum as "R1", EMP.\* from EMP) where "R1">=5 and "R1"<=7; (will, display range of record’s)*

*SQL>* *SELECT \* from (SELECT rownum as "R1", EMP1.\* from EMP1) where "R1" = (SELECT max (rownum) from EMP1); (will, display last entered record in table)*

*SQL>* *SELECT \* from EMP where rownum < 9 minus SELECT \* from EMP where rownum <4; (will, display range of record’s)*

*SQL>* *UPDATE newemp set (eno, dno) = (SELECT 10, 10 from DUAL) where enm='saleel'*

*SQL>* *SELECT dbms\_rowid.rowid\_block\_number (rowid), dbms\_rowid.rowid\_relative\_fno (rowid), dbms\_rowid.rowid\_row\_number (rowid) from EMP;*

*SQL>* *SELECT \* from EMP1 where dbms\_rowid.rowid\_row\_number (rowid) > (SELECT min (dbms\_rowid.rowid\_row\_number (rowid)) from EMP1 b where b.empno=emp1.empno)*

*SQL>* *SELECT \* from (SELECT ename, sal, RANK () OVER (ORDER BY SAL Desc) as R1 FROM EMP ORDER BY SAL Desc) WHERE R1 < 6*

*SQL>* *SELECT \* from (SELECT emp.\*, rank () over (order by sal) as "R1" from EMP) where R1=4*

*SQL>* *whenever sqlerror EXIT;*

*SQL>* *SELECT \* from EMP where (rowid, 0) in (SELECT rowid, mod (rownum, 2) from EMP);*

*SQL>* *SELECT \* from (SELECT emp.\*, mod (rownum, 2) as "R1" from EMP) where "R1"=0;*

*SQL>* *SELECT \* from NLS\_DATABASE\_PARAMETERS;*

*SQL>* *SELECT \* from NLS\_SESSION\_PARAMETERS;*

*SQL>* *SELECT emp1.deptno, dname, sum (sal) as "Cube" from EMP1, DEPT where emp1.deptno = dept.deptno group by cube (dname, emp1.deptno);*

*SQL>* *SELECT job, sal, rank () over (partition by job order by sal) from EMP order by job, sal*

*SQL>* *SELECT job, deptno, sal, rank () over (partition by job order by sal) as "R1", case (rank () Over (partition by job order by sal)) when 1 then 'First' when 2 then 'Second' When 3 then 'Third' else 'Not Required' end from EMP;*

*SQL>* *SELECT \* from (select dense\_rank () over (partition by job order by sal desc) as "R1", job, Sal from EMP) where "R1" in (1, 2); (will, display first 2 highest salary job wise)*

*SQL>* *SELECT job, sal, rank () over (partition by job order by sal) As "R1" from EMP where (Empno, 1) in (SELECT empno, rank () over (partition by job order by sal) as "R1" FROM EMP) or (empno, 2) in SELECT empno, rank () over (partition by job order by sal) as "R1" from emp) or (empno, 3) in SELECT empno, rank () over (partition by job order by sal) as "R1" from emp) order by job (will, display job wise first three record’s)*

*SQL>* *INSERT into DEPT SELECT max (deptno) + 10,'HRD', 'PUNE' from DEPT;*

*SQL>* *SELECT ename, salary, city\_code, rank () over (partition by city\_code order by salary desc) as “Range" from "byRange" partition (spart2);*

*SQL>* *SELECT job, Sal, rank () over (partition by job order by sal) as "Rank", DENSE\_RANK () OVER (PARTITION BY job order by sal) as "Dense Rank" from EMP1;*

*SQL>* *UPDATE NEWEMP set dname = (SELECT dname from DEPT where dept.deptno = newemp.dno);*

*SQL>* *SELECT sum (decode (deptno, 10, sal)) as "Dept No. 10”, sum (decode (deptno, 20, sal)) as "Dept No. 20", sum (decode (deptno, 30, sal)) as "Dept No. 30" from EMP;*

*SQL>* *SELECT max (sal) from EMP where level=3 connect by prior sal>sal; (will display max Nth row from the table)*

*SQL>* *SELECT ename, (case when sal <1500 then 'aaa' when sal <=2500 then 'bbbb' end) as "Case" from EMP;*

*SQL>* *SELECT \* FROM (SELECT ROWNUM as “R1”, e.\* FROM EMP e) WHERE “R1” = 25;*

*SQL>* *SELECT \* FROM (SELECT \* from EMP where dbms\_rowid.rowid\_row\_number (rowid) =25);*

*SQL>* *SELECT sum (case when comm>0 then comm else null end) as "+tv", sum (case when comm <0 then comm else Null end) as "-tv" from EMP;*

*SQL>* *SELECT deptno, ename, sal, MIN (sal) keep (DENSE\_RANK FIRST ORDER BY sal) OVER (PARTITION BY deptno) "Lowest", MAX (sal) keep(DENSE\_RANK LAST ORDER BY sal) OVER (PARTITION BY deptno) "Highest" FROM emp ORDER BY deptno, sal*

*(To display the salary of each employee, along with the lowest and highest within their department.)*

*SQL>* *SELECT to\_date ('20071212','YYYYMMdd') from DUAL;*

*SQL>* *SELECT comm,* ***decode (comm, NULL, 'No Commission', comm)*** *from EMP;*

*SQL>* *GRANT connect, resource, dba, sysdba to sharmin identified by sharmin;*

*SQL>* *SELECT unique job, sal,"R1" from (SELECT job, sal, dense\_rank () over (partition by job order by sal) as "R1" from EMP1) where "R1" in (2, 3) order by job, sal;*

*SQL>* *SELECT rownum, case when rownum between 1 and 5 then '1 to 5' when rownum between 6 and 10 then '6 to 10' else 'Remaining' end "R1" from EMP;*

*SQL>* *INSERT into v$Time values (2, to\_date (to\_char (sysdate,'dd-Mon-yy hh: mi: ss’),'dd-Mon-yy hh: mi: ss’))*

*SQL>* *SELECT \* from EMP order by length (ename);*

*SQL>* *SELECT sysdate, systimestamp, extract(day from sysdate) as "Day", extract(month from sysdate) as "month" ,extract(year from sysdate) as "Year", extract(hour from systimestamp) as "Hour" , extract(minute from systimestamp) as "Minutes", extract(second from systimestamp) as "Seconds", extract(timezone\_hour from systimestamp) as "TimeZone Hour" from DUAL;*

*SQL>* *SELECT \* from EMP where (empno, 5) in (SELECT empno, rank () over (order by sal desc) as "R1" from EMP); (to display 5th record)*

*SQL>* *SELECT ename from EMP3 group by ename having count (ename) =1;*

*SQL>* *SELECT max (sal) from EMP where sal < (SELECT max (sal) from EMP where sal < (SELECT max (sal) from EMP)); (Third highest salary)*

*SQL>* *ALTER user u1 account lock;*

*SQL>* *ALTER user u1 account unlock;*

*SQL>* *INSERT into t values ('This is ' || Chr (10) || ' the test'); (Line Brake)*

*SQL>* *SELECT 'L' || chr (38) || 'T' from DUAL; (will put ‘&’ sign)*

*SQL>* *SELECT rownum, job, row\_number () over (partition by job order by job desc) from EMP;*

*SQL>* *SELECT \* from TT where id1='y' or id in ((SELECT id from tt intersect SELECT id from TT where id1='n') minus (SELECT id from TT intersect SELECT id from TT where id1='y')) order by id;*

*SQL>* *SELECT \* from DEPT where* ***substr (dname, 1, 1) = upper (substr (dname, 1, 1));***

*SQL> SELECT emp.deptno, dname, count (\*) from EMP, dept where emp.deptno = dept.deptno group by emp.deptno, dname having count (emp.deptno) = (SELECT max (count (deptno)) from EMP group by deptno)*

*SQL>* *SELECT \* from EMP1 where job in (SELECT job from EMP1 group by job having count (job) = (SELECT max (count (job)) from EMP1 group by job));*

*SQL>* *SELECT max (length (e.ename))-max (length (ee.ename)) from EMP e, EMP ee where length (e.ename) > length (ee.ename);*

*SQL>* *SELECT terminal, machine from v$session;*

*SQL>* *SELECT ename, rpad (ename, 12, dbms\_random.value ()) as "R1” from EMP;*

*SQL>* *SELECT sum (nvl (sal, 0)), nvl (sum (sal), 0) from EMP;*

*SQL>* *SELECT , ORACLE, from EMP;*

*SQL>* *SELECT mod (empno, 5), empno from EMP where mod (empno, 5) = 0;*

MOD (EMPNO, 5) EMPNO

------------ ----------

0 5

0 10

0 15

0 20

0 25

0 30

0 35

0 40

0 45

9 rows selected.

*SQL>* *SELECT loc, count (empno) from EMP right join dept on emp.deptno = dept.deptno group by loc;*

LOC COUNT (EMPNO)

---------- -----------------------

NEW YORK 3

CHICAGO 6

BOSTON 0

DALLAS 5

4 rows selected.

*SQL>* *SELECT sal, trunc (sal,-3) / 1000 from EMP order by 2 desc;*

SAL TRUNC (SAL,-3)/1000

---------- -----------------------------

5000 5

3000 3

2975 2

1100 1

950 0

5 rows selected.

*SQL>* *SELECT trunc (sal,-3), count (trunc (sal,-3)/1000) from EMP group by trunc (sal,-3) order by 1;*

TRUNC (SAL,-3) COUNT (TRUNC (SAL,-3)/1000)

--------------------- --------------------------------------------

0 2

1000 6

2000 3

3000 2

5000 1

5 rows selected.

*SQL>* *SELECT sum (sal) - (SELECT sum (sal) from EMP where deptno=20) from EMP where deptno=10 group by deptno;*

**OR**

*SQL>* *SELECT (SELECT sum (sal) from EMP where deptno=10) - (SELECT sum (Sal) from EMP where deptno=20) from DUAL;*

**OR**

*SQL>* *SELECT sum (decode (deptno, 10, sal)) – sum (decode (deptno, 20, sal)) from EMP;*

**OR**

*SQL>* *With*

*A as (SELECT sum (sal) as "R1" from EMP where deptno=10),*

*B as (SELECT sum (sal) as "R2" from EMP where deptno=20),*

*C as (SELECT "R2" - "R1" from A, B)*

*SELECT \* from C*

*SQL>* *SELECT distinct (a.sal) from EMP A where &n = (SELECT count (distinct (b.sal)) from EMP B where a.sal <= b.sal)*

***Truncate***

*SQL>* *TRUNCATE table "Emp";* (Will, remove all records from the table.)

*SQL> TRUNCATE table DEPT;*

*TRUNCATE table DEPT*

*\**

***ERROR at line 1:***

***ORA-02266: unique/primary keys in table referenced by enabled foreign keys***

*SQL>* *TRUNCATE table DEPT* ***CASCADE; (Table must be in PK/FK)***

Note: - In case of TRUNCATE, Trigger doesn't get fired.

Note: - ROLLBACK is not possible on Truncated Table.

***High Water Mark***

+---- H**igh Water Mark of newly created table**

|

V

+--------------------------------------------------------+

| | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | |

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

**High Water Mark after inserting 10,000 rows**

|

V

+--------------------------------------------------------+

|x |x |x |x |x |x |x |x |x |x |x |x | | | | | | | |

|x |x |x |x |x |x |x |x |x |x |x |x | | | | | | | |

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

**High Water Mark after inserting 10,000 rows**

|

V

+--------------------------------------------------------+

|x |x |x |x |x |x |x | | | | | | | | | | | | |

|x |x |x |x |x |x |x | | | | | | | | | | | | |

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

**"HWM” is increased when we insert data. But it will not decrease automatically when we delete data.**

*SQL>* *ALTER table EMP* ***enable row movement;***

*SQL>* *ALTER table EMP* ***shrink space;***

***Escape***

*SQL> SELECT \* from DEPT where dname like '%ss\\_%'* ***escape '\';***

Output

DEPTNO DNAME LOC

------------- -------------- -------------

1. ss\_d dd

*SQL> SELECT \* from EMP where 'KING' like ename;*

*SQL> SELECT \* from EMP where 'KING' = ename;*

***REGEXP\_LIKE***

***Note: - REGEXP\_LIKE***

*SQL> SELECT ename from EMP where regexp\_like* ***(ename, '([SA])’);***

*SQL> SELECT ename from EMP where regexp\_like* ***(ename, '(^ [SA])’);***

*SQL> SELECT ename from EMP where regexp\_like* ***(ename, '^…[A]');*** *(4th char must be A)*

*SQL> SELECT \* from EMP where regexp\_like* ***(first\_name, '^. [L]');*** *(2nd char L)*

*SQL> SELECT \* from EMP where regexp\_like* ***(first\_name, '^ [A||B||c]', 'i');*** *(starts with 'A' or 'B' or 'c' and ignore case)*

*SQL> SELECT \* from EMP where regexp\_like* ***(first\_name, '[A||B]');*** *(having char 'A' or 'B' in name)*

*SQL> SELECT \* from EMP where regexp\_like* ***(first\_name, 'h$', 'i');*** *(ends with char 'h' and ignore case)*

*SQL> SELECT \* from EMP where regexp\_like* ***(first\_name, 'h.$', 'i');*** *(second last char 'h' and ignore case)*

*SQL> SELECT regexp\_like* ***(ename, '[[:alpha:]]+') as "R1”,*** *ename from TEMP;*

***^ As the start***

***$ As end respectively***

***c Case sensitive matching***

***I Case insensitive matching***

***[ ] Char sets***

***|| OR***

***'[[:digit:]]+'***

***'[[:alnum:]]’***

***'[[:alpha:]]’***

***'[[:lower:]]’***

***'[[:upper:]]’***

***regexp\_count***

*SQL> SELECT ename,* ***regexp\_count (ename, 'A')*** *from EMP;*

*SQL>**SELECT ename,* ***regexp\_count (ename,'[AS]')*** *from EMP;*

*SQL>**SELECT ename,* ***regexp\_count (ename,'[a]', 1,'i')*** *from EMP;*

***Rules of Precedence***

|  |  |
| --- | --- |
| **Order Evaluated** | **Operator** |
| 1 | Arithmetic operators |
| 2 | Concatenation operator |
| 3 | Comparison conditions |
| 4 | Is [NOT] Null, Like, [NOT] IN |
| 5 | [NOT] Between |
| 6 | NOT Logical condition |
| 7 | And Logical condition |
| 8 | Or Logical condition |

***Select \* from EMP where job='MANAGER' or job='SALESMAN' and sal >=1500;***

Note: - “Select the row if then EMP is ‘SALESMAN’ and earns more than 1500, or if the EMP is a ‘MANAGER’”

***Conversion Functions***

|  |  |
| --- | --- |
| **Function** | **Result** |
| To\_date |  |
| To\_char |  |
| To\_number |  |
| CAST ('1234' as number)  CAST (1234 as varchar2(10))  CAST (sysdate as varchar2(10)) |  |

***Format number in select***

*SQL>**SELECT* ***to\_char (sal,'999, 999, 00')*** *from EMP;*

*SQL>**SELECT* ***to\_char (sal,'$999, 999, 00')*** *from EMP;* (Displays a dollar sign.)

*SQL>**SELECT* ***to\_char (sal, 'L99, 999')*** *from EMP;* (Displays the local currency symbol.)

*SQL>**SELECT* ***to\_char (sal,'S999, 999, 00')*** *from EMP;* (Displays a trailing minus or plus sign.)

*SQL>**SELECT* ***to\_char (sal,'999, 999, 00MI')*** *from EMP;* (Displays a trailing minus sign only.)

***Number Format’s***

|  |  |
| --- | --- |
| **Formats** | |
| 9 | Represents a number **e.g. To\_char(sal, '9999')** |
| 0 | Forces a zero to be displayed **e.g. To\_char(sal, '09999 ')** |
| $ | Places a dollar sign **e.g. To\_char(sal, '$9999')** |
| L | **Local currency symbol**  **e.g.**  **alter session set nls\_currency='Rs.';**  **select To\_char(sal, 'L9999') from emp;** |
| **.** (dot) / D | Print a decimal point. **To\_char (sal, '9999.99')** |
| **,** (comma) / G | Print a comma as thousands indicator **To\_char (sal, '9,999.99')**  **To\_char (sal, '9G999d99')** |
| S | Returns the negative or positive value. **To\_char (sal, 's9,999.99')** |
| MI | Minus sign to right (negative values) **To\_char (sal, '9,999.99mi')** |
| C | Currency **USD12345**  **To\_char (12345, 'C99999')** |
| PR | Returns negative value in <angle brackets>.  Returns positive value with a leading and trailing blank.  Restriction: The PR format element can appear only in the last position of a number format model.  **To\_char (-1000,'9999PR')** |
| RN / r n | Returns a value as Roman numerals in uppercase.  Returns a value as Roman numerals in lowercase.  Value can be an **integer between 1 and 3999.**  **To\_char (4,'RN')** |

***String Functions***

|  |  |
| --- | --- |
| **Function** | **Result** |
| LOWER ('SQL Course') | sql course |
| NLS\_Lower ('SQL Course') | sql course |
| UPPER ('SQL Course') | SQL COURSE |
| NLS\_Upper ('SQL Course') | SQL COURSE |
| INITCAP ('SQL Course') | Sql Course |
| NLS\_Initcap ('SQL Course') | Sql Course |

|  |  |
| --- | --- |
| **Function** | **Result** |
| CONCAT ('Hello', ' World') | Hello World |
| SUBSTR ('Hello World', 1,5)  SUBSTR ('Hello World',**- 5**) | Hello  World |
| LENGTH ('Hello World') | 10 |
| INSTR ('salilbagde@gmail.com', '@') | 11 |
| LPAD (sal, 10,’\*’) / LPAD(ename,25) | \*\*\*\*\*24000 |
| RPAD (sal, 10,’\*’) / RPAD(ename,25) | 24000\*\*\*\*\* |
| TRIM (' HelloWorld ') | HelloWorld |
| LTRIM ('HelloWorld', 'H') | elloWorld |
| RTRIM ('HelloWorld', 'd') | HelloWorl |
| DUMP (‘ABC’) (The dump function returns a varchar2 value that includes the datatype code, the length in bytes) | Typ=96 Len=3: 65,66,67 |
| Reverse ('Hello') | olleH |
| ASCII(‘A’) | 65 |
| REPLACE (ename, 'S', 'x') |  |
| TRANSLATE (ename, 'S', 'x') |  |
| CHR(65) | A |
| REMAINDER(5,2) / MOD(5,2) | 1 |
| ROUND (n1,n2) / TRUNC (n1,n2) |  |
| GREATEST (1, 2, 3, 4) | 4 |
| LEAST (1, 2, 3, 4) | 1 |

Note: - Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds

Note: - SYSDATE is a date-function that returns the current database server date and time. (Will, Display only date)

Note: - SYSTIMESTAMP is a date-function that returns the current database server date and time. (Will, Display only date & time both)

|  |  |  |
| --- | --- | --- |
| **Operation** | **Result** | **Description** |
| date + number | Date | Adds a number of days to a date |
| date - number | Date | Subtracts a number of days to a date |
| date – date | Number of days | Subtracts one date from another |
| date + number / 24 | Date | Adds a number of hours to a date |
| date + number / 1440 | Date | Adds a number of minutes to a date |
| date + number / 86400 | Date | Adds a number of seconds to a date |

***Date Functions***

|  |  |
| --- | --- |
| **Function** | **Result** |
| MONTHS\_BETWEEN ('01-SEP-95', '11-JAN-94') | 19.6774194 |
| ADD\_MONTHS ('01-SEP-95', 6) | 01-MAR-96 |
| NEXT\_DAY ('01-SEP-95', 'FRIDAY') | 08-SEP-95 |
| LAST\_DAY ('01-FEB-95') | 28-FEB-95 |

***Date Format’s***

|  |  |
| --- | --- |
| **Formats** | |
| D | Numeric day of the week. |
| DD | Numeric day of the month. |
| DDD | Numeric day of the year. |
| Dy | Three-letter abbreviation of the week. |
| Day | Full name of the day of the week. |
| fmDay | Formatted Day, which suppresses blank padding, the length of the return value may vary |
| W | Week of month 1-5 |
| Ww | Week of year 1-52 |
| MM | Two-digit value for the month. |
| Mon | Three-letter abbreviation of the month. |
| Month | Full name of the month. |
| fmMonth | Formatted Month, which suppresses blank padding, the length of the return value may vary |
| YY | Two-digit value for the year. |
| YYYY | Full year in numbers. |
| Year | Year spelled out (in English). |
| DL | Long date format |
| DS | Short date format |
| CC |  |
| RM | Roman numerical month. |
| Q | Quarter of year. |
| Th | Ordinal number. Eg 4th **ordinal suffix (st, nd, rd or th.)** |
| Sp | Spelled-out number. |
| HH | Hour of the day. |
| HH12 | Hour of the day (1-12). |
| HH24 | Hour of the day (0-23). |
| MI | Minute (0-59). |
| SS | Seconds (0-59). |
| SSSSS | Seconds after midnight (0 – 86399) |
| AM or PM | Meridian indicator |
| -  /  ,  .  ;  :  "text" | Punctuation and quoted text is reproduced in the result. |

*SQL>* **SELECT '"' || to\_char (hiredate, 'Month') ||'"','"' || to\_char (hiredate, 'fmMonth') ||'"' from EMP;**

|  |  |
| --- | --- |
| **Description** | **Date Expression** |
| Now | SYSDATE / **Current\_date** |
| Tomorrow / next day | SYSDATE + 1 |
| Seven days from now | SYSDATE + 7 |
| One hour from now | SYSDATE + 1 / 24 |
| Three hours from now | SYSDATE + 3 / 24 |
| An half hour from now | SYSDATE + 1 / 48 |
| 10 minutes from now | SYSDATE + 10 / 1440 |
| 30 seconds from now | SYSDATE + 30 / 86400 |
| Tomorrow at 12 midnight | TRUNC (SYSDATE + 1) |
| Tomorrow at 8 AM | TRUNC (SYSDATE + 1) + 8 / 24 |
| Next Monday at 12:00 noon | NEXT\_DAY (TRUNC (SYSDATE), 'MONDAY') + 12 / 24 |
| First day of next month at 12 midnight | TRUNC (LAST\_DAY (SYSDATE) + 1) |
| First day of the current month | TRUNC (LAST\_DAY (ADD\_MONTHS (SYSDATE, -1))) + 1 |
| The next Monday, Wednesday or Friday at 9 am | TRUNC (LEAST (NEXT\_DAY (sysdate, "MONDAY"), NEXT\_DAY (sysdate, "WEDNESDAY"), NEXT\_DAY (sysdate, "FRIDAY"))) + (9/24) |

***Sub Set Operators***

|  |  |
| --- | --- |
| **Sub Set** | |
| In Oracle | In MS-Server |
| UNION | UNION |
| UNION ALL | UNION ALL |
| INTERSECT | INTERSECT |
| MINUS | EXCEPT |

***Set System Variable***

|  |  |
| --- | --- |
| **Set** | |
| Set feedback *n*, on /off | Set feedback on, Set feedback 4 |
| Set numwidth <size> | Set numwidth 8 |
| Set Linesize <size> | Set linesize 100 |
| Set PageSize <size> | Set pagesize 40 |
| Set autotrace on/off | Set autotrace on |
| Set heading on/off | Set heading on |
| Set pause on/off | Set pause on |
| Set sqlcase <MIXED/ UPPER/LOWER> | Set sqlcase Upper (data stored will be in upper case) |
| Set sqlprompt <String> | Set sqlprompt Saleel> |
| Set timing on/off | Set timing on (measuring the running time of SQL commands.) |
| Set time on/off | Set time on |
| Set verify on/off | Set verify off (will hide the :OLD and :NEW value prompt.) |
| Set define on/off/char | Set define off (to deactivate ‘&’ sign.) |
| Set Null *text* | Set Null 'NULL' |
| Set serveroutput on/off | Set serveroutput on |
| Set autocommit on/off | Set autocommit on |
| Set long size | Set long 100000 (1-2000000000) (to change to size of LONG datatype column.) |
| Set errorlogging on/off | All error information is written in sperrorlog table.  select \* from sperrorlog; |

|  |  |
| --- | --- |
| **Data Dictionary** | **Column’s** |
| V$session | Sid, serial#, username |
| V$instance | Instance\_number, instance\_name, host\_name, version |
| V$version | Banner |
| product\_component\_version | Product, version, status |
| user\_catalog | Table\_name, table\_type |
| user\_free\_space | tablespace\_name, file\_id, block\_id, bytes, blocks, relative\_fno |
| user\_synonyms (syn) | Synonym\_name, table\_owner, table\_name, db\_link |
| User\_sequences (seq) | Sequence\_name, min\_value, max\_value, increment\_by, cycle\_flag |
| User\_resource\_limits | Composite\_limit, sessions\_per\_user, cpu\_per\_session |
| Global\_name | Global\_name (Will Display Host String Name) |
| Cols |  |
| Col | Tname, cname, coltype, width, scale, precision, nulls, defaultval |
| User\_triggers | Trigger\_name, table\_owner, table\_name, column\_name |
| User\_ts\_quotas | Tablespace\_name, bytes, max\_bytes, blocks, max\_blocks |
| User\_source | Name, type, line, text |
| User\_views | View\_name, text\_length, text, type\_text\_length, type\_text |
| **Dictionary** | **Table\_name, comments** |

***Tablespace***

|  |  |
| --- | --- |
| V$tablespace | Ts#, name, included\_in\_database\_backup |
| User\_tablespaces | Tablespace\_name, status |
| Dba\_tablespaces | Tablespace\_name, status |
| All\_tab\_tablespaces | table\_owner, table\_name, partition\_name, tablespace\_name, high\_value |

***DataFiles***

|  |  |
| --- | --- |
| dba\_data\_files | file\_name, file\_id, tablespace\_name, relative\_fno, bytes, blocks, status |
| v$datafile | name, status |

***Objects***

|  |  |
| --- | --- |
| user\_objects | object\_name, object\_type, created, last\_ddl\_time, timestamp |
| all\_objects | object\_name, subobject\_name, object\_type, created |
| dba\_objects | owner, object\_name, subobject\_name, object\_type, created |
| user\_tab\_columns (cols) | table\_name, column\_name, data\_type, data\_precision, data\_scale, HIDDEN\_COLUMN , VIRTUAL\_COLUMN |
| user\_tables | table\_name, tablespace\_name, partitioned |

***Roles and Privileges***

|  |  |
| --- | --- |
| dba\_role\_privs | grantee, granted\_role, admin\_option, default\_role |
| dba\_sys\_privs | grantee, privilege, admin\_option |
| user\_role\_privs | username, granted\_role, admin\_option, default\_role, os\_granted |
| user\_tab\_privs | grantee, owner, table\_name, grantor, privilege, grantable |
| user\_col\_privs | owner, table\_name, column\_name, grantor, privilege, grantable |
| user\_sys\_privs | username, privilege, admin\_option |
| all\_tab\_privs | grantor, grantee, table\_schema, table\_name, privilege, grantable |
| all\_col\_privs | grantor, grantee, table\_schema, table\_name, column\_name, privilege, grantable |
| user\_tab\_privs\_recd | owner, table\_name, grantor, privilege, grantable, hierarchy |
| user\_col\_privs\_recd | owner, table\_name, column\_name, grantor, privilege, grantable |

