Certificate in

Client Server Technology

CCST

Study Material

Youth Computer Training Centre

Topics Covered:

- **WINDOWS NT SERVER**
- **ORACLE**

Warning and Disclaimer

This study material is designed to provide information about 'Certificate in Client Server Technology'. While every effort has been made to make this study material as complete, comprehensive and as accurate as possible, no warranty or fitness is to be implied. The information is provided on an "as is" basis. Hewlett Packard Enterprise shall not have any liability or responsibility to any person or entity with respect to any loss or damages arising from the information contained in this study material. Internet websites offered as citations and/ or sources for further information may have changed or disappeared between the time this study material is written and the time when it may be read by the reader.

Contents

Chapter 1: WINDOWS NT Server

About NT Server

An NT Server refers to Microsoft Windows NT Server, an operating system developed by Microsoft as part of its Windows NT (New Technology) family. Windows NT Server was designed to serve as a network operating system (NOS) for managing and supporting enterprise-level networking and computing environments.

Key Features of NT Server

Domain Controller:

Windows NT Server could function as a domain controller, managing authentication and user accounts in a domain-based network.

User and Group Management:

Centralized control of user accounts and security policies via domains.

File and Print Sharing:

Acted as a file server or print server for networked clients.

Scalability:

Supported enterprise environments with features like multiprocessing and large-scale memory support.

Security:

Utilized an Access Control List (ACL)-based permissions model.

Provided user authentication using NTLM (NT LAN Manager) protocols.

Support for Networking Protocols:

Integrated support for TCP/IP, NetBIOS, and other protocols for connecting to different devices.

Built-in Services:

Provided DNS, DHCP, WINS, and other essential services for networking.

Windows NT Server Versions

The Windows NT Server family included:

Windows NT 3.1 (1993):

The first version of Windows NT.

Windows NT 3.5 and 3.51 (1994-1995):

Improved performance and compatibility.

Windows NT 4.0 (1996):

Included a user interface similar to Windows 95 and became popular in enterprise environments.

Windows 2000 Server:

Although technically not branded as "Windows NT" it was built on NT technology and replaced NT Server 4.0.

WINDOWS NT Server in Modern Context

While Windows NT Server itself is no longer in use, its legacy continues in modern Windows Server operating systems, such as:

- Windows Server 2008
- Windows Server 2012
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

These newer versions incorporate and expand upon the foundational principles of Windows NT Server.

Windows NT File System

In any OS, every partition of the Hard Disk must be formatted with a supported file system before it can be addressed by that OS. Windows NT supports three file systems: FAT, NTFS and CDFS. CDFS is a read-only file system for CD-ROMs. Here we will discuss only FAT & NTFS because these are the most widely used file systems.

FAT File System

FAT was originally invented for MS-DOS, but Windows NT, Windows 95/98 and OS/2 now supports this file system, making it the most universally accepted file system. But FAT has its own disadvantages:

FAT is slow. FAT uses an unsorted linked list directory structure. As files are added to a directory, they are appended to the end of the directory list. Windows NT takes longer to find files at the end of the list.Even at the file level, when data is added to an existing file, FAT links the file sector by sector, often fragmenting the disk in the process. For partitions greater than 200 MB in size, FAT performance degrades quickly.

FAT is small. The maximum file, directory, or partition size is only 4 GB in NT and 2GB in DOS.

FAT is insecure. Administrator cannot protect FAT resources with Windows NT user-level security.

FAT is unsafe. In case of power failure during a disk transaction, the FAT file system may be corrupted, resulting in cross-linked files or orphan clusters.

NTFS File System.

The New Technology File System (NTFS) was introduced with Windows NT 3.1

NTFS has advantages over FAT in the following areas:

NTFS is big. This is a 64-bit file system, which means that NTFS files and partitions can be inordinately large. Maximum partition size is 2 TB and maximum file size is 64 GB. This helps in supporting large database files.

NTFS is secure. Individual files and directories can be protected with user-level security Access or attempts to access NTFS resources can be audited.

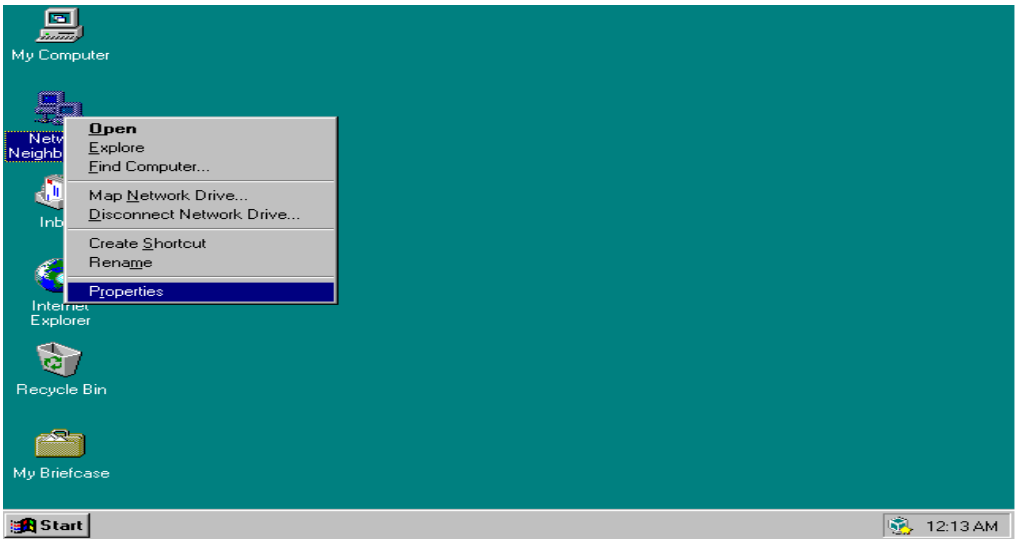
NTFS is robust. NTFS supports sector sparing, also known as hot Fixing. If a sector fails on an NTFS partition, NTFS attempts to write the data to a good sector (if the data is still in memory) and map out the bad sector so that it will not be re-used. If the power fail, leaving NTFS in a corrupt state, the CHKDSK command, executed when the system boots, would attempt to redo the transaction (in the case of a delete, for example) or undo the transaction (in the case of a file write where the data would no longer be in memory).

NTFS is fast. In general, files don't get fragmented with NTFS. After a file is committed to disk, NTFS always finds a contiguous run of disk space for the file (if one is available). The only instance in which files become fragmented is if the drive is almost full and there is not a contiguous space large enough for the file, or if a saved file is modified and expanded.

NTFS has its own drawbacks too:

NTFS has a lot of overhead. Typically, NTFS requires somewhere in the range of 4.5—10 MB for the file system itself. So, floppy disks cannot be formatted with NTFS.NTFS is only available to a Windows NT-based operating system. If computer boots between Windows 95 and Windows NT, we cannot access our NTFS partition under Windows 95

NETWORKING WINDOWS NT



After completion of this chapter the student will be able to:

Install and Configure Network Adapter

Add Network Protocols

Configure TCP/IP

Introduction

In this section we will discuss how to set the commonly used networking configuration in Windows NT.

Adding New Network Adapter

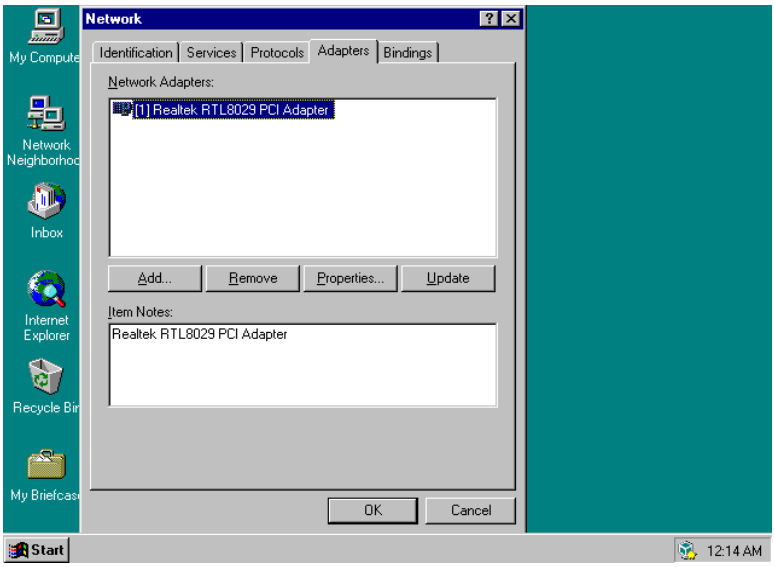
Adding a New Network protocol

For each of these configuration you may either click on the Network Neighborhood "icon" on the desktop or by opening the Network in the control panel.

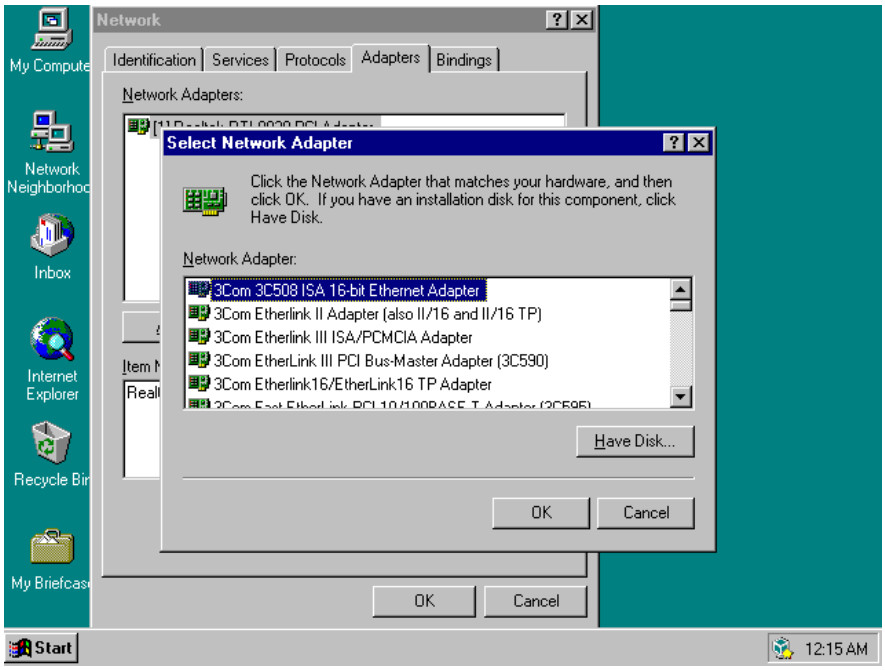
Add New Network Adapter

Right click the "Network Neighborhood" icon in your desktop and choose Properties from the menu. Another way to access it is by opening the Network Control Panel this will open the next screen.

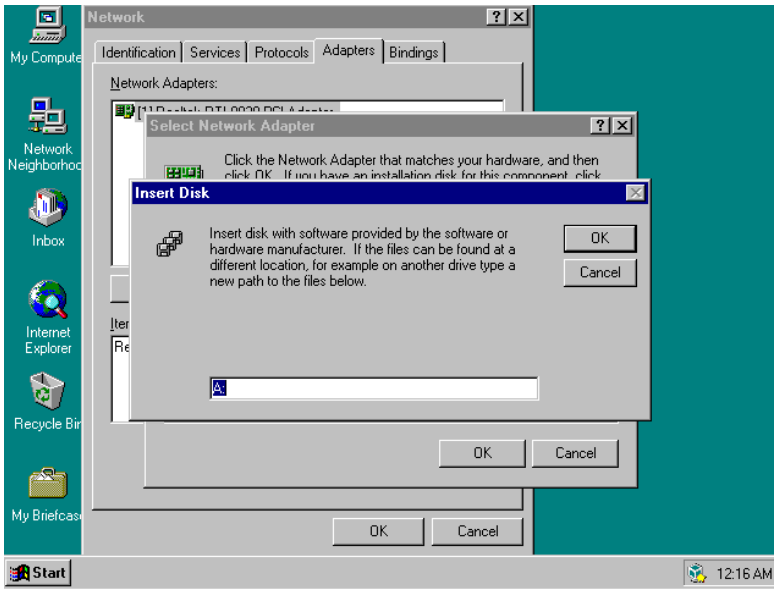
Click the "Adapters" tab to open it. Click the "'Add" button to add a new Adapter



Choose the Adapter you have installed in your computer and then click ok. If your adapter is not in the list, click “HAVE DISK” and follow the instructions to install it from the disks supplied by the manufacturer of your adapter.

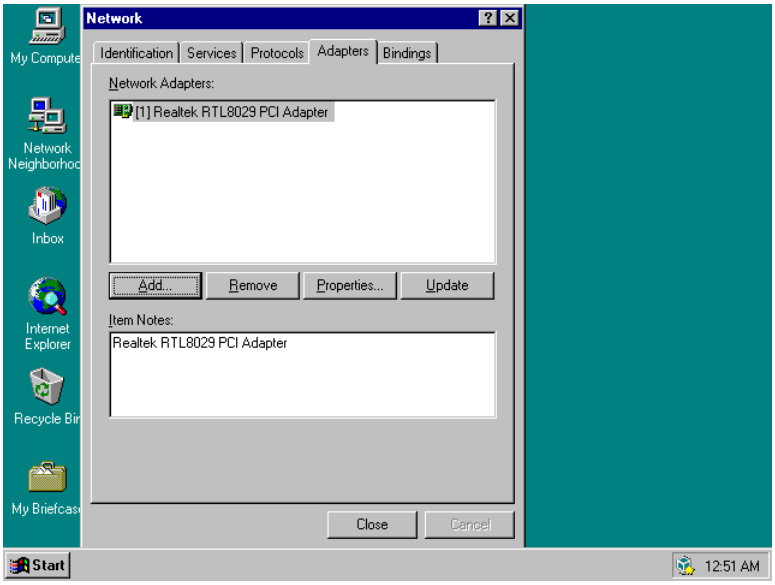


Type in the path to the Windows NT CD-ROM or the location of "I386" folder, which can be copied from your NT CD-ROM, and then click "Continue"

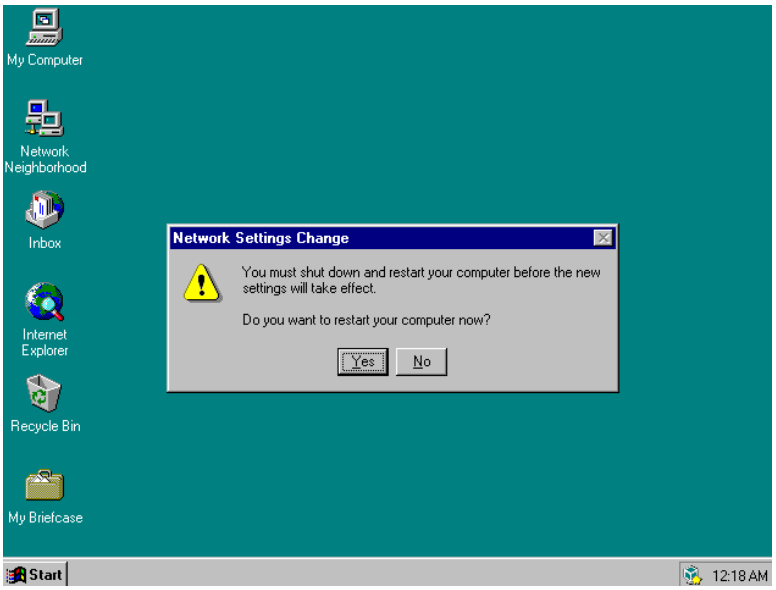


Check to see if the new Adapter is installed and

Then click the "close" button



When you close the Network Properties window, Windows NT prompts restart your computer. Click "Yes" to restart now to make the new setting effective.

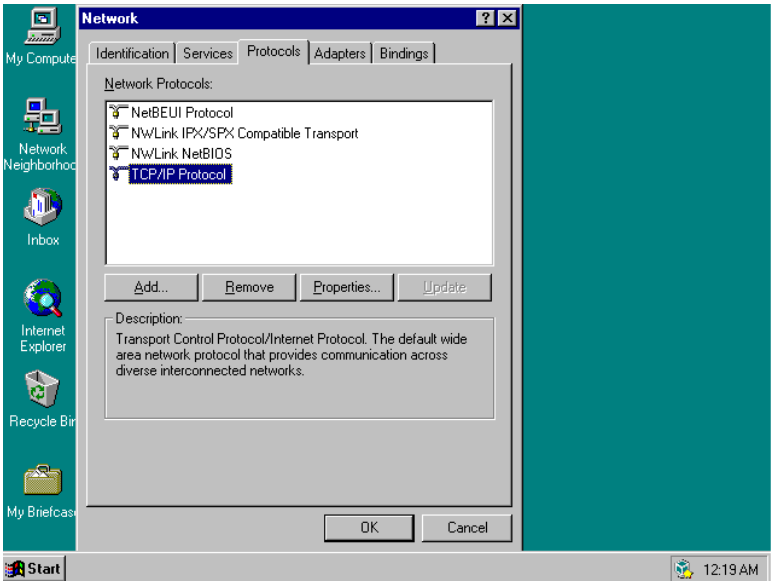


Add a New Network Protocol

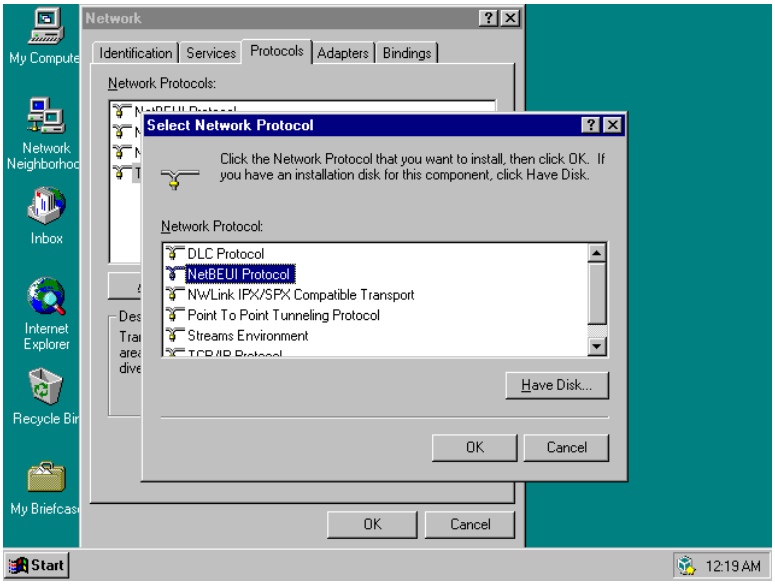
Network protocols are sets of rules that organize the way two devices communicate in the network. For two computers to communicate successfully they must use the same network protocol loaded.

Right click the "Network Neighborhood" icon in your desktop and choose “Properties" from the menu. Another way to access it is by opening the Network Control Panel.

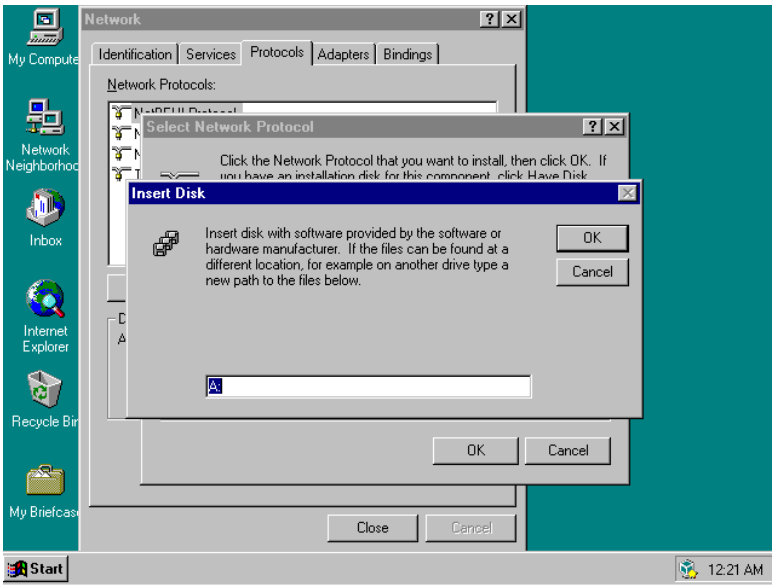
Click once the "Protocol" tab. Click the "Add" button to add a new network protocol.



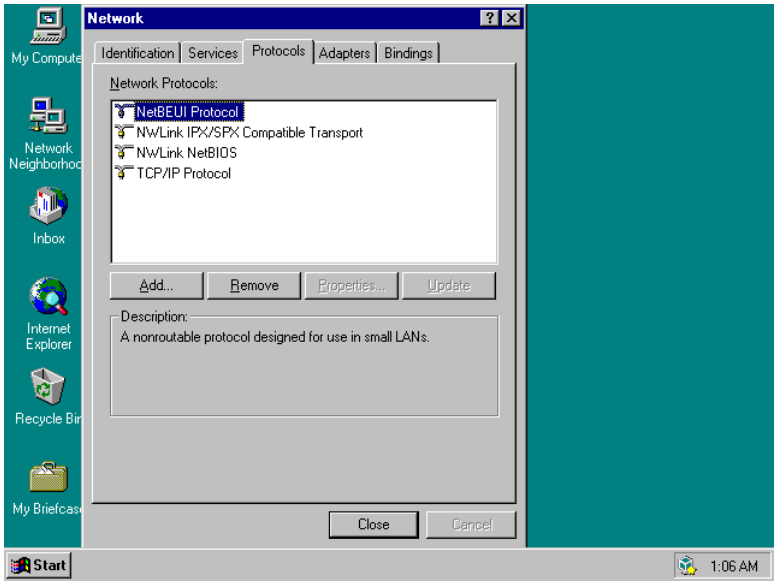
Choose the protocol you want to use from the list of the network protocols available and then click OK.



