Oracle 11g Database

Oracle SQL Basics

Agenda

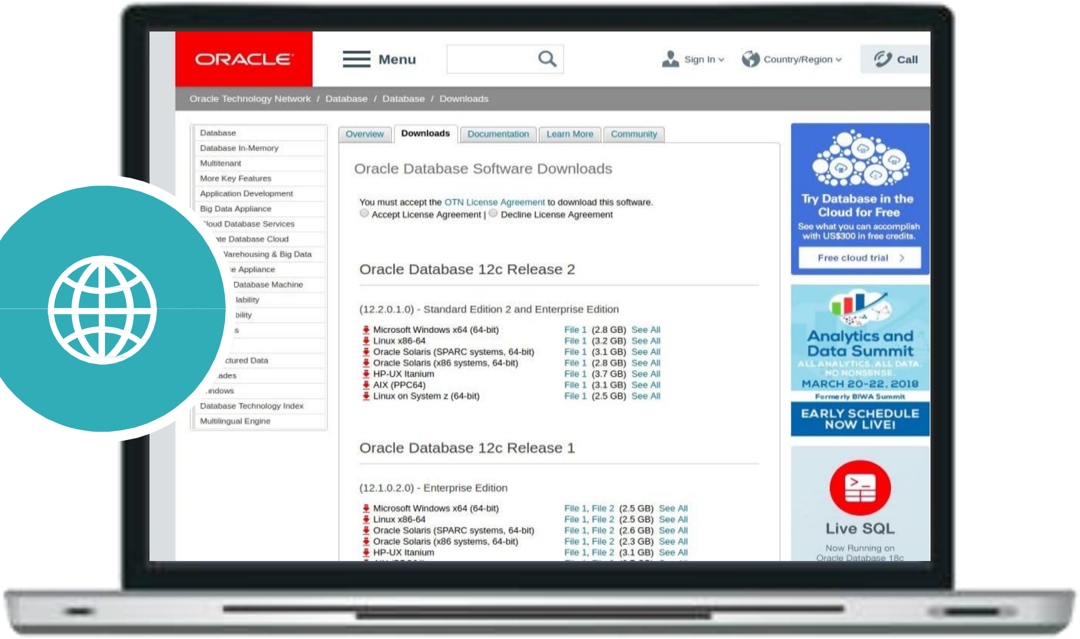
- Software Installation
- Database Concepts
- Database Fundamentals

SOFTWARE INSTALLATION

Installation Guide

Install Oracle Database:

- Dowanload oracle database from Oracle.com and install it.



Oracle Database Software Downloads

You must accept the OTN License Agreement to download this software.
Accept License Agreement | Decline License Agreement

Oracle Database 12c Release 2

(12.2.0.1.0) - Standard Edition 2 and Enterprise Edition

Microsoft Windows x64 (64-bit)	File 1	(2.8 GB)	See All
Linux x86-64	File 1	(3.2 GB)	See All
Oracle Solaris (SPARC systems, 64-bit)	File 1	(3.1 GB)	See All
Oracle Solaris (x86 systems, 64-bit)	File 1	(2.8 GB)	See All
HP-UX Itanium	File 1	(3.7 GB)	See All
AIX (PPC64)	File 1	(3.1 GB)	See All
Linux on System z (64-bit)	File 1	(2.5 GB)	See All

Oracle Database 12c Release 1

(12.1.0.2.0) - Enterprise Edition

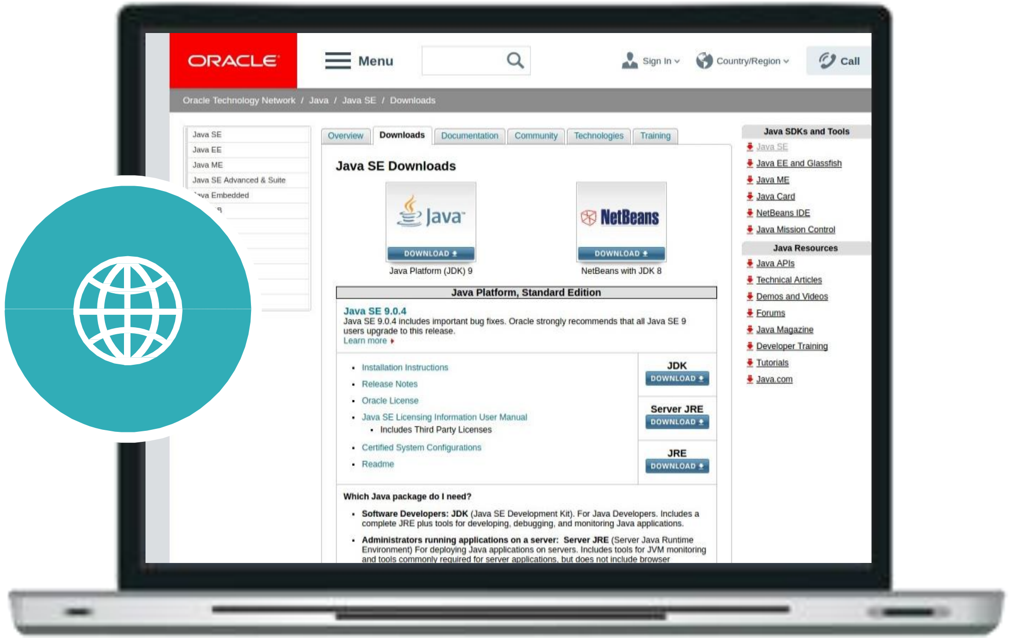
Microsoft Windows x64 (64-bit)	File 1, File 2	(2.5 GB)	See All
Linux x86-64	File 1, File 2	(2.5 GB)	See All
Oracle Solaris (SPARC systems, 64-bit)	File 1, File 2	(2.6 GB)	See All
Oracle Solaris (x86 systems, 64-bit)	File 1, File 2	(2.3 GB)	See All
HP-UX Itanium	File 1, File 2	(3.1 GB)	See All