***Constraints***

|  |  |
| --- | --- |
| user\_constraints | constraint\_name, constraint\_type, table\_name, owner |
| user\_cons\_columns | owner, constraint\_name, table\_name, column\_name, position |
| dba\_constraints | owner, constraint\_name, table\_name, column\_name, position |
| dba\_cons\_columns | owner, constraint\_name, table\_name, column\_name, position |
| all\_constraints | owner, constraint\_name, constraint\_type, table\_name |
| all\_cons\_columns | owner, constraint\_name, table\_name, column\_name, position |
| all\_ind\_columns | index\_owner, index\_name, table\_owner, table\_name, column\_name, column\_position |

***Users***

|  |  |
| --- | --- |
| user\_users | username, user\_id |
| all\_users | username, user\_id, created |
| dba\_users | username, user\_id, password, account\_status, lock\_date, expiry\_date, default\_tablespace, temporary\_tablespace, created, profile |

***Profile***

|  |  |
| --- | --- |
| product\_user\_profile | product, userid, attribute, char\_value |
| dba\_profiles | profile, resource\_name, resource\_type, limit |

***Partitions***

|  |  |
| --- | --- |
| user\_tab\_partitions | table\_name, partition\_name, tablespace\_name |
| dba\_tab\_partitions | table\_owner, table\_name, partition\_name, high\_value, tablespace\_name |
| user\_part\_tables | table\_name, partitioning\_type |
| dba\_part\_tables | owner, table\_name, partitioning\_type, subpartitioning\_type, partition\_count |

***Index***

|  |  |
| --- | --- |
| user\_indexes |  |
| user\_ind\_columns |  |

***Type***

|  |  |
| --- | --- |
| user\_types |  |
| all\_types |  |
| dbs\_types |  |

*SQL> SELECT* ***SYS\_CONTEXT ('USERENV','SESSION\_USER')*** *from DUAL;*

|  |
| --- |
| **SYS\_CONTEXT** |
| SYS\_CONTEXT ('USERENV','TERMINAL') |
| SYS\_CONTEXT ('USERENV','LANGUAGE') |
| SYS\_CONTEXT ('USERENV','SESSIONID') |
| SYS\_CONTEXT ('USERENV','INSTANCE') |
| SYS\_CONTEXT ('USERENV','ENTRYID') |
| SYS\_CONTEXT ('USERENV','ISDBA') |
| SYS\_CONTEXT ('USERENV','NLS\_TERRITORY') |
| SYS\_CONTEXT ('USERENV','NLS\_CURRENCY') |
| SYS\_CONTEXT ('USERENV','NLS\_CALENDAR') |
| SYS\_CONTEXT ('USERENV','NLS\_DATE\_FORMAT') |
| SYS\_CONTEXT ('USERENV','NLS\_DATE\_LANGUAGE') |
| SYS\_CONTEXT ('USERENV','NLS\_SORT') |
| SYS\_CONTEXT ('USERENV','CURRENT\_USER') |
| SYS\_CONTEXT ('USERENV','CURRENT\_USERID') |
| SYS\_CONTEXT ('USERENV','SESSION\_USER') |
| SYS\_CONTEXT ('USERENV','SESSION\_USERID') |
| SYS\_CONTEXT ('USERENV','PROXY\_USER') |
| SYS\_CONTEXT ('USERENV','PROXY\_USERID') |
| SYS\_CONTEXT ('USERENV','DB\_DOMAIN') |
| SYS\_CONTEXT ('USERENV','DB\_NAME') |
| SYS\_CONTEXT ('USERENV','HOST') |
| SYS\_CONTEXT ('USERENV','OS\_USER') |
| SYS\_CONTEXT ('USERENV','EXTERNAL\_NAME') |
| SYS\_CONTEXT ('USERENV','IP\_ADDRESS') |
| SYS\_CONTEXT ('USERENV','NETWORK\_PROTOCOL') |
| SYS\_CONTEXT ('USERENV','BG\_JOB\_ID') |
| SYS\_CONTEXT ('USERENV','FG\_JOB\_ID') |
| SYS\_CONTEXT ('USERENV','AUTHENTICATION\_TYPE') |
| SYS\_CONTEXT ('USERENV','AUTHENTICATION\_DATA') |
| SYS\_CONTEXT ('USERENV','CURRENT\_SQL') |
| SYS\_CONTEXT ('USERENV','CLIENT\_IDENTIFIER') |
| SYS\_CONTEXT ('USERENV','GLOBAL\_CONTEXT\_MEMORY') |

***IMP Notes***



* *The CREATE Table . . . . . . As SELECT . . . . . command will not work if one of the selected columns used the LONG datatype.*
* *Only the functions CONCAT, DECODE, DUMP, NVL, NVL2 and REPLACE can return non-NULL values when called with a NULL argument.*
* *Table in Oralce9i can have* ***1000******columns****.*
* *Table in Oralce9i can have* ***32*** *columns as a* ***PRIMARY KEY****.*
* *The* ***ORDER BY*** *clause cannot have more than* ***255*** *columns or expressions.*
* *The limit to level of* ***nesting*** *is* ***255*** *sub queries.*
* *The ORDER BY clause cannot be given in subqueties.*
* *The WHEN clause is* ***CASE*** *can be repeated for* ***128*** *times only.*
* *The DUAL table is a dummy table in Oracle with one column and one row.*
* *Using DESCRIPE on a stored program such as procedure or function show the parameters that need to be passed In / Out, its datatype and if there is a default value.*
* *The default length of a CHAR datatype column is 1 CHAR*
* *The default precision for fractional seconds in a TIMESTAMP datatype column is 6*
* *You can create PRIMARY KEY, FOREIGN KEY and UNIQUE KEY constraints on a view. The constraints on views are not enforced by Oracle. To enforce a constraint it must be defined on a table.*
* *When you DROP a table then Indexes, constraints, triggers and privileges on the table are also dropped.*
* *When you DROP a table then Oracle does not drop view, materialized views and other stored programs that reference the table.*
* *When you RENAME a table then Oracle automatically transfers integrity constraints, index and grants to the new table.*
* *You can RENAME tables, views, synonyms and stored procedures and functions.*
* *Environment functions (such as SYSDATE, USER, USERENV and UID) and Pseudo-columns (such as ROWNUM, CURRVAL, NEXTVAL and LEVEL) cannot be used to evaluate the check condition.*
* *You can only drop tables or index from read only tablespace.*
* *Data type or length of a table partitioning column may not be changed*
* *The last two digits of the current year are 00 to 49, then the first two digits as the current year*
* *The last two digits of the current year are 50 to 99, then the first two digits as the current year -1*
* *MAX and MIN function cannot be used with LOB of LONG data types.*

***Note: Updateable views cannot include:***

* *Set Operators (INTERSECT, MINUS, UNION, UNION ALL)*
* *DISTINCT*
* *Group Aggregate Functions (AVG, COUNT, MAX, MIN, SUM, etc.)*
* *GROUP BY Clause*
* *ORDER BY Clause*
* *CONNECT BY Clause*
* *START WITH Clause*
* *Collection Expression In A Select List*
* *Subquery In A Select List*
* *Join Query*

***Note: - If pseudo columns are present they cannot be included in an update statement.***

* *The names of Oracle identifiers, such as tables and columns, must not exceed 30 characters in length. The first character must be a letter, but the rest can be any combination of letters, numerals, dollar signs* ***($), pound signs (#), and underscores (\_).***
* *Synonym can be one of the following:*

1. *Table*
2. *Package*
3. *View*
4. *Materialized view*
5. *Sequence java class*
6. *Schema object*
7. *Stored procedure*
8. *User-defined object*
9. *Function*
10. *Synonym*

* ***Truncate will drop the entire table then recreate the structure.***
* ***In-Line VIEW*** *eg. SELECT \* from (SELECT \* from EMP);*
* ***'\*'*** *is called Meta character / wildcard.*

***SQL10g***



***Recyclebin***

*SQL> Show Recyclebin*

*SQL> ALTER SYSTEM SET RECYCLEBIN = OFF/ON;*

*SQL> ALTER SESSION SET RECYCLEBIN = OFF/ON;*

***Purge***

*SQL> PURGE Table EMP; (Drop table from recyclebin)*

*SQL> PURGE RECYCLEBIN; (Drop all table from recyclebin)*

*SQL> PURGE DBA\_RECYCLEBIN;*

*SQL> PURGE Tablespace tspace1;*

**Note: - If we DROP Table, it will get stored in RECYCLEBIN Data Dictionary.**

*SQL> DROP table EMP* ***cascade constraints purge;***

***Flashback Table***

* ***FLASHBACK TABLE test TO BEFORE DROP [RENAME TO test1]***

**Note: - Rollback Table from RECYCLEBIN Data Dictionary using FLASHBACK.**

*SQL > DROP table v$range1;*

*SQL > FLASHBACK table v$range1 to BEFORE DROP;*

*SQL > FLASHBACK table "BIN$hBJrmLxk49DgQAB/AQAmMA==$0" to BEFORE DROP;*

***Row level recovery using Flashback Query***

***Flashback Pseudo Columns***

* versions\_xid
* versions\_operation
* versions\_startscn
* versions\_starttime
* versions\_endscn
* versions\_endtime

***Show Parameter***

*SQL> Show parameter*

*SQL> ALTER SYSTEM SET RECYCLEBIN = OFF/ON;*

*SQL> ALTER SESSION SET RECYCLEBIN = OFF/ON;*

***Different SELECT statement***

*SQL>* *UPDATE EMP set job=DEFAULT; (where DEFAULT is constraint)*

*SQL>* *SELECT decode (grouping (job), 1,'All Jobs’, Job), sum (sal), max (sal) from EMP group by rollup (job) (GROUPING world only with ROLLUP & CUBE)*

*SQL>* *Set linesize 100*

*SQL>* *Store set my\_settings.sql replace*

*SQL>* *ALTER table v$LOCKTABLE disable table lock; (DDL will not work)*

*SQL>* *ALTER table V$LOCKTABLE enable table lock; (DDL will work)*

**Note: - A join is a query that combines rows from two or more tables or views. Oracle Database performs a join whenever multiple tables appear in the FROM clause of the query. The select list of the query can select any columns from any of these tables. If any two of these tables have a column name in common, then you must qualify all references to these columns throughout the query with table names to avoid ambiguity.**

*SQL> SELECT emp.deptno, dname, count (\*) from EMP, DEPT where emp.deptno = dept.deptno group by emp.deptno, dname;*

*DEPTNO DNAME COUNT (\*)*

*---------- ------------------- ---------------*

*10 ACCOUNTING 3*

*20 RESEARCH 5*

*30 SALES 6*

*SQL > SELECT \* from EMP* ***natural join*** *DEPT; (NATURAL JOIN)*

*SQL > SELECT \* from EMP* ***cross join*** *DEPT; (CROSS JOIN)*

*SQL > SELECT \* from EMP* ***inner join*** *DEPT on EMP.deptno = DEPT.deptno; (INNER JOIN)*

*SQL > SELECT \* from EMP* ***join*** *DEPT on emp.deptno=dept.deptno; (INNER JOIN)*

*SQL > SELECT \* from EMP* ***simple join*** *DEPT on emp.deptno=dept.deptno; (SIMPLE JOIN / INNER JOIN)*

*SQL > SELECT \* from EMP* ***simple join*** *DEPT using (deptno); (SIMPLE JOIN / INNER JOIN)*

**Note: - Equiejoins are also called simple join or inner join**

*SQL > SELECT emp.\*, dname, locationname from EMP* ***inner join*** *dept1 on emp.deptno = dept1.deptno inner join location on location.locationid = dept1.locationid; (NESTED INNER JOIN)*

*SQL > SELECT \* from EMP inner join DEPT* ***using (deptno)*** *(INNER JOIN WITH USING)*

*SQL > SELECT \* from EMP1 join DEPT1* ***using (deptno) where job='SALESMAN';*** *(INNER JOIN WITH USING & WHERE)*

*SQL > SELECT \* from EMP, DEPT where emp.deptno = dept.deptno* ***(+)****; (LEFT OUTER JOIN)*

*SQL > SELECT \* from EMP* ***left outer join*** *DEPT on emp.deptno=dept.deptno; (LEFT OUTER JOIN)*

*SQL > SELECT \* from EMP, DEPT where emp.deptno* ***(+)*** *= dept.deptno; (RIGHT OUTER JOIN)*

*SQL > SELECT \* from EMP* ***right outer join*** *DEPT on emp.deptno=dept.deptno; (RIGHT OUTER JOIN)*

*SQL > SELECT \* from EMP* ***full outer join*** *DEPT* ***using (deptno);*** *(FULL OUTER JOIN)*

*SQL > SELECT extract (timezone\_hour from date3) from v$datetime;*

*SQL > SELECT extract (day from sysdate-6) from dual;*

*SQL > SELECT extract (hour from systimestamp) from dual;*

*SQL > SELECT extract (timezone\_hour from systimestamp) from dual;*

*SQL >* ***Var x number****;*

*UPDATE EMP set comm = 100 where comm is NULL returning sum (comm) into* ***:x****;*

***Print :x OR Exec dbms\_output.put\_line (:x)***

*SQL >* ***Var*** *(To display user defined variables)*

***Eg. Var x varchar2 (10);***

***Eg. Exec :x := 'Saleel…';***

*SQL >* ***Define*** *(To display user defined, defined variables)*

*Eg Define x = ‘MANAGER’*

*SQL > Comment on table EMP is 'Important file’; (Add a comment to the table) (user\_tab\_comments)*

*SQL > Comment on table EMP is ' ’; (Remove a comment from the table) (user\_col\_comments)*

*SQL > ALTER session set nls\_currency ='Rs.';*

*Column sal heading "New | Salary" format 'L99,999.99'*

*SELECT \* from EMP;*

*SQL > ALTER session set recyclebin=OFF / ON;*

*SQL > SELECT* ***dbms\_metadata.get\_ddl ('TABLE','DEPT')*** *from DUAL;*

*SQL> SELECT* ***dbms\_metadata.get\_ddl ('TABLE', 'EMP', 'C##SALEEL')*** *from DUAL;*

*SQL > ALTER table EMP move tablespace tspace2;*

*SQL > SELECT ' ''' || SAL || ''' ' from EMP;* ***Output is*** *'****1000****'*

*SQL > INSERT into DEPT values (50,****'Sale''s Man'****,'Pune');*

*SQL > INSERT into DEPT values (99,* ***q'{O'Hara A}'****,'Pune');*

*SQL > SELECT ename ||* ***q'{'s}'*** *from EMP;*

*SQL > SELECT ename ||* ***q'['s]'*** *from EMP;*

*SQL > SELECT ename ||* ***q'<'s>'*** *from EMP;*

*SQL > SELECT ename ||* ***q'('s)'*** *from EMP;*

*SQL > SELECT* ***q'"Isn't this cool"'*** *as "R1" from DUAL;*

*SQL > Exec dbms\_output.put\_line (‘This is the test by Saleel.’)*

*SQL > INSERT into DEPT values (50,'Sale''s Man', 'PUNE')* ***returning rowid into :x;***

*SQL > UPDATE DEPT set deptno=50 where deptno=1* ***returning rowid into :x;***

*SQL > DELETE from DEPT where deptno=50* ***returning rowid into :x;***

*SQL > SELECT rownum, row\_number () over (order by sal desc) from EMP;*

*SQL > CREATE or REPLACE function* ***F1 (newSAL number)*** *return number as*

*xSal number;*

*begin*

*xSal := xSal \* 100;*

*return xSal;*

*end;*

*/*

*SQL > SELECT sal,* ***F1 (sal)*** *from EMP;*

*SQL >* ***Exec :xx := F1 (100);***

*SQL >* ***Call F1 (100) into :xx***

*SQL > SELECT ename, job FROM* ***(SELECT ename, job FROM EMP ORDER BY dbms\_random.value) WHERE rownum = 1;*** (Will display ename, job randomly)

*SQL > ALTER table DEPT drop constraint pk\_deptno* ***cascade;*** (Removes the Primary Key constraint and drop the associated foreign key constraint)

*SQL > SELECT to\_char (hiredate, 'Day') from EMP*

*Order by*

***Case when to\_char (hiredate,'fmDay') =‘Monday’ then 1***

***When to\_char (hiredate,'fmDay') =‘Tuesday’ then 2***

***When to\_char (hiredate,'fmDay') =‘Wednesday’ then 3***

***When to\_char (hiredate,'fmDay') ='Thursday' then 4***

***When to\_char (hiredate,'fmDay') ='Friday' then 5***

***When to\_char (hiredate,'fmDay') =‘Saturday’ then 6***

***When to\_char (hiredate,'fmDay') =‘Sunday’ then 7***

***End;*** (Order by weekday starting with Monday)

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Description** | **Example** |
| Ora\_client\_ ip\_address | Varchar | Returns the IP address of the client in a LOGON event, when the underlying protocol is TCP/IP | If ora\_sysevent=’LOGON’ then addr := ora\_client\_ip\_address; end if; |
| Ora\_database\_ name | Varchar2 (50) | Database Name | Declare db\_name varchar (50); begin db\_name := ora\_database\_name; end; |
| Ora\_des\_ encrypted\_ password | Varchar2 | The DES encrypted password of the user begin Created or Altered. | If ora\_dict\_obj\_type=’USER’ then insert into event\_table (ora\_des\_encrypted\_password); end if; |
| Ora\_dict\_obj\_ name | Varchar (30) | Name of the dictionary object on which the DDL operation occurred. | Insert into event\_table (‘Changed object is ‘ || ora\_dict\_obj\_name’); |
| Ora\_dict\_obj\_ name\_list (name\_list OUT ora\_name\_list\_t) | Binary\_ integer | Returns the list of object name of objects being modified in the event | If ora\_sysevent=’ ASSOCIATE STATISTICS’ then number\_modified := ora\_dict\_obj\_name\_list (name\_list); end if; |
| Ora\_dict\_obj\_ owner | Varchar (30) | Owner of the dictionary object on which the DDL operation occurred | Insert into event\_table (‘object owner is ’ || ora\_dict\_obj\_owner’); |
| Ora\_dict\_obj\_ owner\_list (owner\_list OUT ora\_name\_list\_t) | Binary\_ integer | Returns the list of object owner of object being modify in the event | If ora\_sysevent=’ ASSOCIATE STATISTICS’ then number\_of\_modified\_objects := ora\_dict\_obj\_owner\_list (owner\_list); end if; |
| Ora\_dict\_obj\_ type | Varchar (20) | Type of the dictionary object on which then DDL operation occurred. | Insert into event\_table (‘This object is a ‘ || ora\_dict\_obj\_type’); |
| Ora\_grantee (user\_list OUT ora\_name\_list\_t) | Binary\_ integer | Returns the grantees of a grant event in the OUT parameter; returns the number of grantees in the return value. | If ora\_sysevent=’GRANT’ then number\_of\_user := ora\_grantee (user\_list); end if; |
| Ora\_instance\_ num | Number | Instance number. | If ora\_instance\_num = 1 then insert into event\_table (‘1’); end if; |
| Ora\_is\_alter\_ column (column\_name is varchar2) | Boolean | Returns True if the specified column is Altered. | If ora\_sysevent =’ALTER’ and ora\_dict\_obj\_type=’TABLE’ then alter\_column := ora\_is\_alter\_column (‘FOO’); end if |
| Ora\_is\_drop\_column (column\_name in varchar2 | Boolean | Returns True if the specified column is Dropped. | If ora\_sysevent =’ALTER’ and ora\_dict\_obj\_type=’TABLE’ then drop\_column := ora\_is\_drop\_column (‘FOO’); end if |
| Ora\_is\_servererror | Boolean | Returns True if given error is on error stack, False otherwise | If ora\_is\_servererror (error\_number) then insert into event\_table (‘Server Error’); end if; |
| Ora\_login\_user | Varchar2 | Login user name | Select Ora\_login\_user from dual; |
| Ora\_sysevent | Varchar2 | System event firing the trigger: Event name is same as that in the syntax | Insert into event\_table (ora\_sysevents); |

***PL/SQL***



***PL/SQL is blocked programming language.***

1. ***Named block* (Procedure, Function, etc. which get stored in database.)**
2. ***Unnamed block* (Anonymous-block)**

***Syntax for declaring a variable is:***

***variable\_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial\_value]***

***EG.***

***declare***

***x number;***

***begin***

***x := 3\*\*3;***

***dbms\_output.put\_line(x);***

***end;***

***/***

***EG. Error In code (Variable name and column name is same)***

***declare***

***deptno number := 40;***

***begin***

***update dept set deptno = 1 where dept.deptno=deptno;***

***end;***

***/***

***EG.***

***declare***

***x number := '&Number';***

***o varchar2(1000);***

***begin***

***for xx in 1 .. length(x)***

***loop***

***if substr(x,xx,1) = 1 then***

***o := o || ('One ');***

***elsif substr(x, xx, 1) = 2 then***

***o := o || ('Two ');***

***elsif substr(x, xx, 1) = 3 then***

***o := o ||('Three ');***

***elsif substr(x, xx, 1) = 4 then***

***o := o || ('Four ');***

***elsif substr(x, xx, 1) = 5 then***

***o := o || ('Five ');***

***elsif substr(x, xx, 1) = 6 then***

***o := o || ('Six ');***

***elsif substr(x, xx,1) = 7 then***

***o := o || ('Seven ');***

***elsif substr(x, xx, 1) = 8 then***

***o := o || ('Eight ');***

***elsif substr(x, xx, 1) = 9 then***

***o := o ||('Nine ');***

***elsif substr(x, xx, 1) = 0 then***

***o := o || ('Zero ');***

***end if;***

***end loop;***

***dbms\_output.put\_line (o);***

***end;***

***/***

***Type of Cursor***

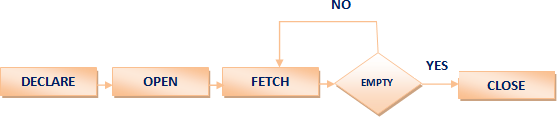
* *Implicit:-*

***If oracle engine open a cursor for its internal processing then it is known as implicit cursor. A simple SELECT....... INTO, UPDATE, DELETE etc, are the examples of implicit cursor.***

* *Explicit:-*

***The following points to be remembered about cursor.***

1. *Do not open an already open cursor.*
2. *If you do not use the loop in the 1st, then the fetch would have retrieved only one row.*
3. *Need to close an open cursor explicitly otherwise it will exceed the limit of "open\_cursor" default value 50 and hence "too\_many\_open\_cursor" error.*
4. *Use exit when cursor\_name%NotFound immediately after fetch statement this will avoid ORA-1002, fetch out of sequence error.*
5. *Don't fetch from already close cursor this will return the ORA-1002, ORA-1001 invalid cursor.*
6. *The "i" in cursor for loop "for i in c\_emp loop" is declared internally and data type is c\_emp%rowtype.*
7. *Column name is specified in FOR update clause, if no column name specified then any column in the underlying table of the cursor can be modified.*
8. *For this example given sql column (sql) is specified and hence sal only can be modified.*



***DECLARE*** *the cursor in the declaration section.*

***OPEN*** *the cursor in the Execution Section.*

***LOOP***

***FETCH*** *the data from cursor into PL/SQL variables or*

***Exit*** *when*

***END LOOP***

***CLOSE*** *the cursor in the Execution Section before you end the PL/SQL Block.*

***The AS keyword is used instead of the IS keyword for creating standalone procedure.***

***IF Condition (in, like, and between)***

*declare*

*x1 number := 1;*

*str varchar2(10) := 'saleel';*

*begin*

***if x1 between 1 and 10 then***

*dbms\_output.put\_line('between');*

*end if;*

***if x1 in (1,2 ,3, 4, 5) then***

*dbms\_output.put\_line ('in');*

*end if;*

***if str like 's%' then***

*dbms\_output.put\_line ('like');*

*end if;*

*end;*

*/*

***Cursor Syntax***

*CURSOR cursor\_name*

*IS*

*SELECT statement*

*FOR UPDATE [OF column\_list] [NOWAIT];*

***Cursor Attributes***

|  |  |  |
| --- | --- | --- |
| ***Attributes*** | | |
| *Found* | *SQL%Found* | *C1%Found* |
| *NotFound* | *SQL%NotFound* | *C1%NotFound* |
| *RowCount* | *SQL%RowCount* | *C1%RowCount* |
| *IsOpen* | *SQL%IsOpen* | *C1%IsOpen* |
|  | *SQL%BULK\_ROWCOUNT* |  |
|  | *SQL%BULK\_EXCEPTIONS* |  |

***A record variable declared with %ROWTYPE or %TYPE, the initial value of each field is NULL***

***SQL%FOUND returns:***

*NULL if no SELECT or DML statement has run*

*TRUE if a SELECT statement returned one or more rows or a DML statement affected one or more rows*

*FALSE otherwise*

***SQL%NOTFOUND (the logical opposite of SQL%FOUND) returns:***

*NULL if no SELECT or DML statement has run*

*FALSE if a SELECT statement returned one or more rows or a DML statement affected one or more rows*

*TRUE otherwise*

***SQL%ROWCOUNT returns:***

*NULL if no SELECT or DML statement has run*

*Otherwise, the number of rows returned by a SELECT statement or affected by a DML statement (a PLS\_INTEGER)*

***EG.***

*begin*

***UPDATE EMP set COMM = -1 where DEPTNO=10;***

*if* ***SQL%RowCount > 8*** *then*

***ROLLBACK;***

*end if;*

*end;*

*/*

***where current of c1***

***NOTE:*** *The WHERE CURRENT OF clause is used in some* ***UPDATE and DELETE statements.***

***UPDATE table\_name***

***SET set\_clause***

***WHERE CURRENT OF cursor\_name;***

***DELETE***

***FROM table\_name***

***WHERE CURRENT OF cursor\_name;***

***EG for UPDATE***

*declare*

***cursor c1 is select \* from EMP where comm is null for update of comm;***

*x number;*

*begin*

*for xRow in c1*

*loop*

*if xRow.deptno = 10 then*

*x := -1;*

*elsif xRow.deptno=20 then*

*x := -2;*

*elsif xRow.deptno=30 then*

*x := -3;*

*end if;*

***UPDATE EMP set comm = x where current of c1;***

*end loop;*

*end;*

*/*

***EG for DELETE***

*declare*

*cursor c1 is select \* from EMP3* ***where deptno=20 for UPDATE of deptno;***

*xRecord EMP3%RowType;*

*begin*

*open c1;*

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

*delete from EMP3* ***where current of c1;***

*dbms\_output.put\_line (xRecord.ename || ' ' || xRecord.deptno);*

*end loop;*

*end;*

*/*

***Current of with UPDATE statement***

*declare*

***CURSOR c1 is SELECT ROWID, sal from EMP where deptno = 30 FOR UPDATE;***

*xRecord c1%RowType;*

*begin*

*open c1;*

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

***/\**** *if xRecord.sal < 1500 then*

***UPDATE EMP set sal = -1 where ROWID = xRecord.rowid;***

*else*

***UPDATE EMP set sal = -2 where ROWID = xRecord.rowid;***

*end if;* ***\*/***

*if xRecord.sal < 1500 then*

***UPDATE EMP set sal = -1 where current of c1;***

*else*

***UPDATE EMP set sal = -2 where current of c1;***

*end if;*

*end loop;*

*close c1;*

*end;*

*/*

***Exceptions***

|  |  |  |
| --- | --- | --- |
| ***Exception*** | ***Oracle Error*** | ***SQLCODE Value*** |
| *ACCESS\_INTO\_NULL* | *ORA-06530* | ***-6530*** |
| *COLLECTION\_IS\_NULL* | *ORA-06531* | ***-6531*** |
| *CURSOR\_ALREADY\_OPEN* | *ORA-06511* | ***-6511*** |
| *DUP\_VAL\_ON\_INDEX* | *ORA-00001* | ***-1*** |
| *INVALID\_CURSOR* | *ORA-01001* | ***-1001*** |
| *INVALID\_NUMBER* | *ORA-01722* | ***-1722*** |
| *LOGIN\_DENIED* | *ORA-01017* | ***-1017*** |
| *NO\_DATA\_FOUND* | *ORA-01403 + 100* | ***+100*** |
| *NOT\_LOGGED\_ON* | *ORA-01012 1012* | ***-1012*** |
| *PROGRAM\_ERROR* | *ORA-06501* | ***-6501*** |
| *ROWTYPE\_MISMATCH* | *ORA-06504* | ***-6504*** |
| *SELF\_IS\_NULL* | *ORA-30625* | ***-30625*** |
| *STORAGE\_ERROR* | *ORA-06500* | ***-6500*** |
| *SUBSCRIPT\_BEYOND\_COUNT* | *ORA-06533* | ***-6533*** |
| *SUBSCRIPT\_OUTSIDE\_LIMIT* | *ORA-06532* | ***-6532*** |
| *SYS\_INVALID\_ROWID* | *ORA-01410* | ***-1410*** |
| *TIMEOUT\_ON\_RESOURCE* | *ORA-00051* | ***-51*** |
| *TOO\_MANY\_ROWS* | *ORA-01422* | ***-1422*** |
| *TRANSACTION\_BACKED\_OUT* | *ORA-00061* | ***61*** |
| *VALUE\_ERROR* | *ORA-06502* | ***-6502*** |
| *ZERO\_DIVIDE* | *ORA-01476* | ***-1476*** |
| *CASE\_NOT\_FOUND* | *ORA-06592* | ***-6592*** |