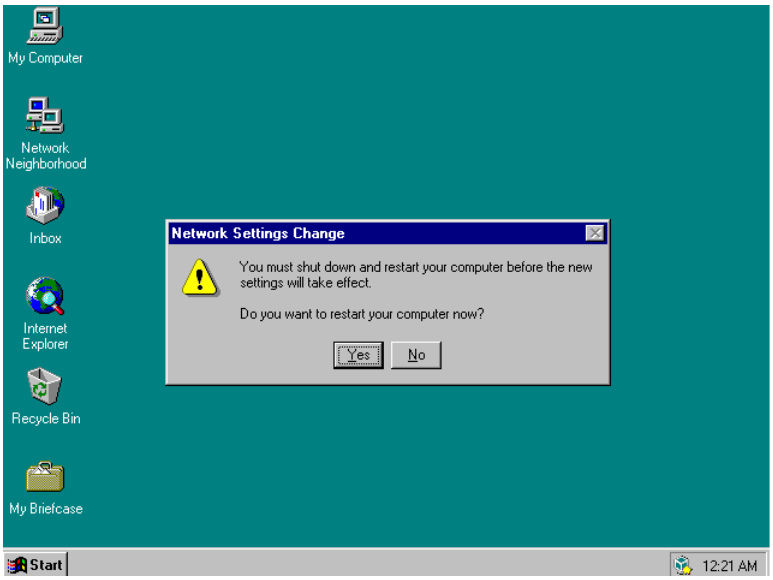
Type in the path to the Windows NT CD-ROM or the location of "1386" folder. This folder can be copied from your NT CD-ROM. Click on "Continue".



Check to see that the new protocol is displayed in the list of protocols and then click the "Close" button to close the window



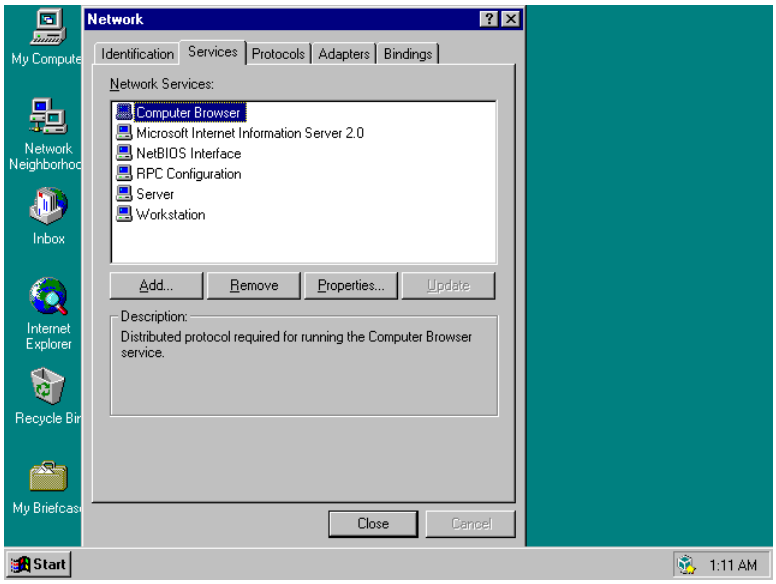
When you close the Network Properties window, Windows NT prompts you to restart your computer. Click "Yes" to restart to make the new settings effective



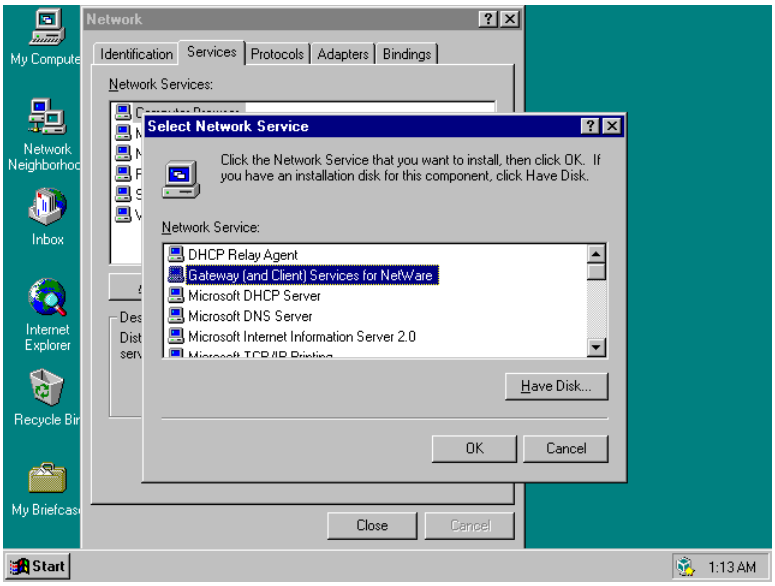
Adding a New Service

Right click the "Network Neighborhood" icon in your desktop and choose Properties from the menu.

Click on the "Services" and Click on the "Add" button to add a new service.

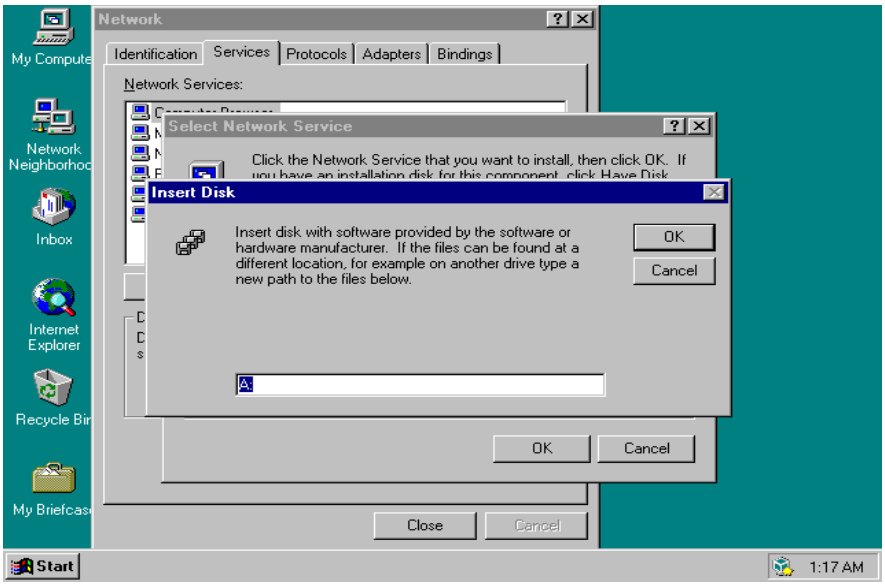


Choose the Service you want to use from the list and then click ok.



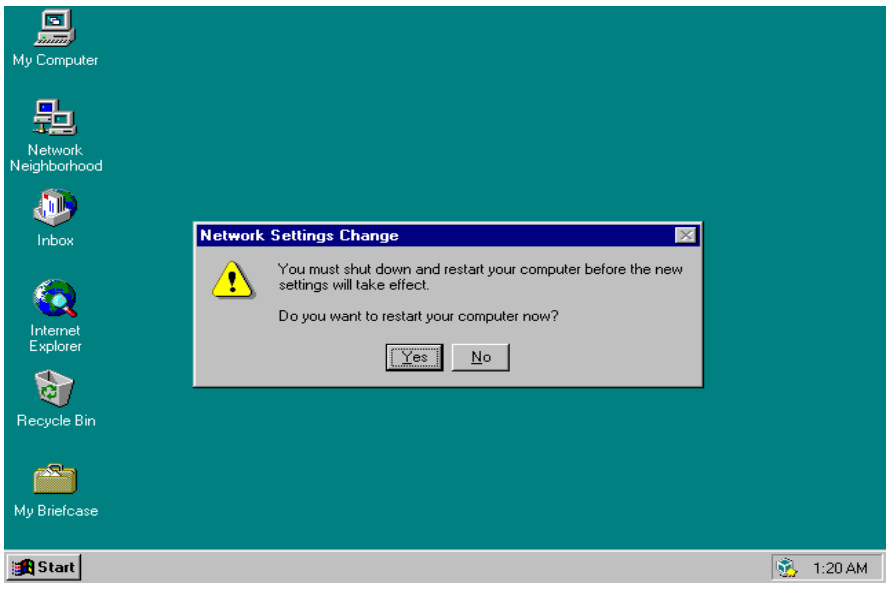
Type in the path to the Windows NT CD-ROM or the location of “I386” folder.

This folder can be copied from your NT CD-ROM. Click on “Continue”.



Check to see if the new Service is in the list of Services and then click the “ok” button to close the window.

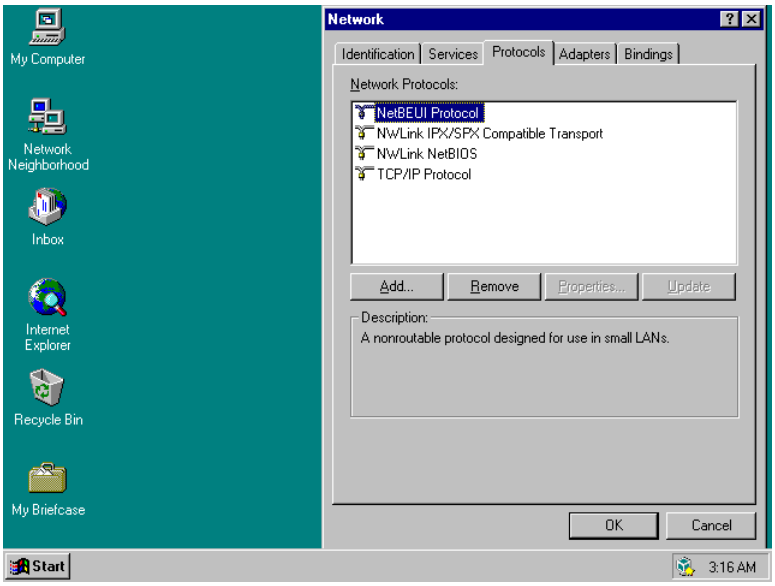
When you close the Network Properties window, Windows NT prompts you to restart your computer. Click "Yes" to restart now to make the new settings effective.



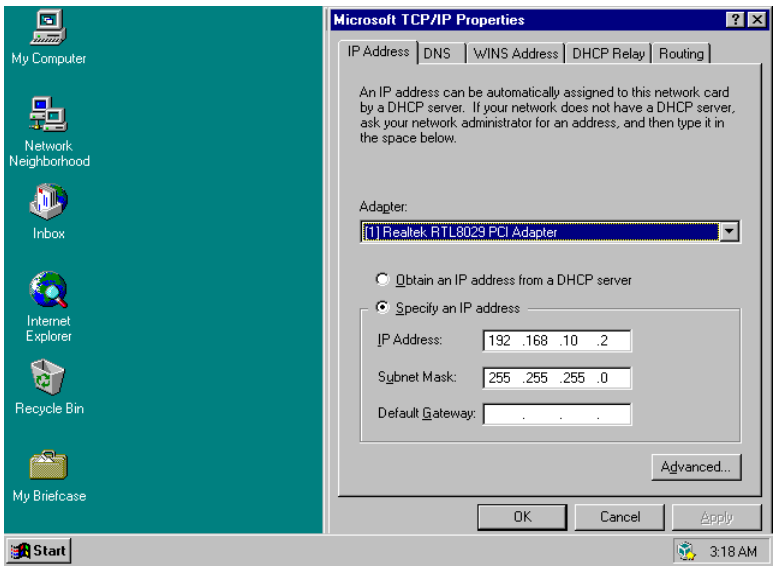
Configuring TCP/IP

Right click the "Network Neighborhood" icon in your desktop and choose Properties from the menu.

Click the Protocols tab. Select TCP/IP. If it’s not there, install TCP/IP as described under "Add New Network Protocol". Click the "Properties" button.

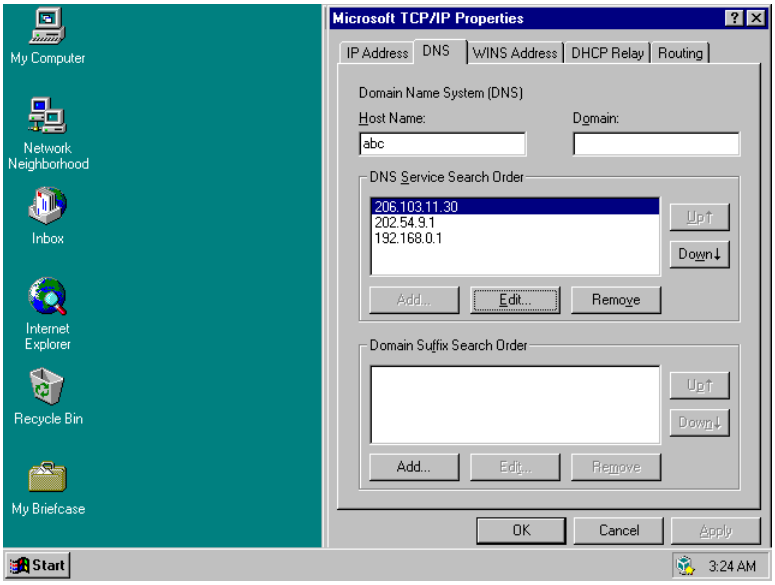


If your network has a DHCP server, then select the radio button next to "Obtain an IP address from DHCP server." If not, contact your network administrator or ISP to obtain an IP address and related information. Once you have this information, you can select the radio button next to "Specify an IP address" and enter your IP address. Subnet Mask and Default Gateway in the specified fields.



Click the “DNS” tab

In the DNS tab choose a Host name for your computer a domain, and enter a DNS server IP address in the specified field.



Concept of Domain

Domains are the central unit of management for a Windows NT network. A domain is a collection of computers and user accounts. A domain can contain all the computers and users for an entire organization or it can be used to provide a division of management, users and resources within an organization. For example, a company can create a domain for its Northern division and a domain for its Eastern division.

Each domain has its own administrator account. While each domain may have its own administrator, a single person can be the top-level administrator for all domains. Each domain also has its own set of user accounts. Users in one domain, however, can be allowed to access resources in another domain if a trust relationship exists between those domains.

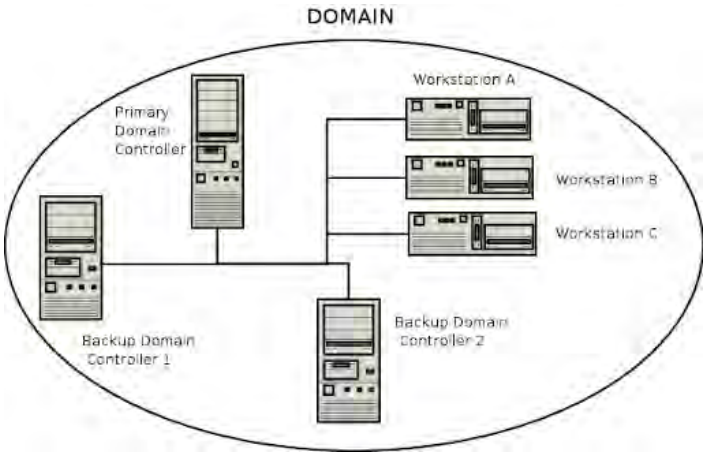
When setting up a Windows NT Server, the installer is given a choice to create a new domain or joining an existing domain. While setting the first Windows NT Server, former option is chosen and name of the domain server is specified the server then becomes a primary domain controller (PDC) for the domain created. A PDC holds the master directory database for a domain. Other Windows NT Servers can be installed in the domain and designated as backup domain controllers (BDCs). Any changes that administrators make to accounts are recorded in the PDC and then replicated to BDCs. BDCs can be located at remote sites to provide localized account information. They can also be used as PDC if the existing PDC goes down.

Concept of Server Manager

Server Manager is the tool used to manage the domain servers and workstations. It's a useful tool for checking the services running in all domain computers from one location, spending messages to connected users and monitoring connections to your servers.

From the server manager you can view all the users connected to your domain, the duration for which they are connected, their idle time and from which computer they are logged on. You can also view which network resources they are connected to (which shares).

The Domain Architecture



Convert a Windows NT Server from a workgroup to a domain controller

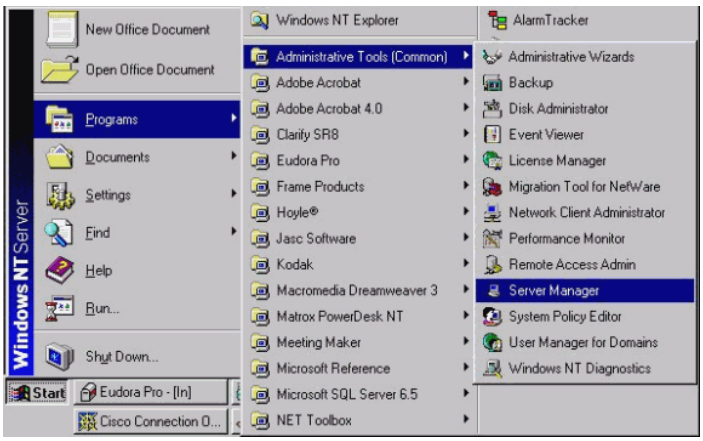
To convert a **Windows NT Server** from a **workgroup** to a **domain controller**, you need to promote the server and configure it to act as a **Primary Domain Controller (PDC)**. Here's how to do it:

Steps to Upgrade a NT Server To a Domain Controller

Open the Server Manager

Log in to the server with the **Administrator** account.

Open the **Server Manager** from the **Administrative Tools** menu.



Promote the Server to a Primary Domain Controller (PDC)

- In the **Server Manager**, select **Computer** and then **Create a Domain**.
- Choose **Primary Domain Controller (PDC)** during the configuration.
- Enter the name of the domain you want to create.

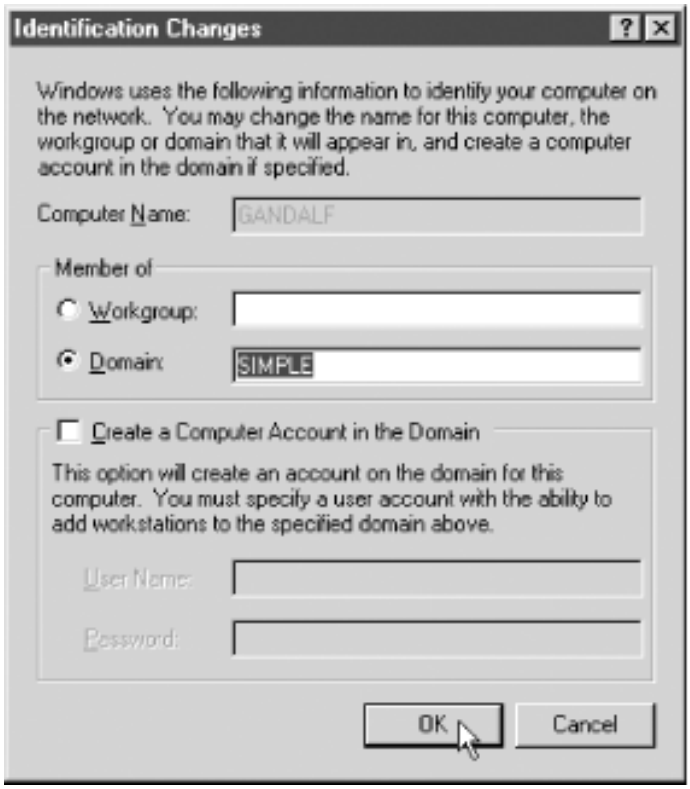
For example:

- Example domain: SIMPLE.
- Confirm the creation and click **OK**.
- Adding computer to a Domain

On a client computer, attempt to join the domain:

- Go to the **Network Neighborhood Properties**.
- Enter the domain name and provide the administrator credentials.

Reboot the client and ensure it logs into the domain.



Additional Notes

- **Backup Data:** Before converting the server to a domain, ensure all critical data is backed up.
- **Trust Relationships:** If you're connecting multiple domains, configure trust relationships in the **Domain**

Configuration utility.

- **Secondary Domain Controller (BDC):** For redundancy, you can configure another server as a Backup Domain Controller.