Link: http://www.oracle.com/technetwo rk/database/enterprise-edition/ downloads/index.html

Installation Guide

Install Java SDK:

- Download latest Java SDK from Oracle.com and install it.Set Environment Path for SDK.



Java SE Downloads

Java Platform (JDK) 9

NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 9.0.4

Java SE 9.0.4 includes important bug fixes. Oracle strongly recommends that all Java SE 9 users upgrade to this release.
Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

JDK DOWNLOAD

Server JRE DOWNLOAD

JRE DOWNLOAD

Which Java package do I need?

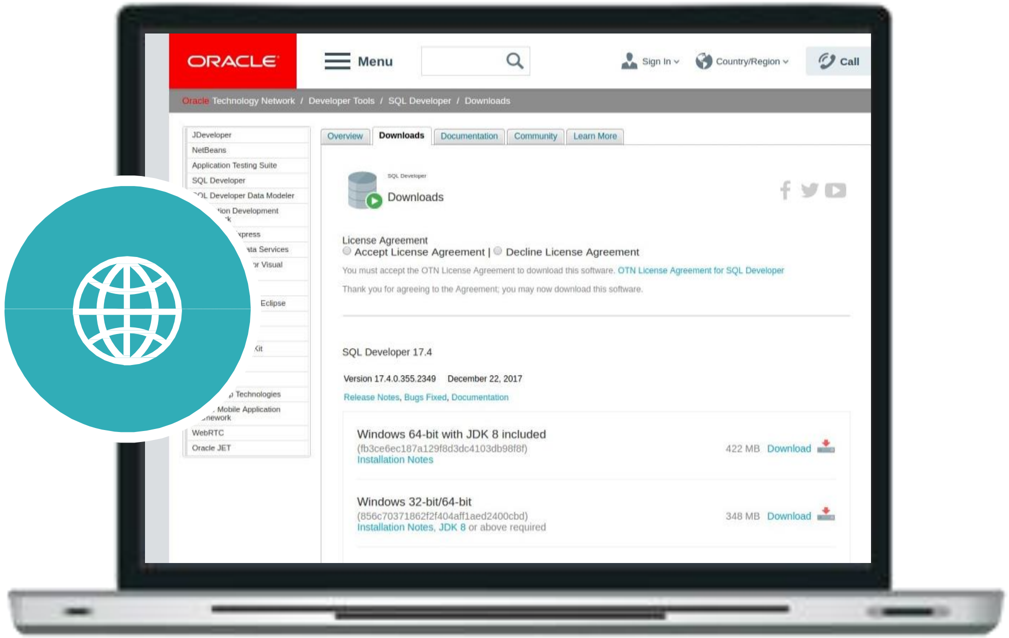
- **Software Developers: JDK** (Java SE Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
- **Administrators running applications on a server: Server JRE** (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser

Link: http://www.oracle.com/technetw ork/java/javase/downloads/inde x.html

Installation Guide

Install SQL Developer:

- Download SQL Developer from Oracle.com and install it. SQL Developer is tool to execute and create SQL Queries.



SQL Developer

Downloads

License Agreement
Accept License Agreement | Decline License Agreement

You must accept the OTN License Agreement to download this software. OTN License Agreement for SQL Developer

Thank you for agreeing to the Agreement; you may now download this software.