**Note: - You cannot declare an exception twice in the same block. You can, however, declare the same exception in two different blocks.**

***INVALID\_CURSOR Exception***

***ORA-01001: invalid cursor error occurs when you tried to reference a cursor that does not yet exist.***

*A few scenarios given below.*

*1. FETCH cursor before opening the cursor.*

*2. CLOSE cursor before opening the cursor.*

*3. FETCH cursor after closing the cursor.*

***EG.***

*SQL> CREATE table TEMP (c1 number not null);*

*SQL> INSERT into TEMP values (null);*

*INSERT into TEMP values (null)*

*\**

*ERROR at line 1:*

*ORA-01400: cannot insert NULL into ("C##SALEEL"." TEMP "."C1")*

***PRAGMA EXCEPTION\_INIT***

*DECLARE*

***NULL\_VALUES EXCEPTION;***

***PRAGMA EXCEPTION\_INIT (NULL\_VALUES, -1400);***

*BEGIN*

*INSERT INTO TEMP VALUES (NULL);*

*dbms\_output.put\_line ('null value not authorized 1');*

*EXCEPTION*

*WHEN* ***null\_values*** *THEN*

*dbms\_output.put\_line ('null value not authorized 2 ' || sqlcode);*

*END;*

*/*

**Dbms\_output.put\_line**

***\* Exec dbms\_output.disable***

***\* Exec dbms\_output.enable***

***Transaction Block (Works with exceptions only)***

***EG.1***

*declare*

*x number;*

*begin*

***SAVEPOINT sp1;***

*INSERT into dept values (1, 1, 1, 1);*

*INSERT into dept values (1, 1, 1, 1);*

***COMMIT;***

*exception*

*when others then*

***ROLLBACK to sp1;***

*dbms\_output.put\_line (dbms\_utility.format\_error\_stack);*

*end;*

*/*

***EG.2***

*declare*

*str varchar2(100);*

*begin*

***SAVEPOINT sp1;***

*str := 'INSERT into DEMO values (:p1, :p2, :p3, :p4)';*

*execute immediate str using 1, 1, 1, 1;*

*str := 'INSERT into DEPT values (:p1, :p2, :p3, :p4)';*

*execute immediate str using 1,1,1,1;*

***COMMIT;***

*exception*

*when others then*

*dbms\_output.put\_line('others exception raised....');*

***ROLLBACK TO SAVEPOINT SP1;***

*end;*

*/*

***Raise\_application\_error***

***EG.***

*Raise\_application\_error (-20001,’Message’);*

*begin*

*RAISE\_APPLICATION\_ERROR (-20000, 'Force and exception');*

*exception*

*WHEN OTHERS THEN*

*dbms\_output.put\_line ('Force and exception');*

*end;*

*/*

***Note: - The error number which must be between (–20000 and –20999)***

***SQLCODE***

**Note: - SQLCODE is an error function that returns the latest error number, but outside of an exception handle. SQLCODE returns the value of the error number for the last error encountered.**

***dbms\_utility***

***dbms\_utility.exec\_ddl\_statement ('Create table EMP (empno number, ename varchar2 (10))');***

***dbms\_utility.format\_call\_stack***

***dbms\_utility.format\_error\_backtrace***

***dbms\_utility.format\_error\_stack (SQLERRM)***

***PLS\_INTEGER with RANGE option***

***EG.***

*declare*

***lv\_integer PLS\_INTEGER RANGE 5..10; -- will only store values between 5 to 10.***

*begin*

***lv\_integer := 91; --an exception will be generated as the value is outside the limit.***

*dbms\_output.put\_line (lv\_integer);*

*exception*

*WHEN value\_error THEN*

*dbms\_output.put\_line ('error'); --this gets printed.*

*end;*

*/*

***Cursor for Loop***

***EG.***

*begin*

***for x in (SELECT \* from EMP)***

*Loop*

*dbms\_output.put\_line* ***(x.ename);***

*End loop;*

*end;*

*/*

***EG.***

*declare*

***cursor c1 is SELECT \* from EMP;***

*begin*

***for x in c1***

*loop*

*dbms\_output.put\_line* ***(x.ename);***

*end loop;*

*end;*

*/*

***EG.***

*CREATE or REPLACE PROCEDURE PRO1* ***(deptno number)*** *is*

***cursor c1 (para1 number) is SELECT \* from EMP where deptno = para1;***

*begin*

***for xRecord in c1 (deptno)***

*loop*

*dbms\_output.put\_line (xRecord.ename ||' ' || xRecord.job ||' ' || xRecord.deptno);*

*end loop;*

*end;*

*/*

***EG.***

*begin*

***for d in (SELECT \* from dept)***

*loop*

***for e in (SELECT \* from EMP where deptno=d.deptno)***

*loop*

*dbms\_output.put\_line (‘Employee ' || e.ename || ' in department ' || d.dname);*

*end loop;*

*end loop;*

*end;*

*/*

***Collection’s (Record)***

***TYPE type\_name IS RECORD***

***(field1 data\_type1 [NOT NULL] := [DEFAULT VALUE],***

***field2 data\_type2 [NOT NULL] := [DEFAULT VALUE],***

***...***

***fieldn data\_type3 [NOT NULL] := [DEFAULT VALUE]***

***);***

***EG. For INSERT***

*declare*

***TYPE RecordType is record (***

***C1 NUMBER,***

***C2 NUMBER,***

***C3 NUMBER,***

***C4 NUMBER***

***);***

***xRecord RecordType;***

*begin*

*for x in 1..20*

*loop*

***xRecord.c1 := x;***

***INSERT into temp values xRecord;***

*end loop;*

*end;*

*/*

***EG. For UPDATE***

*declare*

***TYPE TypeRecord is RECORD (***

***C1 NUMBER,***

***C2 VARCHAR2 (25),***

***C3 VARCHAR2 (25),***

***C4 VARCHAR2 (25)***

***);***

*xRecord TypeRecord;*

*begin*

*for i in 1..20*

*loop*

*xRecord.c1 := i;*

*xRecord.c2 := 'File ' || i;*

*xRecord.c3 := 'Uploaded on ' || (sysdate + i);*

*xRecord.c4 := 'Image File ' || (i\*10) || '.jpeg';*

***UPDATE temp set ROW = xRecord where c1 = i;***

*--INSERT into temp values xRecord;*

*end loop;*

*end;*

*/*

***EG. For UPDATE***

*declare*

*xRow dept%RowType;*

*begin*

*xRow.deptno := 1;*

*xRow.dname := 1;*

*xRow.loc := 1;*

***UPDATE DEPT set Row = xRow where deptno=40;***

*end;*

*/*

***EG.***

*declare*

***TYPE empRecord is RECORD (***

***Empno emp.empno%Type,***

***Ename emp.ename%Type***

***);***

***xRecord empRecord;***

*begin*

*SELECT empno, ename into* ***xRecord*** *from EMP where empno=7788;*

*dbms\_output.put\_line* ***(xRecord.empno);***

*dbms\_output.put\_line* ***(xRecord.ename);***

*end;*

*/*

***EG.***

*declare*

***TYPE deptRecord is RECORD (***

***deptno number,***

***dname varchar2 (20),***

***loc varchar2 (20),***

***walletid number);***

***xRecord deptRecord;***

*begin*

***xRecord.deptno := 1;***

***xRecord.dname := 1;***

***xRecord.loc := 1;***

***xRecord.walletid := 7;***

*INSERT into dept* ***values xRecord;***

*end;*

*/*

***EG.***

*declare*

***TYPE EMPRecord is record (***

***c1 number not null :=11,***

***c2 varchar2 (10) default 'saleel',***

***c3 varchar2 (12),***

***c4 number***

***);***

***xRecord EMPRecord;***

*STR varchar2 (1000);*

*begin*

*xRecord.c3 := 'Pune';*

*xRecord.c4 := 101;*

*STR :='INSERT into DEPT values (:para1, :para2, :para3, :para4)';*

***execute immediate STR using xRecord.c1, xRecord.c2, xRecord.c3, xRecord.c4;***

*end;*

*/*

***Subtype Variables***

***EG.***

*declare*

***subtype x is varchar2 (100);***

***y x;***

*begin*

*y := 'a';*

*Exception*

*When others then*

*dbms\_output.put\_line* ***(dbms\_utility.format\_error\_stack);***

*end;*

*/*

***Cursor BULK COLLECT***

***EG.***

*declare*

***TYPE Type1 IS TABLE OF EMP%ROWTYPE INDEX BY BINARY\_INTEGER;***

***Obj Type1;***

*begin*

*SELECT \** ***bulk collect*** *into obj from EMP;*

*for elements in 1 .. obj.count*

*loop*

*dbms\_output.put\_line* ***(Obj (elements).ENAME || ' ' || Obj (elements).SAL);***

*end loop;*

*end;*

*/*

***EG.***

*declare*

***TYPE EMPRecord is RECORD (***

*EMPNO NUMBER (4), ENAME VARCHAR2 (12),*

*JOB VARCHAR2 (18), MGR NUMBER (4),*

*HIREDATE DATE, SAL NUMBER (7, 2),*

*COMM NUMBER (7, 2), DEPTNO NUMBER (4),*

*HOBBYID NUMBER (3), WALLETID NUMBER (5),*

*TOTAL NUMBER*

*);*

***TYPE EMPTable is table of EMPRecord;***

***newEMPTable EMPTable;***

***--xRecord EMPRecord;***

*STR varchar2 (1000);*

*begin*

*STR := 'SELECT \* from EMP1';*

***execute immediate STR bulk collect into newEMPTable;***

*for i in newEMPTable.first .. newEMPTable.last*

*loop*

*dbms\_output.put\_line (newEMPTable (i).ENAME);*

*end loop;*

*exception*

*when others then*

*dbms\_output.put\_line (dbms\_utility.format\_error\_stack);*

*end;*

*/*

***EG.***

*declare*

*cursor c1 is SELECT \* from emp1;*

***TYPE EMPTable is table of EMP1%RowType;***

***newTable EMPTable;***

*begin*

*open c1;*

*loop*

***fetch c1 bulk collect into newTable limit 10 ;***

***for i in 1.. newTable.count***

*loop*

***dbms\_output.put\_line (i || ' ' || newTable (i).ename);***

*end loop;*

*dbms\_output.put\_line ('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_' ||* ***newTable.count****);*

*exit when c1%NotFound;*

*end loop;*

*close c1;*

*exception*

*when others then*

*dbms\_output.put\_line (dbms\_utility.format\_error\_stack);*

*end;*

*/*

***EG.***

*Declare*

***TYPE tmp\_tbm is table of EMP%Rowtype;***

***emp\_tbl tmp\_tbm;***

*Begin*

***execute immediate 'SELECT \* FROM EMP' bulk collect INTO emp\_tbl;***

*For i in 1 .. Emp\_tbl.COUNT loop*

*dbms\_output.put\_line* ***(emp\_tbl (i).ename || ' ' || emp\_tbl (i).job);***

*End loop;*

*End;*

*/*

***EG.***

*declare*

***TYPE EMPRecord is record*** *(*

*EMPNO NUMBER (4), ENAME VARCHAR2 (12),*

*JOB VARCHAR2 (12), MGR NUMBER (4),*

*HIREDATE DATE, SAL NUMBER (7, 2),*

*COMM NUMBER (7, 2), DEPTNO NUMBER (2),*

*WALLETID NUMBER (5), TOTAL NUMBER,*

*deptDEPTNO NUMBER, deptDNAME varchar2 (12),*

*deptLOC varchar2 (12), deptWALLETID NUMBER*

*);*

***TYPE EMPTable is table of EMPRecord;***

***newTable EMPTable;***

*begin*

*SELECT \** ***bulk collect into newTable*** *from EMP, DEPT where emp.deptno=dept.deptno;*

***for i in 1.. newTable.count***

*loop*

*dbms\_output.put\_line* ***(newTable (i).ENAME || newTable (i).deptDNAME);***

*end loop;*

*end;*

***EG.***

*declare*

***TYPE EMPRecord is record (***

*EMPNO NUMBER (4), ENAME VARCHAR2 (12),*

*JOB VARCHAR2 (18), MGR NUMBER (4),*

*HIREDATE DATE, SAL NUMBER (7, 2),*

*COMM NUMBER (7, 2), DEPTNO NUMBER (4),*

*HOBBYID NUMBER (3), WALLETID NUMBER (5),*

*TOTAL NUMBER*

*);*

***TYPE EMPTable is table of EMPRecord;***

***newTable EMPTable;***

*cursor c1 is SELECT \* from EMP1;*

*begin*

*open c1;*

*loop*

***fetch c1 bulk collect into newTable;***

***for i in 1..newTable.count***

*loop*

*dbms\_output.put\_line* ***(newTable (i).ename);***

*end loop;*

*exit when c1%NotFound;*

*end loop;*

*close c1;*

*end;*

*/*

***EG.***

*CREATE or REPLACE PROCEDURE PL1 IS*

***TYPE t\_tab IS TABLE OF EMP.deptno%TYPE;***

***l\_tab t\_tab;***

*begin*

*UPDATE EMP*

*SET deptno = rownum* ***RETURNING deptno BULK COLLECT INTO l\_tab;***

***for i IN l\_tab.first .. l\_tab.last loop***

***dbms\_output.put\_line ('UPDATE ID=' || l\_tab (i));***

***end loop;***

*end PL1;*

*/*

***forall in BULK COLLECT***

***EG.***

*declare*

*TYPE EMPTable is table of EMP1%ROWTYPE;*

*c1 EMPTable;*

*begin*

*SELECT \* BULK COLLECT INTO c1 from EMP1;*

***forall x in c1.First .. c1.Last***

***INSERT into TEMP values c1(x);***

*end;*

*/*

***dbms\_assert.enquote\_literal***

***EG.***

*CREATE or REPLACE Procedure INSERETRECORD (dno number, dnm varchar2, lo varchar2, wid number) is*

*Vsql varchar2 (100);*

*begin*

*Vsql := 'INSERT into DEPT values (:dno,’;*

***Vsql := vsql || dbms\_assert.enquote\_literal (dnm) || ',';***

***Vsql := vsql || dbms\_assert.enquote\_literal (lo) || ',';***

*Vsql := Vsql || wid || ')';*

*dbms\_output.put\_line (Vsql);*

*exception when others then*

*dbms\_output.put\_line* ***(dbms\_utility.format\_error\_stack);***

*end INSERETRECORD;*

*/*

***Function returns %RowType***

***EG.***

*CREATE or REPLACE FUNCTION F1(X number)* ***return EMP%RowType*** *is*

*xRow EMP%Rowtype;*

*begin*

***SELECT \* into xRow from EMP where empno = x;***

*If SQL%Found then*

***return (xRow);***

*End IF;*

*exception*

*when no\_data\_found then*

***return (null);***

*end F1;*

*/*

**How to call**

*Declare*

*xRow1 EMP%RowType;*

*begin*

***xRow1 := F1 (7788);***

*dbms\_output.put\_line (xRow1.ename);*

*end;*

*/*

***REF CURSOR***

***Strongly Typed***

***EG.***

*declare*

***TYPE emp1 is ref cursor return EMP%Rowtype;***

***c1 emp1;***

***xRecord EMP%RowType;***

*begin*

***open c1 for SELECT \* from EMP;***

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

*dbms\_output.put\_line (xRecord.ename);*

*end loop;*

*close c1;*

*end;*

*/*

***Weakly Typed***

***EG.***

*declare*

***TYPE emp1 is ref cursor;***

***c1 emp1;***

***xRecord EMP%RowType;***

*begin*

***open c1 for SELECT \* from EMP;***

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

*dbms\_output.put\_line (xRecord.ename ||' ' || xRecord.job);*

*end loop;*

*close c1;*

*end;*

*/*

***EG.***

***CREATE or REPLACE procedure PL1 (varDeptno number) is***

***TYPE empCursor is ref cursor;***

***c1 empCursor****;*

*str varchar2(1000);*

*begin*

***--str := 'SELECT \* from emp where deptno = :para1';***

***open c1 for 'SELECT \* from EMP where deptno = :para1' using varDeptno;***

*end PL1;*

*/*

***EG.***

*CREATE or REPLACE procedure PRO1* ***(STR varchar2)*** *is*

*xRow emp%RowType;*

***TYPE MYCURSOR is ref cursor;***

***C1 MYCURSOR;***

*BEGIN*

***open C1 for str;***

*loop*

*fetch C1 into xRow;*

*exit when c1%NotFound;*

*dbms\_output.put\_line (xRow.ename ||' ' || xRow.job || ' ' || xRow.deptno);*

*end loop;*

*close C1;*

*END;*

*/*

***SYS\_REFCURSOR***

***EG.***

*declare*

***TYPE empRecord is record (***

*varEmpno number, varEname varchar2 (12),*

*varJob varchar2 (12), varMgr number,*

*varHiredate date, varSal number,*

*varComm number, varDeptno number,*

*varWalletid number, varTotal number*

*);*

***xRecord empRecord;***

***c1 sys\_refcursor;***

*str varchar2 (1000);*

*begin*

***str := 'SELECT \* from EMP where deptno = :para1';***

***open c1 for str using 20;***

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

*dbms\_output.put\_line (xRecord.varEname);*

*end loop;*

*close c1;*

*end;*

*/*

***EG.How to return sys\_refcursor from FUNCTION***

***EG.***

*CREATE or REPLACE function F1 return* ***SYS\_REFCURSOR*** *is*

***myCursor SYS\_REFCURSOR;***

*begin*

***open myCursor for SELECT \* from EMP;***

***return myCursor;***

*end F1;*

*/*

**How to call**

*declare*

***xx SYS\_REFCURSOR;***

***xRow EMP%Rowtype;***

*begin*

***xx := F1;***

*loop*

***fetch xx into xRow;***

*exit when xx%NotFound;*

*dbms\_output.put\_line (xRow.ENAME);*

*End loop;*

*end;*

*/*

***EG.***

*CREATE or REPLACE function F1(x varchar2) return* ***SYS\_REFCURSOR*** *is*

***C1 SYS\_REFCURSOR;***

*begin*

***open c1 for 'SELECT \* from :para1' using x;***

***return c1;***

*end F1;*

*/*

*CREATE or REPLACE function F1(x number) return* ***sys\_refcursor*** *is*

***c1 sys\_refcursor;***

*str varchar2 (1000);*

*begin*

***str := 'SELECT \* from EMP where deptno = :para1';***

***open c1 for str using x;***

***return (c1);***

*end F1;*

*/*

***EG.***

*declare*

***TYPE EMP is record (***

*varEMPNO NUMBER (4), varENAME VARCHAR2 (12),*

*varJOB VARCHAR2 (12), varMGR NUMBER (4),*

*varHIREDATE DATE, varSAL NUMBER (7, 2),*

*varCOMM NUMBER (7, 2), varDEPTNO NUMBER (2),*

*varWALLETID NUMBER (5), varTOTAL NUMBER*

*);*

***c1 sys\_refcursor;***

***xRecord EMP;***

*begin*

***c1 := F1 (10);***

*loop*

*fetch c1 into xRecord;*

*exit when c1%notFound;*

*dbms\_output.put\_line (xRecord.varENAME);*

*end loop;*

*end;*

*/*

***EG.How to pass sys\_refcursor in PROCEDURE***

*CREATE or REPLACE procedure PL1* ***(x sys\_refcursor)*** *is*

*xRecord emp%RowType;*

*begin*

*loop*

***fetch x into xRecord;***

*exit when x%NotFound;*

*dbms\_output.put\_line (xRecord.ENAME);*

*end loop;*

*end PL1;*

*/*

**How to call**

*declare*

***c1 sys\_refcursor;***

*begin*

***open c1 for SELECT \* from EMP;***

***pl1(c1);***

*end;*

*/*

***dbms\_sql.return\_result***

***EG.***

*CREATE or REPLACE procedure PL1 (x in number) is*

***c1 sys\_refcursor;***

*STR varchar2 (1000);*

*begin*

*STR := 'SELECT \* from EMP1 where DEPTNO =* ***:para1';***

***open c1 for str using x;***

***dbms\_sql.return\_result (c1);***

*end PL1;*

*/*

***CALL***

*begin*

***pl1 (10);***

*end;*

*/*

***ACCESSIBLE BY clause***

*The* ***ACCESSIBLE BY*** *clause can be added to* ***packages, procedures, functions and types*** *to specify which objects are able to reference the PL/SQL object directly.*

***EG.***

*CREATE or replace procedure PRO1* ***ACCESSIBLE BY (PRO2)*** *is*

*begin*

*dbms\_output.put\_line ('ACCESSIBLE BY clause demo’);*

*end;*

*/*

*CREATE or REPLACE procedure* ***PRO2*** *is*

*begin*

*PRO1;*

*end;*

*/*

***Procedure***

***EG.***

*CREATE or REPLACE PROCEDURE PRO1 (C1 number, C2 VARCHAR2, C3 VARCHAR2) IS*

***xRow1 T1%ROWTYPE;***

*begin*

***xRow1.deptno := C1;***

***xRow1.dname := C2;***

***xRow1.loc := C3;***

***INSERT into T1 values xRow1;***

*end PRO1;*

*/*

***EG.***

*CREATE or REPLACE PROCEDURE* ***REMOVE\_DEPT (X NUMBER)*** *AS*

*BEGIN*

*DELETE FROM dept WHERE dept.deptno =* ***REMOVE\_DEPT.X;***

*END;*

*/*

***Procedure / Function Overloading***

***EG.***

*declare*

*x1 number := 1;*

*x2 number := 2;*

***PROCEDURE P1 (c1 number) is***

*begin*

*dbms\_output.put\_line (c1+1);*

*end p1;*

***PROCEDURE P1(c1 number, c2 number) is***

*begin*

*dbms\_output.put\_line (c1 + c2);*

*end p1;*

*begin*

*P1(x1);*

*P1(x1, x2);*

*end;*

*/*

***EXECUTE IMMEDIATE***

***SQL> GRANT CREATE TABLE to c##saleel; -- If execute immediate is not working***

***EXECUTE IMMEDIATE Syntax***

***EXECUTE IMMEDIATE dynamic\_string***

***[INTO {define\_variable [, define\_variable]... | record}]***

***[USING [IN | OUT | IN OUT] bind\_argument***

***[, [IN | OUT | IN OUT] bind\_argument]...]***

***[{RETURNING | RETURN} INTO bind\_argument [, bind\_argument]...];***

***INTO ...***

*Used only for single-row queries, this clause specifies the variables or record into which column values are retrieved. For each value retrieved by the query, there must be a corresponding, type-compatible variable or field in the INTO clause.*

***RETURNING INTO ...***

*Used only for DML statements that have a RETURNING clause (without a BULK COLLECT clause), this clause specifies the bind variables into which column values are returned. For each value returned by the DML statement, there must be a corresponding, type-compatible variable in the RETURNING INTO clause.*

***EG.1***

*declare*

***ROW EMP%RowType;***

*begin*

***Execute immediate 'SELECT \* from EMP where empno = 7788' INTO ROW;***

*dbms\_output.put\_line (row.ename);*

*end;*

*/*

***EG.2***

*declare*

***xRow emp%RowType;***

*STR varchar2 (100);*

*begin*

***STR := 'SELECT \* from EMP where empno=7788';***

***execute immediate STR into xRow;***

*dbms\_output.put\_line (xRow.ename ||' ' || xRow.job ||' ' || xRow.sal);*

*end;*

*/*

***EG.2***

*CREATE OR REPLACE* ***FUNCTION*** *F1 RETURN EMP%ROWTYPE AS*

*STR1 VARCHAR2 (100);*

*X EMP%ROWTYPE;*

*BEGIN*

*STR1 := 'SELECT \* FROM EMP WHERE EMPNO=7788';*

***EXECUTE IMMEDIATE STR1 INTO X ;***

*RETURN(X);*

*END;*

*/*

*DECLARE*

*X1 EMP%ROWTYPE;*

*BEGIN*

*X1:= F1;*

*dbms\_output.put\_line (x1.ename);*

*END;*

*/*

***EG. (Table name more than 10 records)***

*declare*

***CURSOR c1 is SELECT tname from TAB where tabtype='TABLE' order by 1;***

*xRecord tab.tname%Type;*

*xx number;*

*begin*

*open c1;*

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

***execute immediate 'SELECT count (\*) from ' || xRecord into xx;***

*if xx >=10 then*

*dbms\_output.put\_line (xRecord || ' has -> ' || xx || ' Record(s)');*

*end if;*

*end loop;*

*close c1;*

*end;*

*/*

***EG. (Table name more than 10 records)***

*declare*

*xx number;*

*begin*

***for xRecord in (SELECT tname from TAB where tabtype='TABLE' order by 1)***

*loop*

***execute immediate 'SELECT count (\*) from ' || xRecord.tname into xx;***

*if xx >=10 then*

*dbms\_output.put\_line (xRecord.tname || ' has -> ' || xx || ' Record(s)');*

*end if;*

*end loop;*

*end;*

*/*

***EG. (Drop all Table)***

*begin*

*for xRow in (SELECT TABLE\_NAME from user\_tables where TABLE\_NAME like 'Z%')*

*loop*

***execute immediate*** *(****'DROP table ' || xRow.table\_name || ' cascade constraints purge');***

*end loop;*

*end;*

*/*

***EG. (Drop all Table)***

*CREATE OR REPLACE PROCEDURE PRO1* ***(C CHAR)*** *IS*

*TYPE MYCURSOR IS REF CURSOR;*

*C1 MYCURSOR;*

*XROW USER\_TABLES.TABLE\_NAME%TYPE;*

*BEGIN*

*OPEN C1 FOR 'SELECT TABLE\_NAME FROM USER\_TABLES WHERE* ***TABLE\_NAME LIKE :P' USING C ||'%';***

*LOOP*

*FETCH C1 INTO XROW;*

*EXIT WHEN C1%NOTFOUND;*

***execute immediate 'DROP table ' || xrow || ' cascade constraints purge';***

*END LOOP;*

*CLOSE C1;*

*END;*

*/*

***EG.***

*declare*

***rows1 rowid;***

*STR varchar2 (100);*

*begin*

*STR :=* ***'INSERT*** *into DEPT (deptno) values (1)* ***returning rowid into :rows1' ;***