User and Group Management in Windows NT Server

- Explain the concept of users in Windows NT
- Create and maintain different users and groups in a Windows NT Domain
- Explain the concept of local and global group

Introduction

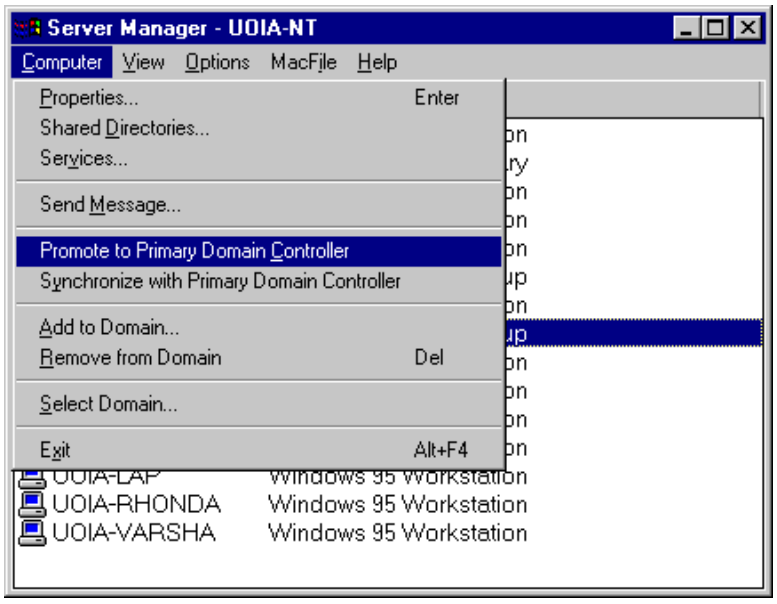
For a better understanding of different security features of NT, we must have a very clear idea about the user and group management in NT environment. The following section deals with user and group management in Windows NT

User and Groups in the Domain

In Windows NT Server 4.0, there are two types of accounts that administrators have to manage:

- User
- Groups

A group is a collection of user accounts. A user must be a part of a global group to be a part of a domain. When Windows NT Server is installed, there are a couple of default user accounts created by default. These accounts are the administrator account and the guest account. By default the guest account is disabled, so that no one can access the network without being assigned a username and password. It is up to the administrator to enable this account



There are two types of groups:

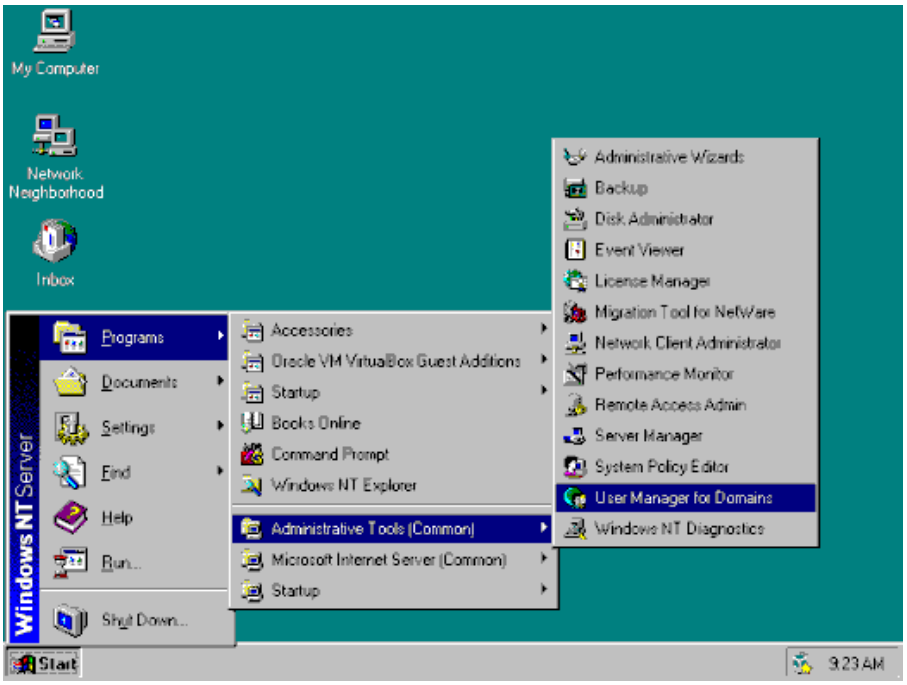
- Global
- Local

Global groups exist only on domain controllers. Standalone servers do not contain any global group. Users who log on to the computer should be a part of a local group. A local group is local to the machine. Global groups are specific to a domain. More about groups will come soon.

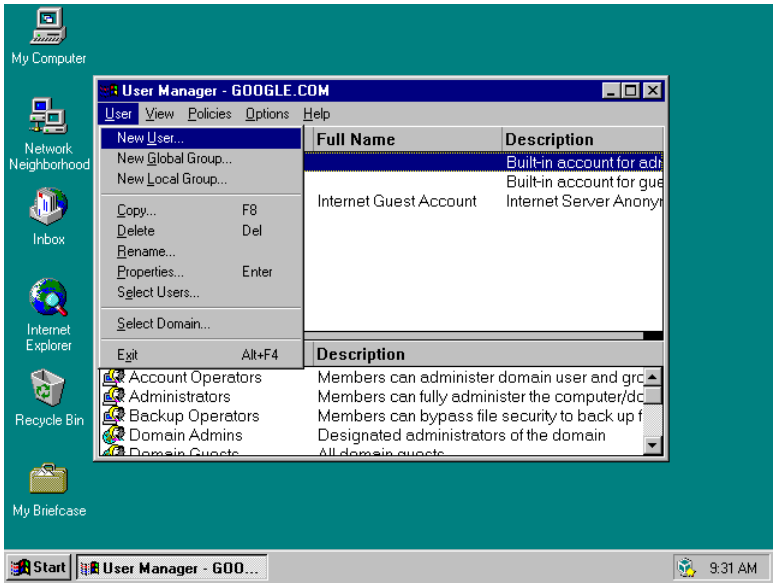
Creating a new user in an NT Domain Controller

Creating a new user in an NT Domain Controller (Windows NT or Active Directory Domain) involves using the **Active Directory Users and Computers (ADUC)** tool or a command-line utility. Here's a step-by-step guide:

- Using Active Directory Users and Computers (GUI Method)
- Log in to the Domain Controller:
- Use an account with permissions to create users (e.g., Domain Admin or equivalent)
- Select Start -» Programs » Administrative Tools (Common) > User Manager for Domains.



Select user Menu-»New user. The new User dialog box appears as shown below

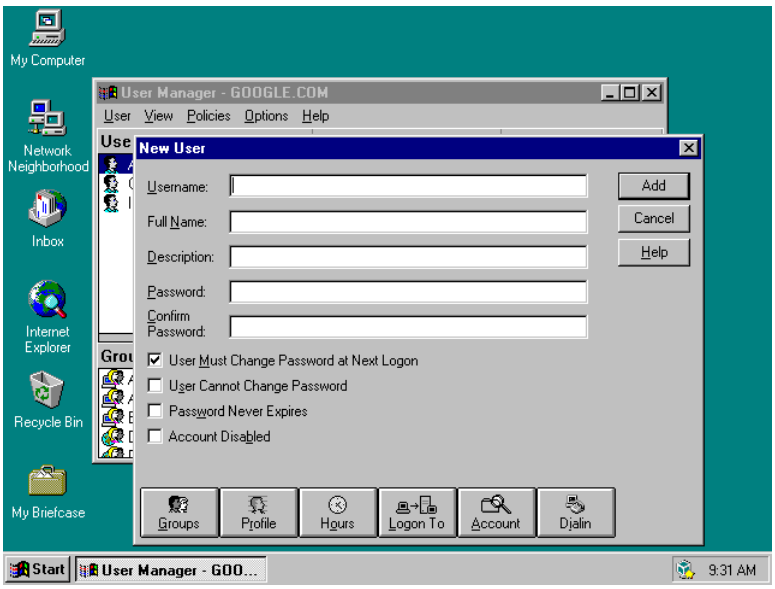


Fill in the User Information:

Provide the following:

- ✓ User Name
- ✓ Full Name

Uncheck the user must change password at next logon checkbox.



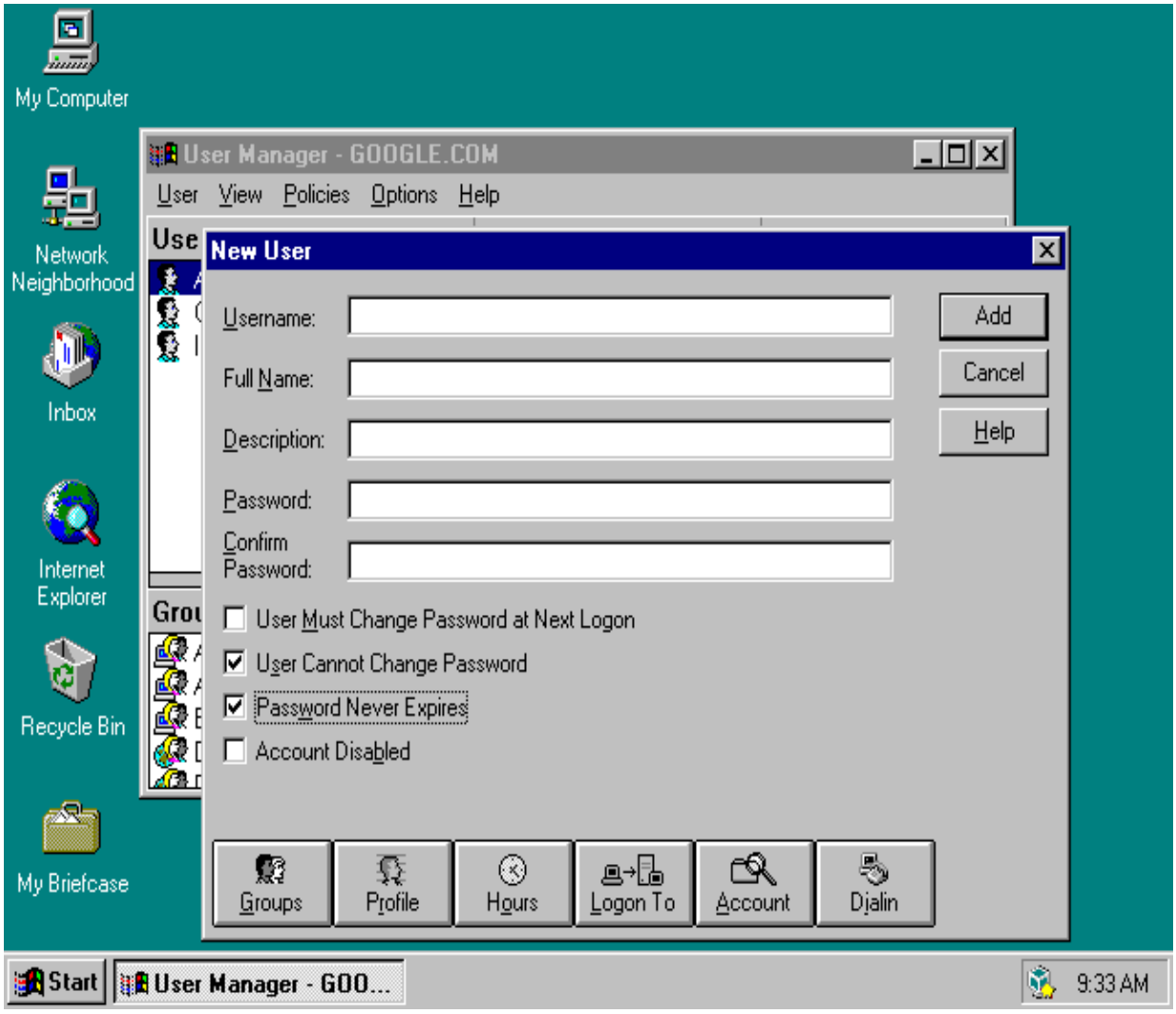
Set a Password:

Enter and confirm the password.

Configure options like:

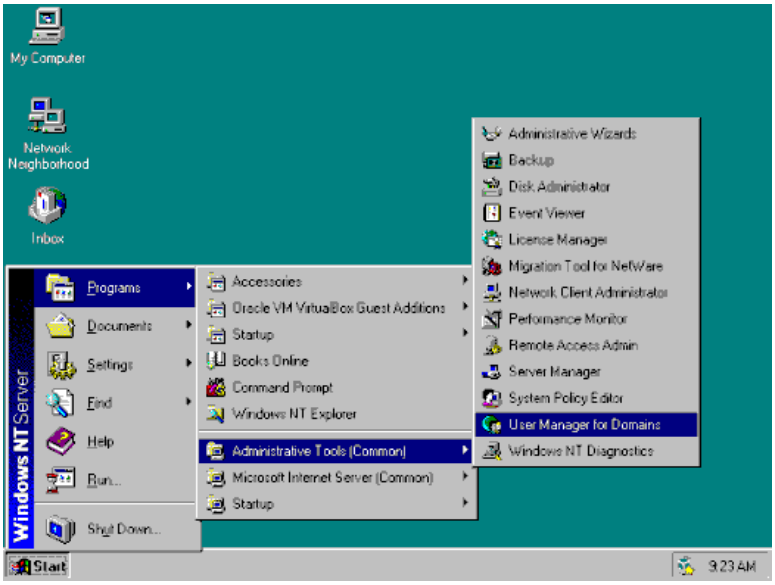
- ✓ User must change password at next logon.
- ✓ Password never expires.

Click Add

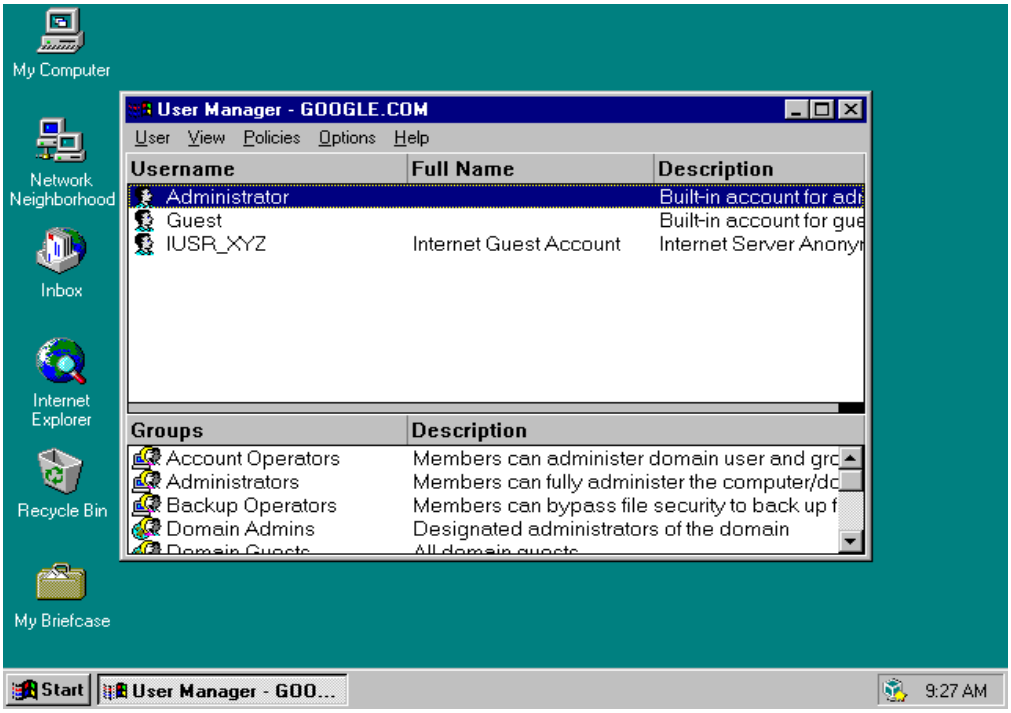


Disabling a User Account

Select Start -» Programs » Administrative Tools (Common) > User Manager for Domains.

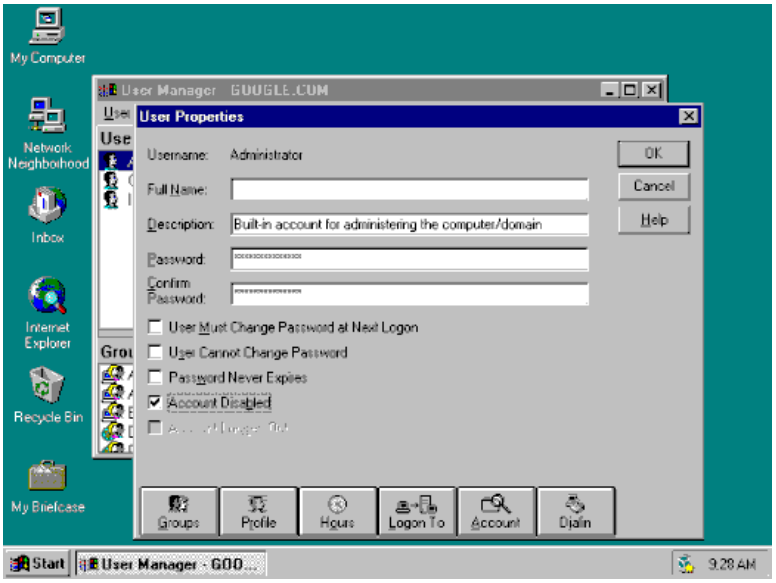


Double-click User Name (e.g. YOUTH)



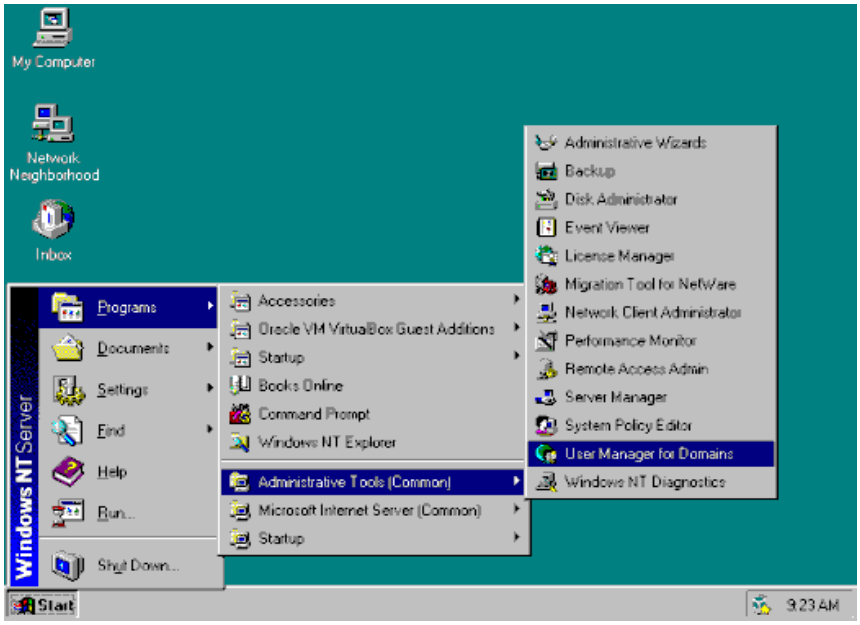
In the User dialog box check the Account Disabled option.

This will disable the particular user account

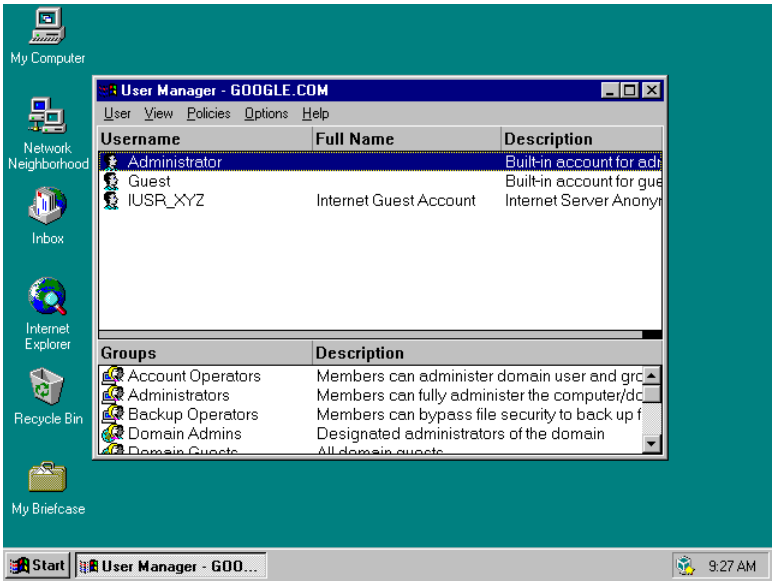


Deleting a User Account

Select Start > Programs » Administrative Tools (Common) » User Manager for Domains.