SQL Developer 17.4

Version 17.4.0.355.2349 December 22, 2017

Release Notes, Bugs Fixed, Documentation

Windows 64-bit with JDK 8 included
(fb3ce6ec187a129f8d3dc4103db98f8f)
Installation Notes
422 MB Download

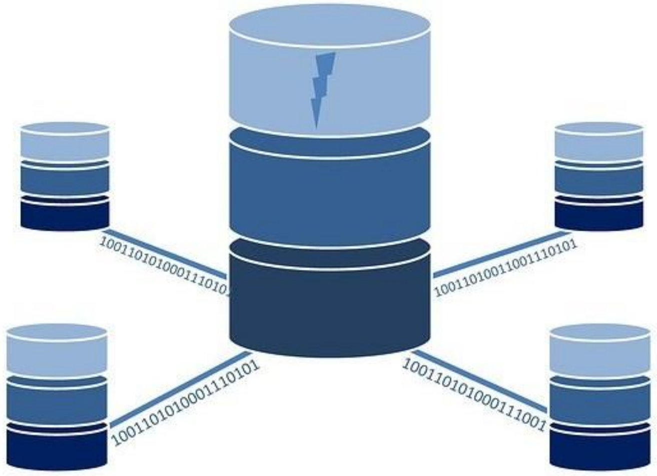
Windows 32-bit/64-bit
(856c70371862f2f404aff1aed2400cbd)
Installation Notes, JDK 8 or above required
348 MB Download

Link: http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html

DATABASE CONCEPTS

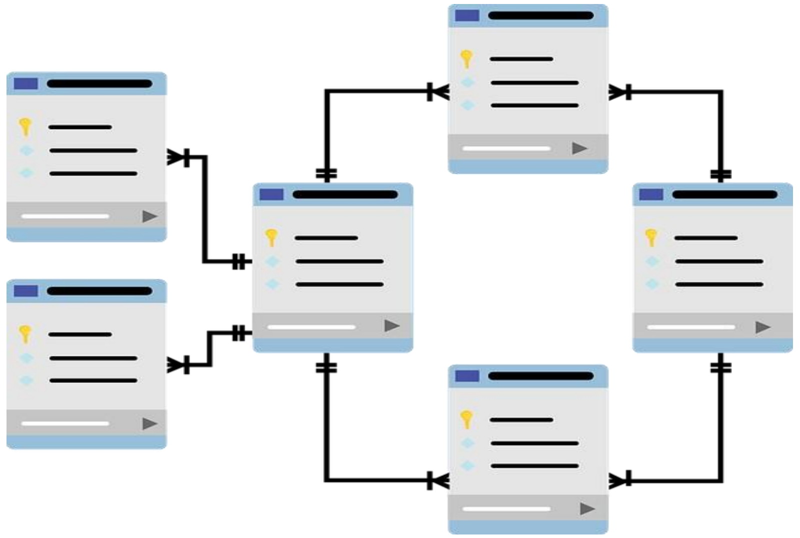
What is Database?

- A database is a collection of information that is well oraganised so that it can be easily accessed, managed and updated. It is a repository which stores the tables.



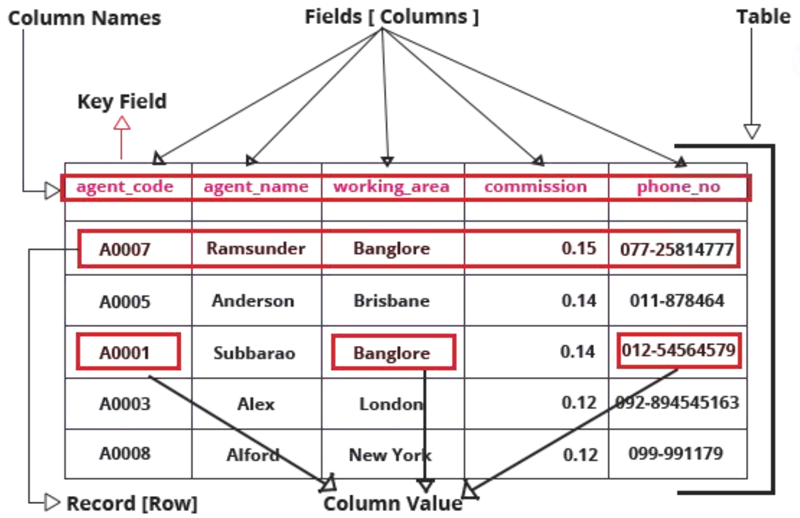
What is Relational Database (RDBMS)?

- RDBMS stores the data into collection of tables which might be realted by common fields(columns).



What is a Table?