***execute immediate STR RETURNING into rows;***

*dbms\_output.put\_line (rows);*

*end;*

*/*

***EG.***

*declare*

*SQL1 varchar2 (100);*

***x rowid;***

*id number;*

*begin*

***SELECT max (deptno) + 1 into id from DEPT;***

***SQL1 := 'INSERT into DEPT values (:para1, :para2, :para3) returning rowid into :para4';***

***execute immediate SQL1 using id,'1','1' returning into x;***

***SQL1 := 'INSERT into R1 values (:para1, :para2)';***

***execute immediate SQL1 using id, x;***

*end;*

*/*

***EG.***

*CREATE or REPLACE Procedure PL1 (p\_deptno number, p\_dname varchar2, p\_loc varchar2, p\_walletid number) is*

*SQLstring varchar2 (100);*

***v\_rowid varchar2 (100);***

*begin*

*SQLstring := 'INSERT into DEPT values (:p1, :p2, :p3, :p4) returning rowid into :p\_rowid';*

***execute immediate SQLstring using p\_deptno, p\_dname, p\_loc, p\_walletid returning into v\_rowid;***

*dbms\_output.put\_line (v\_rowid);*

*end PL1;*

*/*

***EG.***

*CREATE or REPLACE procedure PRO1 (c1 in number, c2 in varchar2, c3 in varchar2, c4 number, c5 date, c6 number, c7 number, c8 number, c9 int) is*

*STR varchar2 (2000);*

*paraROWID rowid;*

*begin*

***SAVEPOINT s1;***

*STR := 'INSERT into EMP values (:para1, :para2, :para3, :para4, :para5, :para6, :para7, :para8, :para9, default ) returning rowid into :paraROWID';*

*execute immediate STR using c1, c2, c3, c4, c5, c6, c7, c8, c9 returning into paraROWID;*

*execute immediate STR using c1, c2, c3, c4, c5, c6, c7, c8, c9 returning into paraROWID;*

*dbms\_output.put\_line (paraROWID);*

***COMMIT;***

*exception*

*when others then*

*dbms\_output.put\_line (dbms\_utility.format\_error\_stack);*

***ROLLBACK to s1;***

***end PRO1;***

***/***

***Triggers***

*CREATE [OR REPLACE] TRIGGER trigger\_name*

*{BEFORE | AFTER | INSTEAD OF}*

*{INSERT [OR] | UPDATE [OR] | DELETE}*

*[OF col\_name]*

*ON table\_name*

***[REFERENCING OLD AS o NEW AS n]***

*[FOR EACH ROW]*

*WHEN (condition)*

*BEGIN*

*--- sql statements*

*END;*

***The trigger cannot include this statement:***

***:NEW := NULL; (ERROR)***

***Mutating error***

***if the trigger attempts to SELECT or MODIFY the table while the trigger has not completed (ie. table is in transition), then mutating trigger error occurs.*** *because the table is in middle of a transaction so it causes the trigger to mutate. you can change the trigger to statement level and apply the logic there.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***DEPT***  ***Table*** | ***Before trigger events*** | | | ***After trigger events*** | | |
| *Insert* | *Update* | *Delete* | *Insert* | *Update* | *Delete* |
| *Insert* | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** |
| *Update* | ***✓*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** |
| *Delete* | ***✓*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** |
| *Select* | ***✓*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** | ***🗴*** |

***Before Trigger***

*CREATE or REPLACE trigger DEMO* ***before*** *INSERT on DEPT for each row*

*begin*

***dbms\_output.put\_line ('Record Inserted');***

*end;*

*/*

*SQL> INSERT into DEPT values (1, 1, 1, 1);*

***ERROR***

***Record Inserted***

*INSERT into DEPT values (1, 1, 1, 1)*

*\**

*ERROR at line 1:*

*ORA-00001: unique constraint (C##SALEEL.DEPT\_PRIMARY\_KEY) violated*

***After Trigger***

*CREATE or REPLACE trigger DEMO* ***after*** *INSERT on DEPT for each row*

*begin*

*dbms\_output.put\_line ('Record Inserted');*

*end;*

*/*

*SQL> INSERT into DEPT values (1, 1, 1, 1);*

***ERROR***

*INSERT into DEPT values (1, 1, 1, 1)*

*\**

*ERROR at line 1:*

*ORA-00001: unique constraint (C##SALEEL.DEPT\_PRIMARY\_KEY) violated*

***Note: In BEFORE message “Record Inserted” will print then the trigger will terminate and in the case of AFTER message will not print.***

***Row Level and Table/Statement Level trigger Difference***

***EG. (Row Level Trigger)***

*CREATE or REPLACE trigger DEMO after UPDATE or DELETE on dept for each row*

*begin*

*dbms\_output.put\_line ('Done');*

*end;*

*/*

*SQL> UPDATE DEPT set loc='a' where deptno=0;*

***Row level trigger will not execute, if UPDATE or DELETE command fails to update or delete the rows.***

***EG. (Table/Statement Level Trigger)***

*CREATE or REPLACE trigger DEMO after UPDATE or DELETE on dept*

*begin*

*dbms\_output.put\_line ('Done');*

*end;*

*/*

*SQL> UPDATE DEPT set loc='a' where deptno=0;*

***Statement/Table level trigger will execute even if UPDATE or DELETE command fails to update or delete the rows.***

***2 digits DEPTNO***

*CREATE or REPLACE trigger DEMO before INSERT on DEPT for each row*

*declare*

*x number:=0;*

*begin*

***if length(:new.deptno) <> 2 then***

*raise\_application\_error (-20000,' Minimum 2 Digits');*

*end if;*

*end;*

*/*

***Trigger Action***

***IF Deleting / Inserting / Updating***

*CREATE or REPLACE trigger DEMO after* ***UPDATE of sal, comm*** *on EMP for each row*

*declare*

*begin*

*if* ***updating ('sal')*** *then*

*dbms\_output.put\_line ('Salary');*

*elsif* ***updating ('comm')*** *then*

*dbms\_output.put\_line ('Commission');*

*end if;*

*end DEMO;*

*/*

***Updating OF column Trigger***

*The trigger will execute / fire when we update comm column for remaining columns the trigger will not fire.*

*CREATE or REPLACE trigger DEMO after* ***UPDATE of comm*** *on EMP for each row*

*begin*

*if updating then*

*dbms\_output.put\_line ('Updated...');*

*else*

*dbms\_output.put\_line ('Not update...');*

*end if;*

*end DEMO;*

*/*

***Enable and Disable triggers***

***\* ALTER trigger DEMO enable / disable***

***\* ALTER table EMP enable / disable [ALL TRIGGERS]***

***Triggers***

***Note: - In case of TRUNCATE, Trigger doesn't get fired if it is a row level trigger.***

***Note: - INSTEAD OF triggers are valid only for views. You cannot specify an INSTEAD OF trigger on a table.***

***Note: - (Mutation error) the error is encountered when a row-level trigger accesses the same table on which it is based, while executing. The table is said to be mutating.***

***Mutation error (Always on DELETE command)***

*CREATE or REPLACE trigger DEMO* ***before DELETE*** *on DEPT for each row*

*declare*

*x number := 0;*

*begin*

*--SELECT count (\*) into x from DEPT;*

*--INSERT into DEPT values (3, 3, 3, 3);*

*--UPDATE DEPT set walletid = 10 where deptno=2;*

*--DELETE from DEPT where deptno=2;*

*dbms\_output.put\_line(x);*

*end DEMO;*

*/*

***Restrictions on BEFORE triggers:***

***You cannot specify a BEFORE trigger on a view or an object view.***

***You can write to the :NEW value but not to the :OLD value.***

***Restrictions on AFTER triggers:***

***You cannot specify an AFTER trigger on a view or an object view.***

***You cannot write either the :OLD or the :NEW value.***

1. *Hitting the ORA-4091 in an* ***INSERT trigger or an UPDATE*** *trigger where you only need access to the* ***:new values***
2. *Hitting the ORA-4091 in a* ***DELETE trigger or an UPDATE*** *trigger where you need to access the* ***:old*** *values*

***PRAGMA AUTONOMOUS\_TRANSACTION;***

Description of autonomous_transaction.gif follows

*CREATE or REPLACE trigger DEMO after INSERT on dept for each row*

*declare*

***PRAGMA AUTONOMOUS\_TRANSACTION;***

*begin*

*insert into t values('aaa',1,1,1);*

***commit;***

*end;*

*/*

***pragma exception\_init;***

Description of exception_init_pragma.gif follows

*CREATE or REPLACE trigger DEMO after INSERT on dept for each row*

*declare*

***ex exception;***

***pragma exception\_init(ex,-01722);***

*begin*

*INSERT into TEMP values ('aaa', 1, 1, 1);*

*exception*

***when ex then***

***dbms\_output.put\_line ('sss');***

*end;*

*/*

***Follows Trigger***

*CREATE or REPLACE trigger DEMO1 before INSERT on DEPT for each row*

*declare*

*begin*

*dbms\_output.put\_line (q'"First trigger fired"');*

*end DEMO1;*

*/*

*CREATE or REPLACE trigger DEMO2 before INSERT on DEPT for each row*

***Follows DEMO1***

*declare*

*begin*

*dbms\_output.put\_line (q'"Second trigger fired"');*

*end DEMO2;*

*/*

***Trigger on DDL [Statement level]***

***Note: This trigger will always execute once even if no records are affected.***

***Note: - Schema trigger can be given on CREATE, ALTER, DROP, AUDIT, RENAME, TRUNCATE and REVOKE***

***EG.***

*CREATE or REPLACE trigger DEMO* ***before RENAME or TRUNCATE on schema***

*begin*

*dbms\_output.put\_line* ***(Ora\_dict\_obj\_name ||' ' || Ora\_dict\_obj\_type || ' ' || Ora\_sysevent);***

*end DEMO;*

*/*

***EG.***

*CREATE or REPLACE trigger DEMO before* ***DROP on schema***

*declare*

*begin*

*raise\_application\_error (-20001,'No');*

*end DEMO;*

*/*

***Trigger on database***

*CREATE OR REPLACE TRIGGER DEMO* ***AFTER STARTUP ON DATABASE***

*BEGIN*

***EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN';***

*END open\_pdbs;*

*/*

***REFERENCING Clause***

Create a table with the same name as a correlation name, new, and then creates a trigger on that table. To avoid conflict between the table name and the correlation name, the trigger references the correlation name as Newest.

***EG.***

*CREATE table* ***NEW*** *(*

*NEW NUMBER,*

*OLD VARCHAR2 (20)*

*);*

*CREATE or REPLACE trigger DEMO before UPDATE on* ***NEW referencing NEW AS N for each row***

*begin*

***dbms\_output.put\_line (:N.NEW);***

***dbms\_output.put\_line (:N.OLD);***

*end DEMO;*

*/*

***EG.***

*CREATE table* ***NEW*** *(*

*C1 NUMBER,*

*C2 NUMBER,*

*C3 NUMBER,*

*C4 NUMBER,*

***NEW VARCHAR2 (10)***

*);*

*CREATE or REPLACE trigger DEMO before insert on* ***NEW referencing NEW as N for each row***

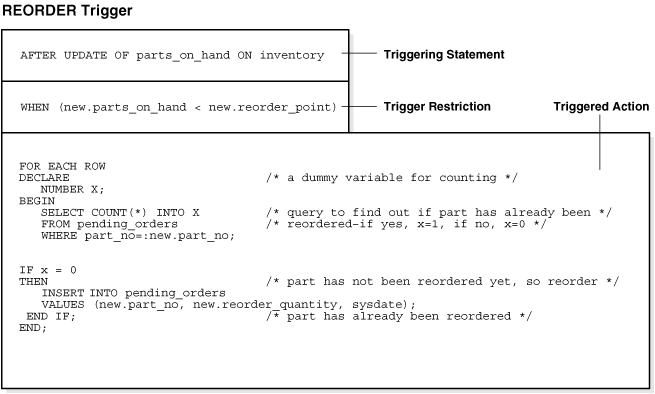
*begin*

***:N.NEW := UPPER (:N.NEW);***

*end DEMO;*

*/*

***WHEN option***



***EG.***

*CREATE or REPLACE trigger DEMO before UPDATE on EMP for each row* ***when (new.comm < old.comm)***

*declare*

*begin*

*raise\_application\_error (-20001,'New commission is less than old commission');*

*end DEMO;*

*/*

***EG.***

*CREATE or REPLACE trigger DEMO before INSERT on NEW* ***referencing new as N for each row when (N.new = 'saleel')***

*begin*

***:N.new := upper (:N.new);***

*end DEMO;*

*/*

|  |  |
| --- | --- |
| DDL Events | |
| **Event or DDL statement** | **When allowed or applicable** |
| STARTUP | AFTER |
| SHUTDOWN | BEFORE |
| SERVERERROR | AFTER |
| LOGON | AFTER |
| LOGOFF | BEFORE |
| CREATE | BEFORE and AFTER |
| DROP | BEFORE and AFTER |
| ALTER | BEFORE and AFTER |

***COMPOUND TRIGGER***

***CREATE OR REPLACE TRIGGER compound\_trigger\_name***

***FOR [INSERT|DELETE] UPDATE [OF column] ON table***

***COMPOUND TRIGGER***

***-- Declarative Section (optional)***

***-- Variables declared here have firing-statement duration.***

***--Executed before DML statement***

***BEFORE STATEMENT IS***

***BEGIN***

***NULL;***

***END BEFORE STATEMENT;***

***--Executed before each row change- :NEW, :OLD are available***

***BEFORE EACH ROW IS***

***BEGIN***

***NULL;***

***END BEFORE EACH ROW;***

***--Executed after each row change- :NEW, :OLD are available***

***AFTER EACH ROW IS***

***BEGIN***

***NULL;***

***END AFTER EACH ROW;***

***--Executed after DML statement***

***AFTER STATEMENT IS***

***BEGIN***

***NULL;***

***END AFTER STATEMENT;***

***END compound\_trigger\_name;***

***DBMS\_DDL.WRAP***

**DECLARE**

**x VARCHAR2 (4000);**

**BEGIN**

***x := 'CREATE or REPLACE procedure PRO1 as ';***

***x := x || 'x1 number := 1000; ';***

***x := x || 'begin ';***

***x := x || 'dbms\_output.put\_line(x1); ';***

***x := x || 'end PRO1; ';***

**dbms\_output.put\_line(x);**

***EXECUTE IMMEDIATE DBMS\_DDL.WRAP(x);***

**pro1;**

**END;**

**/**

***Dynamic SQL (using Native)***

***EG.***

*CREATE or REPLACE procedure PRO1* ***(t varchar2, a number, b varchar2, c varchar2)*** *as*

*STRINSERT varchar2 (200);*

*Begin*

*STRINSERT := 'INSERT into '|| t || ' values (:a, :b, :c) ';*

***Execute immediate STRINSERT using a, c, b;***

*End PRO1;*

*/*

*Note: - exec dbms\_output.enable (10000);*

***@, Start, Run to execute .SQL file***

***EG.***

*declare*

*Statement1 CHAR (46);*

*begin*

***Statement1:= 'INSERT INTO "EMPTY"."DEPT" VALUES (:0, :1, :2)';***

*EXECUTE IMMEDIATE statement1 USING 10, 'ACCOUNTING', 'NEW YORK';*

*EXECUTE IMMEDIATE statement1 USING 20, 'RESEARCH', 'DALLAS';*

*EXECUTE IMMEDIATE statement1 USING 30, 'SALES', 'CHICAGO';*

*EXECUTE IMMEDIATE statement1 USING 40, 'OPERATIONS', 'BOSTON';*

end;

/

**Note: - OUT and IN OUT formal parameters may not have default expressions**

**Note: - Varchar2 string length constraints must be in range (1 .. 32767)**

***Dynamic SQL (using Package)***

***EG.***

*CREATE or REPLACE PROCEDURE PL3 AS*

*C1 integer;*

*RC integer;*

*begin*

*C1 := DBMS\_SQL.OPEN\_CURSOR;*

*DBMS\_SQL.PARSE (C1,'CREATE TABLE X (Y varchar2 (10))', DBMS\_SQL.NATIVE);*

*RC := DBMS\_SQL.EXECUTE (C1);*

*dbms\_output.put\_line (RC);*

*DBMS\_SQL.CLOSE\_CURSOR (C1);*

*end PL3;*

*/*

***EG.***

*CREATE or REPLACE procedure pl3 (eno number) as*

*C1 number;*

*Row number;*

*begin*

*C1 := dbms\_sql.open\_cursor;*

*dbms\_sql.parse (c1,'DELETE from emp1 where empno = :eno', dbms\_sql.native);*

*dbms\_sql.bind\_variable (c1, 'eno', eno);*

*Row := dbms\_sql.execute (c1);*

*Exception*

*When others then*

*dbms\_sql.close\_cursor (c1);*

*end;*

*/*

**Exception Examples**

***EG.***

*CREATE or REPLACE function getDname (deptno IN NUMBER) RETURN VARCHAR2 is*

*Begin*

***CASE***

***WHEN deptno = 1 then return ‘Sales’;***

***WHEN deptno = 2 then return ‘Purchase’;***

***END CASE;***

*Exception*

***When CASE\_NOT\_FOUND then***

*return 'Exception - Case Not Found for score - '|| deptno;*

*End getDname;*

*/*

***EG.***

*DECLARE*

*x number := 10;*

*y number:= 0;*

*BEGIN*

***CASE X***

***when 10 then y := y + 1001;***

***when 20 then y := y + 2001;***

***end case;***

*dbms\_output.put\_line(y);*

*Exception*

***When CASE\_NOT\_FOUND then***

*dbms\_output.put\_line ('Exception - Case Not Found...');*

*END;*

*/*

**Prog 1. For Array**

***EG.***

*CREATE or REPLACE PROCEDURE P2 AS*

***TYPE arr1 is varray (10) of varchar2 (10);***

***arr arr1 := arr1 ();*** *// Object of VARRAY TYPE*

*begin*

*for x in 1..10*

*Loop*

*arr.extend;*

*arr (x) := x;*

*End loop;*

*end P2;*

*/*

**Prog 2. : To check for null value in variable**

***EG.***

*declare*

*x number :=0;*

*begin*

*x := ‘&x';*

***If x is null then***

*dbms\_output.put\_line ('null');*

*Else*

*dbms\_output.put\_line ('not null');*

*End if;*

*end;*

*/*

**Prog 3. : To return Boolean value.**

***EG.***

*CREATE or REPLACE function* ***F1(x number) return Boolean as***

*begin*

*If x=100 then*

*return True;*

*Else*

*return False;*

*End if;*

*end;*

*/*

**How to call**

***EG.***

*begin*

***If F1 (200) then***

*dbms\_output.put\_line (‘true');*

*Else*

*dbms\_output.put\_line ('false');*

*End if;*

*end;*

*/*

**Boolean value**

***EG.***

*CREATE or REPLACE procedure* ***PL1 (Var\_X in boolean, Var\_Y in boolean, Var\_Z in out boolean)*** *as*

*begin*

***Var\_Z := Var\_X or Var\_Y;***

*end PL1;*

*/*

**How to call**

*Declare*

*ans boolean;*

*begin*

***PL1 (null, false, ans);***

*If ans then*

*dbms\_output.put\_line ('true');*

*Else*

*dbms\_output.put\_line ('false');*

*End if;*

*end;*

*/*

**Prog 4. : To find missing number from Sr. No. column.**

*declare*

*cursor c1 is SELECT no from N;*

*xx number;*

*flg number :=0;*

*begin*

*for x in 1..10*

*loop*

*open c1;*

*loop*

*fetch c1 into xx;*

*exit when c1%NotFound;*

*If xx = x then*

*flg :=0;*

*Exit;*

*Else*

*flg :=1;*

*End if;*

*end loop;*

*close c1;*

*If flg = 1 then*

*dbms\_output.put\_line(x);*

*end if;*

*end loop;*

*end;*

*/*

**Prog 5. : Multiple exceptions.**

*declare*

*x number;*

***Ex1 exception;***

***Ex2 exception;***

*begin*

*x := '&b';*

*If x>10 then*

*Raise ex1;*

*Else*

*Raise ex2;*

*End if;*

*exception*

***When Ex1 or Ex2 then***

*dbms\_output.put\_line ('Error Message…');*

*end;*

*/*

**Prog 6. : Pragma\_Exception\_Init.**

*declare*

*X number;*

***Ex exception;***

***Ex1 exception;***

***Pragma exception\_init (ex, 100);***

***Pragma exception\_init (ex1,-6502);***

*begin*

*SELECT ename into x from EMP where empno=7788;*

*exception*

***When ex OR ex1 then***

***dbms\_output.put\_line ('Saleel’);***

***--when ex1 then***

***-- dbms\_output.put\_line ('Saleel’);***

*end;*

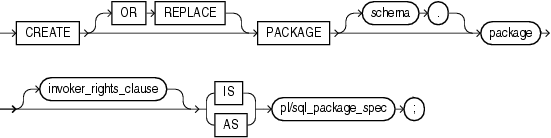
*/*

***Packages***

*Procedures, functions, cursors, and variables that are declared in the package specification are* ***global****.*

*When you create the package body, make sure that each procedure that you define in the body has the same parameters, by name, datatype, and mode, as the declaration in the package specification. For functions in the package body, the parameters and the return type must agree in name and type.*

***Create Package***



***Drop Package***

drop_package

**Prog 7. : Packages Header**

*CREATE or REPLACE package PKG1 is*

*Procedure P1;*

*Function F1(x number, y number) return number;*

*Function F1(x number, y number, z number) return number;*

*end PKG1;*

*/*

**Packages Body**

*CREATE or REPLACE package body PKG1 as*

*Procedure P1 is*

*begin*

*dbms\_output.put\_line ('Hello.');*

*end P1;*

*Function F1(x number, y number) return number is*

*begin*

*Return (x+y);*

*end F1;*

*Function F1(x number, y number, z number) return number is*

*begin*

*Return (x + y + z);*

*end F1;*

*end PKG1;*

*/*

***EG.***

***Note this package contents***

1. ***RECORD TYPE***
2. ***FUNCTION***
3. ***PROCEDURE***
4. ***EXCEPTION***

***CREATE or REPLACE package PKG1 is***

***TYPE EMPRecord is record (***

*VAREMPNO NUMBER (4), VARENAME VARCHAR2 (12),*

*VARJOB VARCHAR2 (12), VARMGR NUMBER (4),*

*VARHIREDATE DATE, VARSAL NUMBER (7, 2),*

*VARCOMM NUMBER (7, 2), VARDEPTNO NUMBER (2),*

*VARWALLETID NUMBER (5), VARTOTAL NUMBER*

);

***procedure PRO1 (varEMPNO in number);***

***function F1 return number;***

***ex exception;***

end PKG1;

/

***CREATE or REPLACE package body PKG1 is***

***xRecord EMPRecord;***

*function F1 return number is*

*xx number := 10;*

*begin*

*if xx = 10 then*

***raise ex;***

*else*

*return (1001);*

*end if;*

*exception*

*when ex then*

*dbms\_output.put\_line ('ex exception raised...');*

*return null;*

*end f1;*

*procedure pro1 (varEMPNO in number) is*

*str varchar2 (1000);*

*begin*

*str := 'SELECT \* from EMP where empno=:para1';*

***execute immediate str into xRecord using varEMPNO;***

*dbms\_output.put\_line (xRecord.VARENAME);*

*end pro1;*

*end PKG1;*

*/*

***EG.***

***CREATE or REPLACE package PKG1 is***

*procedure PL1;*

*end PKG1;*

*/*

***CREATE or REPLACE package body PKG1 is***

***x number := 0;***

*procedure PL1 is*

*begin*

***x := x + 1;***

***dbms\_output.put\_line(x);***

*end PL1;*

*end PKG1;*

*/*

**Prog 8. : Random Number.**

*CREATE or REPLACE PROCEDURE P3 AS*

*TYPE Arr1 is varray (100) of number;*

*char1 varchar2 (1000);*

*arr arr1 := arr1 ();*

*cnt number := 1;*

*xValue number := 0;*

*flg number := 1;*

*y number;*

*BEGIN*

*SELECT round (dbms\_random.value (1,500)) into xValue from DUAL;*

*arr.extend;*

*arr (cnt) := xValue;*

*loop*

*Exit when cnt >=100;*

*xValue:=0;*

*SELECT round (dbms\_random.value (1,500)) into xValue from DUAL;*

*For y in 1..arr.count*

*loop*

*if arr(y) = xValue then*

*flg := 1;*

*exit;*

*else*

*flg :=0;*

*end if;*

*end loop;*

*If flg = 0 then*

*cnt := cnt +1;*

*arr.extend;*

*arr (cnt) := xValue;*

*end if;*

*end loop;*

*for yy in 1..arr.count*

*loop*

*dbms\_output.put\_line (arr (yy));*

*char1 := char1 || arr (yy) || ',’;*

*--insert into prn values (arr (yy));*

*end loop;*

*dbms\_output.put\_line ('Work done...' || char1);*

*END P3;*

**Prog 9. : Default parameter.**

***EG.***

*CREATE or REPLACE* ***function F1 (x number default 100)*** *return number as*

*begin*

*return x+100;*

*end F1;*

*/*

***EG.***

*CREATE or REPLACE procedure P4* ***(x number := 5, y number default 550)*** *as*

*begin*

*dbms\_output.put\_line(x+y);*

*end;*

*/*

**Prog 10. : Structure in PL/SQL.**

*declare*

***TYPE T1 is record***

***(***

***X emp.empno%Type,***

***Y emp.ename%Type***

***);***

***Obj T1;***

*begin*

*SELECT empno, ename**into* ***obj*** *from EMP where empno=7788;*

*dbms\_output.put\_line* ***(obj.X ||*** *'**'* ***|| obj.Y);***

*end;*

*/*

**Function with RESULT CACHE**

***EG.***

*CREATE or REPLACE FUNCTION F1 (C1 IN VARCHAR2, C2 IN VARCHAR2)*

*RETURN VARCHAR2* ***RESULT\_CACHE*** *IS*

*begin*

*return (C1 || ' ' || C2);*

*end F1;*

*/*

**Prog 11. : Function with in, out and in out parameters.**

*CREATE or REPLACE function F1* ***(no in number, no1 out number, no2 out varchar2)*** *return number as*

*begin*

*no1:= no;*

***no2:='Saleel' || Null || 'Bagde';***

***return no;***

*end F1;*

*/*

**Call Function**

*declare*

*X number :=0;*

*Y number :=0;*

*Z varchar2 (20);*

*begin*

***X := F1 (1500, Y, Z);***

*dbms\_output.put\_line (X || ' ' || Y || ' ' || Z);*

*end;*

*/*

**Prog 12. : Function with in, out parameters.**

*CREATE or REPLACE function F1(x number, y number, z out number) return number is*

*begin*

*Z := 1000;*

*Return x + y;*

*end;*

*/*

**Call Function**

*declare*

*X1 number :=10;*

*Y1 number :=10;*

*Z number := 0;*

*Z1 number := 0;*

*begin*

***Z := F1(x1, y1, z1);***

***Z := F1(x1, y=>y1, z =>z1);***

***Z := F1(x=>x1, y=>y1, z=>z1);***

*dbms\_output.put\_line (z || ' ' || z1);*

*end;*

*/*

**Prog 13 :**

*declare*

***"End" number := 1001;***

*begin*

***dbms\_output.put\_line ("End");***

*end;*

*/*

***Methods for Passing Parameters***

***Actual parameters could be passed in three ways:***

* *Positional notation*
* *Named notation*
* *Mixed notation*

**POSITIONAL NOTATION**

*In positional notation, you can call the procedure as:*

***EG.***

***FindMin (a, b, c, d);***

*In positional notation, the first actual parameter is substituted for the first formal parameter; the second actual parameter is substituted for the second formal parameter, and so on. So, a is substituted for x, b is substituted for y, c is substituted for z and d is substituted for m.*