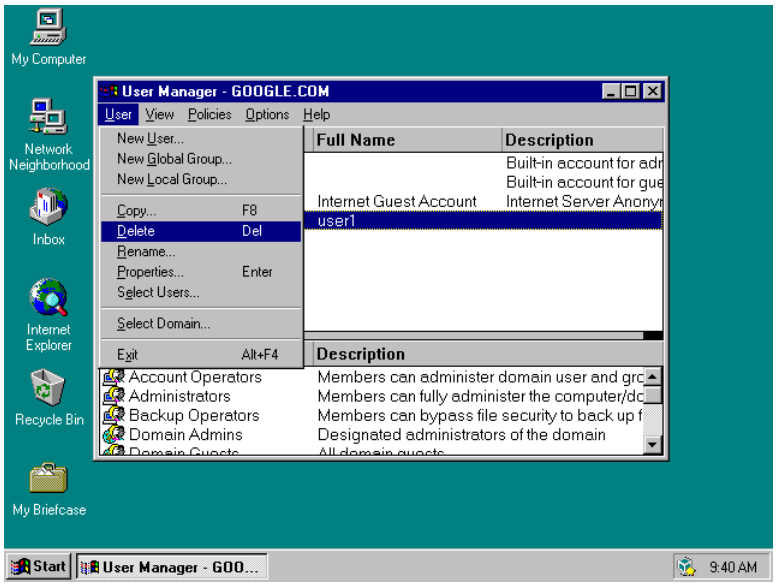
Click User Name (e.g. YOUTH)



Click the User Menu » Delete

You will get message saying once you delete this user account, it will be erased forever, click OK

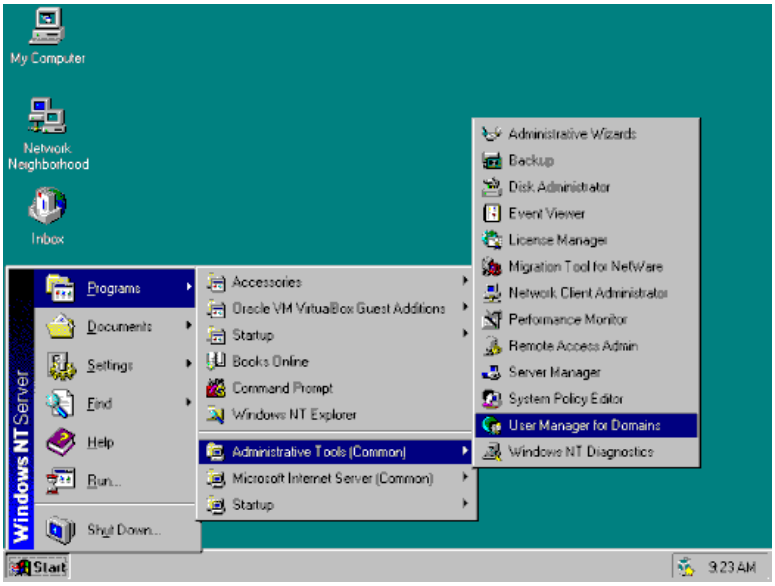
One last chance to change your mind! Click yes to delete the particular user account.



Renaming a User Account

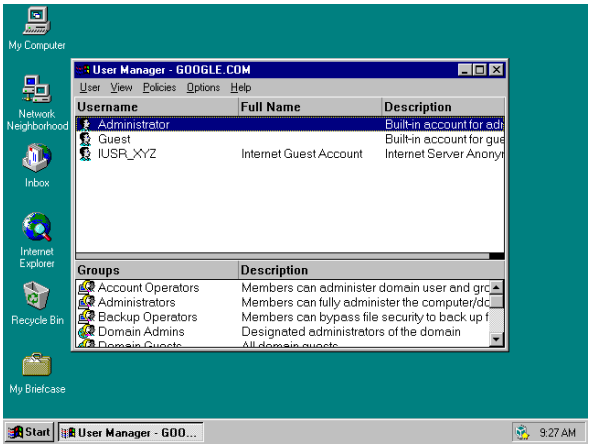
Select Start > Programs » Administrative Tools (Common) > User Manager for Domains.

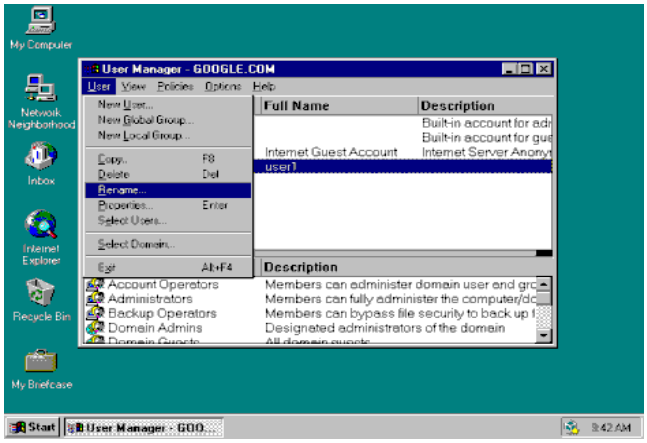
Click User (e.g. YOUTH)



Click the User Menu » Rename.

The dialog box will prompt you with Change to: Type a new user name and Click OK.





Check your Understanding:

Question 1. What does NT in Windows NT stand for?

a) New Technology
b) Network Technology
c) Next Technology
d) Net Task

Question 2. Windows NT is designed primarily for which type of environment?

a) Gaming
b) Networked environments
c) Personal home use
d) Embedded systems

Question 3. What file system is commonly associated with Windows NT?

a) FAT16
b) NTFS
c) HFS
d) EXT4

Question 4. Which file system is used in the Windows 98?

a) FAT
b) HPFS
c) ExFAT
d) NTFS

Question 5. Which of the following is not a Windows NT Server edition?

a) Windows NT Server 3.51
b) Windows NT Server 4.0
c) Windows NT Server 5.1
d) Windows NT Server 2000

Question 6. Which process is responsible for managing logins in Windows NT?

a) Winlogon
b) Explorer
c) Services.exe
d) Task Manager

Question 7. Which command in Windows NT is used to check the IP configuration?

a) Ipconfig
b) Ifconfig
c) Netconfig
d) Networkcfg

Question 8. What is the primary protocol used for communication in Windows NT networks?

a) HTTP
b) IPX/SPX
c) TCP/IP
d) NetBEUI

Question 9. In Windows NT, the "Task Manager" is primarily used to:

a) Manage network settings
b) Monitor system performance
c) Edit the registry
d) Manage disk partitions

Question 10. Which is a characteristic of NTFS over FAT file systems?

a) Larger file size support
b) Journaling for recovery
c) File permissions
d) All of the above

Chapter 2: Oracle

About Oracle

Oracle database is a relational database management system (RDBMS) from Oracle Corporation. This article will explain a complete overview of the Oracle database, features, history, and editions. Before discussing the oracle, we will first need to know about the database.

Concept of Database

A database refers to the organized collection of structured data stored electronically in a device. It allows us to access, manage, and find relevant information frequently. The flat file structure was extensively used to store data before the database system was invented. The relational database approach becomes popular in comparison to the flat file model because it eliminates redundant data. For example, suppose we have an employee and contact information stored in the same file. In such a case, the employees with multiple contacts will show up in many rows.

The RDBMS system manages the relational data. Oracle Database is the most famous relational database system (RDBMS) because it shares the largest part of a market among other relational databases. Some other popular relational databases are MySQL, DB2, SQL Server, PostgreSQL, etc.

Editions of Oracle database

Oracle database is compatible with a wide range of platforms such as Windows, UNIX, Linux, and macOS. It supports several operating systems like IBM AIX, HP-UX, Linux, Microsoft Windows Server, Solaris, SunOS, macOS, etc. In the late **1990s**, Oracle began supporting open platforms like GNU/Linux.

A list of Oracle database editions in order of priority:

Enterprise Edition: It is the most robust and secure edition. It offers all features, including superior performance and security.

Standard Edition: It provides the base functionality for users that do not require Enterprise Edition's robust package.

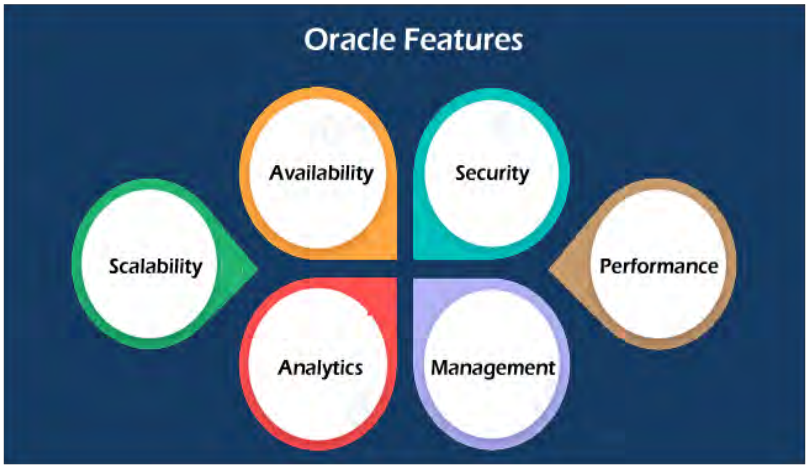
Express Edition (XE): It is the lightweight, free and limited Windows, and Linux edition.

Oracle Lite: It is designed for mobile devices.

Personal Edition: It's comparable to the Enterprise Edition but without the Oracle Real Application Clusters feature.

Oracle Database Features

Oracle database manages data with the help of an open, complete, and integrated approach. The following are features that complete the demand for powerful database management:



Availability: It is never offline or out of service that means supported 24*7 availability of the database. It provides high availability of databases because of the Oracle Data Guard functionality. This functionality allows using of the secondary database as a copy of the primary database during any failure. As a result, all normal processes such as backups and partial failures do not interrupt the database from being used.

Security: Oracle has a mechanism for controlling and accessing the database to prevent unauthorized access. It provides high security because of the Oracle Advanced Security features. It offers two solutions to protect databases that are TDE (Transparent Data Encryption) and Data Redaction. TDE supports data encryption both at the source and after export. Redaction is performed at the application level. Oracle has some other security features like Oracle Database Vault that regulates user privileges and Oracle Label Security.

Scalability: It provides features like RAC (Real Application Cluster) and Portability, which makes an Oracle database scalable based on usage. In a clustered environment, it includes capabilities such as rolling instance migrations, performing upgrades, maintaining application continuity, quality of service management, etc.

Performance: Oracle provides performance optimization tools such as Oracle Advanced Compression, Oracle Database In-Memory, Oracle Real Application Testing, and Oracle Times Ten Application-Tier Database Cache. Their main objective is to improve system performance to the highest possible level.

Analytics: Oracle has the following solutions in the field of analytics:

- OLAP (Oracle Analytic Processing): It is an implementation of Oracle for doing complicated analytical calculations on business data.
- Oracle Advanced Analytics: It is a technical combination of Oracle R Enterprise and Oracle Data Mining that assists customers in determining predictive business models through data and text mining, as well as statistical data computation.

Management: Oracle Multitenant is a database management tool that combines a single container database with many pluggable databases in a consolidated design.

Advantages of Oracle Database

Performance: Oracle has procedures and principles that help us to get high levels of database performance. We can increase query execution time and operations with the use of performance optimization techniques in its database. This technique helps to retrieve and alter data faster.

Portability: The Oracle database can be ported on all different platforms than any of its competitors. We can use this database on around 20 networking protocols as well as over 100 hardware platforms. This database makes it simple to write an Oracle application by making changes to the OS and hardware in a secure manner.

Backup and Recovery: It is always better to take a proper backup of your entire oracle online backup and recovery. The Oracle database makes it easy to accomplish recovery quickly by using the. RMAN (Recovery Manager) functionality. It can recover or restore database files during downtime or outages. It can be used for online backups, archived backups, and continuous archiving. We can also use SQL* PLUS for recovery, which is known as user-managed recovery.

PL/SQL: One of the greatest benefits of using the Oracle database is to support PL/SQL extension for procedural programming.

Multiple Database: Oracle database allows several database instances management on a single server. It provides an instance caging approach for managing CPU allocations on a server hosting database instances. The database resource management and instance caging can work together to manage services across multiple instances.

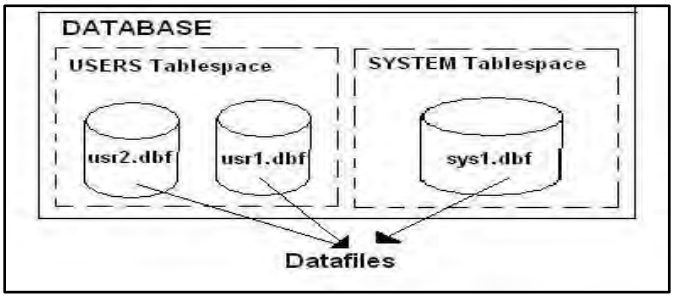
Flashback Technology: This advantage comes with the recent Oracle version. It allows us to recover those data that are incorrectly deleted or lost by human errors like accidental deletion of valuable data, deleting the wrong data, or dropping the table.

Disadvantages of Oracle Database

Complexity: Oracle is not recommended to use when the users are not technically savvy and have limited technical skills required to deal with the Oracle Database. It is also not advised to use if the company is looking for a database with limited functionality and easy to use.

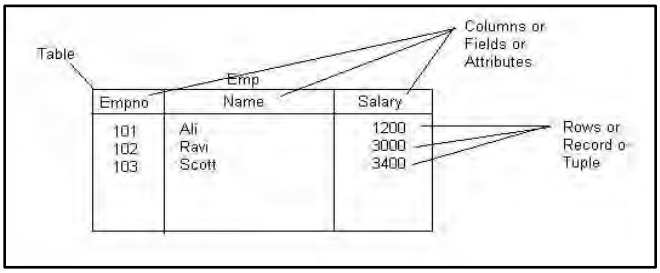
Cost: The price of Oracle products is very high in comparison to other databases. Therefore users are more likely to choose other less expensive options such as MS SQL Server, MySQL, etc.

Difficult to manage: Oracle databases are often much more complex and difficult in terms of the management of certain activities.



Oracle Database

Every Oracle Database Contains Logical and Physical Structures. Logical Structures are tablespaces, Schema objects, extents and segments. Physical Structures are Datafiles, Redo Log Files, Control File.



A database is divided into logical storage units called tablespaces, which group related logical structures together. Each Tablespace in turn consists of one are more datafiles.

In relational database system all the information is stored in form of tables. A table consists of rows and columns

All the tables and other objects in Oracle are stored in tablespace logically, but physically they are stored in the datafiles associated with the tablespace.

Every Oracle database has a set of two or more redo log files. The set of redo log files for a database is collectively known as the database's redo log. A redo log is made up of redo entries (also called redo records). The primary function of the redo log is to record all changes made to data. If a failure prevents modified data from being permanently written to the data files, the changes can be obtained from the redo log so work is never lost.

Every Oracle database has a control file. A control file contains the database name and locations of all data files and redo log files.

Every Oracle database also has a Parameter File. Parameter file contains the name of the Database, Memory Settings and Location of Control file.Data types and Creating Tables.

A table is the data structure that holds data in a relational database. A table is composed of rows and columns.

A table in Oracle Ver. 7.3 can have maximum 255 Columns and in Oracle Ver. 8 and above a table can have maximum

1000 columns. Number of rows in a table is unlimited in all the versions.

A table can represent a single entity that you want to track within your system. This type of a table could represent a list of the employees within your organization, or the orders placed for your company's products.

A table can also represent a relationship between two entities. This type of a table could portray the association between employees and their job skills, or the relationship of products to orders. Within the tables, foreign keys are used to represent relationships.

Although some well-designed tables could represent both an entity and describe the relationship between that entity and another entity, most tables should represent either an entity or a relationship.

Designing Tables

Guidelines to designing your tables**:**

- Use descriptive names for tables, columns, indexes, and clusters.
- Table Names, Columns Names can contain maximum of 30 characters and they should start with an alphabet.
- Be consistent in abbreviations and in the use of singular and plural forms of table names and columns.
- Select the appropriate data type for each column.
- Arrange columns that can contain NULL Values in the last, to save storage space.

Before creating a table, you should also determine whether to use integrity constraints. Integrity constraints can be defined on the columns of a table to enforce the business rules of your database automatically.

Before creating a Table you also have to decide what type of data each column can contain. This is known as data type. Let's discuss what data types are available in Oracle.

Data types

A data type associates a fixed set of properties with the values that can be used in a column of a table or in an argument of a procedure or function. These properties cause Oracle to treat values of one data type differently from values of another data type. For example, Oracle can add values of NUMBER data type, but not values of RAW data type.

Oracle supplies the following built-in data types:

Character data types

- CHAR
- NCHAR
- VARCHAR2 and VARCHAR
- NVARCHAR2
- CLOB
- NCLOB
- LONG

Numeric data types

- NUMBER

Time and date data types

- DATE
- INTERVAL DAY TO SECOND
- INTERVAL YEAR TO MONTH
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE

Binary data types

- BLOB
- BFILE
- RAW
- LONG RAW

Another data type, ROWID, is used for values in the ROWID pseudocolumn, which represents the unique address of each row in a table.

The following table summarizes the information about each Oracle built-in datatype.

Datatype	Description	Column Length and Default
CHAR (size [BYTE | CHAR])	Fixed-length character data of length size bytes or characters.	Fixed for every row in the table (with trailing blanks); maximum size is 2000 bytes per row, default size is 1 byte per row. Consider the character set (single-byte or multibyte) before setting size.
VARCHAR2 (size [BYTE | CHAR])	Variable-length character data, with maximum length size bytes or characters.	Variable for each row, up to 4000 bytes per row. Consider the character set (single-byte or multibyte) before setting size. A maximum size must be specified.
NCHAR (size)	Fixed-length Unicode character data of length size characters.	Fixed for every row in the table (with trailing blanks). Column size is the number of characters. (The number of bytes is 2 times this number for the AL16UTF16 encoding and 3 times this number for the UTF8 encoding.) The upper limit is 2000 bytes per row. Default is 1 character.
NVARCHAR2 (size)	Variable-length Unicode character data of length size characters. A maximum size must be specified.	Variable for each row. Column size is the number of characters. (The number of bytes may be up to 2 times this number for a the AL16UTF16 encoding and 3 times this number for the UTF8 encoding.) The upper limit is 4000 bytes per row. Default is 1 character.
CLOB	Single-byte character data	Up to 232 - 1 bytes, or 4 gigabytes.
NCLOB	Unicode national character set (NCHAR) data.	Up to 232 - 1 bytes, or 4 gigabytes.
LONG	Variable-length character data.	Variable for each row in the table, up to 232 - 1 bytes, or 2 gigabytes, per row. Provided for backward compatibility.
NUMBER (p, s)	Variable-length numeric data. Maximum precision p and/or scale s is 38.	Variable for each row. The maximum space required for a given column is 21 bytes per row.

DATE	Fixed-length date and time data, ranging from Jan. 1, 4712 B.C.E. to Dec. 31, 4712 C.E.	Fixed at 7 bytes for each row in the table. Default format is a string (such as DD-MON-RR) specified by the NLS_DATE_FORMAT parameter.
INTERVAL YEAR (precision) TO MONTH	A period of time, represented as years and months. The precision value specifies the number of digits in the YEAR field of the date. The precision can be from 0 to 9, and defaults to 2 for years.	Fixed at 5 bytes.
INTERVAL DAY (precision) TO SECOND (precision)	A period of time, represented as days, hours, minutes, and seconds. The precision values specify the number of digits in the DAY and the fractional SECOND fields of the date. The precision can be from 0 to 9, and defaults to 2 for days and 6 for seconds.	Fixed at 11 bytes.
TIMESTAMP (precision)	A value representing a date and time, including fractional seconds. (The exact resolution depends on the operating system clock.) The precision value specifies the number of digits in the fractional second part of the SECOND date field. The precision can be from 0 to 9, and defaults to 6	Varies from 7 to 11 bytes, depending on the precision. The default is determined by the NLS_TIMESTAMP_FORMAT initialization parameter.
TIMESTAMP (precision) WITH TIME ZONE	A value representing a date and time, plus an associated time zone setting. The time zone can be an offset from UTC, such as '-5:0', or a region name, such as 'US/Pacific'.	Fixed at 13 bytes. The default is determined by the NLS_TIMESTAMP_TZ_FORMAT initialization parameter.
TIMESTAMP (precision) WITH LOCAL TIME	Similar to TIMESTAMP WITH TIME ZONE, except that the data is normalized to the database time	Varies from 7 to 11 bytes, depending on the precision. The default is determined by the

ZONE	zone when stored, and adjusted to match the client's time zone when retrieved.	NLS_TIMESTAMP_FORMAT initialization parameter.
BLOB	Unstructured binary data	Up to 232 - 1 bytes, or 4 gigabytes.
BFILE	Binary data stored in an external file	Up to 232 - 1 bytes, or 4 gigabytes.
RAW (size)	Variable-length raw binary data	Variable for each row in the table, up to 2000 bytes per row. A maximum size must be specified. Provided for backward compatibility.
LONG RAW	Variable-length raw binary data	Variable for each row in the table, up to 231 - 1 bytes, or 2 gigabytes, per row. Provided for backward compatibility.
ROWID	Binary data representing row addresses	Fixed at 10 bytes (extended ROWID) or 6 bytes (restricted ROWID) for each row in the table.

- Oracle CREATE TABLE

In Oracle, CREATE TABLE statement is used to create a new table in the database.

To create a table, you have to name that table and define its columns and datatype for each column.

Syntax**:**

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

Parameters used in syntax

- Table name: It specifies the name of the table which you want to create.
- column1, column2, column n: It specifies the columns which you want to add in the table. Every column must have a datatype. Every column should either be defined as "NULL" or "NOT NULL". In the case, the value is left blank; it is treated as "NULL" as default.

Oracle CREATE TABLE Example

Here we are creating a table named customers. This table doesn't have any primary key.