- A table is a collection of related data held in a structured format within a database
- It consists of Fields(Columns) and Records(Rows)
- Every Column has a datatype- Table follows rules like if a column of number datatype can only hold number values so the data is in structured format

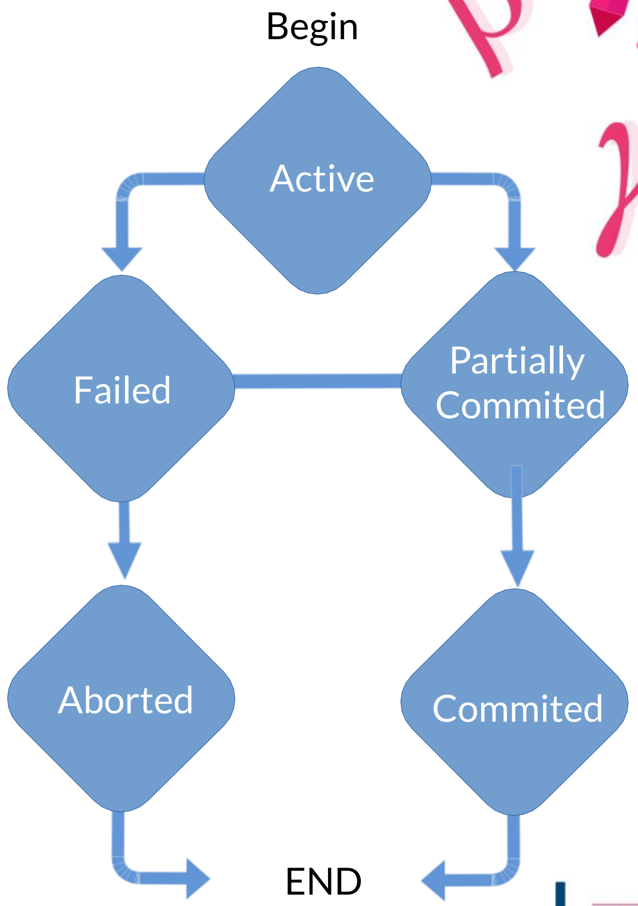


agent_code	agent_name	working_area	commission	phone_no
A0007	Ramsunder	Banglore	0.15	077-25814777
A0005	Anderson	Brisbane	0.14	011-878464
A0001	Subbarao	Banglore	0.14	012-54564579
A0003	Alex	London	0.12	092-894545163
A0008	Alford	New York	0.12	099-991179

What is Transaction?

- A transaction comprises of a Unit of work performed within a system against a database and is performed in a reliable way independent of other transactions.

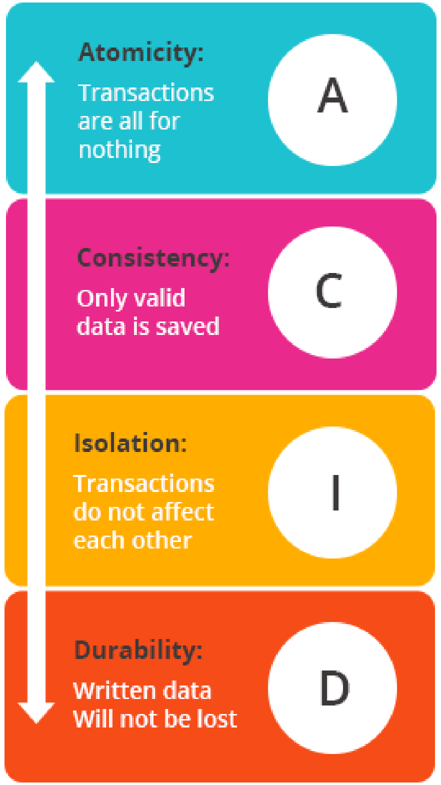
- A transaction is reliable, means if any step during a transaction gets failed then whole transaction would get failed



ACID Properties

ACID refers to the basic properties of a database transaction. All oracle datbase comply with ACID properties.

- **Atomicity:** The 'All' or 'Nothing' property.The entire seq of actions must be completed or aborted.
- **Consistency:** The transaction takes the resource from one steady state to another steady state.
- **Isolation:** A transaction's effect is not
- Visible to other transaction until the transaction is commited. A transaction can not interfere with another transaction
- **Durability:** Chnages made by the commited transaction are permanent and must survive the system failure.


Atomicity:
Transactions are all for nothing
A
Consistency:
Only valid data is saved
C
Isolation:
Transactions do not affect each other
I
Durability:
Written data Will not be lost
D

DATABASE FUNDAMENTALS

Enter The Database

How do we interact with a Database?

- SQL- Structured Query Langauge is a computer language used for storing, manipulating data stored in a relational database
- It supports all relational operator
- Can be embedded in any procedural language Used for insert and create data
- Very easy as the English language

Let's See what tables do I own?