**NAMED NOTATION**

*In named notation, the actual parameter is associated with the formal parameter using the arrow symbol (=>). So the procedure call would look like:*

***EG.***

***FindMin (x=>a, y=>b, z=>c, m=>d);***

**MIXED NOTATION**

*In mixed notation, you can mix both notations in procedure call; however, the positional notation should precede the named notation.*

**The following call is legal:**

***EG.***

***FindMin (a, b, c, m=>d);***

**But this is not legal:**

***EG.***

***FindMin (x=>a, b, c, d);***

**Prog 13. : Function for Calculation**

*CREATE or REPLACE function* ***NEW\_USER.CALC*** *(pi\_val varchar2) return number is*

*v\_return number;*

*begin*

*Execute immediate 'SELECT '|| pi\_val ||' from dual' INTO v\_return;*

*Return v\_return;*

*end;*

*/*

*SQL> SELECT* ***new\_user.calc*** *('2+3') from dual;*

**Prog 14. : Use of case**

*declare*

*x number:=&X;*

*begin*

***case when x = 100 then***

***dbms\_output.put\_line ('1');***

***when x = 200 then***

***dbms\_output.put\_line ('2');***

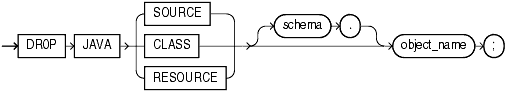
***end case;***

*end;*

*/*

## Java methods from oracle

***Method 1***



*SQL> SELECT* ***dbms\_java.get\_jdk\_version*** *from dual;*

*SQL>* ***exec dbms\_java.dropjava ('F1');***

**Step 1: Create file "HellWorld.java" file**

import java.util.\*;

public class HelloWorld {

**public static String HelloWorld (String x, String y) {**

**return (x + " " + y);**

**}**

}

**Step 2: Compile the file.**

C:\> javac HelloWorld.java

**Step 3: Use loadJava**

C:\> **loadjava** –user saleel/sony C:\HelloWorld.class –v

C:\> **dropjava** –user saleel/sony C:\HelloWorld.class -v

**Step 4: Write function / procedure in PL/SQL**

**Create or replace function Helloworld (agr1 varchar2, arg2 varchar2) return varchar2 as language java name 'HelloWorld.HelloWorld   (java.lang.String, java.lang.String) return java.lang.String';**

**Step 5: Execute the function / procedure**

SQL> Select **HelloWorld** (**'**Saleel**', '**Bagde**'**) from dual;

**Drop Java Class / Java File**

SQL> ***Drop java class "myClass";***

SQL> ***Drop*** ***java source "myJavaSource"***

***EG.***

public class addMethod {

public static Integer addMethod (Integer arg1, Integer arg2) {

return (arg1 + arg2);

}

}

## Java multiple methods program

public class HelloWorld {

public static String HelloWorld1 () {

return "Hello World...";

}

public static String StringConcat (String arg1, String arg2) {

return (arg1 +" "+ arg2);

}

public static Integer AddNumber (Integer arg1, Integer arg2) {

try {

return (arg1 + arg2);

}

catch (**NullPointerException** ex)

{

return 0;

}

/\* if (**arg2 == null**)

{

return 555;

}

else {

return (arg1 + arg2);

} \*/

}

}

## Package to call java program

**Note: - Oracle packages are used to call multiple methods of java class file.**

Create or replace package pk as

Function F1 returns varchar2;

Function F2 (arg1 varchar2, arg2 varchar2) return varchar2;

Function F3 (arg1 number, arg2 number) return number;

End pk;

/

Create or replace package body pk as

Function F1 return varchar2 as language java name 'HelloWorld.HelloWorld1 () return java.lang.String';

Function F2 (arg1 varchar2, arg2 varchar2) return varchar2 as language java name 'HelloWorld. StringConcat (java.lang.String, java.lang.String) return java.lang.String';

Function F3 (arg1 number, arg2 number) return number as language java name ‘HelloWorld.AddNumber (java.lang.Integer, java.lang.Integer) return java.lang.Integer';

End pk;

/

## Java overload methods program

public class Java2 {

public static String Function1 (String arg1) {

return (arg1);

}

public static String Function1 (String arg1, String arg2) {

return (arg1 +" “+ arg2);

}

}

## Package to call java Program

Create or replace package pk1 as

Function F4 (arg1 varchar2) return varchar2;

Function F4 (arg1 varchar2, arg2 varchar2) return varchar2;

End pk1;

/

Create or replace package body pk1 as

Function F4 (arg1 varchar2) return varchar2 as language java name 'Java2.Function1 (java.lang.String) return java.lang.String';

Function F4 (arg1 varchar2, arg2 varchar2) return varchar2 as language java name 'Java2.Function1 (java.lang.String, java.lang.String) return java.lang.String';

End pk1;

/

***Method 2***

***EG.***

*CREATE or REPLACE and COMPILE java source named Class1 as*

*public class Class1 {*

*public static java.lang.String F1 () {*

*return ("Record Inserted...");*

*}*

*public static java.lang.String F1 (java.lang.String x) {*

*return (x);*

*}*

*}*

*/*

## Package to call java Program

*CREATE or REPLACE PACKAGE PKG1 is*

*FUNCTION F1 return varchar2;*

*FUNCTION F1(x varchar2) return varchar2;*

*end PKG1;*

*/*

*CREATE or REPLACE PACKAGE PKG1 is*

*FUNCTION F1 return varchar2 is language java name ‘Class1.F1 () return java.lang.String';*

*FUNCTION F1 (x varchar2) return varchar2 is language java name ‘Class1.F1 (java.lang.String) return java.lang.String';*

*end PKG1;*

*/*

**Java Scanner (Input number)**

import java.util.\*;

public class HelloWorld {

public static String HelloWorld (String x, String y) {

return (x + " " + y);

}

public static void main (String agrs []) {

Scanner s = new Scanner (System.in);

String x1 = s.nextLine ();

String x2 = s.nextLine ();

String str = HelloWorld (x1, x2);

System.out.println (str);

}

}

|  |  |
| --- | --- |
| Predefined Exceptions in PL/SQL | |
| **Oracle Error/SQLCODE** | **Description** |
| CURSOR\_ALREADY\_OPEN  ORA-6511 SQLCODE= -6511 | You tried to OPEN a cursor that was already OPEN. You must CLOSE a cursor before you try to OPEN or re-OPEN it. |
| DUP\_VAL\_ON\_INDEX  ORA-00001 SQLCODE= -1 | Your INSERT or UPDATE statement attempted to store duplicate values in a column or columns in a row which is restricted by a unique index. |
| INVALID\_CURSOR  ORA-01001 SQLCODE= -1001 | You made reference to a cursor that did not exist. This usually happens when you try to FETCH from a cursor or CLOSE a cursor before that cursor is OPENed. |
| INVALID\_NUMBER  ORA-01722 SQLCODE = -1722 | PL/SQL executes a SQL statement that cannot convert a character string successfully to a number. This exception is different from the VALUE\_ERROR exception, as it is raised only from within a SQL statement. |
| LOGIN\_DENIED  ORA-01017 SQLCODE= -1017 | Your program tried to log onto the Oracle RDBMS with an invalid username-password combination. This exception is usually encountered when you embed PL/SQL in a 3GL language. |
| NO\_DATA\_FOUND  ORA-01403 SQLCODE= +100 | This exception is raised in three different scenarios: (1) You executed a SELECT INTO statement (implicit cursor) that returned no rows. (2) You referenced an uninitialized row in a local PL/SQL table. (3) You read past end of file with UTL\_FILE package. |
| NOT\_LOGGED\_ON  ORA-01012 SQLCODE= -1012 | Your program tried to execute a call to the database (usually with a DML statement) before it had logged into the Oracle RDBMS. |
| PROGRAM\_ERROR  ORA-06501 SQLCODE= -6501 | PL/SQL encounters an internal problem. The message text usually also tells you to "Contact Oracle Support." |
| STORAGE\_ERRORORA-06500 SQLCODE= -6500 | Your program ran out of memory or memory was in some way corrupted. |
| TIMEOUT\_ON\_RESOURCEORA-00051 SQLCODE= -51 | A timeout occurred in the RDBMS while waiting for a resource. |
| TOO\_MANY\_ROWSORA-01422 SQLCODE= -1422 | A SELECT INTO statement returned more than one row. A SELECT INTO can return only one row; if your SQL statement returns more than one row you should place the SELECT statement in an explicit CURSOR declaration and FETCH from that cursor one row at a time. |
| TRANSACTION\_BACKED\_OUTORA-00061 SQLCODE= -61 | The remote part of a transaction is rolled back, either with an explicit ROLLBACK command or as the result of some other action. |
| VALUE\_ERRORORA-06502 SQLCODE= -6502 | PL/SQL raises the VALUE\_ERROR whenever it encounters an error having to do with the conversion, truncation, or invalid constraining of numeric and character data. This is a very general and common exception. If this same type of error is encountered in a SQL DML statement within a PL/SQL block, then the INVALID\_NUMBER exception is raised. |
| ZERO\_DIVIDEORA-01476 SQLCODE= -1476 | Your program tried to divide by zero. |

***IMP Notes***



* DROP TABLE can have the PURGE clause but DROP INDEX and DROP TABLESPACE can not.

***Oracle12c new features***



***Default constraint using SEQUENCE***

***EG.***

*SQL> CREATE table TEMP*

*(COL1 NUMBER* ***DEFAULT S1.NEXTVAL,***

*COL2 NUMBER* ***DEFAULT ON NULL S2.NEXTVAL,***

*ENAME VARCHAR2 (10));*

***IDENTITY***

***[GENERATED {ALWAYS | BY DEFAULT} AS IDENTITY***

***[(START WITH IntegerConstant***

***[, INCREMENT BY IntegerConstant] ) ] ] ]***

*There are three types of identity that can be created*

1. *GENERATED ALWAYS AS IDENTITY*
2. *GENERATED BY DEFAULT AS IDENTITY*
3. *GENERATED BY DEFAULT ON NULL AS IDENTITY*

*Rules*

1. *A table can only contain one identity column*
2. *The column must be a numeric data type*
3. *If an identity clause is specified for a column, then a default clause cannot be specified for that column*
4. *NOT NULL and NOT DEFERRABLE constraints are created automatically for the identity column if they do not already exist. Conflicting constraints will raise an error.*
5. *CREATE TABLE AS SELECT statements do not inherit the identity property of a column in the source table.*

***Identity column will automatically create a SEQUENCE starting by the name ISEQ$$\_95101 to generate the number.***

***EG.***

*SQL> CREATE table TEMP*

*(ID NUMBER* ***generated always as identity,***

*DESCRIPTION VARCHAR2 (30));*

***EG.***

*SQL> CREATE table TEMP*

*(ID NUMBER* ***generated by default as identity,***

*DESCRIPTION VARCHAR2 (30)*

***EG.***

*SQL> CREATE table TEMP*

*(ID NUMBER* ***generated by default on null as identity,***

*DESCRIPTION VARCHAR2 (30)*

***EG.***

*SQL> CREATE table TEMP (col1 number* ***generated always as identity (start with 1000 increment by 10));***

***Reset generated (identity column) with 0***

*SQL> ALTER table TEMP* ***modify c1 number generated always as identity restart;***

***Row Limiting Clause***

*SQL> SELECT \* from EMP* ***FETCH FIRST 2 ROWS ONLY;***

*SQL> SELECT \* from EMP* ***order by sal desc fetch first 5 rows only;***

*SQL> SELECT \* from EMP* ***Fetch First 2 Rows with Ties;***

*SQL> SELECT \* from EMP* ***OFFSET 5 ROW fetch next 5 rows only; // Offset will leave those number or records and then fetch the rows.***

*SQL> SELECT \* from EMP* ***offset (SELECT count(\*) -1 from EMP) row fetch next 1 row only;*** ***// Offset will leave those number or records and then fetch the rows.***

***Invisible columns***

*SQL>* ***desc USER\_TAB\_COLS (data dictionary)***

***Following operations don’t see invisible columns***

* *SELECT \* FROM statements in SQL*
* *DESCRIBE commands in SQL\*Plus*
* *%ROWTYPE attribute declarations in PL/SQL*
* *Describes in Oracle Call Interface (OCI)*

***SQL> set colinvisible on/off***

*SQL> CREATE table EMP (*

*empno number (6),*

*ename varchar2 (40),*

***sal number (9)******Invisible);***

*SQL> ALTER table EMP MODIFY* ***(sal VISIBLE / INVISIBLE);***

*SQL> ALTER table EMP* ***modify salary visible;***

*SQL> ALTER table EMP* ***modify salary invisible;***

***Value can be inserted in the invisible columns by giving column list***

*SQL> INSERT into EMP (empno, ename,* ***sal****) values (1, 'saleel', 7000);*

*SQL> SELECT column\_name,* ***hidden\_column, virtual\_column,*** *histogram from* ***USER\_TAB\_COLS*** *where table\_name='EMP';*

*SQL> SELECT COLUMN\_NAME, HIDDEN\_COLUMN, VIRTUAL\_COLUMN, DEFAULT\_ON\_NULL, IDENTITY\_COLUMN from* ***USER\_TAB\_COLS*** *where table\_name='NEW';*

***Multiple indexes on the same column***

Displays the name of the Container to which you are connected when connected to a Consolidated Database. For non-consolidated database, it will return "Non Consolidated".

***SQL Tuning***



**Different SELECT statement**

**\*** Alter system flush buffer\_cache;

**\*** alter system flush shared\_pool;

**Listing 2: Documented Oracle Hints:**

|  |  |  |
| --- | --- | --- |
| **ALL\_ROWS** | **INDEX\_SS\_ASC** | **PARALLEL** |
| **AND\_EQUAL** | **INDEX\_SS\_DESC** | **PARALLEL\_INDEX** |
| **ANTIJOIN** | **INLINE** | **PIV\_GB** |
| **APPEND** | **LEADING** | **PIV\_SSF** |
| **BITMAP** | **LIKE\_EXPAND** | **PQ\_DISTRIBUTE** |
| **BUFFER** | **LOCAL\_INDEXESMATERIALIZE** | **PQ\_MAP** |
| **BYPASS\_RECURSIVE\_CHECK** | **MERGE** | **PQ\_NOMAP** |
| **BYPASS\_UJVC** | **MERGE\_AJ** | **PUSH\_PRED** |
| **CACHE** | **MERGE\_SJ** | **PUSH\_SUBQ** |
| **CACHE\_CB** | **MV\_MERGE** | **REMOTE\_MAPPED** |
| **CACHE\_TEMP\_TABLE** | **NESTED\_TABLE\_GET\_REFS** | **RESTORE\_AS\_INTERVALS** |
| **CARDINALITY** | **NESTED\_TABLE\_SET\_REFS** | **REWRITE** |
| **CHOOSE** | **NESTED\_TABLE\_SET\_SETID** | **RULE** |
| **CIV\_GB** | **NL\_AJ** | **SAVE\_AS\_INTERVALS** |
| **COLLECTIONS\_GET\_REFS** | **NL\_SJ** | **SCN\_ASCENDING** |
| **CPU\_COSTING** | **NO\_ACCESS** | **SELECTIVITY** |
| **CUBE\_GB** | **NO\_BUFFER** | **SEMIJOIN** |
| **CURSOR\_SHARING\_EXACT** | **NO\_EXPAND** | **SEMIJOIN\_DRIVER** |
| **DEREF\_NO\_REWRITE** | **NO\_EXPAND\_GSET\_TO\_UNION** | **SKIP\_EXT\_OPTIMIZER** |
| **DML\_UPDATE** | **NO\_FACT** | **SQLLDR** |
| **DOMAIN\_INDEX\_NO\_SORT** | **NO\_FILTERING** | **STAR** |
| **DOMAIN\_INDEX\_SORT** | **NO\_INDEX** | **STAR\_TRANSFORMATION** |
| **DRIVING\_SITE** | **NO\_MERGE** | **SWAP\_JOIN\_INPUTS** |
| **DYNAMIC\_SAMPLING** | **NO\_MONITORING** | **SYS\_DL\_CURSOR** |
| **DYNAMIC\_SAMPLING\_EST\_CDN** | **NO\_ORDER\_ROLLUPS** | **SYS\_PARALLEL\_TXN** |
| **EXPAND\_GSET\_TO\_UNION** | **NO\_PRUNE\_GSETS** | **SYS\_RID\_ORDER** |
| **FACT** | **NO\_PUSH\_PRED** | **TIV\_GB** |
| **FIRST\_ROWS** | **NO\_PUSH\_SUBQ** | **TIV\_SSF** |
| **FORCE\_SAMPLE\_BLOCK** | **NO\_QKN\_BUFF** | **UNNEST** |
| **FULL** | **NO\_SEMIJOIN** | **USE\_ANTI** |
| **GBY\_CONC\_ROLLUP** | **NO\_STATS\_GSETS** | **USE\_CONCAT** |
| **GLOBAL\_TABLE\_HINTS** | **NO\_UNNEST** | **USE\_HASH** |
| **HASH** | **NOAPPEND** | **USE\_MERGE** |
| **HASH\_AJ** | **NOCACHE** | **USE\_NL** |
| **HASH\_SJ** | **NOCPU\_COSTING** | **USE\_SEMI** |
| **HWM\_BROKERED** | **NOPARALLEL** | **USE\_TTT\_FOR\_GSETS** |
| **IGNORE\_ON\_CLAUSE** | **NOPARALLEL\_INDEX** |  |
| **IGNORE\_WHERE\_CLAUSE** | **NOREWRITE** |  |
| **INDEX\_ASC** | **OR\_EXPAND** |  |
| **INDEX\_COMBINE** | **ORDERED** |  |
| **INDEX\_DESC** | **ORDERED\_PREDICATES** |  |
| **INDEX\_FFS** | **OVERFLOW\_NOMOVE** |  |
| **INDEX\_JOIN** |  |  |
| **INDEX\_RRS** |  |  |
| **INDEX\_SS** |  |  |

**Undocumented Hints:**

|  |  |  |
| --- | --- | --- |
| **BYPASS\_RECURSIVE\_CHECK** | **IGNORE\_ON\_CLAUSE** | **OVERFLOW\_NOMOVE** |
| **BYPASS\_UJVC** | **IGNORE\_WHERE\_CLAUSE** | **PIV\_GB** |
| **CACHE\_CB** | **INDEX\_RRS** | **PIV\_SSF** |
| **CACHE\_TEMP\_TABLE** | **INDEX\_SS** | **PQ\_MAP** |
| **CIV\_GB** | **INDEX\_SS\_ASC** | **PQ\_NOMAP** |
| **COLLECTIONS\_GET\_REFS** | **INDEX\_SS\_DESC** | **REMOTE\_MAPPED** |
| **CUBE\_GB** | **LIKE\_EXPAND** | **RESTORE\_AS\_INTERVALS** |
| **CURSOR\_SHARING\_EXACT** | **LOCAL\_INDEXES** | **SAVE\_AS\_INTERVALS** |
| **DEREF\_NO\_REWRITE** | **MV\_MERGE** | **SCN\_ASCENDING** |
| **DML\_UPDATE** | **NESTED\_TABLE\_GET\_REFS** | **SKIP\_EXT\_OPTIMIZER** |
| **DOMAIN\_INDEX\_NO\_SORT** | **NESTED\_TABLE\_SET\_REFS** | **SQLLDR** |
| **DOMAIN\_INDEX\_SORT** | **NESTED\_TABLE\_SET\_SETID** | **SYS\_DL\_CURSOR** |
| **DYNAMIC\_SAMPLING** | **NO\_EXPAND\_GSET\_TO\_UNION** | **SYS\_PARALLEL\_TXN** |
| **DYNAMIC\_SAMPLING\_EST\_CDN** | **NO\_FACT** | **SYS\_RID\_ORDER** |
| **EXPAND\_GSET\_TO\_UNION** | **NO\_FILTERING** | **TIV\_GB** |
| **FORCE\_SAMPLE\_BLOCK** | **NO\_ORDER\_ROLLUPS** | **TIV\_SSF** |
| **GBY\_CONC\_ROLLUP** | **NO\_PRUNE\_GSETS** | **UNNEST** |
| **GLOBAL\_TABLE\_HINTS** | **NO\_STATS\_GSETS** | **USE\_TTT\_FOR\_GSETS** |
| **HWM\_BROKERED** | **NO\_UNNEST** |  |
|  | **NOCPU\_COSTING** |  |

**\*** Explain plan for Select \* from xemp;

**\*** Select \* from table (dbms\_xplan.display);

**\*** Set autotrace traceonly

**\*** Set autotrace on explain

**Roles**

1. Create role MyRole; (will, Create New Role)
2. Grant create table,..,..,.. to MyRole (will, Grant Role To New Role)
3. Grant connect, resource, MyRole to saleel identified by saleel (Will, Create New User With Grant Options and Roles)
4. Select username, granted\_role from user\_role\_privs; (see, Roles Of The Current User)
5. Password saleel (To, change the password)

**\*** USER\_ROLE\_PRIVS, DBA\_ROLE\_PRIVS (Oracle Data Dictionary)

**\*** Select ora\_login\_user, ora\_client\_ip\_address from dual; (Will, return current user and IP address)

**Profiles**

**\*** Create profile ProfileInfo limit

FAILED\_LOGIN\_ATTEMPTS 1; (If there are 1 consecutive failed connects to saleel account, the account will be automatically locked by Oracle)

**\*** Alter user saleel account UNLOCK identified by password; (Unlock account)

**\*** Alter user saleel account LOCK; (To locked the account)

**\*** Alter user saleel Password EXPIRE; (Force user to choose a new password)

**\*** Alter user saleel profile ProfileInfo; (Granting profile to saleel user)

**\*** Select \* from dba\_profiles;

|  |  |  |  |
| --- | --- | --- | --- |
| Oracle Languages | | | |
| us AMERICAN | cs CZECH | din GERMAN DIN | esa LATIN AMERICAN SPANISH |
| ar ARABIC | dk DANISH | d GERMAN | lv LATVIAN |
| bn BENGALI | nl DUTCH | el GREEK | lt LITHUANIAN |
| ptb BRAZILIAN PORTUGUESE | eg EGYPTIAN | iw HEBREW | ms MALAY |
| bg BULGARIAN | gb ENGLISH | hu HUNGARIAN | esm MEXICAN SPANISH |
| frc CANADIAN FRENCH | et ESTONIAN | is ICELANDIC | n NORWEGIAN |
| ca CATALAN | sf FINNISH | in INDONESIAN | pl POLISH |
| zhs SIMPLIFIED CHINESE | f FRENCH | i ITALIAN | pt PORTUGUESE |
| hr CROATIAN | ko KOREAN | ja JAPANESE | ro ROMANIAN |
| ru RUSSIAN | sk SLOVAK | sl SLOVENIAN | e SPANISH |
| s SWEDISH | th THAI | tr TURKISH | zht TRADITIONAL CHINESE |
| uk UKRAINIAN | vn VIETNAMESE |  |  |

**Product\_user\_profile (Data Dictionary)**

Note: - If **Product\_user\_profile** table is not present then run **pupbld.sql** Script

**\*** Insert into product\_user\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'DELETE', 'DISABLED');

**\*** Insert into product\_user\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'INSERT', 'DISABLED');

**\*** Insert into product\_user\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'SELECT', 'DISABLED');

**\***  Insert into product\_user\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'UPDATE', 'DISABLED');

**\*** Insert into product\_user\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'CREATE', 'DISABLED');

**Commands that can be prevented**

***This feature allows disabling of SQL, PL/SQL and SQL\*PLUS commands.***

1. SQL: ALTER, AUDIT, ANALYZE, CREATE, DELETE, DROP, INSERT, LOCK, NOAUDIT, RENAME, SELECT, UPDATE, VALIDATE, TRUNCATE, GRANT, REVOKE, SET ROLE, SET TRANSACTION
2. PL/SQL: DECLARE, BEGIN
3. SQL\*PLUS: COPY, HOST, SET, EDIT, PASSWORD, SPOOL, EXECUTE, QUIT, START, EXIT, RUN, GET, SAVE

**System.product\_profile (Data Dictionary)**

**\*** Insert into system.product\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'DECLARE', 'DISABLED');

**\*** Insert into system.product\_profile (product, userid, attribute, char\_value)

Values ('SQL\*Plus', 'SCOTT', 'BEGIN', 'DISABLED');

***MS-SQL Server***



**Data Types**

|  |  |
| --- | --- |
| Data Types | Size |
|  |  |
|  |  |

**The INFORMATION\_SCHEMA views allow you to retrieve metadata about the objects within a database.**

**Following is a list of each of the views that exist.**

* INFORMATION\_SCHEMA.CHECK\_CONSTRAINTS
* INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE
* INFORMATION\_SCHEMA.COLUMN\_PRIVILEGES
* INFORMATION\_SCHEMA.COLUMNS
* INFORMATION\_SCHEMA.CONSTRAINT\_COLUMN\_USAGE
* INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE
* INFORMATION\_SCHEMA.DOMAIN\_CONSTRAINTS
* INFORMATION\_SCHEMA.DOMAINS
* INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE
* INFORMATION\_SCHEMA.PARAMETERS
* INFORMATION\_SCHEMA.REFERENTIAL\_CONSTRAINTS
* INFORMATION\_SCHEMA.ROUTINE\_COLUMNS
* INFORMATION\_SCHEMA.ROUTINES
* INFORMATION\_SCHEMA.SCHEMATA
* INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS
* INFORMATION\_SCHEMA.TABLE\_PRIVILEGES
* INFORMATION\_SCHEMA.TABLES
* INFORMATION\_SCHEMA.VIEW\_COLUMN\_USAGE
* INFORMATION\_SCHEMA.VIEW\_TABLE\_USAGE
* INFORMATION\_SCHEMA.VIEWS

***Different SQL statement***

**\*** begin Transaction (This is used to start the transaction for Commit, Rollback)

***Stored Procedure***

* sp\_password Null,'saleel','saleel' **(Null, 'New Password', 'User Name')**
* sp\_columns 'EMP'
* sp\_tables
* sp\_tables 'EMP'
* sp\_pkeys 'EMP' **(Primary key details)**

***System DataTime***

**\*** Select SYSDATETIME(), GETDATE()

***Create Table***

Create table TEMP (srno int identity ,empno numeric , ename varchar(20))

Select \* from INFORMATION\_SCHEMA.COLUMNS where TABLE\_NAME='TEMP'

***Alter Table***

Alter table TEMP ADD City varchar (50), State varchar (50);