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50)
);
```

This table contains three columns

Customer_id: It is the first column created as a number datatype (maximum 10 digits in length) and cannot contain null values.

Customer name: it is the second column created as a varchar2 datatype (50 maximum characters in length) and cannot contain null values.

City: This is the third column created as a varchar2 datatype. It can contain null values.

Oracle CREATE TABLE Example with primary key

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);
```

Primary key

A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

CREATE TABLE AS Statement

The CREATE TABLE AS statement is used to create a table from an existing table by copying the columns of existing table.

Syntax:

CREATE TABLE new_table

AS (SELECT * FROM old_table);

Create Table Example: copying all columns of another table

In this example, we are creating a "newcustomers" table by copying all the columns from the already existing table "Customers".

CREATE TABLE new customers

AS (SELECT * FROM customers WHERE customer_id < 5000);

Table created.

This table is named as "new customers" and having the same columns of "customers" table.

Create Table Example: copying selected columns of another table

Syntax:

CREATE TABLE new_table

AS (SELECT column_1, column2, ... column_n

FROM old_table);

Let's take an example:

CREATE TABLE newcustomers2

AS (SELECT customer_id, customer_name

```
FROM customers
WHERE customer_id < 5000);
```

The above example will create a new table called "newcustomers2". This table includes the specified columns customer_id and customer_name from the customers table.

Create Table Example: copying selected columns from multiple tables

Syntax:

```
CREATE TABLE new_table
AS (SELECT column_1, column2, ... column_n
   FROM old_table_1, old_table_2, ... old_table_n);
```

Let's take an example: Consider that you have already created two tables "regularcustomers" and "irregular customers".

The table "regularcustomers" has three columns rcustomer_id, rcustomer_name and rc_city.

```
CREATE TABLE  "regularcustomers"
  (    "RCUSTOMER_ID" NUMBER(10,0) NOT NULL ENABLE,
   "RCUSTOMER_NAME" VARCHAR2(50) NOT NULL ENABLE,
   "RC_CITY" VARCHAR2(50)
  )
/
```

The second table "irregular customers" has also three columns ircustomer_id, ircustomer_name and irc_city.

```
CREATE TABLE  "irregularcustomers"
  (    "IRCUSTOMER_ID" NUMBER(10,0) NOT NULL ENABLE,
   "IRCUSTOMER_NAME" VARCHAR2(50) NOT NULL ENABLE,
   "IRC_CITY" VARCHAR2(50)              );
```

In the following example, we will create a table name "newcustomers3" form copying columns from both tables.

Example:

```
CREATE TABLE newcustomers3
  AS (SELECT regularcustomers.rcustomer_id, regularcustomers.rc_city, irregularcustomers.ircustomer_name
     FROM regularcustomers, irregularcustomers
     WHERE regularcustomers.rcustomer_id = irregularcustomers.ircustomer_id
     AND regularcustomers.rcustomer_id < 5000);
```

Oracle ALTER TABLE Statement

In Oracle, ALTER TABLE statement specifies how to add, modify, drop or delete columns in a table. It is also used to rename a table.

Add column in a table

Syntax:

```
ALTER TABLE table_name
  ADD column_name column-definition;
```

Example**:**

Consider that already existing table customers. Now, add a new column customer_age into the table customers.

```
ALTER TABLE customers
ADD customer_age varchar2(50);
```

Now, a new column "customer_age" will be added in customers table.

Add multiple columns in the existing table

Syntax:

```
ALTER TABLE table_name
  ADD (column_1 column-definition,
```

column_2 column-definition,

...

column_n column_definition);

Example

```
ALTER TABLE customers
  ADD (customer_type varchar2(50),
      customer_address varchar2(50));
```

Modify column of a table

Syntax:

```
ALTER TABLE table_name
  MODIFY column_name column_type;
```

Example:

```
ALTER TABLE customers
  MODIFY customer_name varchar2(100) not null;
```

Drop column of a table

Syntax:

```
ALTER TABLE table_name
  DROP COLUMN column_name;
```

Example:

```
ALTER TABLE customers
  DROP COLUMN customer_name;
```

This will drop the customer_name column from the table.

Rename column of a table

Syntax:

ALTER TABLE table_name

RENAME COLUMN old_name to new_name;

Example:

ALTER TABLE customers

RENAME COLUMN customer_name to cname;

This will rename the column customer_name into cname.

Rename of a table

Syntax:

ALTER TABLE table_name

RENAME TO new_table_name;

Example:

ALTER TABLE customers

RENAME TO retailers;

This will rename the customer table into "retailers" table.

Oracle DROP TABLE Statement

Oracle DROP TABLE statement is used to remove or delete a table from the Oracle database.

Syntax

DROP [schema_name].TABLE table_name

[CASCADE CONSTRAINTS]

[PURGE];

Parameters

Schema name: It specifies the name of the schema that owns the table.

Table name: It specifies the name of the table which you want to remove from the Oracle database.

CASCADE CONSTRAINTS: It is optional. If specified, it will drop all referential integrity constraints as well.

PURGE: It is also optional. If specified, the table and its dependent objects are placed in the recycle bin and can't be recovered.

If there are referential integrity constraints on table_name and you do not specify the CASCADE CONSTRAINTS option, the DROP TABLE statement will return an error and Oracle will not drop the table.

DROP TABLE Example

DROP TABLE customers;

This will drop the table named customers.

Oracle Queries

You can execute many queries in oracle database such as insert, update, delete, alter table, drop, create and select.

Oracle Select Query

Oracle select query is used to fetch records from database.

Example:

SELECT * from customers;

Oracle SELECT Statement

The Oracle SELECT statement is used to retrieve data from one or more than one tables, object tables, views, object views etc.

Syntax

SELECT expressions

FROM tables

WHERE conditions;

Parameters

Expressions: It specifies the columns or calculations that you want to retrieve.

Tables: This parameter specifies the tables that you want to retrieve records from. There must be at least one table within the FROM clause.

Conditions: It specifies the conditions that must be followed for selection.

Select Example: select all fields

Let's take an example to select all fields from an already created table named customers

```
SELECT *
FROM customers;
```

Output

NAME	AGE	ADDRESS	SALARY
mohan	21	ghaziabad	20000
rohan	22	delhi	22000
sohan	25	noida	24000
alex	28	Paris	40000

4 rows returned in 0.02 seconds

Select Example: select specific fields

Example

```
SELECT age, address, salary
FROM customers
WHERE  age < 25
AND salary > '20000'
```

ORDER BY age ASC, salary DESC;

AGE	ADDRESS	SALARY
22	delhi	22000

1 rows returned in 0.00 seconds

Select Example: select fields from multiple tables (JOIN)

SELECT customers.name, courses.trainer

FROM courses

INNER JOIN customers

ON courses.course_id = course_id

ORDER BY name;

Output

NAME	TRAINER
alex	sonoo jaiswal
alex	dd sharma
alex	swati uniyal
alex	rashmi desai
alex	mahesh sharma
mohan	mahesh sharma
mohan	rashmi desai
mohan	swati uniyal
mohan	sonoo jaiswal
mohan	dd sharma
More than 10 rows available. Increase rows selector to view more rows.	

10 rows returned in 0.02 seconds

Oracle Insert Query

Oracle insert query is used to insert records into table. For example:

insert into customers values(101,'rahul','delhi');

Oracle Insert Statement

In Oracle, INSERT statement is used to add a single record or multiple records into the table.

Syntax: (Inserting a single record using the Values keyword):

INSERT INTO table

(column1, column2, ... column_n)

VALUES

(expression1, expression2, ... expression_n);

Syntax: (Inserting multiple records using a SELECT statement):

INSERT INTO table

(column1, column2, ... column_n)

SELECT expression1, expression2, ... expression_n

FROM source_table

WHERE conditions;

Parameters:

- Table: The table to insert the records into.
- Column1, column2, Column_n:
- The columns in the table to insert values.
- Expression1, Expression2, Expression:
- The values to assign to the columns in the table. So column1 would be assigned the value of expression1, column2 would be assigned the value of expression2, and so on.
- Source table:
- The source table when inserting data from another table.
- Conditions:
- The conditions that must be met for the records to be inserted.

Oracle Insert Example: By VALUE keyword

It is the simplest way to insert elements to a database by using VALUE keyword.

See this example:

Consider here the already created suppliers table. Add a new row where the value of supplier_id is 23 and supplier_name is Flipkart.

See this example:

INSERT INTO suppliers

(supplier_id, supplier_name)

VALUES

(50, 'Flipkart');

Output:

1 row(s) inserted.

0.02 seconds

Oracle Insert Example: By SELECT statement

This method is used for more complicated cases of insertion. In this method insertion is done by SELECT statement. This method is used to insert multiple elements.

See this example:

In this method, we insert values to the "suppliers" table from "customers" table. Both tables are already created with their respective columns.

Execute this query:

INSERT INTO suppliers

(supplier_id, supplier_name)

SELECT age, address

FROM customers

WHERE age > 20;

Output:

4 row(s) inserted.

0.00 seconds

You can even check the number of rows that you want to insert by following statement:

```
SELECT count(*)
FROM customers
WHERE age > 20;
```

Output:

4

Oracle Update Query

Oracle update query is used to update records of a table. For example:

```
update customers set name='bob', city='london' where id=101;
```

Oracle UPDATE Statement

In Oracle, UPDATE statement is used to update the existing records in a table. You can update a table in 2 ways.

Traditional Update table method

Syntax:

```
UPDATE table
SET column1 = expression1,
   column2 = expression2,
   ...
   column_n = expression_n
WHERE conditions;
```

Update Table by selecting records from another table

Syntax:

```
UPDATE table1
```

```
SET column1 = (SELECT expression1
        FROM table2
        WHERE conditions)
WHERE conditions;
```

Parameters:

- column1, column2, ... column_n:
- It specifies the columns that you want to update.
- expression1, expression2, ...expression_n:
- This specifies the values to assign to the column1, column2, ?. column_n.
- Conditions: It specifies the conditions that must be fulfilled for execution of UPDATE stateme.

Oracle Update Example: (Update single column)

```
UPDATE suppliers
SET supplier_name = 'Kingfisher'
WHERE supplier_id = 2;
```

This example will update the supplier_name as "Kingfisher" where "supplier_id" is 2.

Oracle Update Example:

The following example specifies how to update multiple columns in a table. In this example, two columns supplier_name and supplier_address is updated by a single statement.

- Oracle Delete Query
- Oracle update query is used to delete records of a table from database.
- For example: Delete from customers where id=101;

Oracle DELETE Statement

Syntax

```
DELETE FROM table_name
WHERE conditions;
```

Parameters

- Table name**:** It specifies the table which you want to delete.
- Conditions**:** It specifies the conditions that must met for the records to be deleted.

Oracle Delete Example: On one condition

DELETE FROM customers

WHERE name = 'Sohan';

This statement will delete all records from the customer table where name is "Sohan".

Oracle Delete Example: On multiple conditions

DELETE FROM customers

WHERE last_name = 'Maurya'

AND customer_id > 2;

This statement will delete all records from the customers table where the last_name is "Maurya" and the customer_id is greater than 2.

Oracle Truncate Query

Oracle update query is used to truncate or remove records of a table. It doesn't remove structure. For example:

truncate table customers;

Oracle TRUNCATE TABLE

In Oracle, TRUNCATE TABLE statement is used to remove all records from a table. It works same as DELETE statement but without specifying a WHERE clause. It is generally used when you don’t have to worry about rolling back

Once a table is truncated, it can’t be rolled back. The TRUNCATE TABLE statement does not affect any of the table’s indexes, triggers or dependencies.

Syntax

TRUNCATE TABLE [schema_name.]table_name

Parameters

- schema_name: This parameter specifies the name of the schema that the table belongs to. It is optional.

- table_name: It specifies the table that you want to truncate.

Oracle TRUNCATE Table Example

- Consider a table named "customers" and execute the following query to truncate this
- TRUNCATE TABLE customers;

Output

Table truncated.

1.11 seconds

Now check the customers table, you will find that there is no data available in that table. It is equally similar to DELETE TABLE statement in Oracle.

Oracle DELETE Table Example

DELETE TABLE customers;

TRUNCATE TABLE Vs. DELETE TABLE

Both the statements will remove the data from the "customers" table but the main difference is that you can roll back the DELETE statement whereas you can't roll back the TRUNCATE TABLE statement.

Oracle Drop Query

Oracle drop query is used to drop a table or view. It doesn't have structure and data. For example:

drop table customers;

Oracle DROP TABLE Statement

Oracle DROP TABLE statement is used to remove or delete a table from the Oracle database.

Syntax

DROP [schema_name].TABLE table_name

[CASCADE CONSTRAINTS]

[PURGE];

Parameters

schema_name: It specifies the name of the schema that owns the table.

table_name: It specifies the name of the table which you want to remove from the Oracle database.

CASCADE CONSTRAINTS: It is optional. If specified, it will drop all referential integrity constraints as well.

PURGE: It is also optional. If specified, the table and its dependent objects are placed in the recycle bin and can't be recovered.

If there are referential integrity constraints on table_name and you do not specify the CASCADE CONSTRAINTS option, the DROP TABLE statement will return an error and Oracle will not drop the table.

DROP TABLE Example

DROP TABLE customers;

This will drop the table named customers.

DROP TABLE Example with PURGE parameter

DROP TABLE customers PURGE

This statement will drop the table called customers and issue a PURGE so that the space associated with the customers table is released and the customers table is not placed in recycle bin. So, it is not possible to recover that table if required.

Oracle Create Query

Oracle create query is used to create a table, view, sequence, procedure and function. For example:

CREATE TABLE customers

(id number(10) NOT NULL, name varchar2(50) NOT NULL, city varchar2(50),

CONSTRAINT customers_pk PRIMARY KEY (id));

Oracle Create Table

In Oracle, CREATE TABLE statement is used to create a new table in the database.

To create a table, you have to name that table and define its columns and datatype for each column.

Syntax:

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
);
```

Parameters used in syntax

table_name: It specifies the name of the table which you want to create.

column1, column2, ... column n: It specifies the columns which you want to add in the table. Every column must have a datatype. Every column should either be defined as "NULL" or "NOT NULL". In the case, the value is left blank; it is treated as "NULL" as default.

Oracle CREATE TABLE Example

Here we are creating a table named customers. This table doesn't have any primary key.

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
  customer_name varchar2(50) NOT NULL,
  city varchar2(50)
);
```

This table contains three columns

customer_id: It is the first column created as a number datatype (maximum 10 digits in length) and cannot contain null values.

customer_name: it is the second column created as a varchar2 datatype (50 maximum characters in length) and cannot contain null values.

city: This is the third column created as a varchar2 datatype. It can contain null values.

Oracle CREATE TABLE Example with primary key

```
CREATE TABLE customers
( customer_id number(10) NOT NULL,
```

```
  customer_name varchar2(50) NOT NULL,
  city varchar2(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);
```

Primary key

A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

Oracle Alter Query

Oracle alter query is used to add, modify, delete or drop columns of a table. Let's see a query to add column in customers table:

```
ALTER TABLE customers
ADD age varchar2(50);
```

Oracle ALTER TABLE Statement

In Oracle, ALTER TABLE statement specifies how to add, modify, drop or delete columns in a table. It is also used to rename a table.

Add column in a table

Syntax:

```
ALTER TABLE table_name
  ADD column_name column-definition;
```

Example:

Consider that already existing table customers. Now, add a new column customer_age into the table customers.

```
ALTER TABLE customers
ADD customer_age varchar2(50);
```

Now, a new column "customer_age" will be added in customers table.

Add multiple columns in the existing table

Syntax**:**

```
ALTER TABLE table_name
  ADD (column_1 column-definition,
      column_2 column-definition,
      ...
      column_n column_definition
);
```

Example

```
ALTER TABLE customers
  ADD (customer_type varchar2(50),
      customer_address varchar2(50));
```

Now, two columns customer_type and customer_address will be added in the table customers.

Modify column of a table

Syntax**:**

```
ALTER TABLE table_name
  MODIFY column_name column_type;
```

Example:

```
ALTER TABLE customers
MODIFY customer_name varchar2(100) not null;
```

Now the column column_name in the customers table is modified to varchar2 (100) and forced the column to not allow null values.

Modify multiple columns of a table

Syntax**:**

```
ALTER TABLE table_name
  MODIFY (column_1 column_type,
      column_2 column_type,
      ...
```

```
        column_n column_type);
```

Example**:**

```
ALTER TABLE customers
  MODIFY (customer_name varchar2(100) not null,
        city varchar2(100)
);
```

This will modify both the customer_name and city columns in the table.

Drop column of a table

Syntax**:**

```
ALTER TABLE table_name
  DROP COLUMN column_name;
```

Example:

```
ALTER TABLE customers
  DROP COLUMN customer_name;
```

This will drop the customer_name column from the table.

Rename column of a table

Syntax**:**

```
ALTER TABLE table_name
  RENAME COLUMN old_name to new_name;
```

Example:

```
ALTER TABLE customers
RENAME COLUMN customer_name to cname;
```

This will rename the column customer_name into cname.

How to rename table

Syntax:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Example:

```
ALTER TABLE customers
RENAME TO retailers;
```

This will rename the customer table into "retailers" table.

Oracle DISTINCT Clause

Oracle DISTINCT clause is used to remove the duplicate records from the result set. It is only used with SELECT statement.

Syntax:

```
SELECT DISTINCT expressions
FROM tables
WHERE conditions;
```

Parameters:

Expressions: It specifies the columns that you want to retrieve.

Tables: It specifies the table from where you want to retrieve records.

Conditions: It specifies the conditions that must be fulfilled.

Oracle DISTINCT Example: (with single expression)

Let's take a table "customers"

Customer table**:**

```
CREATE TABLE  "CUSTOMERS"
   (    "NAME" VARCHAR2(4000),
    "AGE" NUMBER,
    "SALARY" NUMBER,
    "STATE" VARCHAR2(4000)          );
```

Execute this query:

```
SELECT DISTINCT state
FROM customers
WHERE name = 'charu';
```

Output:

STATE
delhi

Oracle DISTINCT Example: (with multiple expressions)

```
SELECT DISTINCT name, age, salary
FROM customers
WHERE age >= '60';
```

Output:

This example specifies distinct name, age and salary of the customer where age is greater than or equal to 65.

Oracle FROM Clause

FROM clause is a mandatory clause in SELECT expression. It specifies the tables from which data is to be retrieved.

Syntax:

EDIT	NAME	AGE	SALARY	STATE
	mohan	16	12000	bihar
	rohan	21	25000	bihar
	teipratap	26	56000	bihar
	chanchal	21	25000	delhi
	charu	21	23000	delhi
	chanda	27	27000	delhi
	chandni	31	31000	delhi
	pralaynath	56	76000	tamilnadu
	gundaswami	65	87000	tamilnadu
	hina	18	18000	maharashtra
			row(s) 1 - 10 of 10	

```
FROM table_name...
Expressions...
```

Oracle FROM Clause Example: (with one table)

Let's take an example to explain how to use FROM clause to retrieve data from one table. Consider a table "customers".

NAME	AGE	SALARY	STATE
charu	21	23000	delhi
chanchal	21	25000	delhi
rohan	21	25000	bihar
chanda	27	27000	delhi
chandni	31	31000	delhi
tejpratap	26	56000	bihar
pralaynath	56	76000	tamilnadu
gundaswami	65	87000	tamilnadu

row(s) 1 - 10 of 10

Customer table:

```
CREATE TABLE  "CUSTOMERS"
("NAME" VARCHAR2(4000),
"AGE" NUMBER,
  "SALARY" NUMBER,
  "STATE" VARCHAR2(4000)
 ) ;
```

Execute this query:

NAME	AGE	SALARY
gundaswami	65	87000

```
SELECT *
FROM customers
WHERE salary >= 20000
ORDER BY salary ASC;
```

Output:

Oracle ORDER BY Clause

In Oracle, ORDER BY Clause is used to sort or re-arrange the records in the result set. The ORDER BY clause is only used with SELECT statement.

Syntax:

```
SELECT expressions
FROM tables
WHERE conditions
ORDER BY expression [ ASC | DESC ];
```

Parameters:

expressions: It specifies columns that you want to retrieve.

tables: It specifies the table name from where you want to retrieve records.

conditions: It specifies the conditions that must be fulfilled for the records to be selected.

SUPPLIER_ID	FIRST_NAME	LAST_NAME
2	kanchabhai	Motwani
3	lallu	lal
1	Dhirubhai	Ambani
	row(s) 1 - 3 of 3	

ASC: It is an optional parameter that is used to sort records in ascending order.

DESC: It is also an optional parameter that is used to sort records in descending order.

Oracle ORDER BY Example: (without ASC/DESC attribute)

Let's take a table "supplier"

Supplier table:

```
CREATE TABLE  "SUPPLIER"

   (    "SUPPLIER_ID" NUMBER,

    "FIRST_NAME" VARCHAR2(4000),

    "LAST_NAME" VARCHAR2(4000)

   ) ;
```

Execute this Query:

```
SELECT *
FROM supplier
ORDER BY last_name;
```

Output:

EDIT	SUPPLIER_ID	FIRST_NAME	LAST_NAME
	1	dhirubhai	ambani
	2	kanchabhai	motwani
	3	lallu	lal
		row(s) 1 - 3 of 3	

The above example returns the first_name ordered by last_name in ascending order.