- Using CAT data dictionary SELECT * FROM CAT;
- CAT - Catalouge of all the tables owned by the users

Let's Limit The Data

Use of Where clause for filtering numeric value

- SELECT * FROM sales WHERE total_amount > 1000;
- SELECT * FROM sales WHERE total_amount != 44;
- SELECT * FROM sales WHERE total_amount^44;
- SELECT * FROM sales WHERE quantity <= 10;

Use of Where clause for filtering text value

- SELECT * FROM sales WHERE sales_date = ’09-feb-2015’;
- SELECT * FROM product WHERE color = ‘RED’;

Use of Where clause for comparing column values

- SELECT * FROM sales WHERE total_amount > sales_amount;

Logical Operators

Use of BETWEEN and NOT BETWEEN

- SELECT * FROM sales WHERE total_amount NOT BETWEEN 1 and 100;
- SELECT * FROM sales WHERE total_amount BETWEEN 1 and 100;

Use of IN

- SELECT * FROM sales WHERE quantity IN (20,2,10);

Use of LIKE

- SELECT * FROM product WHERE product_name LIKE 'Mob%';
- SELECT * FROM product WHERE product_name LIKE '%Mob';
- SELECT * FROM product WHERE product_name LIKE 'Mob_Device';

Logical Operators

Use of ALL

- SELECT * FROM sales WHERE total_amount > ALL (50,100,200);

Use of ANY

- SELECT * FROM sales WHERE total_amount > ANY (50,100,200);

Use of NULL

- SELECT * FROM product WHERE color IS NULL;

Use of AND

- SELECT * FROM sales WHERE total_amount > 100 AND quantiy < 20;

Arithmetic Operator

- **Addition (+) :** SELECT 100/ 20 FROM DUAL; --5
- **Substraction (-) :** SELECT 100+20 FROM DUAL; --120
- **Multiplication (*) :** SELECT 100-20 FROM DUAL; --80
- **Division (/) :** SELECT 100*20 FROM DUAL; --2000
- **Modulus (%) :** SELECT 10%100 FROM DUAL; --10

Let's Sort the Data

Order By Clause

- SELECT sales_date, product_id, order_id, sales_amount, tax_amount FROM sales ORDER BY tax_amount;

- SELECT sales_date, product_id, order_id, sales_amount, tax_amount FROM sales ORDER BY sales_amount, tax_amount;

- SELECT order_id, sales_date, product_id, sales_amount, tax_amount FROM sales ORDER BY order_id DESC;

How NULL values are treated while sorting the Data?

- NULL values are treated as very large value by Oracle. So NULL data will sort to the bottom of the sort is in ascending order and to the top of the sort is in descending order.

Set Operators

- Set operators combines the result of two component queries into a single unit. Queries containing the set operators are called compound queries

- The data type of columns should be same to use the set operators

- Types of Set Operators: UNION, UNION ALL, INTERSECT, and MINUS

Set Operators

UNION ALL

- SELECT order_id FROM sales UNION ALL SELECT order_id FROM sales_history;

UNION

- SELECT order_id FROM sales UNION SELECT order_id FROM sales_history;

INTERSECT

- SELECT order_id FROM sales INTERSECT SELECT order_id FROM sales_history;

MINUS

- SELECT order_id FROM sales MINUS SELECT order_id FROM sales_history;

Let's group the data

Aggregate/Summary Functions

- Aggregate Funcitons returns a single result row based on the group of rows.
- Some of the functions are: MIN(), MAX(), COUNT(), SUM(), AVG()

SUM Function

- SELECT sales_date, SUM(total_amount) FROM sales GROUP BY sales_date;

Let's group the data

MAX Funciton

- SELECT sales_date, order_id, MAX(total_amount) FROM sales GROUP BY sales_date, order_id;

MIN Function with Having Clause

- SELECT sales_date, MIN(total_amount) FROM sales GROUP BY sales_date HAVING MIN(total_amount)< 100;

Difference Between Where and Having?

- Where clause is used to filter the detailed result data whereas Having clause is used to filter the aggregated result.

JOINS

Joins are used to join one or more table in database using a common column in tables.

Why JOINS?

- To get data from two or more tables in single SQL statement To avoid the unwanted duplication of data, we split the single table into multiple table and join them on the basis of common columns

Interesting Things

CASE Statements:

- The CASE Statements evaluated a single expression and compares it against several potential values, or evaluates multiple boolean expression and choose the first one that is true.

Alias Name:

- We can provide different titles to the column name. Spaces are not allowed in an alias name.
- If required, then use double quotes : SELECT SUM(AMOUNT) "TOTAL AMOUNT" FROM SALES;

Pseudo Columns in Oracle:

- A pseudo column is an Oracle assigned value used in the same context as column value but not stored on disk.
- SYSDATE: Returns Current Date
- USER: Returns the current timestamp
- ROWNUM: It indicates a number indicating the order of the row selected from the table
- ROWID: Returns the RowID(Binary Address) of a row in a database table

Data Definition Language DDL

1. Create Table statement

CREATE TABLE movies (Movie_number number, Movie_name varchar2(100), Movie_type varchar2(40), Movie_release_date date);

2. Add column to table

ALTER TABLE movies ADD (movie_language varchar2(30));

3. Modify Column attributes

ALTER TABLE movies MODIFY (movie_type varchar2(50));

4. Drop Table

DROP TABLE movies;

Data Definition Language DDL

5. Insert Values into a table

INSERT INTO movies VALUES (01, 'TERMINATOR', 'ACTION', '12-
JAN-2015'); COMMIT;

6. Update a record

UPDATE movies set movie_release_date = '14-
jan-2015' COMMIT;

WHERE movie_number = 101;

7. Delete a record

DELETE from movies WHERE movie_name = 'RUSH
HOUR'; COMMIT;

8. Truncate Statement

TRUNCATE TABLE
SALES;

Data Definition Language DDL

Difference between DELETE and TRUNCATE?