Alter table TEMP Alter Column no varchar

Alter table TEMP DROP Column City, State

***MySQL***

**Data Types**

|  |  |  |
| --- | --- | --- |
| MySQL Datatypes | | |
| **Ty p e** | **S i z e** | **D e s c r i p t i o n** |
| CHAR[Length] | Length bytes | A fixed-length field from 0 to 255 characters long. |
| VARCHAR(Length) | String length + 1 bytes | A fixed-length field from 0 to 255 characters long. |
| TINYTEXT | String length + 1 bytes | A string with a maximum length of 255 characters. |
| TEXT | String length + 2 bytes | A string with a maximum length of 65,535 characters. |
| MEDIUMTEXT | String length + 3 bytes | A string with a maximum length of 16,777,215 characters. |
| LONGTEXT | String length + 4 bytes | A string with a maximum length of 4,294,967,295 characters. |
| TINYINT[Length] | 1 byte | Range of -128 to 127 or 0 to 255 unsigned. |
| SMALLINT[Length] | 2 bytes | Range of -32,768 to 32,767 or 0 to 65535 unsigned. |
| MEDIUMINT[Length] | 3 bytes | Range of -8,388,608 to 8,388,607 or 0 to 16,777,215 unsigned. |
| INT[Length] | 4 bytes | Range of -2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 unsigned. |
| BIGINT[Length] | 8 bytes | Range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 unsigned. |
| FLOAT | 4 bytes | A small number with a floating decimal point. |
| DOUBLE[Length, Decimals] | 8 bytes | A large number with a floating decimal point. |
| DECIMAL[Length, Decimals] | Length + 1 or Length + 2 bytes | A DOUBLE stored as a string, allowing for a fixed decimal point. |
| DATE | 3 bytes | In the format of YYYY-MM-DD. |
| DATETIME | 8 bytes | In the format of YYYY-MM-DD HH:MM:SS. |
| TIMESTAMP | 4 bytes | In the format of YYYYMMDDHHMMSS; acceptable range ends inthe year 2037. |
| TIME | 3 bytes | In the format of HH:MM:SS |
| ENUM | 1 or 2 bytes | Short for enumeration, which means that each column can have one of several possible values. |
| SET | 1, 2, 3, 4, or 8 bytes | Like ENUM except that each column can have more than one of several possible values. |

The INFORMATION\_SCHEMA CHARACTER\_SETS Table

The INFORMATION\_SCHEMA COLLATIONS Table

The INFORMATION\_SCHEMA COLLATION\_CHARACTER\_SET\_APPLICABILITY Table

The INFORMATION\_SCHEMA COLUMNS Table

The INFORMATION\_SCHEMA COLUMN\_PRIVILEGES Table

The INFORMATION\_SCHEMA ENGINES Table

The INFORMATION\_SCHEMA EVENTS Table

The INFORMATION\_SCHEMA FILES Table

The INFORMATION\_SCHEMA GLOBAL\_STATUS and SESSION\_STATUS Tables

The INFORMATION\_SCHEMA GLOBAL\_VARIABLES and SESSION\_VARIABLES Tables

The INFORMATION\_SCHEMA KEY\_COLUMN\_USAGE Table

The INFORMATION\_SCHEMA ndb\_transid\_mysql\_connection\_map Table

The INFORMATION\_SCHEMA OPTIMIZER\_TRACE Table

The INFORMATION\_SCHEMA PARAMETERS Table

The INFORMATION\_SCHEMA PARTITIONS Table

The INFORMATION\_SCHEMA PLUGINS Table

The INFORMATION\_SCHEMA PROCESSLIST Table

The INFORMATION\_SCHEMA PROFILING Table

The INFORMATION\_SCHEMA REFERENTIAL\_CONSTRAINTS Table

The INFORMATION\_SCHEMA ROUTINES Table

The INFORMATION\_SCHEMA SCHEMATA Table

The INFORMATION\_SCHEMA SCHEMA\_PRIVILEGES Table

The INFORMATION\_SCHEMA STATISTICS Table

The INFORMATION\_SCHEMA TABLES Table

The INFORMATION\_SCHEMA TABLESPACES Table

The INFORMATION\_SCHEMA TABLE\_CONSTRAINTS Table

The INFORMATION\_SCHEMA TABLE\_PRIVILEGES Table

The INFORMATION\_SCHEMA TRIGGERS Table

The INFORMATION\_SCHEMA USER\_PRIVILEGES Table

The INFORMATION\_SCHEMA VIEWS Table

InnoDB INFORMATION\_SCHEMA Tables

Thread Pool INFORMATION\_SCHEMA Tables

Extensions to SHOW Statements

**By default MySQL queries are not case-sensitive.**

**mysql>** SHOW databases;

**mysql**> SELECT database();

**mysql**> SELECT user();

**mysql**> STATUS;

**mysql**> SHOW tables

**mysql**> DESC EMP;

**mysql**> SELECT curdate(); **//2016-11-05**

**mysql**> SELECT sysdate(); **// 2016-11-05 13:55:33**

**mysql**> CREATE table EMP (c1 int, c2 enum ('saleel', 'sharmin', 'vrushali'));

**mysql**> INSERT into EMP values (1, 2); **// We can also give the enum value number**

**// instead of actual**

**// values (1 – ‘Saleel’, 2 – ‘sharmin’, 3 – ‘vrushali’)**

**mysql**> CREATE table EMP (id int **auto\_increment primary key**, ename varchar (10));

**mysql**> ALTER TABLE EMP AUTO\_INCREMENT = 100;

**Insert multiple records**

**mysql**> INSERT into EMP values **(default,'a1'), (default,'a2'), (default,'a3')**;

**mysql**> SELECT \* FROM information\_schema.TABLE\_CONSTRAINTS,

**mysql**> desc EMP;

**mysql**> show columns from EMP;

**mysql**> show fields from EMP;

**mysql**> explain EMP;

**mysql**> show create table EMP;

**mysql**>

**mysql**>

**mysql**>

***Hadoop***



***Hadoop Distributed File System (HDFS)***

***Hadoop*** *is an open source framework for writing and running distributed applications that process large amounts of data.*

***Hadoop*** *runs on large clusters of commodity machines or on cloud computing services.*

*A* ***Hadoop*** *cluster has many parallel machines that store and process large data sets. Client computers send jobs into this computer cloud and obtain results.*

***Start Process for Hadoop and Hbase***

***Start Hadoop***

*$ su [-] hduser*

*$ /home/hduser/Hadoop/bin/Start-all.sh*

*give password 4 times (linux)*

*jpc to check services for hadoop*

***Start Hbase***

*$ /home/hduser/hbase/bin/Start-hbase.sh*

***Hbase Shell***

*$ /home/hduser/hbase/bin/hbase shell*

***Stop Process for Hadoop and Hbase***

***Stop Hbase***

*$ /home/hduser/hbase/bin/Stop-hbase.sh*

***Stop Hadoop***

*$ /home/hduser/Hadoop/bin/Stop-all.sh*

***HBase***



***HBase*** *is built on top of* ***Apache Hadoop & Apache ZooKeeper.*** *You use the HBase shell to interact with HBase from the command line.*

*Just like tables in a relational database, tables in HBase are organized into* ***rows and columns****. HBase treats columns a little differently than a relational database.* ***Columns*** *in HBase are organized into groups called* ***column families.*** *A table in HBase must have at least one column family.*

***Note: - Objects are case sensitive in HBase.***

***list tables***

hbase(main):001:0> ***list /\* list all the table of current database\*/***

***display structure***

hbase(main):004:0> ***describe 'employee' /\* displays structure \*/***

***disable / enable table***

hbase(main):005:0> ***disable 'employee' /\* disable table, [ get ] will not work \*/***

hbase(main):005:0> ***enable 'employee' /\* enable table \*/***

***Column Family***

***Columns in Apache HBase are grouped into column families.***

***create table with one column family***

hbase(main):058:0> ***create 'mytable', 'cf' /\* create a table called mytable with a***

***Single column family \*/***

***Table name***

***Column family***

***create table with multiple column family***

hbase(main):015:0> ***create 'mytable', 'cf1', 'cf2' /\* create a table called mytable***

***with a*** ***multiple column family \*/***

***multiple column family***

***put (Insert record in 1 column family)***

hbase(main):058:0> ***create 'mytable', 'cf'***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf* :  *empno', 7788***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf* : *ename', 'saleel'***

***put (Insert record in multiple column family)***

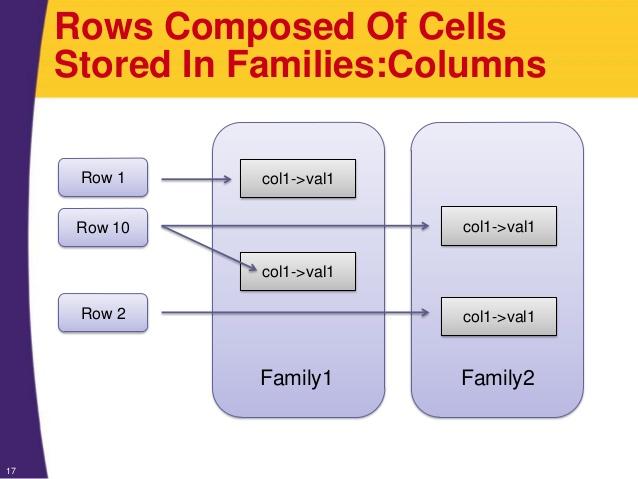
hbase(main):015:0> ***create 'mytable', 'cf1', 'cf2'***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf1* :  *empno', 1234***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf1* : *ename', 'saleel'***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf2* :  *job', 'manager'***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf2* : *salary', 7500***



***Note: - put is also used to update the cells.***

***put (Update record)***

hbase(main):069:0> ***put 'mytable', 'row1', 'cf* :  *empno', 1234***

***/\* put is also used for updating the cell \*/***

***delete rows***

hbase(main):069:0> ***delete 'mytable', 'row1', 'cf* :  *empno'***

***/\* will delete empno cell from row1 \*/***

***drop table***

***Drop the named table. Table must first be disabled.***

hbase(main):005:0> ***disable 'employee'***

hbase(main):006:0> ***drop 'employee' /\* drop table \*/***

***Column-Oriented Database Systems***

A columnar database is a database management system (DBMS) that stores data in columns instead of rows.

The goal of a columnar database is to efficiently write and read data to and from hard disk storage in order to speed up the time it takes to return a query.

Here is an example of a simple database table with 4 columns and 3 rows.

**ID         Last      First      Bonus**

1          Doe      John       8000

2          Smith    Jane      4000

3          Beck     Sam      1000

In a row-oriented database management system, the data would be stored like this:

***1, Doe, John, 8000; 2, Smith, Jane, 4000; 3, Beck, Sam, 1000;***

***001: 1, Doe, John, 8000; 002: 2, Smith, Jane, 4000; 003: 3, Beck, Sam, 1000;***

In a column-oriented database management system, the data would be stored like this:

***1, 2, 3; Doe, Smith, Beck; John, Jane, Sam; 8000, 4000, 1000;***

***001: 1, 002: 2, 003: 3; 001: Doe, 002: Smith, 003: Beck; 001: John, 002: Jane,***

***003: Sam; 001: 8000, 002: 4000, 003: 1000;***

***NoSQL Database Systems***

NoSQL systems are also referred to as "Not only SQL" to emphasize that they do in fact allow SQL-like query languages to be used.

**NoSQL MongoDB *a product of 10gen***

***MongoDB is Document Database***

1. **No JOINS support.**
2. **No TRANSACTION support.**
3. **No CONSTRAINTS support (no relation).**
4. **No schema.**

***MongoDB is Scalable, open-source, high-perform, document-oriented database.***

***Core MongoDB Operations (CRUD)***

***CRUD*** stands for ***create, read, update, and delete***, which are the four core database operations used in database driven application development.

***Queries in MongoDB are BSON objects that use a set of query operators.***

***BSON*** is a computer data interchange format used mainly as a data storage and network transfer format in the MongoDB database. It is a binary form for representing simple data structures and associative arrays (called objects or documents in MongoDB). ***The name "BSON" is based on the term JSON and stands for "Binary JSON".***

***JSON: JavaScript Object Notation.***

***JSON*** (JavaScript Object Notation) ***is a lightweight data-interchange format.*** It is easy for humans to read and write.

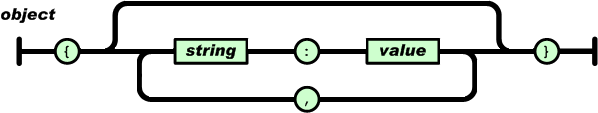
***Document Format***

* ***MongoDB stores documents on disk in the BSON serialization format. BSON is a binary representation of JSON documents, though it contains more data types than JSON.***

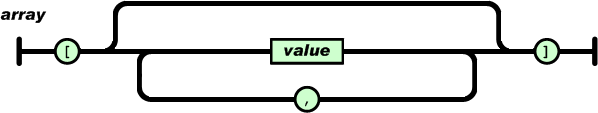
***JSON is built on two structures:***

* ***A collection of name / value pairs.*** In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
* ***An ordered list of values.*** In most languages, this is realized as an array, vector, list, or sequence.

***An object is an unordered set of name/value pairs.*** An object begins with { (left brace) and ends with } (right brace). Each name is followed by **: (colon)** and the name/value pairs are separated by **, (comma).**



***An array is an ordered collection of values.*** An array begins with [(left bracket) and ends with] (right bracket). Values are separated by **, (comma)**.



|  |  |
| --- | --- |
| **RDBMS** | **mongoDB** |
| database | database |
| tables | collections |
| rows | Documents  or BSON document |
| column | Field |

**Performance**

**Functionality**

**e**

We use **$** in front of field name to distinguish fields from string literals in expression [**"$name" *vs* "name"]**

***Working with DateFormates***

|  |  |
| --- | --- |
| ***e.g 1***  > *var startDate = new Date()*  >***startDate***  ISODate ("2013-08-27T06:19:05.980Z")  > *a.getFullYear()*  2013  ***e.g 2***  > *var startDate = new Date('Aug 27, 1926')*  > ***startDate***  ISODate("1926-08-26T18:30:00Z")  ***e.g 3***  > *var startDate = new Date (2013, 12, 12)*  ***e.g 4***  > *var* *x = ISODate ('2013-03-23')* | |
| **Name** | **Description** |
| a.getDate () + 1 | Gets day. |
| a.getMonth () + 1 | Gets month. |
| a.getFullYear ( ) | Get year. |
| a.getHours () + 1 | Get hour. |
| a.getMinutes () | Get minutes. |
| a.getSeconds () | Get seconds. |

***Start DB Server***

* ***By default, MongoDB stores the data files in the /data/db (C:\data\db on Windows) directory and listens for requests on port 27017.***

Create a folder ***C:\data\db***

C:\> *mongod*

**Or**

C:\> *mongod --dbpath "e:\MongoDB\mongoData" --journal*

***Start Shell***

C:\> *mongo* **// *It is a JavaScript Console.***

C:\> ***mongo db1***

C:\> ***mongo --host "192.168.100.17" --port "27017"***

*C:\ >* ***mongo "192.168.100.20/ db1"***

*C:\ >* ***mongo "192.168.100.17/ db2"***

*C:\ >* ***mongo "192.168.100.126/db3"***

***Default database***

1. ***local***
2. ***test***

***Remote DB connection (imp)***

*>* ***db = connect ("192.168.100.17:27017/primaryDB")***

*>* ***getHostName()***

*>* ***db.serverStatus()***

***Define the EDITOR environment variable***

***EDITOR="\"C:\\Program Files (x86)\\Notepad++\\notepad++.exe\""***

> *cls* **//** **clear screen (Windows)**

> *Ctrl + L* **//** **clear screen in shell (Linux)**

***ObjectId***

**ObjectId is a 12-byte BSON type, constructed using:**

*• 4-byte value representing the seconds since the UNIX epoch,*

*• 3-byte machine identifier,*

*• 2-byte process id, and*

*• 3-byte counter, starting with a random value.*

***new ObjectId***

> *var obj = new ObjectId ()* **//** **creates new ObjectId.**

***Load JavaScript in MongoDB***

> *load* ***("E:\\MongoDB\\MongoData\\scripts\\a.js")***

**// loads external JavaScript file.**

***Import data from .csv file***

***E:\> mongoimport -db info --collection emp --type csv c:\export.csv -headerline***

**// loads external data in MongoDB.**

***E:\> mongoimport -db info --collection emp --type csv "c:\MongoDB Data\emp.csv" -headerline***

***E:\> mongoimport -db info --collection emp --type csv c:\export.csv -f EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO***

**// loads external data in MongoDB.**

***E:\> mongoimport -db info --collection emp --type csv c:\export.csv -f EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO --upsertFields id***

**// loads external data in MongoDB without**

**//duplicates.**

***Import data from JSON file***

***E:\> mongoimport --db info --collection emp --file e:\emp.json --journal***

**// loads external data which is in JSON**

**// format**

***Import data from JSON file to remort database computer***

***E:\>mongoimport --host "192.168.100.20" --port "27017" --db "db1" --collection population --file "d:\MongoDB\emp.json"***

***E:\> mongoimport --host "192.168.100.17" --port "27017" --db "db2" --collection "population" --file "d:\MongoDB\population.json"***

***Export data to JSON format***

***E:\ mongoexport --db info --collection emp --out emp.json --journal***

***Current database***

> *db* **//** **show current database used.**

> *db.getName ()* **//** **show current database used.**

> *printjson (db)* **//** **show current database used.**

> *show dbs* **//** **show all database names.**

***Create / Connect to database***

*Note:* ***MongoDB****didn’t provide any command to create* ***"database".***

> ***use info*** **//** **open or create database.**

> *db.dropDatabase ()* **// drop database.**

***printjson***

> *printjson (10+10)*

*> printjson ('saleel bagde')*

**Collections (Table) *collection names are case sensitive***

> *db.getCollectionNames ()* **//** **show collections of current database.**

> *db.****getSiblingDB('db2')****.getCollectionNames();*

**// to access another database without**

**// switching databases**

> *show collections* **//** **show collections of current database.**

> *db.createCollection ("emp")* **//** **creates the collection (Table).**

> *db.createCollection ("\_emp")* **//** **creates the collection (Table).**

> *db.emp.drop ()* **//** **drops the collection.**

> *db.emp.renameCollection ("newemp")* **//** **rename the collection.**

## *Query Selectors*

### *Comparison*

|  |  |
| --- | --- |
| **Name** | **Description** |
| ***$eq*** | ***Matches values that are equal to a specified value.*** |
| ***$gt*** | ***Matches values that are greater than a specified value.*** |
| ***$gte*** | ***Matches values that are greater than or equal to a specified value.*** |
| ***$lt*** | ***Matches values that are less than a specified value.*** |
| ***$lte*** | ***Matches values that are less than or equal to a specified value.*** |
| ***$ne*** | ***Matches all values* that *are not equal to a specified value.*** |
| ***$in*** | ***Matches any of the values specified in an array.*** |
| ***$nin*** | ***Matches none of the values specified in an array.*** |

### *Logical*

|  |  |
| --- | --- |
| **Name** | **Description** |
| ***$or*** | ***Joins query clauses with a logical OR returns all documents that match the conditions of either clause.*** |
| ***$and*** | ***Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.*** |
| ***$not*** | ***Inverts the effect of a query expression and returns documents that do not match the query expression.*** |
| ***$nor*** | ***Joins query clauses with a logical NOR returns all documents that fail to match both clauses.*** |

***Comparison Query Operators***

***$eq***

***Syntax:* *{field: {$eq: value}}***

> *db.emp.find (****{deptno: {$eq: 3}}****)*

***$ne***

***Syntax:* *{field: {$ne: value}}***

> *db.emp.find (****{deptno: {$ne: 2}}****)*

***$gt***

***Syntax:* *{field: {$gt: value}}***

> *db.emp.find (****{deptno: {$gt: 3}}****)*

***$gte***

***Syntax:* *{field: {$gte: value}}***

> *db.emp.find* (***{deptno: {$gte: 3}}***)

***$lt***

***Syntax:* *{field: {$lt: value}}***

> *db.emp.find (****{deptno: {$lt: 2}}****)*

***$lte***

***Syntax:* *{field: {$lte: value}}***

> *db.emp.find (****{deptno: {$lte: 2}}****)*

***$in***

***Syntax:* *{field: {$in: [<value1>, <value2>, ..., <valueN>]}}***

> *db.emp.find (****{deptno: {$in: [2, 4, 6]}}****)*

> *db.publisher.find (****{publisheraddress :{ $not: {$in: ['pune', 'baroda']}}}****).forEach (printjson)*

***Logical Query Operators***

***$or***

***Syntax:* *{ $or: [ { <expression1> }, { <expression2> }, ... , { <expressionN> } ] }***

> *db.emp.find (****{$or: [{deptno: 1}, {ename: 'sharmin'}]}****)*

> *db.emp.find (****{$or: [{deptno: 1}, {sal: {$gt: 5000}}]}****)*

> *db.emp.find (****{$or: [{\_id:15}, {\_id:16}]}****).forEach (printjson)*

> *db.publisher.find (****{$or: [{publisheraddress: 'baroda'}, {publisheraddress: 'pune'}]}****).forEach (printjson)*

> *db.emp.find (****{$or: [{deptno: 10}, {deptno: 20}]}).sort ({job: true}****)*

***$and***

***Syntax:* *{ $and: [ { <expression1> }, { <expression2> } , ... , { <expressionN> } ] }***

> *db.emp.find (****{$and: [{empno: 7788}, {sal: 5000}]}****)*

> *db.publisher.find (****{$and: [{publisheraddress: {$ne: 'baroda'}}, {publisheraddress: {$ne: 'pune'}}]}****).forEach (printjson)*

*> db.emp.find (****{$and: [{\_id: {$gte: 4}}, {\_id: {$lte: 8}}]}****)*

*> db.getCollection ('emp').find (****{$and: [{sal: {$gt: 4500}}, {sal: {$lt: 5000}}]}****);*

*> db.emp.find (****{$and :[{ sal: {$gt: 3000}}, {sal: {$lt: 3500}}]}****)*

***$not***

***Syntax:* *{field: {$not: {<operator-expression>}}}***

> *db.publisher.find (****{publisheraddress: {$not: {$in: ['pune', 'baroda']}}}****).forEach (printjson)*

***$mod***

***Syntax*:** ***{field: {$mod: [divisor, remainder]}}***

* ***the $mod selects the documents where the field value divided by the divisor has the specified remainder.***

> *db.emp.find (****{deptno: {$mod: [2, 0]}}****)*

> *db.emp.find (****{deptno: {$mod: [2, 1]}}****)*

***Insert document (Insert Record)***

***Syntax:* *db.collection.insert (document)***

***Note: -***

* ***MongoDB uniquely identifies each document in a collection using the ObjectId. The ObjectId for a document is stored as the \_id attribute of that document. While inserting a record, any unique value can be set as the ObjectId.***

> *db.users.save ({\_id: 1, username: "goggle", password: "google123"})*

**//** **If a document does not exist with the**

**// specified \_id value, the *save ()* method**

**// performs an *Insert.***

**//** **If a document does exist with the**

**// specified \_id value, the *save ()* method**

**// performs an *Update.***

> *db.users.insert ({\_id:1, username: "google", password: "google123"})*

**//** **A document will be Inserted with user**

**//** **defined \_id.**

> *db.users.insert ({username: "google", password: "google123"})*

**//** **A document will be Inserted with system**

**//** **generated \_id.**

> *db.products.insert (****[***

***{empno: 1001, ename: 'saleel', address: 'paud road'},***

***{empno: 1002, ename: 'vrushali', address: 'paud road'},***

***{empno: 1003, ename: 'sharmin', address: 'paud road'}***

***]****)*

**//** **Multiple document will be inserted in the**

**// the collection**

*> db.dt.insert ({ename: 'vrushali',* ***hiredate: ISODate ('1970-07-19')****})*

***Insert NULL data***

> *db.users.save ({\_id: 1, username: "goggle",* ***password: NULL****})*

> *db.users.find ({****password: NULL****})*

> *db.users.find ({****password: {$type: 10}}*)**

***Insert into \_emp collection***

> *db.createCollection ("****\_emp****")* **//** **creates the collection.**

> *db.getCollection ('****\_emp****').insert ({ename:'saleel'})*

> *db.getCollection ('****\_emp****').find ()*

> *db.getCollection ('****\_emp****').find ({ename:'saleel'})*

***Insert Array***

> *db.project.insert (*

*{*

*\_id: 1,*

***technology: {***

***os: ['CentOS', 'Linux'],***

***webtech: ['php', 'java', '.NET'],***

***database: ['oracle', 'db2', 'SQLServer']***

***},***

***projects: ['project1', 'project2', 'project3']***

*}*

*)*

> *db.address.insert (* **// Two address of same person using**

*{* **// arrays.**

*\_id:1,*

*ename: 'saleel',*

***address: [***

***{***

***street: 'paud road',***

***city: 'pune',***

***state: 'maharashtra'***

***},***

***{***

***street: 'kothrud #1',***

***city: 'pune',***

***state: 'maharashtra'***

***}***

***],***

*mobile: 9850884228*

*}*

*)*

***Java Document Object Example***

> *delete doc.comment* **// delete elements from object {}**

***e.g 1***

> ***var doc = {};* // JavaScriptobject**

> *doc.title = "MongoDB Tutorial"*

> *doc.url = "http://mongodb.org"*

> *doc.comment = "Good tutorial video"*

> *doc.tags = ['tutorial', 'noSQL']*

> *doc.saveondate = new Date ()*

> ***doc.meta = {}* // JavaScriptsub objectwithin doc object {}**

> *doc.meta.browser = 'Google Chrome’*

> *doc.meta.os = 'Microsoft Windows7'*

> *doc.meta.mongodbversion = '2.4.0.0'*

> ***doc***

> ***db.book.insert (doc);***

***result:***

{

"title": "MongoDB Tutorial",

"url": "http://mongodb.org",

"comment": "Good tutorial video",

**"tags": [**

**"tutorial", // Array within an object**

**"NoSQL"**

**],**

"saveondate": ISODate ("2013-08-19T04:44:41.903Z"),

**"meta": {**

**"browser": "Google Chrome", // Object: *doc*, Sub Object: *meta***

**"os": "Microsoft Windows7",   // Sub object inside an object**

**"mongodbversion": "2.4.0.0"**

**}**

}

***e.g 2***

> ***var doc = {};* //JavaScript object**

> *doc.title = "MongoDB Tutorial"*

> *doc.url = "http://mongodb.org"*

> *doc.comment = "Good tutorial video"*

> *doc.tags = ['tutorial', 'video tutorial']*

> *doc.saveondate = new Date ()*

> *doc.published\_date =* ***ISODate ('2010-09-24')***

> ***doc.author = []***

> *doc.author =* ***[***

*{name: 'smith', email: 'smith.mongodb@gmail.com' },*

*{name: 'james', email: 'james.mongodb@hotmail.com' },*

*{name: 'joe', email: 'joe.mongodb@yahoomail.com' }*

***]***

> ***doc.meta = {}***

> *doc.meta.browser = 'Google Chrome’*

> *doc.meta.os = 'Microsoft Windows7 Ultimate'*

> *doc.meta.mongodbversion = 'v2.4.5'*

> ***doc***

> ***db.book.insert (doc);***

***result:***

{

"title": "MongoDB Tutorial",

"url": "http://mongodb.org",

"comment": "Good tutorial video",

**"tags": [**

**"tutorial", // Array within an object**

**"video tutorial"**

**],**

"saveondate": ISODate ("2013-08-19T13:38:51.106Z"),

"published\_date": ISODate ("2010-09-24T00:00:00Z"),

**"author": [**

**{**

**"name": "smith",**

**"email": "smith.mongodb@gmail.com"**

**},**

**{**

**"name": "james",**

**"email": "james.mongodb@hotmail.com"**

**},**

**{**

**"name": "joe",**

**"email": "joe.mongodb@yahoomail.com"**

**}**

**],**

**"meta": {**

**"browser": "Google Chrome", // Object: *doc*, Sub Object: *meta***

**"os": "Microsoft Windows7 Ultimate", // Sub object inside an object**

**"mongodbversion": "v2.4.5"**

**}**

}

>*db.emp.find ({},* ***{'skills.database': true}****).forEach (printjson)*

>*db.emp.find* ***({'skills.database':'mongodb'})****.forEach (printjson)*

> *db.bookinfo.find ({},* ***{'author.name': true, 'author.email': true, \_id: false}****).forEach (printjson)*