Oracle ORDER BY Example: (sorting in descending order)

If you want to sort your result in descending order, you should use the DESC attribute in your ORDER BY clause:

Execute this Query:

SELECT *

FROM supplier

ORDER BY last_name **DESC**;

ITEM	Total Sales
Belts	105
Sari	5210
Shoes	1165
Medicines	250
Computer	210

Output

The above example returns the first_name ordered by last_name in descending order.

Oracle GROUP BY Clause

In Oracle GROUP BY clause is used with SELECT statement to collect data from multiple records and group the results by one or more columns.

Syntax:

SELECT expression1, expression2, ... expression_n,

aggregate_function (aggregate_expression)

FROM tables

WHERE conditions

SUPPLIER_ID	FIRST_NAME	LAST_NAME
1	dhirubhai	ambani
3	lallu	lal
2	kanchabhai	motwani
	row(s) 1 - 3 of 3	

GROUP BY expression1, expression2, ... expression_n;

Parameters:

expression1, expression2 ... expression_n: It specifies the expressions that are not encapsulated within aggregate function. These expressions must be included in GROUP BY clause.

- aggregate function: It specifies the aggregate functions i.e. SUM, COUNT, MIN, MAX or AVG functions.
- aggregate expression: It specifies the column or expression on that the aggregate function is based on.
- tables: It specifies the table from where you want to retrieve records.
- conditions: It specifies the conditions that must be fulfilled for the record to be selected.

Oracle GROUP BY Example: (with SUM function)

Let's take a table "salesdepartment"

Salesdepartment table:

CREATE TABLE "SALESDEPARTMENT"

STATE	Number Of Customers
maharashtra	1
delhi	4
tamilnadu	2
bihar	3

("ITEM" VARCHAR2(4000),

"SALE" NUMBER,

"BILLING_ADDRESS" VARCHAR2(4000)

) ;

EDIT	NAME	AGE	SALARY	STATE
	mohan	16	12000	bihar
	rohan	21	25000	bihar
	tejpratap	26	56000	bihar
	chanchal	21	25000	delhi
	charu	21	23000	delhi
	chanda	27	27000	delhi
	chandni	31	31000	delhi
	pralavnath	56	76000	tamilnadu
	gundaswami	65	87000	tamilnadu
	hina	18	18000	maharashtra

row(s) 1 - 10 of 10

Execute this query:

SELECT item, SUM(sale) AS "Total sales"

FROM salesdepartment

GROUP BY item;

Output

The above example will show the total sales of every individual item.

Oracle GROUP BY Example: (with COUNT function)

Let's take a table "customers"

Here we are creating a table named customers. This table doesn't have any primary key.

Customer table:

CREATE TABLE "CUSTOMERS"

("NAME" VARCHAR2(4000),

"AGE" NUMBER,

"SALARY" NUMBER,

"STATE" VARCHAR2(4000)

) ;

Execute this query:

DEPARTMENT	Lowest Salary
Mechanical	12000
Software	10000
hardware	15000

```
SELECT state, COUNT(*) AS "Number of customers"
FROM customers
WHERE salary > 10000
GROUP BY state;
```

Output:

EDIT	EMP_ID	NAME	AGE	DEPARTMENT	SALARY
	1	aladin	21	Mechanical	12000
	2	billu	23	hardware	15000
	3	chhava	22	Software	24000
	4	Dinesh	34	Software	32000
	5	ramesh	21	Software	20000
	6	lallu	18	Software	10000

row(s) 1 - 6 of 6

Oracle GROUP BY Example: (with MIN function)

Let's take a table "employees"

Employees table:

```
CREATE TABLE  "EMPLOYEES"
  (    "EMP_ID" NUMBER,
   "NAME" VARCHAR2(4000),
   "AGE" NUMBER,
   "DEPARTMENT" VARCHAR2(4000),
   "SALARY" NUMBER
  );
```

EDIT	ITEM	SALE	BILLING_ADDRESS
	Shoes	120	Agra
	Belts	105	Kolkata
	Shoes	45	Allahabad
	Sari	210	Varanasi
	Sari	5000	Chennai
	Medicines	250	Salem
	Computer	210	Delhi
	Shoes	1000	Kanpur

row(s) 1 - 8 of 8

Execute this query:

```
SELECT department,
MIN(salary) AS "Lowest salary"
FROM employees
```

GROUP BY department;

Output:

Oracle GROUP BY Example: (with MAX function)

In this example, we are using "employees" table that is given above.

Execute this query:

SELECT department,

MAX(salary) AS "Highest salary"

FROM employees

GROUP BY department;

Oracle HAVING Clause

In Oracle, HAVING Clause is used with GROUP BY Clause to restrict the groups of returned rows where condition is TRUE.

Syntax:

SELECT expression1, expression2, ... expression_n,

aggregate_function (aggregate_expression)

FROM tables

WHERE conditions

GROUP BY expression1, expression2, ... expression_n

HAVING having_condition;

Parameters:

expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within aggregate function. These expressions must be included in GROUP BY clause.

aggregate_function: It specifies the aggregate functions i.e. SUM, COUNT, MIN, MAX or AVG functions.

aggregate_expression: It specifies the column or expression on that the aggregate function is based on.

tables: It specifies the table from where you want to retrieve records.

conditions: It specifies the conditions that must be fulfilled for the record to be selected.

having_conditions: It specifies the conditions that are applied only to the aggregated results to restrict the groups of returned rows.

Oracle HAVING Example: (with GROUP BY SUM function)

Let's take a table "salesdepartment"

Salesdepartment table:

CREATE TABLE "SALESDEPARTMENT"

("ITEM" VARCHAR2(4000),

"SALE" NUMBER,

"BILLING_ADDRESS" VARCHAR2(4000)

) ;

ITEM	Total Sales
Belts	105
Medicines	250
Computer	210

Execute this query:

```
SELECT item, SUM(sale) AS "Total sales"
FROM salesdepartment
GROUP BY item
HAVING SUM(sale) < 1000;
```

Output:

Oracle HAVING Example: (with GROUP BY COUNT function)

Let's take a table "customers"

Customer table:

EDIT	ITEM	SALE	BILLING_ADDRESS
	Shoes	120	Agra
	Belts	105	Kolkata
	Shoes	45	Allahabad
	Sari	210	Varanasi
	Sari	5000	Chennai
	Medicines	250	Salem
	Computer	210	Delhi
	Shoes	1000	Kanpur

row(s) 1 - 8 of 8

CREATE TABLE "CUSTOMERS"

("NAME" VARCHAR2(4000),

"AGE" NUMBER,

"SALARY" NUMBER,

"STATE" VARCHAR2(4000)

) ;

Execute this query:

SELECT state, COUNT(*) AS "Number of customers"

FROM customers

WHERE salary > 10000

GROUP BY state

HAVING COUNT(*) >= 2;

Output:

STATE	Number Of Customers
delhi	4
tamilnadu	2
bihar	3

Oracle HAVING Example: (with GROUP BY MIN function)

Let's take a table "employees"

EDIT	EMP_ID	NAME	AGE	DEPARTMENT	SALARY
	1	aladin	21	Mechanical	12000
	2	billu	23	hardware	15000
	3	chhaya	22	Software	24000
	4	Dinesh	34	Software	32000
	5	ramesh	21	Software	20000
	6	lallu	18	Software	10000

row(s) 1 - 6 of 6

Employees table:

CREATE TABLE "EMPLOYEES"

("EMP_ID" NUMBER,

"NAME" VARCHAR2(4000),

"AGE" NUMBER,

"DEPARTMENT" VARCHAR2(4000),

"SALARY" NUMBER)

Execute this query:

SELECT department,

EDIT	NAME	AGE	SALARY	STATE
	mohan	16	12000	bihar
	rohan	21	25000	bihar
	tejpratap	26	56000	bihar
	chanchal	21	25000	delhi
	charu	21	23000	delhi
	chanda	27	27000	delhi
	chandni	31	31000	delhi
	pralaynath	56	76000	tamilnadu
	gundaswami	65	87000	tamilnadu
	hina	18	18000	maharashtra

row(s) 1 - 10 of 10

DEPARTMENT	Lowest Salary
Mechanical	12000
Software	10000

```
MIN(salary) AS "Lowest salary"
FROM employees
GROUP BY department
HAVING MIN(salary) < 15000;
```

Output

Oracle HAVING Example: (with GROUP BY MAX function)

Execute this query:

```
SELECT department,
MAX(salary) AS "Highest salary"
FROM employees
GROUP BY department
HAVING MAX(salary) > 30000;
```

Oracle UNION Operator

In Oracle, UNION operator is used to combine the result sets of two or more Oracle SELECT statements. It combines the both SELECT statement and removes duplicate rows between them./p>

Each SELECT statement within the UNION operator must have the same number of fields in the result sets with similar data types.

Syntax

```
SELECT expression1, expression2, ... expression_n
FROM table1
WHERE conditions
UNION
SELECT expression1, expression2, ... expression_n
FROM table2
WHERE conditions;
```

Parameters

- expression1, expression2, ... expression_n: It specifies the columns that you want to retrieve.
- table1, table2: it specifies the tables from where you retrieve the records.

- conditions: it specifies the conditions that must be fulfilled for the records to be selected.
- Note: The number of expressions must be same in both of the SELECT statements.

Oracle UNION Example: (Fetch single field)

```
SELECT supplier_id
FROM suppliers
UNION
SELECT supplier_id
FROM order_details
```

SUPPLIER_ID
1
2
3
20
21
22
25
28
30
31
More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.07 seconds

OUTPUT

In this example, supplier_id is defined in both of the table "suppliers" and "order_details". After the UNION, it would appear once in the result set because Oracle UNION operator removes duplicate sets.

Note: If you don't want to remove duplicates, use Oracle UNION ALL operator.

Oracle UNION Example: (Using ORDER BY)

The Oracle UNION operator can be used with ORDER BY clause to orders the results of the query.

```
SELECT supplier_id, supplier_name
FROM suppliers
WHERE supplier_id <= 20
UNION
SELECT s_id, s_name
FROM shopkeepers
WHERE s_name = 'dhirubhai'
ORDER BY 1;
```

SUPPLIER_ID	SUPPLIER_NAME
1	Bata shoes
1	dhirubhai
2	Kingfisher
3	VOJO
20	Google

5 rows returned in 0.01 seconds

Output

In the above example, result is sorted by supplier_name/s_name in ascending order, as denoted by ORDER BY 1.

Oracle UNION ALL Operator

In Oracle, the UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It is different from UNION operator in a way that it does not remove duplicate rows between the various SELECT statements. It returns all of the rows.

Each SELECT statement within the UNION ALL must have the same number of fields in the result sets with similar data types.

Difference between UNION and UNION ALL operators

UNION operator removes duplicate rows while UNION ALL operator does not remove duplicate rows.

Syntax

SELECT expression1, expression2, ... expression_n

FROM table1

WHERE conditions

UNION ALL

SELECT expression1, expression2, ... expression_n

FROM table2

WHERE conditions;

Parameters

expression1, expression2, expression_n: It specifies the columns that you want to retrieve. Number of expressions must be same in both SELECT statements.

table1, table2: It specifies the table name that you want to retrieve records from.

conditions: It specifies the conditions that must be fulfilled for the records to be selected.

Oracle UNION ALL Operator Example

- SELECT supplier_id
- FROM suppliers
- UNION ALL
- SELECT supplier_id
- FROM order_details;

The above example will return the supplier_id multiple times in the result set if the same value appeared in both the supplier_id and order_details table.

Output

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIER_ID	NUMBER	Yes	-	-
FIRST_NAME	VARCHAR2(4000)	Yes	-	-
LAST_NAME	VARCHAR2(4000)	Yes	-	-

1 - 3

Oracle INTERSECT Operator

In Oracle, INTERSECT Operator is used to return the results of 2 or more SELECT statement. It picks the common or intersecting records from compound SELECT queries.

Syntax

SELECT expression1, expression2, ... expression_n

FROM table1

WHERE conditions

INTERSECT

SELECT expression1, expression2, ... expression_n

FROM table2

WHERE conditions;

Parameters

expression1, expression2, ... expression_n: It specifies the columns that you want to retrieve.

table1, table2: It specifies the tables that you want to retrieve records from.

conditions: it specifies the conditions that must be fulfilled for the records to be selected.

Oracle INTERSECT Example: (with single expression)

Suppliers Table

Suppliers Data

Order_details Table

SUPPLIER_ID
1
2
3

3 rows returned in 0.02 seconds

Order_details Data

SELECT supplier_id

FROM suppliers

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIER_ID	NUMBER	Yes	-	-
SUPPLIER_NAME	VARCHAR2(4000)	Yes	-	-
SUPPLIER_ADDRESS	VARCHAR2(4000)	Yes	-	-

1 - 3

INTERSECT

EDIT	SUPPLIER_ID	SUPPLIER_NAME	SUPPLIER_ADDRESS
	1	Bata shoes	Agra
	2	Kingfisher	Delhi
	3	VOJO	Lucknow
	4	Apple	-
	5	Flipkart	-

SELECT supplier_id

EDIT	SUPPLIER_ID	SUPPLY_DATE	SUPPLY_ADDRESS
	1	12-OCT-12	Ahmedabad
	2	23-OCT-12	Mumbai
	3	31-OCT-12	delhi

row(s) 1 - 3 of 3

FROM order_details;

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIER_ID	NUMBER	Yes	-	-
SUPPLY_DATE	DATE	Yes	-	-
SUPPLY_ADDRESS	VARCHAR2(4000)	Yes	-	-

1 - 3

In the above example, the supplier_id appears in both the suppliers and order_details table. Now the common entries will be returned in the result set.

Output

Oracle INTERSECT Example: (with multiple expressions)

Supplier Table

Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMER_ID	NUMBER	Yes	-	-
FIRST_NAME	VARCHAR2(4000)	Yes	-	-
LAST_NAME	VARCHAR2(4000)	Yes	-	-
DEPARTMENT_ID	NUMBER	Yes	-	-

1 - 4

Supplier Data

EDIT	CUSTOMER_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
	1	aryan	tomar	-
	2	ajeet	maurya	-
	3	richa	goyal	-
	4	dhirubhai	jotwani	-
	32	alex	pandian	1

row(s) 1 - 5 of 5

Customer Table

Customer Data

EDIT	SUPPLIER_ID	FIRST_NAME	LAST_NAME
	1	dhirubhai	ambani
	2	kanchabhai	motwani
	3	lallu	lal

row(s) 1 - 3 of 3

SELECT supplier_id, last_name, first_name

FROM supplier

```
WHERE first_name <> 'dhirubhai'

INTERSECT

SELECT customer_id, last_name, first_name

FROM customer

WHERE customer_id < 5;
```

LAST_NAME	FIRST_NAME
maurya	ajeet

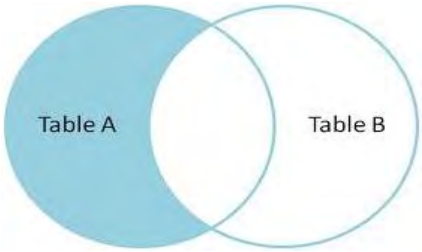
1 rows returned in 0.00 seconds

Output

The above example returns the records from the supplier table where the supplier_id, last_name and first_name values match the customer_id, last_name, and first_name value of customer table.

Oracle MINUS operator

In Oracle, MINUS operator is used to return all rows in the first SELECT statement that are not returned by the second SELECT statement.



Each SELECT statement has a dataset and the MINUS operator returns all documents from the first dataset and then removes all documents from the second dataset.

For example

Syntax

```
SELECT expression1, expression2, ... expression_n

FROM table1

WHERE conditions

MINUS

 SELECT expression1, expression2, ... expression_n

 FROM table2

 WHERE conditions;
```

Parameters

expression1, expression2, ... expression_n: It specifies the columns that you want to retrieve.

table1, table2: it specifies the tables that you want to retrieve records from.

conditions: it specifies the conditions that must be fulfilled for the records to be selected.

Note: The expressions must be same in number for both the SELECT statement and have similar datatype.

Oracle MINUS Example

This example will return one field with the same datatype from two tables "suppliers" and "order_details".

```
SELECT supplier_id
FROM suppliers
MINUS
SELECT supplier_id
FROM order_details;
```

Output

SUPPLIER_ID
20
21
22
25
28
30
31
50

8 rows returned in 0.11 seconds

SQL Functions in Oracle

SQL functions are built into Oracle and are available for use in various appropriate SQL statements. You can also create your own function using PL/SQL.

Single-Row Functions

Single-row functions return a single result row for every row of a queried table or view. These functions can appear in select lists, WHERE clauses, START WITH and CONNECT BY clauses, and HAVING clauses.

Oracle SQL Functions can be divided into following categories

- Number Functions
- Character Functions
- Miscellaneous Single Row Functions
- Aggregate Functions
- Date and Time Functions

Here are the explanation and example of these functions

Number Functions (also known as Math Functions)

Number functions accept numeric input and return numeric values. Most of these functions return values that are accurate to 38 decimal digits.

The number functions available in Oracle are:

ABS ACOS ASIN ATAN ATAN2 BITAND CEIL COS COSH EXP FLOOR LN

LOGMOD POWER ROUND (number) SIGN SIN SINH SQRT TAN TANH TRUNC (number)

ABS

ABS returns the absolute value of n.

The following example returns the absolute value of -87:

```
SELECT ABS(-87) "Absolute" FROM DUAL;

  Absolute
 ----------
        87
```

ACOS

ACOS returns the arc cosine of n. Inputs are in the range of -1 to 1, and outputs are in the range of 0 to pi and are expressed in radians.

The following example returns the arc cosine of .3:

```
SELECT ACOS(.3)"Arc_Cosine" FROM DUAL;

Arc_Cosine
```

1.26610367

Similar to ACOS, you have ASIN (Arc Sine), ATAN (Arc Tangent) functions.

CIEL

Returns the lowest integer above the given number.

Example:

The following function return the lowest integer above 3.456;

select ciel(3.456) “Ciel” from dual;

Ciel

4

FLOOR

Returns the highest integer below the given number.

Example:

The following function return the highest integer below 3.456;

select floor(3.456) “Floor” from dual;

Floor

3

COS

Returns the cosine of an angle (in radians).

Example:

The following example returns the COSINE angle of 60 radians.

select cos(60) “Cosine” from dual;

SIN

Returns the Sine of an angle (in radians).

Example:

The following example returns the SINE angle of 60 radians.

select SIN(60) “Sine” from dual;

TAN

Returns the Tangent of an angle (in radians).

Example:

The following example returns the tangent angle of 60 radians.

```
select Tan(60) “Tangent” from dual;
```

Similar to SIN, COS, TAN functions hyperbolic functions SINH, COSH, TANH are also available in oracle.

MOD

Returns the remainder after dividing m with n.

Example

The following example returns the remainder after dividing 30 by 4.

```
Select mod(30,4) “MOD” from dual;

MOD
---------
       2
```

POWER

Returns the power of m, raised to n.

Example

The following example returns the 2 raised to the power of 3.

```
select power(2,3) “Power” from dual;

POWER
---------
       8
```

EXP

Returns the e raised to the power of n.

Example

The following example returns the e raised to power of 4.

```
SELECT EXP(4) "e to the 4th power"
  FROM DUAL;

e to the 4th power
------------------
          54.59815
```

LN

Returns natural logarithm of n.

Example

The following example returns the natural logarithm of 95.

```
SELECT LN(95) "Natural log of 95"
  FROM DUAL;

Natural log of 95
-----------------
       4.55387689
```

LOG

Returns the logarithm, base m, of n.

Example

The following example returns the log of 100.

```
select log(10,100) from dual;

LOG
---------
       2
```

ROUND

Returns a decimal number rounded of to a given decimal positions.

Example

The following example returns the no. 3.4573 rounded to 2 decimals.

```
select round(3.4573,2) “Round” from dual;

Round
------------
       3.46
```

TRUNC

Returns a decimal number Truncated to a given decimal positions.

Example

The following example returns the no. 3.4573 truncated to 2 decimals.

select round(3.4573,2) “Round” from dual;

Round

3.45

SQRT

Returns the square root of a given number.

Example

The following example returns the square root of 16.

select sqrt(16) from dual;

SQRT

4

Character Functions

Character functions operate on values of dataype CHAR or VARCHAR.

LOWER

Returns a given string in lower case.

select LOWER(‘SAMI’) from dual;

LOWER

---

sami

UPPER

Returns a given string in UPPER case.

select UPPER(‘Sami’) from dual;

UPPER

SAMI

INITCAP

Returns a given string with Initial letter in capital.

select INITCAP(‘mohammed sami’) from dual;

INITCAP

Mohammed Sami

LENGTH

Returns the length of a given string.

select length(‘mohammed sami’) from dual;

LENGTH

13

SUBSTR

Returns a substring from a given string. Starting from position p to n characters.

For example the following query returns “sam” from the string “mohammed sami”.

select substr('mohammed sami',10,3) from dual;

Substr
---
sam

INSTR

Tests whether a given character occurs in the given string or not. If the character occurs in the string then returns the first position of its occurrence otherwise returns 0.