- ROLLBACK after DELETE can work, but not after TRUNCATE.
- TRUNCATE auto commits.
- DELETE generates a small amount of REDO space and a large amount of UNDO space but TRUNCATE generates neither of these two.

Let's Put Some Restrictions

Why constraints?

- Constraints apply the specific rule to data, ensuring the data confirms the requirement defined.
- Example: NOT NULL, UNIQUE, Primary Key, Check, Foreign Key

CHECK:

- Check constraint validates that value in a given column meets specific criteria.
- CREATE TABLE movies (Movie_number number, Movie_name varchar2(100), Movie_type varchar2(40) CHECK (movie_type IN ('ACTION', 'COMEDY')), Movie_release_date date);

Let's Put Some Restrictions

Foreign Key:

- A foreign Key constraint is used to enforce a relationship between two tables
- CREATE TABLE movies (Movie_number number, Movie_name varchar2(100), Movie_type varchar2(40), Movie_release_date date, Movie_director_number number REFERENCES director(director_number));

VIEWS

What is a VIEW?

- A view is simply the representation of a SQL statement that is stored in memory so that it can be easily reused.
    - A view gives a look of a table as defined by the select statement in the view definition
    - A view does not store data separately
    - Only the definition(SQL statement) of the view is stored
    - The data is retrieved from the underlying table based on the view definition

Advantages:

- Security - restrict the access of complete data to all
- Abstraction - Hiding complex logic and just displaying the required output

VIEWS

Create a View

- CREATE VIEW SALES_MOBILE AS SELECT S.SALES_DATE, S.ORDER_ID, S.QUANTITY, S.UNIT_PRICE, S.TOTAL_AMOUNT, P.PRODUCT_NAME, P.PRODUCT_CATEGORY FROM SALES S, PRODUCT P WHERE S.PRODUCT_ID = P.PRODUCT_ID AND PRODUCT_CATEGORY = 'Mobile';

Modify View

- CREATE OR REPLACE VIEW SALES_MOBILE AS SELECT S.SALES_DATE, S.ORDER_ID, S.QUANTITY, S.UNIT_PRICE, S.TOTAL_AMOUNT, P.PRODUCT_NAME, P.PRODUCT_CATEGORY, S.PRODUCT_ID FROM SALES S, PRODUCT P WHERE S.PRODUCT_ID = P.PRODUCT_ID AND PRODUCT_CATEGORY = 'Mobile';

Drop a View

- DROP view SALES_MOBILE;

Other Database Objects

Synonyms

- Synonym is an alternative name for a table, view, sequence, procedure, stored function
- Syntax: CREATE SYNONYM inventory_data FROM SALES;

Sequences

- A Sequence is an object in Oracle that is used to generate a number series(sequence).
- This can be useful when you need to create a Unique number to act as a primary key.
- Syntax: CREATE SEQUENCE VA_RECORD_ID MIN VALUE 1, MAX VALUE 999999, START_WITH , INCEREMENT BY 1, CACHE 10;
- NEXTVAL is used to get next value of sequence and CURRVAL is used to get the current value.

Giving Permissions To Other Users

GRANT

- GRANT command can be used to grant schema object privileges to the role or to the user
- WITH GRANT OPTION enables the user to pass on the privilege to other user or role.
- Syntax: GRANT SELECT ON SALES TO SCOTT WITH GRANT OPTION;

REVOKE

- REVOKE command can be used to take back the granted privileges to the user or role
- Syntax: REVOKE ALL ON SALES FROM SCOTT;

Sub Queries

Sub Suery

- A sub query is a query within a query which can return one or more rows. A subquery executes inner query before the main query

Multiple Column Subqueries

- Pair wise Comparision: Values compared in pair

✓ **Syntax:** SELECT sales_date, order_id, customer_id FROM SALES WHERE (product_id, unit_price) IN (SELECT product_id, unit_price FROM SALESPERSON where sales_date='01-Jan-2009');

- Non Pairwise Comparision: Values are compared individually

Sub Queries

WITH Clause

- To hold the result of a SQL statement using a WITH clause and use it in multiple SQL Statement.
  - ✓ Syntax: WITH st as (SELECT * FROM SALES_TOTAL) SELECT * FROM
  - ✓ SALES s, st WHERE s.sales_date=st.sales_date;

Scaler Subquery

- Scaler sub queries will allow treating the output of a sub query as a column or even an expression within a select statement.
  - ✓ It must return only one row and one column.
  - ✓ Syntax: SELECT s.sales_date, s.id, (SELECT SUM(total_sakes) FROM SALES) as sales_total from SALES s;

Sub Queries

Corelated Subquery

- Correlated sub query is a subquery that uses the values from the outer query and is evaluated once for each row processed by the outer query.