**OR**

> *db.emp.aggregate (****{$project: {ename: 1, deptno: true}}****)*

***Queries document using find () (Select Record)***

***Note: -***

*MongoDB provide a db.collection.find () method. The method accepts both the query criteria and projections and returns a cursor to the matching documents.* ***You can optionally modify the query to impose limits, skips, and sort orders.***

***Syntax:* *db.collection.find (<criteria>, <projection>)***

* ***the <criteria> Optional. Specifies selection criteria using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}), and***
* ***the <projection> Optional. Specifies the fields to return using***[**projection operators**](http://docs.mongodb.org/manual/reference/operator/projection/)***. To return all fields in the matching document, omit this parameter.***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| **criteria** | document | Optional. Specifies selection criteria using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}). |
| **projection** | document | Optional. Specifies the fields to return using projection operators. To return all fields in the matching document, omit this parameter. |

> *var x = db.emp.find()* **// returns result in variable.**

>*printjson (x [0])* **// prints documents**

> *db.emp.find ()****.pretty ()*** **//** **print all documents from collections well**

**// formatted.**

> *db.emp.find ()* **//** **print all documents from collections.**

> *db ['emp'].find* () **//** **print all documents from collections.**

> *db.emp.find ()****.forEach (printjson)*** **//** **print all documents from collections well**

**// formatted.**

> *db.getCollection ("emp").find ({}, {ename: true}).forEach (printjson)*

**// print all document of specified fields**

**// without condition.**

> *db.emp.find () [0]* **// print entire documents of 0th index**

**// number.**

> *db.emp.find () [0].ename* **// print ename of 0th index number.**

> *db.mytable.find (****{media: {$elemMatch: {c1: 111}}}****).forEach (printjson)*

**// search data in array**

> *db.emp.find ({****salary: {$gt: 5000}},*** *{ename: 1, \_id:0})*

**// search salary > 5000**

> *db.emp.find (****{job: 'manager', $and: [{sal: {$gt: 3000}}, {sal: {$lt: 4800}}]}****)*

***Find() Query Modifiers***

***Syntax:* *db.collection.find ({$query: {<query>}, <option>})***

> *db.emp.find (****{$query: {sal: {$gt: 3500}}}****)*

> *db.emp.find ({$query: {sal: {$gt: 3500}},* ***$orderby: {ename:-1}},*** *{ename: 1})*

> *db.emp.find (****{$query: {$or: [{job: 'manager'}, {job: 'salesman'}]}, $orderby: {ename: 1}}****)*

> *db.emp.find (****{$query: {$or: [{deptno: 10}, {deptno: 20}], $and: [{sal: {$lt: 5000}}]}, $orderby: {ename:-1}}****)* **// $and, $or both in query**

* ***$orderby will work only in $query***

***.sort() (Order by clause)***

***Ascending: 1 Descending: -1***

> *db.emp.find ().sort (****{job: 1}****).forEach (printjson)* **// print all document in ASC**

> *db.emp.find ().sort (****{job: -1}****).forEach (printjson)* **// print all document in DESC**

> *db.emp.find ().sort (****{job: -1, ename: 1}****).forEach (printjson)*

> *db.emp.find (****{deptno: {$in: [1, 3]}}****).sort (****{deptno: -1})***

> *db.emp.find(* ***{ $or: [{deptno:10}, {deptno:20}]}, {ename:true, deptno:true}****).sort(* ***{deptno:-1, ename:-1}****)* **//multiple column sort**

***.limit() (Rownum)***

*Use the limit () method on a cursor to specify the maximum number of documents the cursor will return. limit () is analogous to the LIMIT statement in a SQL database.*

> *db.getCollection ("emp").find ().****limit (20)****.forEach (printjson)*

**// print first 20 document**

> *db.getCollection ("emp").find ({deptno: 10}).****limit (20)****.forEach (printjson)*

**// print first 20 document of specified**

**// with condition.**

***.skip()***

> *db.emp.find({}, {ename:true,\_id:false}).****skip(5)***

***Count documents***

***Syntax:* *db.collection.count (<query>)***

* ***the {query} argument corresponds to the WHERE statement.***

> *db.emp.count ()* **// Count total number of documents.**

> *db.emp.count (****{job:'manager'}****)* **// Count total number of documents**

**// according to the condition.**

***db.collection.group ()***

***Syntax:* *db.collection.group ({key, reduce, initial, [keyf,] [cond,] finalize})***

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **key** | **document** | The field or fields to group. Returns a “key object” for use as the grouping key. |
| **reduce** | **function** | An aggregation function that operates on the documents during the grouping operation. These functions may return a sum or a count. The function takes two arguments: the current document and an aggregation result document for that group. |
| **initial** | **document** | Initializes the aggregation result document. |
| **keyf** | **function** | Optional. Alternative to the key field. Specifies a function that creates a “key object” for use as the grouping key. Use keyfinstead of key to group by calculated fields rather than existing document fields. |
| **cond** | **document** | Optional. The selection criteria to determine which documents in the collection to process. If you omit the cond field, db.collection.group() processes all the documents in the collection for the group operation. |
| **finalize** | **function** | Optional. A function that runs each item in the result set beforedb.collection.group() returns the final value. This function can either modify the result document or replace the result document as a whole. |

> *db.emp.group (****{key: {job: 1, deptno: 1}, reduce: function (curr, result) {}, initial: {}}****)*

***Queries document using findOne ()***

***Syntax:* *db.collection.findOne (<criteria>, <projection>)***

***Note: -***

* ***Returns one document that satisfies the specified query criteria. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk.***

> *db.emp.findOne ({deptno: 1})*

> *var x = db.emp.findOne ()*

> *print (tojson (x))*

{

"\_id": 1,

"empno": 7788,

"ename": "saleel",

"sal": 5000,

"deptno": 1

}

> ***print (tojson (x.ename))***

"saleel"

> ***print (tojson (x.ename + ' ' + x.sal))***

"saleel 5000"

***Queries document using $type***

***$type***

***Note: -***

* ***$type selects the documents where the value of the field is the specified BSON type.***

***Syntax:* *{field: {$type: <BSON type>}}***

***Refer to the following table for the available BSON types and their corresponding numbers.***

***Type Number***

1. ***Double***  *1*
2. ***String***  2
3. ***Object***  3
4. ***Array***  4
5. ***Binary data***  5
6. ***Undefined (deprecated)***  6
7. ***Object id***  7
8. ***Boolean***  8
9. ***Date***  9
10. ***Null***  10
11. ***Regular Expressio0n***  11
12. ***JavaScript***  13
13. ***Symbol***  14
14. ***JavaScript (with scope)***  15
15. ***32-bit integer***  16
16. ***Timestamp***  17
17. ***64-bit integer***  18
18. ***Min key***  255
19. ***Max key***  127

> *db.e.find ()* **// consider the following document for the**

**// above examples.**

{ "\_id": 1, "col1": 1, "col2": 1 }

{ "\_id": 2, "col1": 2, "col2": 2 }

{ "\_id": 3, "col1": 3, "col2": 3 }

{ "\_id": 4, "col1": 4, "col2": 4 }

{ "\_id": 5, "col1": 5, "col2": 5 }

{ "\_id": 6, "col1": 6, "col2": 6, "col3" : 6 }

{ "\_id": 7, "col1": 7, "col2": 7, "col3" : 7 }

{ "\_id": 8, "col1": 8, "col2": 8, "col3" : 8 }

{ "\_id": 9, "col1": 9, "col2": 9 }

{ "\_id": 10, "col1": 10, "col2": 10 }

{ "\_id": 11, "col1": 11, "col2": null }

{ "\_id": 12, "col1": 12, "col2": null }

{ "\_id": 13, "col1": "123", "col2": null }

{ "\_id": 14, "col1": 123.55, "col2": null }

{ "\_id": 15, "col1": 123456, "col2": null }

> *db.e.find (****{col1: {$type: 1}}****)* **// col1 having double type data.**

> *db.e.find (****{col1: {$type: 2}}****)* **// col1 having string type data.**

> *db.e.find (****{col2: {$type: 10}}****)* **// col2 having NULL data.**

***Queries document using $where***

***$where:***

***Note: -***

* ***Use the $where operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system.***

> *db.emp.find (****{$where: "this.salary > 8000"}****)*

> *db.emp.find (****{$where:"this.salary > 5000 && this.deptno == 40"}****)*

> *db.emp.find (****{$where: "obj.salary > 5000 && obj.deptno == 40"}****)*

> *db.emp.find ({****$where: 'this.deptno == 10* || *this.deptno == 20'}****)****.sort ({deptno: true})***

> *db.emp.find (****{$where: 'this.deptno == 10 && this.ename == "sharmin"'}).sort ({deptno: true})***

> *db.emp.find (****{$where :"( obj.job=='manager' || obj.job=='salesman') && obj.sal==2200"}****)*

*Reference the document in the JavaScript expression or function using either****this****or****obj****.*

***Function in $where:***

> *db.emp.find* ***({$where: function () {if (obj.balance==5000) {return obj.balance - obj.salary}}})***

***Distinct Clause***

***Syntax:* *db.collection.distinct (field, query)***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| **field** | **string** | The field for which to return distinct values. |
| **query** | **document** | A query that specifies the documents from which to retrieve the distinct values. |

* ***the <field> for which to return distinct values, and***
* ***the <query> that specifies the documents from which to retrieve the distinct values.***

> *db.publisher.distinct ('publisheraddress')* **// distinct documents.**

> *db.publisher.distinct ('publisheraddress',* ***{web: 'java.com'})***

**// distinct documents with condition.**

> *db.emp.distinct (****'job', {sal: {$gt: 4500}}****)* **// distinct documents with condition.**

|  |
| --- |
| ***Syntax*:**  > *db.emp.find(****{condition}, {field list’s}****).forEach(printjson)*  > *db.emp.find(****{query}, {projection}****)* |

* ***the {query} argument corresponds to the WHERE statement, and***
* ***the {projection} argument corresponds to the list of fields to select from the result set.***

> *db.emp.find (****{\_id: 1}****).forEach (printjson)* **// print the entire document matching to**

**// the condition *{\_id: 1}.***

***IMP: How to display specific field data.***

***0 / false: - Exclude fields.***

***1 / true: - Include fields.***

**Note: -** You cannot specify the combination of **0** ***[false]*** ***&*** **1 *[true]*** to display the fields, except in the case of ***\_ID: [objectID:].***

*> db.emp.find ({}, {}).forEach (printjson)* **//** **print all documents, all field from**

**// collections without condition.**

***Output***

{

***"\_id": 1,***

"empno": 3020,

"ename": "sharmin",

"job": "manager",

"hiredate": ISODate ("2013-08-05T03:15:45.012Z"),

"sal": 12000,

"comm": 4500,

"deptno": 20

}

{

***"\_id": 2,***

"empno": 8844,

"ename": "vrushali",

"job": "manager ",

"hiredate": ISODate ("2013-08-05T03:18:14.475Z"),

"sal": 9000,

"comm": 1500,

"deptno": 20,

***"mobile": [***

***9850884228,* // *Array of mobile field.***

***9850443775***

***]***

}

> *db.emp.find ({},* ***{ename: 1, job: 1, sal: 1}****).forEach (printjson)*

**// print all document of specified**

**// fields without condition.**

> *db.getCollection ("emp").find ()****.count ()***

**// Count total number of documents.**

> *db.getCollection ("emp").find ({\_id: 1})****.count ()***

**// Count total number of documents**

**// according to the condition.**

***0 / false: - Exclude fields.***

***1 / true: - Include fields.***

> *db.emp.find (****{\_id: 3}, {ename: 1, job: 1, sal: true}****).forEach (printjson)*

**// print all document of specified fields,**

**// with condition.**

> *db.emp.find (****{\_id: 3}****,* ***{ename: true, job: true, sal: true, \_id: false}****).forEach (printjson)*

**// print all document of specified fields,**

**// without ObjectID with condition.**

> *db.emp.find ({},* ***{ename: 0, job: 0, sal: false, \_id: false}****).forEach (printjson)*

**// print all document other than ename,**

**// job, sal and \_id field.**

***Like***

> *db.emp.find (****{ename: /^s/},*** *{ename: true}).forEach (printjson)*

**// prints all documents whose names starts**

**// with s.**

> *db.emp.find (****{ename: /s/},*** *{ename: true}).forEach (printjson)*

**// prints all documents whose names**

**// contains character s.**

> *db.emp.find (****{ename: /sa/},*** *{ename: true}).forEach (printjson)*

**// prints all documents whose names**

**// contains character s and a.**

> *db.emp.find (****{ename: /[sa]/},*** *{ename: true}).forEach (printjson)*

**// prints all documents whose names**

**// contains character s or a.**

***Aggregation***

***Syntax:* *db.collection.aggregate (pipeline)***

|  |  |
| --- | --- |
| **SQL Terms, Functions, and Concepts** | **MongoDB Aggregation Operators** |
| WHERE | [***$match***](http://docs.mongodb.org/manual/reference/operator/aggregation/match/#pipe._S_match) |
| GROUP BY | [***$group***](http://docs.mongodb.org/manual/reference/operator/aggregation/group/#pipe._S_group) |
| HAVING | [***$match***](http://docs.mongodb.org/manual/reference/operator/aggregation/match/#pipe._S_match) |
| SELECT | [***$project***](http://docs.mongodb.org/manual/reference/operator/aggregation/project/#pipe._S_project) |
| ORDER BY | [***$sort***](http://docs.mongodb.org/manual/reference/operator/aggregation/sort/#pipe._S_sort) |
| LIMIT | [***$limit***](http://docs.mongodb.org/manual/reference/operator/aggregation/limit/#pipe._S_limit) |
| SUM() | [***$sum***](http://docs.mongodb.org/manual/reference/operator/aggregation/sum/#grp._S_sum) |
| COUNT() | [***$sum***](http://docs.mongodb.org/manual/reference/operator/aggregation/sum/#grp._S_sum) |

* ***the aggregation <pipeline> is a framework for performing aggregation tasks. Using this framework, MongoDB passes the documents of a single collection through a pipeline. The pipeline transforms the documents into aggregated results.***

# **$match**

**$match $and**

> *db.emp.aggregate (****{$match: {deptno: 1, sal: 5000}}****)*

> *db.emp.aggregate (****{$match: {deptno: 1, sal: 5000}}****)*

> *db.emp.aggregate (****{$match: {sal: {$gt: 1000, $lt: 3000}}}****)*

> *db.emp.aggregate (****{$match: {deptno: 4}}****,* ***{$project: {ename: true, deptno: true, \_id: 0}}****)*

> *db.emp.aggregate ({$match:* ***{$and: [{sal: 3100}, {job: 'auditor'}]}****}).forEach (printjson)*

> *db.emp.aggregate ({$match:* ***{$and: [{sal: {$gt: 3100}}, {sal: {$lt: 3500}}]}****}).forEach (printjson)*

**$match $or**

> *db.emp.aggregate ({$match:* ***{$or: [{job: 'manager'}, {job: 'salesman'}]}****})*

**$match $and and $or**

> *db.emp.aggregate (****{$match: {$and: [{sal: {$gt: 3000}}, {sal: {$lt: 3500}}]}},******{$match: {$or: [{job: 'manager'}, {job: 'salesman'}]}}****).forEach (printjson)*

**$match $in**

> *db.emp.aggregate (****{$match: {deptno: {$in: [10, 30]}}}****,* ***{$sort: {deptno: -1}}****)*

>*db.emp.aggregate (****{$group: {\_id:'$job', count: {$sum: 1}}}, {$match: {count: {$gt: 4}}}****)***// condition on SUM aggregate**

**// function**

> *db.emp.aggregate (****{$group: {\_id:'$job', totalSalary: {$sum: '$sal'}}}, {$match: {totalSalary: 13000}}****)*

> *db.emp.aggregate ({$group: {\_id: '$job',* ***count: {$sum: 1},******totalSalary: {$sum: '$sal'}****}},* ***{$match: {totalSalary: 13000, count: 5}}****)*

# **$project**

> *db.emp.aggregate (****{$match: {deptno: 1}},******{$project: {ename: true}},******{$sort: {ename: -1}}****)*

> *db.emp.aggregate (****{$project: {ename: 1, sal: 1, comm: 1, hra: 1, da: 1, total: {$add: ["$sal","$comm","$hra"]}}}, {$sort: {total:-1}}, {$match: {total: {$ne: null}}}****)*

> *db.emp.aggregate (****{$project: {ename: '$ename', job: '$job'}}****)*

> *db.emp.aggregate (****{$project: {sal: true, Newsal: {$add:['$sal',1000]}}}, {$match:{Newsal:6000}}****)*

# **$unwind**

> *db.population.aggregate (****{$unwind: '$city'}, {$match: {'city.name': 'Rajauri'}},*** *{$project: {country: 1, state: 1, city: 1, total:* ***{$add: ['$city.male', '$city.female' ]}****}}).forEach (printjson)*

# **$sort**

> *db.emp.aggregate (****{$sort: {deptno: 1}}****)*

> *db.emp.aggregate (****{$sort: {deptno: 1, ename: 1}}****)*

> *db.emp.aggregate (****{$group: {\_id: '$job', Total: {$sum: '$salary'}}}, {$sort: {\_id:-1}}****)*

*> db.emp.aggregate (****{$project: {total: {$ifNull: ['$comm', 0]}, ename: true}}, {$sort: {total: 1}}****)*

***Group by clause in SQL***

# **$group**

> *db.emp.aggregate (****{$group: {\_id: '$deptno'}}****)*

**// group on deptno fields.**

> *db.emp.aggregate (****{$group: {\_id: null, TotalSalary: {$sum: '$sal'}}}****)*

**// group on null, will consider all**

**// documents as single group.**

> *db.emp.aggregate (****{$group: {\_id: {deptno: '$deptno', sal: '$sal'}}}****)*

**// group on multiple fields.**

>*db.emp.aggregate (****{$group: {\_id:'$job', count: {$sum: 1}}}, {$match: {count: {$gt: 4}}}****)***// condition on SUM aggregate**

**// function**

> *db.emp.aggregate (****{$group: {\_id:'$job', totalSalary: {$sum: '$sal'}}}, {$match: {totalSalary: 13000}}****)*

> *db.emp.aggregate ({$group: {\_id: '$job',* ***count: {$sum: 1},******totalSalary: {$sum: '$sal'}****}},* ***{$match: {totalSalary: 13000, count: 5}}****)*

> *db.getCollection ('emp').aggregate (****{$match: {'skills.language': 'java'}}, {$group: {\_id: '$job', count: {$sum: 1}}}****)*

>*db.emp.aggregate (****{$match: {$or: [{job:'manager'}, {job:'salesman'}]}},*** *{$group: { \_id: '$job', Count: {$sum: 1}}},* ***{$match: {Count: 5}}****)*

# **$group on multiple fields**

> *db.emp*.*aggregate* (***{$group: {\_id: {job: '$job', deptno: '$deptno'}, Total: {$sum: '$salary'}}}, {$sort: {\_id:1}}***) **// group on multiple fields with**

**// sorting.**

**$group *[Aggregation Operators]***

|  |  |
| --- | --- |
| **Name** | **Description** |
| ***$addToSet*** | Returns an array of all the *unique* values for the selected field among for each document in that group. |
| ***$first*** | Returns the first value in a group. |
| ***$last*** | Returns the last value in a group. |
| ***$max*** | Returns the highest value in a group. |
| ***$min*** | Returns the lowest value in a group. |
| ***$avg*** | Returns an average of all the values in a group. |
| ***$push*** | Returns an array of *all* values for the selected field among for each document in that group. |
| ***$sum*** | Returns the sum of all the values in a group. |

***$sum***

> *db.emp.aggregate (****{$group: {\_id:' $job', totalSalary: {$sum: '$salary'}}}****)*

> *db.emp.aggregate (****{$group: {\_id: null, totalSalary: {$sum: '$salary'}}}****)*

***$min***

> *db.emp.aggregate (****{$group: {\_id:' $job', minSalary: {$min: '$salary'}}}****)*

> *db.emp.aggregate (****{$group: {\_id: null, minSalary: {$min: '$salary'}}}****)*

***$max***

> *db.emp.aggregate (****{$group: {\_id:' $job', maxSalary: {$max: '$salary'}}}****)*

> *db.emp.aggregate (****{$group: {\_id: null, maxSalary: {$max: '$salary'}}}****)*

***$avg***

> *db.emp.aggregate (****{$group: {\_id:' $job', avgSalary: {$avg: '$salary'}}}****)*

> *db.emp.aggregate (****{$group: {\_id: null, avgSalary: {$avg: '$salary'}}}****)*

***$count***

>*db.emp.aggregate (****{$group: {\_id: '$job', countSalary: {$sum: 1}}}****)*

> *db.emp.aggregate (****{$group: {\_id: null, countSalary: {$sum: 1}}}****)*

# **$sum for Counting**

> *db.emp.aggregate (****{$group: {\_id: '$deptno', CountDEPT: {$sum: 1}}}****)*

> *db.emp.aggregate (****{$match: {deptno: {$in: [10, 20]}}}****,* ***{$group: {\_id: '$deptno', Count: {$sum: 1}}}****)*

***Arithmetic Aggregation Operators***

***$add*** ***// Computes the sum of an array of numbers.***

> *db.emp.aggregate (****{$project: {\_id: true, ename: true, salary: true, comm: true, total: {$add: ["$salary","$comm"]}}}****)*

> *db.emp.aggregate (****{$project: {job: true, total: {$add:["$sal","$comm","$da"]}}}****)*

> *db.emp.aggregate (****{$project: {\_id: 1, ename: true, Total1: {$add: ["$salary", "$comm"]}, Total2: {$add: [10, 20]}}}****)*

> *db.emp.aggregate (****{$project: {ename: 1, sal: 1, comm: 1, hra: 1, da: 1, total: {$add: ["$sal","$comm","$hra"]}}}, {$sort: {total:-1}}, {$match: {total: {$ne: null}}}****)*

***$divide*** ***// Takes two numbers and divides the first number by the second.***

> *db.emp.aggregate (****{$project: {\_id: true, ename: true, salary: true, comm: true, total: {$divide: ["$salary", "$comm"]}}}****)*

***$mod*** ***// Takes two numbers and calculates the modulo of the first number divided by the second.***

> *db.emp.aggregate (****{$project: {\_id: true, ename: true, salary: true, comm: true, total: {$mod: ["$salary", "$comm"]}}}****)*

***$multiply*** ***// Computes the product of an array of numbers.***

*> db.emp.aggregate (****{$project: {\_id: true, ename: true, salary: true, comm: true, total: {$multiply: ["$salary", "$comm"]}}}****)*

***$subtract*** ***// Takes two numbers and subtracts the second number from the first.***

> *db.emp.aggregate (****{$project: {\_id: true, ename: true, salary: true, comm: true, total: {$subtract: ["$salary", "$comm"]}}}****)*

***Nested Arithmetic Aggregation Operator***

> *db.emp.aggregate (****{$project: {sal: true, total: {$add: ["$sal", {$multiply: ["$sal", .25]}]}}}****)*

**$group *[Aggregation Operators]***

### *Date Operators*

|  |  |
| --- | --- |
| **Name** | **Description** |
| ***$dayOfYear*** | Converts a date to a number between 1 and 366. |
| ***$dayOfMonth*** | Converts a date to a number between 1 and 31. |
| ***$dayOfWeek*** | Converts a date to a number between 1 and 7. |
| ***$year*** | Converts a date to the full year. |
| ***$month*** | Converts a date into a number between 1 and 12. |
| ***$week*** | Converts a date into a number between 0 and 53 |
| ***$hour*** | Converts a date into a number between 0 and 23. |
| ***$minute*** | Converts a date into a number between 0 and 59. |
| ***$second*** | Converts a date into a number between 0 and 59. May be 60 to account for leap seconds. |
| ***$millisecond*** | Returns the millisecond portion of a date as an integer between 0 and 999. |

> *db.dt.aggregate (****{$project: {month: {$month: '$hiredate'}}}****)*

***Aggregation Commands***

# **aggregate**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **aggregate** | **string** | The ***name of the collection*** to as the input for the aggregation pipeline. |
| ***pipeline*** | **array** | An array of aggregation pipeline stages that process and transform the document stream as part of the aggregation pipeline. |
| **explain** | **boolean** | Optional. Specifies to return the information on the processing of the pipeline. |
| **allowDiskUse** | **boolean** | Optional. Enables writing to temporary files. When set to true, aggregation stages can write data to the \_tmp subdirectory in the[dbPath](http://docs.mongodb.org/manual/reference/configuration-options/#storage.dbPath) directory. |
| **cursor** | **document** | Optional. Specify a document that contains options that control the creation of the cursor object. |

***Pipeline Aggregation Operators***

|  |  |
| --- | --- |
| **Name** | **Description** |
| ***$project*** | Reshapes a document stream. [$project](http://docs.mongodb.org/manual/reference/operator/aggregation/project/#pipe._S_project) can rename, add, or remove fields as well as create computed values and sub-documents. |
| ***$match*** | Filters the document stream, and only allows matching documents to pass into the next pipeline stage. [$match](http://docs.mongodb.org/manual/reference/operator/aggregation/match/#pipe._S_match) uses standard MongoDB queries. |
| ***$redact*** | Restricts the content of a returned document on a per-field level. |
| ***$limit*** | Restricts the number of documents in an aggregation pipeline. |
| ***$skip*** | Skips over a specified number of documents from the pipeline and returns the rest. |
| ***$unwind*** | Takes an array of documents and returns them as a stream of documents. |
| ***$group*** | Group’s documents together for the purpose of calculating aggregate values based on a collection of documents. |
| ***$sort*** | Takes all input documents and returns them in a stream of sorted documents. |
| ***$geoNear*** | Returns an ordered stream of documents based on proximity to a geospatial point. |
| ***$out*** | Writes documents from the pipeline to a collection. The [$out](http://docs.mongodb.org/manual/reference/operator/aggregation/out/#pipe._S_out) operator must be the last stage in the pipeline. |

> *db.runCommand (****{aggregate: 'emp', pipeline: [{$project: {ename: true, job: true, sal: true, \_id: false}}]}****)*

> *db.runCommand ({aggregate: 'emp'****, pipeline: [{$match: {$or: [{job: 'salesman'}, {job: 'manager'}]}}, {$project: {empid: true, ename: true, job: true, skills: true, \_id: false}}, {$sort: {ename: 1}}, {$limit: 2}]}****)*

***$toUpper (aggregation)***

> *db.emp.aggregate (****{$group: {\_id: $toUpper: '$job'}}}****)*

# **$concat (aggregation)**

> *db.emp.aggregate ({$project:* ***{title: {$concat: ['Mr.', '$ename']}}****})*

***$ifNull (aggregation)***

***Syntax:* *db.< collection name>.aggregate( { $project : { < field name> : { $ifNull : [ <no-null expression>, <null expression>] } } } )***

> *db.emp.aggregate ({****$project: {Price: {$ifNull: ["$comm", 0]}}****})*

> *db.emp.aggregate ({****$project: {\_id: 1, 'myNum': {$add: ["$salary", {$ifNull: ["$comm", 0]}]}}****})*

***Remove() document (Delete Record)***

***Syntax:* *db.collection.remove (query, justOne)***

* ***the <query> Optional. Specifies deletion criteria using query operators. To delete all documents in a collection, omit this parameter or pass an empty document ({})., and***
* ***the <justOne> Optional. To limit the deletion to just one document, set to true. The default value is false..***