Example

The following query tests whether the character “a” occurs in string “mohammed sami”

select instr('mohammed sami','a') from dual;

INSTR

4

REPLACE

Replaces a given set of characters in a string with another set of characters.

Example

The following query replaces “mohd” with “mohammed” .

```
select replace('ali mohd khan','mohd','mohammed') from dual;

REPLACE
---------
ali mohammed khan
```

INSTR

Tests whether a given character occurs in the given string or not. If the character occurs in the string then returns the first position of its occurrence otherwise returns 0.

Example

The following query tests whether the character “a” occurs in string “mohammed sami”

```
select instr('mohammed sami','a') from dual;

INSTR
--------
4
```

REPLACE

Replaces a given set of characters in a string with another set of characters.

Example

The following query replaces “mohd” with “mohammed” .

```
select replace('ali mohd khan','mohd','mohammed') from dual;

REPLACE
---------
ali mohammed khan
```

TRANSLATE

This function is used to encrypt characters. For example you can use this function to replace characters in a given string with your coded characters.

Example

The following query replaces characters A with B, B with C, C with D, D with E,...Z with A, and a with b,b with c,c with d, d with e ....z with a.

```
select translate('interface','ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',
    'BCDEFGHIJKLMNOPQRSTUVWXYZAbcdefghijklmnopqrstuvwxyza') “Encrypt” from dual;

Encrypt
```

---

joufsgbdf

SOUNDEX

This function is used to check pronounciation rather than exact characters. For example many people write names as “smith” or “smyth” or “smythe” but they are pronounced as smith only.

Example

The following example compare those names which are spelled differently but are pronouced as “smith”.

Select ename from emp where soundex(ename)=soundex('smith');

ENAME

Smith
Smyth
Smythe

RPAD

Right pads a given string with a given character to n number of characters.

Example

The following query rights pad ename with '*' until it becomes 10 characters.

select rpad(ename,'*',10) from emp;

Ename

Smith*****
John******
Mohammed**
Sami******

LPAD

Left pads a given string with a given character upto n number of characters.

Example

The following query left pads ename with '*' until it becomes 10 characters.

select lpad(ename,'*',10) from emp;

Ename

*****Smith

******John

**Mohammed

******Sami

LTRIM

Trims blank spaces from a given string from left.

Example

The following query returns string “ Interface “ left trimmed.

```
select ltrim('  Interface  ') from dual;

Ltrim
--------------
Interface
```

RTRIM

Trims blank spaces from a given string from Right.

Example

The following query returns string “ Interface “ right trimmed.

```
select rtrim('  Interface  ') from dual;

Rtrim
------------
  Interface
```

TRIM

Trims a given character from left or right or both from a given string.

Example

The following query removes zero from left and right of a given string.

```
Select trim(0 from '00003443500') from dual;

Trim
----------
34435
```

CONCAT

Combines a given string with another string.

Example

The following Query combines ename with literal string “ is a “ and jobid.

Select concat(concat(ename,' is a '),job) from emp;

Concat

Smith is a clerk
John is a Manager
Sami is a G.Manager

Miscellaneous Single Row Functions

COALESCE

Coalesce function returns the first not null value in the expression list.

Example.

The following query returns salary+commision, if commission is null then returns salary, if salary is also null then returns 1000.

select empno,ename,salary,comm,coalesce(salary+comm,salary,1000) “Net Sal” from emp;

ENAME	SALARY	COMM	NET SAL
SMITH	1000	100	1100
SAMI	3000		3000
SCOTT			1000
RAVI		200	1000

DECODE

DECODE(expr, searchvalue1, result1,searchvalue2,result2,..., defaultvalue)

Decode functions compares an expr with search value one by one. If the expr does not match any of the search value then returns the default value. If the default value is omitted then returns null.

Example

The following query returns the department names according the deptno. If the deptno does not match any of the search value then returns “Unknown Department”

```
select decode(deptno,10,'Sales',20,'Accounts,30,'Production,
        40,'R&D','Unknown Dept') As DeptName from emp;
```

DEPTNAME

Sales
Accounts
Unknown Dept.

Accounts
Production
Sales
R&D
Unknown Dept.

GREATEST

GREATEST(expr1, expr2, expr3,expr4...)

Returns the greatest expr from a expr list.

Example

select greatest(10,20,50,20,30) from dual;

GREATEST

50

select greatest('SAMI','SCOTT','RAVI','SMITH','TANYA') from dual;

GREATEST

TANYA

LEAST

LEAST(expr1, expr2, expr3,expr4...)

It is simillar to greatest. It returns the least expr from the expression list.

select least(10,20,50,20,30) from dual;

LEAST

10

select least('SAMI','SCOTT','RAVI','SMITH','TANYA') from dual;

LEAST

RAVI

NVL

NVL2(expr1,expr2)

This function is oftenly used to check null values. It returns expr2 if the expr1 is null, otherwise returns expr1.

Example

The following query returns commission if commission is null then returns 'Not Applicable'.

Select ename,nvl(comm,'Not Applicable') “Comm” from dual;

ENAME COMM
------ ----
Scott 300
Tiger 450
Sami Not Applicable
Ravi 300
Tanya Not Applicable

NVL2

NVL2(expr1,expr2,expr3)

NVL2 returns expr2 if expr1 is not null, otherwise return expr3.

Example

The following query returns salary+comm if comm is not null, otherwise just returns salary.

select salary,comm,nvl2(comm,salary+comm,salary) “Income” from emp;

SALARY COMM INCOME
------ ---- ------
1000 100 1100
2000 2000
2300 200 2500
3400 3400

NULLIF

NULLIF(expr1, expr2)

Nullif compares expr1 with expr2. If they are equal then returns null, otherwise return expr1.

Example.

The following query shows old jobs of those employees who have changed their jobs in the company by comparing the current job with old job in oldemp table.

Select ename,nullif(e.job,o.job) “Old Job” from emp e, oldemp o where e.empno=o.empno;

ENAME OLD JOB
----- -------
SMITH CLERK

SAMI
SCOTT MANAGER

UID

Returns the current session ID of user logged on.

Example

select uid from dual;

UID

20

USER

Returns the username of the current user logged on.

select user from dual;

USER

SCOTT

SYS_CONTEXT

SYS_CONTEXT returns the value of parameter associated with the context namespace. You can use this function in both SQL and PL/SQL statements.

EXAMPLE

The following query returns the username of the current user.

Select sys_context('USERENV','SESSION_USER') “Username” from dual;

USERNAME

SCOTT

Similar to SESSION_USER parameter for namespace USERENV the other important parameters are

ISDBA :To check whether the current user is having DBA privileges or not.

HOST :Returns the name of host machine from which the client is connected.

INSTANCE :The instance identification number of the current instance

IP_ADDRESS: IP address of the machine from which the client is connected.

DB_NAME : Name of the database as specified in the DB_NAME initialization parameter

VSIZE

VSIZE(expr)

Returns the internal representation of expr in bytes.

Example

The following query return the representation of ename in bytes.

select ename,vsize(ename) as Bytes from emp;

ENAME BYTES
------ ------
SCOTT 5
SAMI 4
RAVI 4
KIRAN 5

Oracle Date Functions and Operators.

Functions to View Current Date and Time

To see the system date and time in Oracle, we can use the following functions:

CURRENT_DATE : Returns the current date in the session time zone, in a value in the Gregorian calendar of datatype
DATE
SYSDATE : Returns the current date and time.
SYSTIMESTAMP : The SYSTIMESTAMP function returns the system date, including fractional seconds and time zone of the database. The return type is TIMESTAMP WITH TIME ZONE.

SYSDATE Example

To see the current system date and time we can use the most simple and widely used SYSDATE function.

To see system date and time give the following query.

select sysdate from dual;

SYSDATE

8-AUG-03

The format in which the date is displayed depends on NLS_DATE_FORMAT parameter.

For example set the NLS_DATE_FORMAT to the following format

alter session set NLS_DATE_FORMAT='DD-MON-YYYY HH:MIpm';

Then give the give the following statement

```
select sysdate from dual;
SYSDATE
------------------
8-AUG-2003 03:05pm
```

The default setting of NLS_DATE_FORMAT is DD-MON-YY

CURRENT_DATE Example

To see the current system date and time with time zone use CURRENT_DATE function

```
ALTER SESSION SET TIME_ZONE = '-4:0';
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-04:00          22-APR-2003 14:15:03

ALTER SESSION SET TIME_ZONE = '-7:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE CURRENT_DATE
--------------- --------------------
-07:00          22-APR-2003 09:15:33
```

SYSTIMESTAMP Example

To see the current system date and time with fractional seconds with time zone give the following statement

```
select systimestamp from dual;

SYSTIMESTAMP
-------------------------------
22-APR-03 08.38.55.538741 AM -07:00
```

Next, How to view Date and Time in different formats.

Formatting Oracle Dates in different Formats

By default, Oracle will display the dates in DD-MON-YY format. If you want to see dates in other format then you can do so by using TO_CHAR() function. The TO_CHAR() function is used to convert date and time into a character string according to the given format. Oracle has provided several formats to convert the date and time to meet all user requirements

To translate the date into a different format string you can use TO_CHAR function with date format. For example, to see the current day you can give the following query

Select to_char(sysdate,'DAY')"Today" FROM DUAL;

TODAY

THURSDAY

Like this "DAY" format model there are many other date format models available in Oracle. The following table list date and time format models.

FORMAT	MEANING
D	Day of the week
DD	Day of the month
DDD	Day of the year
DAY	Full day for ex. 'Monday', 'Tuesday', 'Wednesday'
DY	Day in three letters for ex. 'MON', 'TUE','FRI'
W	Week of the month
WW	Week of the year
MM	Month in two digits (1-Jan, 2-Feb,…12-Dec)
MON	Month in three characters like "Jan", "Feb", "Apr"
MONTH	Full Month like "January", "February", "April"
RM	Month in Roman Characters (I-XII, I-Jan, II-Feb,…XII-Dec)
Q	Quarter of the Month
YY	Last two digits of the year.
YYYY	Full year
YEAR	Year in words like "Nineteen Ninety Nine"
HH	Hours in 12 hour format
HH12	Hours in 12 hour format
HH24	Hours in 24 hour format
MI	Minutes
SS	Seconds
FF	Fractional Seconds
SSSSS	Milliseconds
J	Julian Day i.e. Days since 1st-Jan-4712BC to till-date
RR	If the year is less than 50 Assumes the year as 21ST Century. If the year is greater than 50 then assumes the year in 20th Century.

Suffixes

TH	Returns th, st, rd or nd according to the leading number like 1st , 2nd 3rd 4th
SP	Spells out the leading number
AM or PM	Returns AM or PM according to the time

SPTH	Returns Spelled Ordinal number. For. Example First, Fourth

For example to see the todays date in the following format

Friday, 7th March, 2014

Give the following statement

select to_char(sysdate,'Day, ddth Month, yyyy')"Today" from dual;

TODAY

Friday, 7th March, 2014

For example you want to see hire dates of all employee in the following format

Friday, 8th August, 2003

Then give the following query.

select to_char(hire_date,'Day, ddth Month, yyyy') from emp;

Next, to translate a character value, which is in format other than the default date format, into a date value you can use TO_DATE function with date format to date

TO_DATE Example

TO_DATE() function is used to convert strings into date values. For example you want to see what was the day on 15-aug-1947. For this purpose, we will use the TO_DATE() function to first convert the string into date value and then pass on this value to TO_CHAR() function to extract day.

select to_char(to_date('15-aug-1947','dd-mon-yyyy'),'Day') from dual;
TO_CHAR(

Friday

To see how many days have passed since 15-aug-1947 then give the following query

select sysdate-to_date('15-aug-1947','dd-mon-yyyy') from dual;

Now we want to see which date will occur after 45 days from now

select sysdate+45 from dual;

SYSDATE

06-JUN-2003

ADD_MONTHS

To see which date will occur after 6 months from now, we can use ADD_MONTHS function

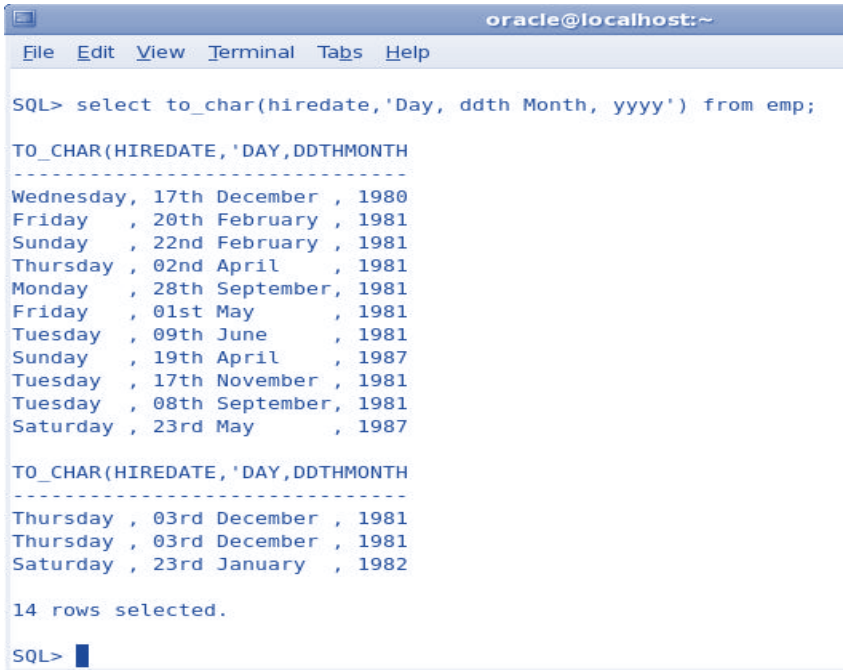
Select ADD_MONTHS(SYSDATE,6) from dual;

ADD_MONTHS

22-OCT-2003

MONTHS_BETWEEN

To see how many months have passed since a particular date, use the MONTHS_BETWEEN function.



For Example, to see how many months have passed since 15-aug-1947, give the following query.

select months_between(sysdate,to_date('15-aug-1947')) from dual;

Months

616.553

To eliminate the decimal value use truncate function

select trunc(months_between(sysdate,to_date('15-aug-1947'))) from dual;

Months

616

LAST_DAY

To see the last date of the month of a given date, Use LAST_DAY function.

select LAST_DAY(sysdate) from dual;

LAST_DAY

31-AUG-2003

NEXT_DAY

To see when a particular day is coming next , use the NEXT_DAY function.

For Example to view when next Saturday is coming, give the following query

select next_day(sysdate) from dual;

NEXT_DAY

09-AUG-2003

EXTRACT

An EXTRACT datetime function extracts and returns the value of a specified datetime field from a datetime or interval value expression. When you extract a TIMEZONE_REGION or TIMEZONE_ABBR (abbreviation), the value returned is a string containing the appropriate time zone name or abbreviation

The syntax of EXTRACT function is

EXTRACT (YEAR / MONTH / WEEK / DAY / HOUR / MINUTE / TIMEZONE FROM DATE)

Example
The following demonstrate the usage of EXTRACT function to extract year from current date.

select extract(year from sysdate) from dual;

EXTRACT

2003

Oracle Joins

Join is a query that is used to combine rows from two or more tables, views, or materialized views. It retrieves data from multiple tables and creates a new table.

Join Conditions

There may be at least one join condition either in the FROM clause or in the WHERE clause for joining two tables. It compares two columns from different tables and combines pair of rows, each containing one row from each table, for which join condition is true.

Types of Joins

- Inner Joins (Simple Join)
- Outer Joins

Left Outer Join (Left Join)

Right Outer Join (Right Join)

Full Outer Join (Full Join)

- Equijoins
- Self Joins
- Cross Joins (Cartesian Products)
- Antijoins
- Semijoins

Oracle INNER JOIN

- Inner Join is the simplest and most common type of join. It is also known as simple join. It returns all rows from multiple tables where the join condition is met.

Syntax

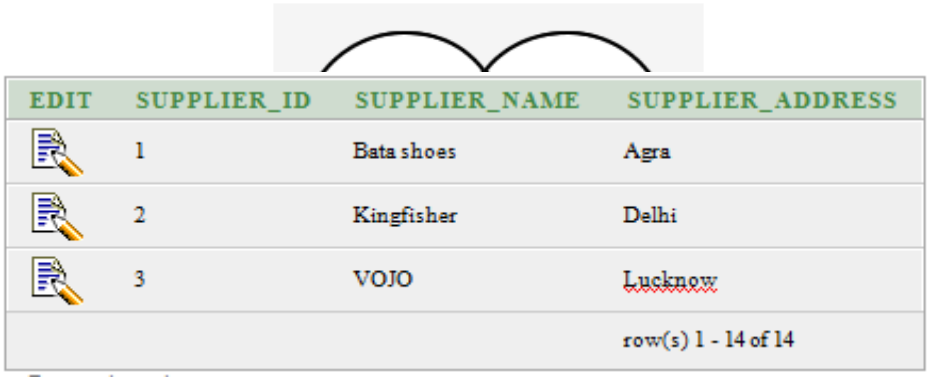
SELECT columns

FROM table1

INNER JOIN table2

ON table1.column = table2.column;

Image representation of Inner Join



EDIT	SUPPLIER_ID	SUPPLIER_NAME	SUPPLIER_ADDRESS
	1	Bata shoes	Agra
	2	Kingfisher	Delhi
	3	VOJO	Lucknow
			row(s) 1 - 14 of 14

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIER_ID	NUMBER	Yes	-	-
SUPPLIER_NAME	VARCHAR2(4000)	Yes	-	-
SUPPLIER_ADDRESS	VARCHAR2(4000)	Yes	-	-
				1 - 3

Oracle INNER JOIN Example

Let's take an example to perform Inner Join on two tables "Suppliers" and "Order1".

Suppliers

Column Name	Data Type	Nullable	Default	Primary Key
ORDER_NUMBER	NUMBER	Yes	-	-
SUPPLIER_ID	NUMBER	Yes	-	-
CITY	VARCHAR2(4000)	Yes	-	-
				1 - 3

Order1

This example will return all rows from "suppliers" and "order1" table where there is a matching supplier_id value in both the suppliers and order1 tables.

Execute the following query

SELECT suppliers.supplier_id, suppliers.supplier_name, order1.order_number

EDIT	ORDER_NUMBER	SUPPLIER_ID	CITY
	101	1	Allahabad
	102	2	Kanpur
		row(s) 1 - 3 of 3	

FROM suppliers

INNER JOIN order1

ON suppliers.supplier_id = order1.supplier_id;

Output

SUPPLIER_ID	SUPPLIER_NAME	ORDER_NUMBER
1	Bata shoes	101
2	Kingfisher	102

2 rows returned in 0.03 seconds

Oracle OUTER JOIN

An outer join is similar to equijoin but it gets also the non-matched rows from the table. It is categorized in Left Outer Join, Right Outer Join and Full Outer Join by Oracle 9i ANSI/ISO 1999 standard.

Left Outer Join

Left Outer Join returns all rows from the left (first) table specified in the ON condition and only those rows from the right (second) table where the join condition is met.

Syntax

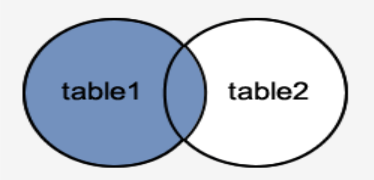
SELECT columns

FROM table1

LEFT [OUTER] JOIN table2

ON table1.column = table2.column;

Image representation of left outer join



Example

In this example, we are performing left outer join on the already created tables ?suppliers? and ?order1?.

The following example would return all records from table ?suppliers? and only those records from table ?order1? where the join fields are equal.

Execute this query

SELECT suppliers.supplier_id, suppliers.supplier_name, order1.order_number

FROM suppliers

LEFT OUTER JOIN order1

ON suppliers.supplier_id = order1.supplier_id;

Output

Right Outer Join

The Right Outer Join returns all rows from the right-hand table specified in the ON condition and only those rows from the other table where the join condition is met.

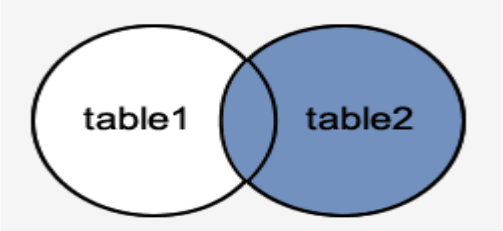
SELECT columns

FROM table1

RIGHT [OUTER] JOIN table2

ON table1.column = table2.column;

Image representation of Right Outer Join



Example

In this example, we are performing right outer join on the already created tables ?suppliers? and ?order1?.

The following example would return all rows from the order1 table and only those rows from the suppliers table where the join condition is met.

Execute this query

```
SELECT order1.order_number, order1.city, suppliers.supplier_name

FROM suppliers

RIGHT OUTER JOIN order1

ON suppliers.supplier_id = order1.supplier_id;
```

Output

ORDER_NUMBER	CITY	SUPPLIER_NAME
101	Allahabad	Bata shoes
102	Kanpur	Kingfisher
105	Ghaziabad	-

3 rows returned in 0.00 seconds

50	Apple	-
3	VOJO	-
21	Microsoft	-
21	ghaziabad	-
30	Google	-
28	Paris	-
22	Kingfisher	-
More than 10 rows available. Increase rows selector to view more rows.		