  ✓ Syntax: SELECT * FROM SALES x where total_amt>(SELECT AVG(sales_amt) FROM SALES y WHERE y.c_id=x.c_id);

Index

Index

- An index is a performance tuning method of allowing faster retrieval of records

Properties:

- Indexes enable faster data acess
- Index stores column values and their location
- The index can be created on multiple columns

DROP Index:

DROP INDEX <index_name>;

Index

Unique Index

A unique Index is an Index no duplicate values are allowed

- Unique Index will not accept duplicate values
- It can have null values until and unless restricted

✓ Syntax: CREATE UNIQUE INDEX cust_idx ON CUSTOMER(cust_id);

Rename Index

ALTER INDEX <index_old_name> RENAME TO new_name_of_index>;

Function

Function

A function is a subprogram that is used to return a single value. You must declare and define a function before invoking it. It can be declared and defined at a same time or can be declared first and defined later in the same block.

Syntax

```
CREATE [OR REPLACE] FUNCTION function_name
  [ (parameter [,parameter]) ]
RETURN return_datatype
IS | AS
 [declaration_section]
BEGIN
  executable_section
[EXCEPTION
  exception_section]
END [function_name];
```

- **IN:** It is a default parameter. It passes the value to the subprogram.
- **OUT:** It must be specified. It returns a value to the caller.
- **IN OUT:** It must be specified. It passes an initial value to the subprogram and returns an updated value to the caller.

Function

Oracle Function Example

Let's see a simple example to **create a function.**

```
create or replace function adder(n1 in number, n2 in number)
return number
is
    n3 number(8);
begin
    n3 :=n1+n2;
    return n3;
end;
/
```

Function

Oracle Function Example

Now write another program to **call the function.**

```
DECLARE
   n3 number(2);
BEGIN
   n3 := adder(11,22);
   dbms_output.put_line('Addition is: ' || n3);
END;
/
```

Output

```
Addition is: 33
Statement processed.
0.05 seconds
```

Procedure

Procedure

- A procedure is a group of PL/SQL statements that can be called by name. The call specification (sometimes called call spec) specifies a java method or a third-generation language routine so that it can be called from SQL and PL/SQL.

Syntax

```
CREATE [OR REPLACE] PROCEDURE procedure_name
   [ (parameter [,parameter]) ]
IS
   [declaration_section]
BEGIN
   executable_section
[EXCEPTION
   exception_section]
END [procedure_name];
```

- **IN:** It is a default parameter. It passes the value to the subprogram.
- **OUT:** It must be specified. It returns a value to the caller.
- **IN OUT:** It must be specified. It passes an initial value to the subprogram and returns an updated value to the caller.

Procedure

Oracle Function Example

In this example, we are going to insert record in the "user" table. So you need to create user table first.

Table creation:

```
create table user(id number(10) primary key,name varchar2(100));
```

Now write the procedure code to insert record in user table.

Procedure Code:

```
create or replace procedure "INSERTUSER"
(id IN NUMBER,
       name IN VARCHAR2)
is
begin
      insert into user values(id,name);
end;
/
```

Procedure

Oracle Function Example

Oracle program to call procedure

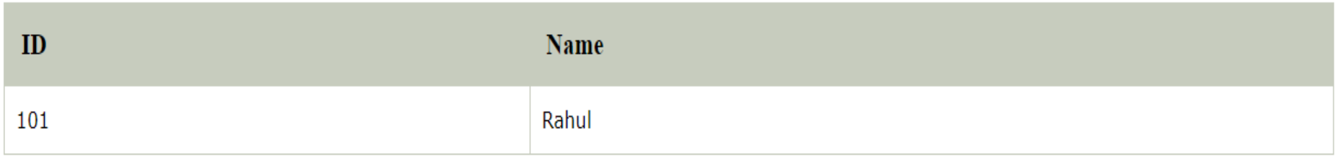
```
BEGIN

  insertuser(101,'Rahul');

  dbms_output.put_line('record inserted successfully');
END;

/
```

Now, see the "USER" table, you will see one record is inserted.



ID	Name
101	Rahul

Triggers

- A database trigger is a stored PL/SQL program unit associated with a specific database table.

- ORACLE executes (fires) a database trigger automatically when a given SQL operation (like INSERT, UPDATE or DELETE) affects the table.

- Unlike a procedure, or a function, which must be invoked explicitly, database triggers are invoked implicitly.

Triggers

Database triggers can be used to perform any of the following:

- Audit data modification
- Log events transparently
- Enforce complex business rules
- Derive column values automatically
- Implement complex security authorizations
- Maintain replicate tables

Triggers

- You can associate up to 12 database triggers with a given table.