> *db.emp.remove (****{}****)* **// remove all documents**

> *db.emp.remove (****{\_id: 1}****)* **// remove documents according to condition.**

> *db.emp.remove (****{\_id: ObjectId ("5215756b6fc384d96d55b1c2")}****)*

**// remove documents according to condition.**

***1 / True***

> *db.food.remove (****{food: 'apple'}, True****)* **// remove only one document, if multiple**

**// documents are getting matched.**

***Update() document (Update Record)***

***Syntax:* *db.collection.update (<query>, <update>, <options>)***

***Syntax:* *db.collection.update (<query>, <update>, <upsert>, <multi>)***

***Field update operators***

***Name Description***

* 1. ***$inc*** *Increments the value of the field by the specified amount.*
  2. ***$rename*** *Renames a field.*
  3. ***$setOnInsert*** *Sets the value of a field upon documentation creation during an upsert. Has no effect on update operations that modify existing documents.*
  4. ***$set*** *Sets the value of a field in an existing document.*
  5. ***$unset*** *Removes the specified field from an existing document.*
* ***the <query> argument corresponds to the WHERE statement, and***
* ***the <update> corresponds to the SET ... statement, and***
* ***the <upsert> Optional. If set to true, creates a new document when no document matches the query criteria. The default value is false, which does not insert a new document when no match is found, and***
* ***the <multi> Optional. If set to true, updates multiple documents that meet the query criteria. If set to false, updates one document. The default value is false.***

***$set:***

> *db.bookinfo.update (****{\_id:1}, {$set: {pages: 1001, language: 'english'}}****)*

**// update multiple fields according to**

**// condition.**

> *db.emp.update (****{}, {$set: {balance: 5000}}****)*

***$unset:***

***Syntax:*** ***db.collection.update ({field: value1}, {$unset: {field1: "" }})***

> *db.emp.update (****{}, {$unset: {project: ''}}, {multi: true}****)*

**// removes the specified field from an**

**// existing document.**

***$rename:***

***Syntax:*** ***{$rename: {<old col1> : <new col1>, <old col2> : <new col2>}}***

> *db.emp.update (****{\_id: 14}, {$rename: {'mobile': 'mob', project: 'proj'}}****)*

**//rename the specified field from an**

**// existing document.**

***$inc:***

> *db.bookinfo.update (****{title: 'TATA'}, {$inc: {pages: 1}}, {multi: true}****)*

**// *{$inc}****)* **will****increase the pages field by**

**// value 1 all documents**

**// according to condition.**

***multi: true***

> *db.bookinfo.update (****{title: 'TATA'}, {$set: {pages: 2001}}, {multi: true}****)*

**// *{multi: true}*****will****update all documents**

**// according to condition.**

> *db.emp.update (****{}, {$set: {salary: 1000}}, {multi: true}****)*

**// *{} and {multi: true}*****will****update all**

**// documents**

***Add and Remove fields using Update***

***Add Fields***

> *db.bookinfo.update (****{\_id: 2},******{$set: {language: 'hindi', title: 'Core Java'}}****)*

**// 2 new fields language and title are added in**

**// bookinfo collection matching to the**

**// condition.**

> *db.emp.update (****{}, {$set: {balance: 5000}}, {multi: true}****)*

**// new field balance is added in emp**

**// collection.**

***Remove Fields***

***$unset:***

***e.g 1***

> *db.emp.update (****{ename: 'sam'}, {$unset: {project: true}}****)*

***Add and Remove fields using save command***

***Add Fields***

***e.g 1***

> ***x*** *= db.emp.findOne (****{ename: 'saleel'}****)* **// Step #1**

> *print (tojson (x))* **// Step #2**

> ***x****.newColumn = 'hello'* **// Step #3 add a new column in x object.**

> *db.emp.save (****x****)* **// Step #4 saves the object x for EMP**

**// document.**

***e.g 2***

> *db.emp.save ({\_id: 1, ename: "saleel", job: "manager", salary: 1,* ***dob: new Date ()****})*

**// DOB is new column**

***Remove Fields***

***e.g 1***

> ***x*** *= db.bookinfo.findOne (****{\_id: 2}****)* **// Step #1**

> *print (tojson (x))* **// Step #2**

> *delete* ***x.title*** **// Step #3 deletes the field title.**

> *db.emp.save (****x****)* **// Step #4 saves the object x for EMP**

**// document.**

***One-to-One Relationships Between Documents***

|  |
| --- |
| Note: - To achieve *One-to-One relationship* here we have create two collections (*bookmaster & authordetails*) to store the data.  ***bookmaster Collection***  {  ***"\_id":******'ISBN-10-10293-991*** ***',***  "title": "Oracle Database r12c The Complete Reference",  "pages": 1220  }  ***authordetails Collection***  {  "\_id": ObjectId ("521da6908d8653ce803a67d5"),  ***"bookid":******'ISBN-10-10293-991*** ***',***  "name": "kevin",  "email": "kevin@gmail.com",  "tags": [  "education",  "book"  ]  }  Note: - To achieve *One-to-One relationship* we have stored in one collection (*bookinfo*) only.  ***bookinfo Collection***  {  ***"\_id":*** ***'ISBN-10-10293-991*** ***',***  "title": "Oracle Database r12c The Complete Reference",  "pages": 1220,  ***"author":* {**  "name": "kevin",  "email": "kevin@gmail.com",  "tags": [  "education",  "book"  ]  **}**  } |

***One-to-Many Relationships Between Documents***

|  |
| --- |
| Note: - To achieve *One-to-Many relationship* here we have create two collections (*bookmaster & authordetails*) to store the data.  ***bookmaster Collection***  {  ***"\_id":******'ISBN-10-10293-991*** ***',***  "title": "Oracle Database r12c The Complete Reference",  "pages": 1220  }  ***authordetails Collection***  {  "\_id": ObjectId ("521ec3e01d3d1d4807fad3e2"), **// documents one**  ***"bookid":******'ISBN-10-10293-991*** ***',***  "name": "kevin",  "email": "kevin@gmail.com",  "tags": [  "education",  "book"  ]  }  {  "\_id": ObjectId ("521ec3ec1d3d1d4807fad3e3"), **// documents two**  ***"bookid":******'ISBN-10-10293-991*** ***',***  "name": "smith",  "email": " smith@gmail.com",  "tags": [  "education",  "book"  ]  }  Note: - To achieve *One-to-Many relationship* we have stored data in one collection (*bookinfo*) only.  ***bookinfo Collection***  {  ***"\_id":*** ***'ISBN-10-10293-991*** ***',***  "title": "Oracle Database r12c The Complete Reference",  "pages": 1220,  ***"author":* [**  **{**  "name": "kevin",  "email": "kevin@gmail.com",  "tags": [  "education",  "book"  ]  **},**  **{**  "name": "smith",  "email": "smith@gmail.com",  "tags": [  "education",  "book"  ]  **}**  **]**  } |

***publisher collection***

***document: 1***

> *db.publisher.insert (*

*{*

*"\_id" : "TATA",*

*"publishername" : "TATA McGraw-Hill",*

*"contectperson" : [*

*"kevin",*

*"roger"*

*],*

*"address" : [*

*{*

*"headoffice" : {*

*"lane" : 1,*

*"city" : "new delhi",*

*"phone" : 9850884228*

*},*

*"corporateoffice" : [*

*{*

*"lane" : 1,*

*"city" : "pune",*

*"phone" : "020-254-20-200"*

*},*

*{*

*"lane" : 3,*

*"city" : "baroda",*

*"phone" : "023-256-30100"*

*}*

*]*

*}*

*],*

*"email" : "tata@gmail.com",*

*"web" : "tataMcGraw-Hill.com"*

*}*

*)*

***document: 2***

> *db.publisher.insert (*

*{*

*"\_id" : "wrox",*

*"publishername" : "wrox",*

*"contectperson" : [*

*"smith",*

*"sam",*

*"richards"*

*],*

*"address" : [*

*{*

*"headoffice" : {*

*"lane" : "#137",*

*"city" : "new york",*

*"phone" : 9850883366*

*},*

*"corporateoffice" : [*

*{*

*"lane" : 1,*

*"city" : "pune",*

*"phone" : "020-254-22-222"*

*},*

*{*

*"lane" : "#1980/1022",*

*"city" : "banglore",*

*"phone" : "011-256-30135"*

*}*

*]*

*}*

*],*

*"email" : "wroxpublication@gmail.com",*

*"web" : "wroxpublisher.com"*

*}*

*)*

***document: 3***

> *db.publisher.insert (*

*{*

*"\_id" : "oracle",*

*"publishername" : "oraclepress",*

*"contectperson" : [*

*"scott",*

*"ivan"*

*],*

*"address" : [*

*{*

*"headoffice" : {*

*"lane" : "#1-37-5621#",*

*"city" : "bombay",*

*"phone" : 9850121212*

*},*

*"corporateoffice" : [*

*{*

*"lane" : 100,*

*"city" : "new delhi",*

*"phone" : "027-254-22-666"*

*},*

*{*

*"lane" : "#2081/7788",*

*"city" : "banglore",*

*"phone" : "011-254-43926"*

*}*

*]*

*}*

*],*

*"email" : "oracle@gmail.com",*

*"web" : "oraclepress.com"*

*}*

*)*

***JavaScript Function in MongoDB***

***e.g 1***

> *var F1 = function (x, y) {*

*... return (x + y);*

*... }*

***e.g 2***

> *function F2 (x, y) {*

*... return (x + y);*

*... }*

***e.g 3***

> *var F1 = function() {*

*... var s = 0 ;*

*... for (var x=1; x<10; x++) {*

*... s = s + (x\*2);*

*... print (x +' ' + s);*

*... }*

*... return x + ' ' + s;*

*... }*

> *F1 (10, 20)* **// Calling of JavaScript function.**

> *F2 (10, 20)* **// Calling of JavaScript function.**

> *F1* ()

***Copy all objects from one collection to another***

> *db.emp.find ().forEach* ***(function (x) {db.e.insert (x)})***

***Map-Reduce***

***Note: -***

* ***Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results.***

***Redis***

**Redis** is an open source, BSD licensed, advanced key-value cache and store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets, sorted sets, bitmaps and hyperloglogs.

*A key difference between Redis and other structured storage systems is that Redis supports not only strings, but also abstract data types:*

1. ***Lists*** *of strings*
2. ***Sets*** *of strings (collections of non-repeating unsorted elements)*
3. *Sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score)*
4. ***Hashes*** *where keys and values are strings*

***How to start redis server and client***

*E:\redis64-latest>* ***redis-server.exe***

***Note: - Set the number of databases from 16 to 26 in redis.windows.conf***

*E:\redis64-latest>* ***redis-server E:/redis64-latest/redis.windows.conf***

*E:\redis64-latest>* ***redis-cli.exe***

***Redis Commands***

*127.0.0.1:6379>* ***clear***

*127.0.0.1:6379>* ***CONFIG GET databases***

*127.0.0.1:6379>* ***select 0 // selects database***

***// 0 default db, total 16 db***

*127.0.0.1:6379>* ***ping***

*127.0.0.1:6379>* ***keys \* // Display all keys***

*127.0.0.1:6379>* ***keys \*c\* // Display keys by the name ‘c’***

*127.0.0.1:6379>* ***set col1 "sharmin bagde"******// key-value pair***

*127.0.0.1:6379>* ***append col1 " saleel" // Appends value***

*127.0.0.1:6379>* ***get col1 // Print’s value of key***

*127.0.0.1:6379>* ***mset k1 1521 k2 "saleel bagde" k3 9850884228 k4 "pune"***

***// Multiple cols set***

*127.0.0.1:6379>* ***mget k1 k2 k3 k4 // Multiple cols get***

*127.0.0.1:6379>* ***set x 1***

*127.0.0.1:6379>* ***incr x // Increments the value by 1***

*127.0.0.1:6379>* ***incrby x 4 // Increment by 4***

*127.0.0.1:6379>* ***decr col3 // Decrement value by 1***

*127.0.0.1:6379>* ***decrby x 2 // Decrements the value by 2***

***// steps***

*127.0.0.1:6379>* ***del col1 // Delete keys***

*127.0.0.1:6379>* ***flushdb //Remove all keys from the current database***

*127.0.0.1:6379>* ***flushall // Remove all keys from all databases***

*127.0.0.1:6379>* ***move k1 1 // Moves key from db [0] to db [1]***

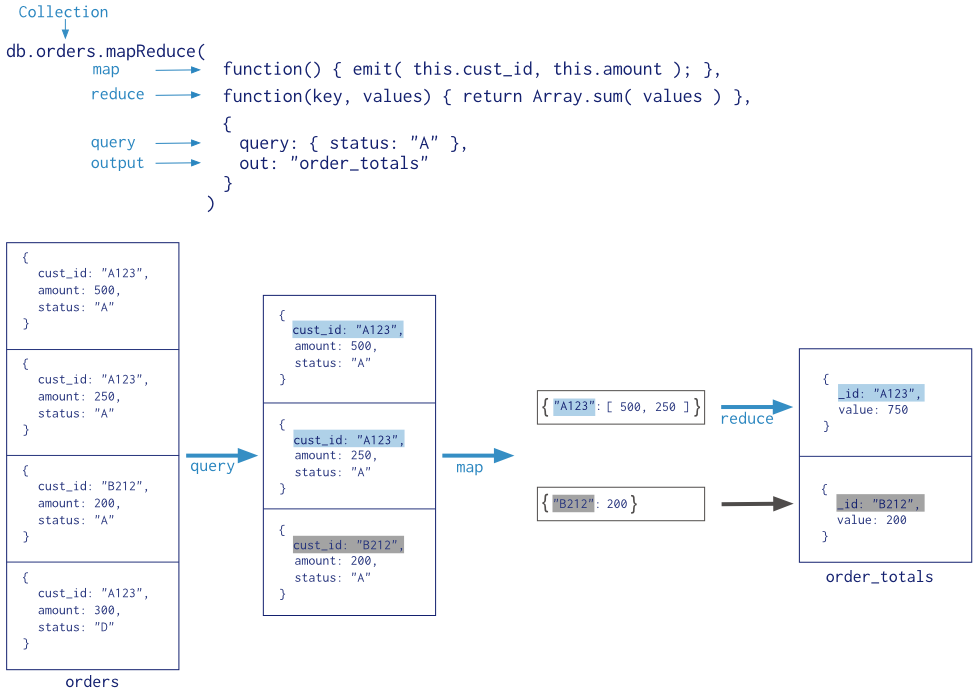
*127.0.0.1:6379>* ***client list***

*127.0.0.1:6379>* ***info***

*127.0.0.1:6379>* ***monitor***

*https://www.youtube.com/watch?v=BLQIyYreK64*

[*http://redis.io/commands*](http://redis.io/commands)



db.emp.find({\_id:3}, {ename:1 , job:1,sal:1})

> db.emp.insert({\_id:5, empno:8844, ename:'vasant', job:'clerk', hiredate:new Date(), sal:5000, comm:1500, deptno:20, mobile:{[9850884228, 9850443775]})

<http://www.youtube.com/watch?v=PlaJsseTgk4>

Primary Key in MongoDB: ObjectId

Primary Key and Foreign Key Concept in MongoDB

SELECT Columns or Fields ( { KEY: VALUE } ): MongoDB

MongoDB: It's Not Just About Big Data

by [handsonerp](http://www.youtube.com/user/handsonerp)

<http://www.youtube.com/watch?v=5R68LI0h6Fk&list=PLdDkCwh_eIfQe_ehPb0JII7d-pmGyDqbm>

<http://www.youtube.com/watch?v=u4BT9-3VN1Y&list=PLgWPwY_mXf4WehvLweImOtYwWPTi5RS-->

**> db.system.js.save({\_id:"addNumbers", value:function(x, y){ return x + y; }});**

**db.eval('addNumbers(17,18)');**

**> db.system.js.save ({\_id:"addNumbers",value:function(){return 2;}});**

**> db.system.js.save ({\_id:"addEmp", value: function (empNo, eName, hireDate){db.emp.insert ({empno: empNo, ename: eName, hiredate: hireDate})}});**

**> db.eval('addEmp(1233,"vipul",new Date())');**

*JSON*

*<html>*

***<script type="text/JavaScript">***

***var obj1 = {};***

***obj1.ename="Saleel";***

***obj1.data = {***

***street : "Paud Road",***

***city : "Pune",***

***state : "Maharashtra"***

***};***

***var obj2 = {};***

***obj2.phone = {***

***mobile: [9850884228, 9922200200],***

***landline: ['020-25420200', '022-8837723']***

***};***

***function F2() {***

***document.write ("Name: " + obj1.ename + "<br>");***

***document.write ("Street: " + obj1.data.street + "<br>");***

***document.write ("City: " + obj1.data.city + "<br>");***

***document.write ("State: " + obj1.data.state + "<br>");***

***document.write ("Mobile: " + obj2.phone.mobile [0] + "<br>");***

***document.write ("Land Line: " + obj2.phone.landline [1] + "<br>");***

***document.write ("<hr color='red' />");***

***}***

***</script>***

*<body>*

*<input type="button" id="B1" name="B1" value="Submit"* ***onclick="F2();"****>*

*</body>*

*</html>*

[*http://www.solarpowerenergyindia.com/home\_package.php*](http://www.solarpowerenergyindia.com/home_package.php)

[*http://www.dieselserviceandsupply.com/Power\_Calculator.aspx*](http://www.dieselserviceandsupply.com/Power_Calculator.aspx)

*> db.emp.aggregate ({$group: {\_id: '$deptno',total: {$sum:'$sal'}}}, {$group: {\_id: '$\_id.deptno', total1:{$max: '$total'}}})*

* ***The Database Writer (DBWR)*** *writes modified blocks from the buffer cache to the database files.*
* ***The Recovery*** *Writer (RVWR) writes modified blocks from the buffer cache to the flashback logs*
* ***The Log Writer (LGWR)*** *writes modified blocks from the log buffer to the redo log.*

***Demo Application on Bank***

*CREATE type ADDRESS as object*

*(addr1 varchar2 (10),*

*addr2 varchar2 (10));*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE table ACHOLDER*

*(acno number,*

*acname varchar2(10),*

*acdate date default sysdate,*

*addr address)*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE table TRANS*

*(srno number,*

*acno number,*

*transdate date,*

*transtype char,*

*amount number)*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE table BALANCE*

*(acno number,*

*total number)*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE or REPLACE trigger T1 before INSERT or DELETE on ACHOLDER for each row*

*declare*

*x number := 0 ;*

*begin*

*if inserting then*

*SELECT max (nvl2 (acno, acno, 1)) + 1 into x from ACHOLDER;*

*if x is null then*

*:new.acno := 1;*

*else*

*:new.acno := x;*

*end if;*

*INSERT into BALANCE values (:new.acno, 1000);*

*dbms\_output.put\_line ('=====================================');*

*dbms\_output.put\_line ('Record inserted successfully...');*

*dbms\_output.put\_line ('Account of ' || upper (:new.acname) || ' is created, A/C No is ' || to\_char (:new.acno,'000000000'));*

*dbms\_output.put\_line ('=====================================');*

*:new.acname := upper(:new.acname);*

*elsif updating then*

*null;*

*elsif deleting then*

*DELETE from BALANCE where acno = :old.acno;*

*DELETE from TRANS where acno = :old.acno;*

*dbms\_output.put\_line ('=====================================');*

*dbms\_output.put\_line ('Record deleted successfully...');*

*dbms\_output.put\_line ('Account of ' || :old.acname || ' is closed, A/C No is ' || :old.acno);*

*dbms\_output.put\_line ('=====================================');*

*end if;*

*end;*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE or REPLACE trigger T2 before INSERT on TRANS for each row follows t3*

*declare*

*x number;*

*y varchar2(20);*

*z number;*

*begin*

*if :new.amount > 0 then*

*SELECT rowid, total into y, z from balance where acno = :new.acno;*

*if SQL%Found then*

*--SELECT max (nvl2 (srno, srno, 1)) + 1 into x from TRANS where acno = :new.acno;*

*SELECT max (srno) + 1 into x from TRANS where acno = :new.acno;*

*if x is null then*

*:new.srno := 1;*

*else*

*:new.srno := x;*

*end if;*

*:new.TRANSTYPE := upper(:new.TRANSTYPE);*

*dbms\_output.put\_line ('=====================================');*

*dbms\_output.put\_line ('Transaction completed successfully...');*

*dbms\_output.put\_line ('=====================================');*

*if :new.TRANSTYPE = 'D' then*

*UPDATE BALANCE set total = total + :new.AMOUNT where acno = :new.acno;*

*elsif :new.TRANSTYPE = 'W' and z > :new.Amount then*

*UPDATE BALANCE set total = total - :new.AMOUNT where acno = :new.acno;*

*else*

*raise\_application\_error (-20000,'Low balance...');*

*end if;*

*if SQL%Found then*

*dbms\_output.put\_line (‘==================================');*

*dbms\_output.put\_line ('Balance updated successfully...');*

*dbms\_output.put\_line ('=====================================');*

*end if;*

*end if;*

*else*

*raise\_application\_error (-20000,'Amount must be more than 0...');*

*end if;*

*exception*

*when no\_data\_found then*

*raise\_application\_error (-20000,'Invalid Account...');*

*end;*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE or REPLACE trigger T3 before INSERT on TRANS for each row follows t4 when (new.TRANSTYPE not in ('D','d','W','w'))*

*begin*

*raise\_application\_error (-20000,'Invalid transaction type [W or D] only...');*

*end;*

*/*

*-------------------------------------------------------------------------------------------------------*

*CREATE or REPLACE trigger T4 before INSERT on TRANS for each row*

*begin*

*if to\_char(sysdate,'fmDay') = 'Sunday' then*

*raise\_application\_error (-20001,'Transaction cannot be done on Monday...');*

*end if;*

*end;*

*/*

*-------------------------------------------------------------------------------------------------------*

*SQL> SELECT n.acno, n.acname, n.acdate, n.addr.addr1, n.addr.addr2, total, cursor (SELECT \* from TRANS where trans.acno = n.acno order by 2, 1) from acholder n, balance where n.acno = balance.acno*

*-------------------------------------------------------------------------------------------------------*

*SQL > SELECT \* from TRANS group by acno, srno, transdate, transtype, amount order by 2;*

*-------------------------------------------------------------------------------------------------------*

*CREATE table t (*

*id number,*

*l\_blob blob );*

*SQL> INSERT into t values (1, utl\_raw.cast\_to\_raw ('Saleel Bagde'));*

*SQL> INSERT into t values (2, utl\_raw.cast\_to\_raw ('C:\img1.jpg'));*

*SQL> SELECT UTL\_RAW.CAST\_TO\_VARCHAR2 (l\_blob) from t;*

*-------------------------------------------------------------------------------------------------------*

*declare*

*--cursor c1 is SELECT tname from tab where tabtype='TABLE' order by 1 ;*

*--xRecord tab.tname%Type;*

*xx number;*

*begin*

*for xRecord in (SELECT tname from tab where tabtype='TABLE' order by 1)*

*loop*

*execute immediate 'SELECT count(\*) from ' || xRecord.tname into xx;*

*if xx >=10 then*

*dbms\_output.put\_line (xRecord.tname || ' has -> ' || xx || ' Record(s)');*

*end if;*

*end loop;*

*/\*open c1;*

*loop*

*fetch c1 into xRecord;*

*exit when c1%NotFound;*

*execute immediate 'SELECT count (\*) from ' || xRecord into xx;*

*if xx >=10 then*

*dbms\_output.put\_line (xRecord || ' has -> ' || xx || ' Record(s)');*

*end if;*

*end loop;*

*close c1; \*/*

*end;*

*/*

*System Information Command*

*C:\Users\administrator>****msinfo32***

***#########################################################***

*SQL> exec dbms\_java.grant\_permission ('C##SALEEL', 'SYS:java.net.SocketPermission', '192.168.100.20:1521', 'listen, resolve’);*

*SQL> call dbms\_java.grant\_permission ('C##SALEEL', 'java.util.PropertyPermission','\*','read, write');*

***#########################################################***

*CREATE or REPLACE and compile java source named CLASS1 as*

*import java.sql.\*;*

*public class Class1 {*

*public static java.lang.String InsertData () {*

*Connection cn = null;*

*Statement stmt = null;*

*try {*

***Class.forName ("oracle.jdbc.driver.OracleDriver");***

***cn = DriverManager.getConnection ("jdbc:oracle:thin:192.168.100.20:1521:orcl", "c##saleel","saleel");***

*stmt = cn.createStatement ();*

*String sql = "INSERT INTO dept VALUES (1, 'Zara', 'Ali', 6)";*

*stmt.executeUpdate (sql);*

*return ("Inserted records into the table...");*

*} catch (Exception ex) {*

*return (ex.getMessage());*

*}*

*}*

*}*

*/*

***#########################################################***

*CREATE or REPLACE function F1 return varchar2 as language java name ‘Class1.InsertData () return java.lang.String';*

*‘---------------------------------------*

***ALTER SESSION SET CONTAINER = PDB22;***

***Create user u1 identified by u1 container=current;***

SQL> CONN system/password@//localhost:1521/pdb1

SQL> connect sys/sony@192.168.100.20:1521/cdb1 as sysdba

***ALTER SESSION SET CONTAINER = PDB21;***

***create user u1 identified by u1 container=current;***

***sys/sony@cdb1 as sysdba***

***show con\_name***

***show parameter db\_name***

***SQL> select name, con\_id, open\_mode from v$pdbs;***

***SQL> alter pluggable database pdb21 open;***

***SQL> alter pluggable database pdb22 open;***

***SQL> alter pluggable database all open;***

ALTER PLUGGABLE DATABASE pdb1, pdb2 OPEN READ ONLY FORCE;

ALTER PLUGGABLE DATABASE pdb1, pdb2 CLOSE IMMEDIATE;

ALTER PLUGGABLE DATABASE ALL OPEN;

ALTER PLUGGABLE DATABASE ALL CLOSE IMMEDIATE;

ALTER PLUGGABLE DATABASE ALL EXCEPT pdb1 OPEN;

ALTER PLUGGABLE DATABASE ALL EXCEPT pdb1 CLOSE IMMEDIATE;

***Lsnrctl help***

***SQL> ALTER USER c##u1 quota unlimited on users;***

***declare***

***x varchar2(30);***

***str varchar2(100);***

***type xx is ref cursor;***

***c1 xx;***

***xRow varchar2(1000);***

***xx1 varchar2 (100);***

***begin***

***xx1 := '&tnm';***

***select cname into x from col where tname=xx1 and colno=2;***

***str := 'SELECT ' || x ||' from ' || xx1;***

***open c1 for str;***

***loop***

***fetch c1 into xRow;***

***exit when c1%NotFound;***

***dbms\_output.put\_line(xRow);***

***end loop;***

***close c1;***

***end;***

***/***

***SQL> CREATE TABLE T (C1 VARCHAR2 (10), C2 VARCHAR2 (4000) GENERATED ALWAYS AS (F1 (C1)) VIRTUAL);***

***CREATE OR REPLACE FUNCTION F1 (X VARCHAR2) RETURN VARCHAR2 DETERMINISTIC IS***

***BEGIN***

***RETURN (X);***

***END;***

***/***

***SQL> set long 32000***

***SQL> variable x1 CLOB***

***SQL> exec dbms\_utility.expand\_sql\_text ('SELECT TABLE\_NAME FROM DBA\_TABLES', :x1);***

***SQL> print x1***

[***http://www.avatto.com/computer-science/test/mcqs/questions-answers/database/73/1.html***](http://www.avatto.com/computer-science/test/mcqs/questions-answers/database/73/1.html)