- A database trigger has three parts: a triggering event, an optional trigger constraint, and a trigger action.

- When an event occurs, a database trigger is fired, and an predefined PL/SQL block will perform the necessary action.

Triggers

SYNTAX:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE|AFTER} triggering_event ON table_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
    Declaration statements
BEGIN
    Executable statements
EXCEPTION
    Exception-handling statements
END;
```

Triggers

- The trigger_name references the name of the trigger.
- BEFORE or AFTER specify when the trigger is fired (before or after the triggering event).
- The triggering_event references a DML statement issued against the table (e.g., INSERT, DELETE, UPDATE).
- The table_name is the name of the table associated with the trigger.
- The clause, FOR EACH ROW, specifies a trigger is a row trigger and fires once for each modified row.
- A WHEN clause specifies the condition for a trigger to be fired.
- Bear in mind that if you drop a table, all the associated triggers for the table are dropped as well.

Types of Triggers

- Triggers may be called BEFORE or AFTER the following events:
- INSERT, UPDATE and DELETE.
- The before/after options can be used to specify when the trigger body should be fired with respect to the triggering statement.
- If the user indicates a BEFORE option, then Oracle fires the trigger before executing the triggering statement.
- On the other hand, if an AFTER is used, Oracle fires the trigger after executing the triggering statement.

Types of Triggers

- A trigger may be a ROW or STATEMENT type. If the statement FOR EACH ROW is present in the CREATE TRIGGER clause of a trigger, the trigger is a row trigger.
- A row trigger is fired for each row affected by an triggering statement.
- A statement trigger, however, is fired only once for the triggering statement, regardless of the number of rows affected by the triggering statement

Types of Triggers

Example: statement trigger

```
CREATE OR REPLACE TRIGGER mytrig1 BEFORE DELETE OR INSERT
        OR UPDATE ON employee
BEGIN
IF     (TO_CHAR(SYSDATE,    'day')    IN    ('sat',    'sun'))    OR
        (TO_CHAR(SYSDATE,'hh:mi')   NOT   BETWEEN   '08:30'   AND
        '18:30') THEN          RAISE_APPLICATION_ERROR(-20500, 'table
        is secured');
END IF;
END;
/
```

- The above example shows a trigger that limits the DML actions to the employee table to weekdays from 8.30am to 6.30pm.
- If a user tries to insert/update/delete a row in the EMPLOYEE table, a warning message will be prompted.

Example – ROW Triggers

```
CREATE OR REPLACE TRIGGER mytrig2
AFTER DELETE OR INSERT OR UPDATE ON employee
FOR EACH ROW
BEGIN
IF DELETING THEN
INSERT INTO xemployee (emp_ssn, emp_last_name,emp_first_name, deldate)
VALUES (:old.emp_ssn, :old.emp_last_name,:old.emp_first_name, sysdate);
ELSIF INSERTING THEN
 INSERT INTO nemployee (emp_ssn, emp_last_name,emp_first_name, adddate)
VALUES (:new.emp_ssn, :new.emp_last_name,:new.emp_first_name, sysdate);
 ELSIF UPDATING('emp_salary') THEN
INSERT INTO cemployee (emp_ssn, oldsalary, newsalary, up_date)
VALUES (:old.emp_ssn,:old.emp_salary, :new.emp_salary, sysdate);    ELSE
INSERT INTO uemployee (emp_ssn, emp_address, up_date)
VALUES (:old.emp_ssn, :new.emp_address, sysdate);
END IF;
END;
/
```

Example – ROW Triggers

- The previous trigger is used to keep track of all the transactions performed on the employee table.

- If any employee is deleted, a new row containing the details of this employee is stored in a table called xemployee.

- Similarly, if a new employee is inserted, a new row is created in another table called nemployee, and so on.

- Note that we can specify the old and new values of an updated row by prefixing the column names with the :OLD and :NEW qualifiers.

Example – ROW Triggers

```
SQL>  DELETE FROM  employee WHERE emp_last_name = 'Joshi';
1 row deleted.
SQL> SELECT * FROM xemployee;


EMP_SSN   EMP_LAST_NAME   EMP_FIRST_NAME DELDATE
-------------  -----------------------   -------------------------- -----------------
999333333  Joshi                     Dinesh               02-MAY-03
```

Enabling, Disabling & Dropping TRIGGERS

SQL>ALTER TRIGGER trigger_name DISABLE;

SQL>ALTER TABLE table_name DISABLE ALL TRIGGERS;

To enable a trigger, which is disabled, we can use the following syntax:

SQL>ALTER TABLE table_name ENABLE trigger_name;

All triggers can be enabled for a specific table by using the following command

SQL> ALTER TABLE table_name ENABLE ALL TRIGGERS;

SQL> DROP TRIGGER trigger_name

Thank You!