10 rows returned in 0.02 seconds

Full Outer Join

The Full Outer Join returns all rows from the left hand table and right hand table. It places NULL where the join condition is not met.

Syntax

SELECT columns

FROM table1

FULL [OUTER] JOIN table2

ON table1.**column** = table2.**column**;

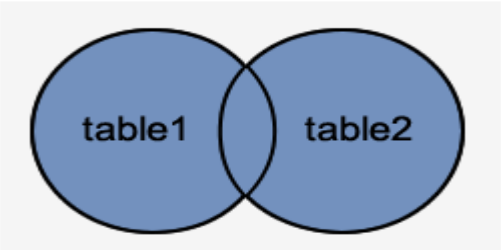


Image representation of Full Outer Join

Example

In this example, we are performing full outer join on the already created tables ?suppliers? and ?order1?.

The following example will return all rows from the ?suppliers? table and all rows from the ?order1? table and whenever the join condition is not met, it places the NULL value.

Execute this query

SELECT suppliers.supplier_id, suppliers.supplier_name, order1.order_number

FROM suppliers

FULL OUTER JOIN order1

ON suppliers.supplier_id = order1.supplier_id;

Output

SUPPLIER_ID	SUPPLIER_NAME	ORDER_NUMBER
1	Bata shoes	101
2	Kingfisher	102
50	Flipkart	-
50	Apple	-
3	VOJO	-
21	Microsoft	-
21	ghaziabad	-
30	Google	-
28	Paris	-
22	Kingfisher	-
More than 10 rows available. Increase rows selector to view more rows.		

10 rows returned in 0.01 seconds

Oracle EQUI JOIN

Oracle Equi join returns the matching column values of the associated tables. It uses a comparison operator in the WHERE clause to refer equality.

Syntax

SELECT column_list

FROM table1, table2....

WHERE table1.column_name =

table2.column_name;

Equijoin also can be performed by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.

Syntax

SELECT *

FROM table1

JOIN table2

[ON (join_condition)]

Oracle EQUI JOIN Example

Let' take two tables "agents" and "customer".

Agents tableAgent data

Column Name	Data Type	Nullable	Default	Primary Key
AGENT_ID	NUMBER	Yes	-	-
AGENT_NAME	VARCHAR2(4000)	Yes	-	-
AGENT_CITY	VARCHAR2(4000)	Yes	-	-

1 - 3

Customer data

EDIT	AGENT_ID	AGENT_NAME	AGENT_CITY
	3	Ajeet	Allahabad
	7	Bond	London
	11	Tejpratap	Patna

row(s) 1 - 3 of 3

Execute this query

SELECT agents.agent_city,customer.last_name,

customer.first_name

FROM agents,customer

WHERE agents.agent_id=customer.customer_id;

Output

EDIT	CUSTOMER_ID	FIRST_NAME	LAST_NAME
	1	aryan	tomar
	2	ajeet	maurya
	3	richa	goyal
	4	dhirubhai	jotwani

row(s) 1 - 4 of 4

Oracle SELF JOIN

Self Join is a specific type of Join. In Self Join, a table is joined with itself (Unary relationship). A self join simply specifies that each rows of a table is combined with itself and every other row of the table.

Syntax

SELECT a.column_name, b.column_name...

FROM table1 a, table1 b

AGENT_CITY	LAST_NAME	FIRST_NAME
Allahabad	goyal	richa

1 rows returned in 0.02 seconds

WHERE a.common_filed = b.common_field;

Oracle SELF JOIN Example

EDIT	NAME	AGE	ADDRESS	SALARY
	Alex	24	NewYork	25000
	Pandian	32	Chennai	32000
	Lalu	45	Bihar	56000
	Bholu	19	Haridwar	12000

row(s) 1 - 4 of 4

Let's take a table "customers".

Join this table using SELF JOIN as follows:

SELECT a.name, b.age, a.SALARY

FROM CUSTOMERS a, CUSTOMERS b

WHERE a.SALARY < b.SALARY;

Output

NAME	AGE	SALARY
Alex	32	25000
Alex	45	25000
Pandian	45	32000
Bholu	24	12000
Bholu	32	12000
Bholu	45	12000

6 rows returned in 0.02 seconds

Oracle Cross Join (Cartesian Products)

The CROSS JOIN specifies that all rows from first table join with all of the rows of second table. If there are "x" rows in table1 and "y" rows in table2 then the cross join result set have x*y rows. It normally happens when no matching join columns are specified.

In simple words you can say that if two tables in a join query have no join condition, then the Oracle returns their Cartesian product.

Syntax

SELECT *

FROM table1

CROSS JOIN table2;

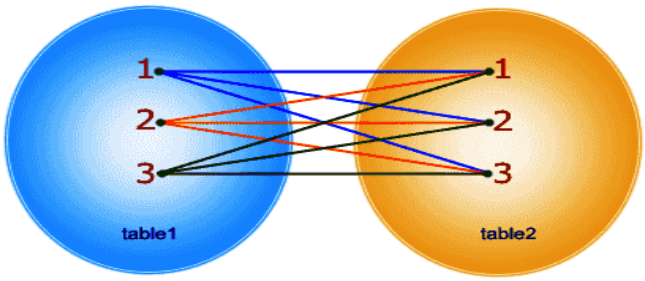
Or

SELECT * FROM table1, table2

Both the above syntax are same and used for Cartesian product.

They provide similar result after execution.

Image representation of cross join



Oracle Cross Join Example

Let's take two tables "customer" and "supplier".

Customer table detail

```
CREATE TABLE  "CUSTOMER"
  (    "CUSTOMER_ID" NUMBER,
   "FIRST_NAME" VARCHAR2(4000),
   "LAST_NAME" VARCHAR2(4000)
  )
```

EDIT	CUSTOMER_ID	FIRST_NAME	LAST_NAME
	1	aryan	tomar
	2	ajeet	maurya
	3	richa	goyal
	4	dhirubhai	jotwani

row(s) 1 - 4 of 4

Supplier table detail

```
CREATE TABLE  "SUPPLIER"
  (    "SUPPLIER_ID" NUMBER,
   "FIRST_NAME" VARCHAR2(4000),
   "LAST_NAME" VARCHAR2(4000)
  ) ;
```

EDIT	SUPPLIER_ID	FIRST_NAME	LAST_NAME
	1	dhirubhai	ambani
	2	kanchabhai	motwani
	3	lallu	lal
		row(s) 1 - 3 of 3	

Execute this query

SELECT * FROM customer,supplier

Output

CUSTOMER_ID	FIRST_NAME	LAST_NAME	SUPPLIER_ID	FIRST_NAME	LAST_NAME
1	aryan	tomar	1	dhirubhai	ambani
2	ajeet	maurya	1	dhirubhai	ambani
3	richa	goyal	1	dhirubhai	ambani
4	dhirubhai	jotwani	1	dhirubhai	ambani
1	aryan	tomar	2	kanchabhai	motwani
2	ajeet	maurya	2	kanchabhai	motwani
3	richa	goyal	2	kanchabhai	motwani
4	dhirubhai	jotwani	2	kanchabhai	motwani
1	aryan	tomar	3	lallu	lal
2	ajeet	maurya	3	lallu	lal
More than 10 rows available. Increase rows selector to view more rows.					

10 rows returned
in 0.06 seconds

Oracle Anti Join

Anti-join is used to make the queries run faster. It is a very powerful SQL construct Oracle offers for faster queries.

Anti-join between two tables returns rows from the first table where no matches are found in the second table. It is opposite of a semi-join. An anti-join returns one copy of each row in the first table for which no match is found.

Anti-joins are written using the NOT EXISTS or NOT IN constructs.

Example

Let's take two tables "departments" and "customer"

Departments table

```
CREATE TABLE  "DEPARTMENTS"
  (    "DEPARTMENT_ID" NUMBER(10,0) NOT NULL ENABLE,
   "DEPARTMENT_NAME" VARCHAR2(50) NOT NULL ENABLE,
    CONSTRAINT "DEPARTMENTS_PK" PRIMARY KEY ("DEPARTMENT_ID") ENABLE)
```

EDIT	DEPARTMENT_ID	DEPARTMENT_NAME
	1	a
	2	b
	3	c
	4	d

row(s) 1 - 4 of 4

Customer table

```
CREATE TABLE  "CUSTOMER"
  (    "CUSTOMER_ID" NUMBER,
   "FIRST_NAME" VARCHAR2(4000),
   "LAST_NAME" VARCHAR2(4000),
   "DEPARTMENT_ID" NUMBER
);
```

EDIT	CUSTOMER_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
	1	aryan	tomar	-
	2	ajeet	maurya	-
	3	richa	goyal	-
	4	dhirubhai	jotwani	-
	32	alex	pandian	1

row(s) 1 - 5 of 5

Execute this query

```
SELECT   departments.department_id, departments.department_name
```

```
FROM     departments
WHERE    NOT EXISTS
(
SELECT 1
 FROM   customer
 WHERE customer.department_id = departments.department_id
 )
 ORDER BY departments.department_id;
```

Output

DEPARTMENT_ID	DEPARTMENT_NAME
2	b
3	c
4	d

3 rows returned in 0.02 seconds

Oracle Semi Join

Semi-join is introduced in Oracle 8.0. It provides an efficient method of performing a WHERE EXISTS sub-query.

A semi-join returns one copy of each row in first table for which at least one match is found.

Semi-joins are written using the EXISTS construct.

EDIT	DEPARTMENT_ID	DEPARTMENT_NAME
	1	a
	2	b
	3	c
	4	d

row(s) 1 - 4 of 4

Oracle Semi Join Example

Let's take two tables "departments" and "customer"

Departments table

```
CREATE TABLE  "DEPARTMENTS"
  (    "DEPARTMENT_ID" NUMBER(10,0) NOT NULL ENABLE,
   "DEPARTMENT_NAME" VARCHAR2(50) NOT NULL ENABLE,
    CONSTRAINT "DEPARTMENTS_PK" PRIMARY KEY ("DEPARTMENT_ID") ENABLE)
```

Customer table

```
CREATE TABLE  "CUSTOMER"
  (    "CUSTOMER_ID" NUMBER,
   "FIRST_NAME" VARCHAR2(4000),
```

"LAST_NAME" VARCHAR2(4000),

"DEPARTMENT_ID" NUMBER)

EDIT	CUSTOMER_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
	1	aryan	tomar	-
	2	ajeet	maurya	-
	3	richa	goyal	-
	4	dhirubhai	jotwani	-
	32	alex	pandian	1

row(s) 1 - 5 of 5

Execute this query

```
SELECT   departments.department_id, departments.department_name

 FROM     departments

 WHERE    EXISTS

 (

 SELECT 1

 FROM   customer

 WHERE customer.department_id = departments.department_id

 ) ;

 ORDER BY departments.department_id;
```

Output

DEPARTMENT_ID	DEPARTMENT_NAME
1	a

1 rows returned in 0.02 seconds

Difference between anti-join and semi-join

While a semi-join returns one copy of each row in the first table for which at least one match is found, an anti-join returns one copy of each row in the first table for which no match is found.

SQL Constraints

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

INTEGRITY CONSTRAINTS

Integrity Constraints are used to prevent entry of invalid information into tables. There are five Integrity Constraints Available in Oracle. They are:

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Uniquely identifies a row/record in another table

CHECK - Ensures that all values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column when no value is specified

NOT NULL Constraint

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field always contain at least a single value

Example

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int
);
```

UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Example:

```
CREATE TABLE Persons (
ID NUMBER UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int
);
```

Note: UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created,

Example > ALTER TABLE Persons ADD UNIQUE (ID);

PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a database table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.

Example:

```
CREATE TABLE Persons (
ID NUMBER PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int
);
```

Note: PRIMARY KEY on ALTER TABLE

- To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following

Example > ALTER TABLE Persons ADD PRIMARY KEY (ID);

FOREIGN KEY Constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field in one table (Child table) that refers to the PRIMARY KEY in another table (Parent table).
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

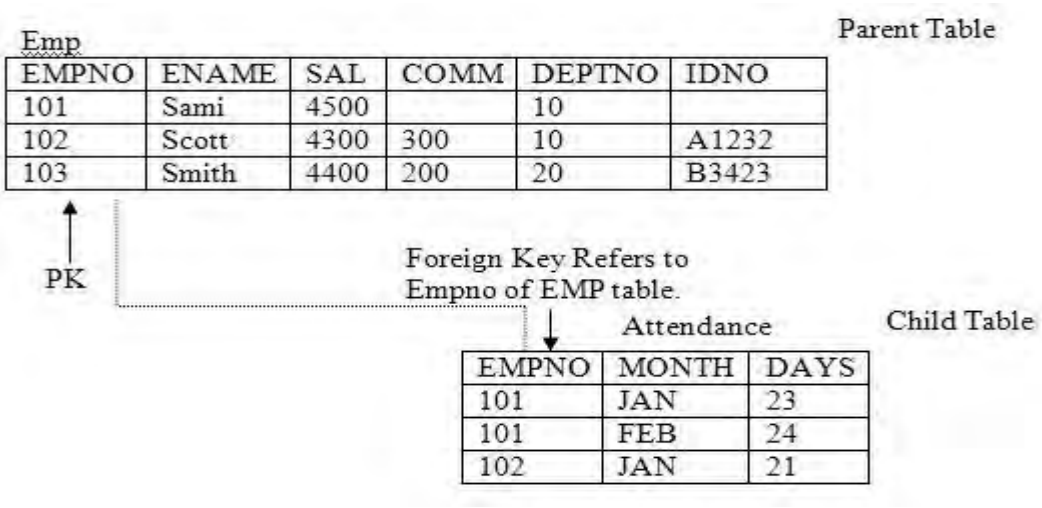
Example:

Creating parent table:

```
CREATE TABLE DEPT (
DEPTNO NUMBER PRIMARY KEY,
DNAME VARCHAR2(10),
LOC VARCHAR2(10)
);
```

Creating child table:

```
CREATE TABLE EMP (
EMPNO NUMBER PRIMARY KEY,
ENAME VARCHAR2(10),
DEPTNO NUMBER REFERENCES DEPT(DEPTNO)
);
```



CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a single column it allows only certain values for this column.

Example:

```
CREATE TABLE TEST (
EMPNO NUMBER,
SAL NUMBER CHECK (SAL>5000)
);
CREATE TABLE TEST (
EMPNO NUMBER,
AGE NUMBER CHECK (AGE>25),
DOB DATE
);
```

SQL DEFAULT Constraint

- The DEFAULT constraint is used to provide a default value for a column.
- The default value will be added to all new records if no other value is specified.

EXAMPLE:

CREATE TABLE Persons (

ID NUMBER NOT NULL,

City varchar (255) DEFAULT 'KOLKATA'

);

Enabling and Disabling Constraints

You can enable and disable constraints at any time. To enable and disable constraints the syntax is

ALTER TABLE <TABLE_NAME> ENABLE/DISABLE CONSTRAINT <CONSTRAINT_NAME>

For example to disable primary key of EMP table give the following statement

alter table emp disable constraint emppk;

And to enable it again, give the following statement

alter table emp enable constraint emppk;

Dropping constraints

You can drop constraint by using ALTER TABLE DROP constraint statement.For example to drop Unique constraint from emp table, give the following statement

alter table emp drop constraint id_unique;

To drop primary key constraint from emp table.

alter table emp drop constraint emppk;

The above statement will succeed only if the foreign key is first dropped otherwise you have to first drop the foreign key and then drop the primary key. If you want to drop primary key along with the foreign key in one statement then CASCADE CONSTRAINT statement like this:

alter table emp drop constraint emppk cascade;

Viewing Information about Constraints

To see information about constraints, you can query the following data dictionary tables.

select * from user_constraints;
select * from user_cons_columns;

Oracle Subqueries

A subquery is a query within another query, the subquery is used to return data that will be used in the main query. Subqueries can be used in various places within a query (such as: SELECT, FROM, WHERE), this tutorial explains how to use subqueries in the Oracle WHERE clause.

Using Oracle Subqueries

Subqueries are widely used to answer a question within another question. For example, which products cost more than product no. 54 ?
To retrieve this information, you need to answer two question, each in a separate query :

What is product no. 54's price?

```
SELECT price
FROM products
WHERE product_id = 54
-- Let's assume that this is the result :
 price
------
62
```

Which products cost more than product no. 54?

```
SELECT product_id, product_name, price
FROM products
WHERE price > 62
```

Instead of executing each query separately, you can combine the two queries, placing one query inside the other.

Basic Oracle Subquery Syntax

```
SELECT …
FROM table
WHERE condition (SELECT … FROM table)
```

Guidelines

```
    SELECT product_id , product_name , price
    FROM products
```

```
WHERE price > ( SELECT price
FROM products
WHERE product_id = 54)
```

The subquery is executed once before the main query, then the result returned by the subquery is submitted to the main query (in the Oracle example provided above, the subquery determines the price of product no. 54, then the main query takes the result of that subquery and uses the result to display all the products that cost more than this amount).

The subquery must be enclosed by round brackets.

Place subqueries on the right side of the comparison condition.

A subquery cannot be placed in the Oracle GROUP BY Clause.

Sub-queries can be divided into two main categories:

Single Row Subqueries – subqueries that return zero or one row to the outer SQL statement.

Multiple Row Subqueries – subqueries that return more than one row to the outer SQL statement.

Oracle Single Row Subquery

You may use comparison operators (also referred as single-row operators) in the outer query to handle a subquery that returns a single value. This Oracle example would retrieve the products whose category number is the same as that of product 64.

```
SELECT product_name , product_id , price , category_id
FROM products
WHERE category_id = (   SELECT category_id
FROM products
WHERE product_id = 64)
```

- The subquery must return a single row; a subquery written without a Oracle WHERE Clause (hence usually returns more than one row) will generate an error.
- The subquery must return a single column; specifying more than one column in the subquery's SELECT clause will result in an error.
- If you need to display all products whose category number is the same as that of product 54, excluding product 54, simply add this condition
- AND product_id <> 54

```
SELECT product_name , product_id , price , category_id
FROM products
WHERE category_id =  (SELECT category_id
FROM products
    WHERE product_id = 54)
AND product_id <> 54
```

- You can use group functions in a subquery to return a single row. For example: retrieve all products that cost more than the average price in category no. 60:

```
SELECT product_id , product_name , price
FROM products
WHERE price > ( SELECT AVG(price)
        FROM products
        WHERE category_id = 60)
```

- A subquery also can be used in the Oracle HAVING Clause. This Oracle example retrieves a summary of the average price for each category, for all categories whose average price is greater than the average price of category no. 90:
- The next Oracle example would retrieve all products that cost more than product no. 54 and their category number equals the category number of product no. 42, not including product no. 42. This Oracle example consists of three queries,

```
SELECT category_id , AVG(price)
FROM products
GROUP BY category_id
HAVING AVG(price)  > (SELECT AVG(price)
            FROM products
            WHERE category_id = 90)
```

main query and two subqueries. The subqueries are executed first, generating the query results. Then the main query is processed and uses the values returned by these subqueries.

```
SELECT product_id , product_name , price
FROM products
WHERE price > (SELECT price
               FROM products
               WHERE product_id = 54 )
AND category_id = (SELECT category_id
                   FROM products
                   WHERE product_id = 42)
AND product_id <> 42
```

Oracle Multiple Row Subquery

- You may use the IN, ANY, or ALL operators (multiple row operators) in the outer query to handle a subquery that returns multiple rows, the multiple row operators expect one or more values.
- The column below represents the prices of different products in category number 80, the following examples will use these values as the multiple row subquery result.

```
SELECT price
FROM products
WHERE category_id = 80
```

price
4300
5200
6700
8200
12500

Oracle IN Operator

- The Oracle IN operator allows comparing a column with a list of values returned from the subquery. This Oracle example would retrieve all products whose price is equal to one of the prices of products in category 80:
- In fact, the main query would look like the following to the Database Server :

```
SELECT product_id , product_name , price
FROM products
WHERE price IN (SELECT price
        FROM products
        WHERE category_id = 80)
```

```
SELECT product_id , product_name , price
FROM products
WHERE price IN (4300,5200,6700,8200,12500)
```

- To display all products whose price is equal to one of the prices of products in category 80, excluding the products in category 80, simply use this condition : AND category_id <> 80.

```
SELECT product_id , product_name , price
FROM products
WHERE price IN (SELECT price
        FROM products
        WHERE category_id = 80)
AND category_id <> 80
```

ANY Operator

- The Oracle ANY operator allows comparing a column with at least one of the values returned from the subequry. When using this operator, it is possible to work with the following comparison methods : >ANY, <ANY, =ANY
- ANY
- The following Oracle example would retrieve all products whose price is greater than at least one of the prices of the products in category 80.
- When seeking to know which value is greater than at least one of the values in a specific list, you actually seek to find the value that is greater than the minimum (because the requested value must be greater than at least one of the values in the list, no matter which value).

```
SELECT product_id , product_name , price
FROM products
WHERE price > ANY (SELECT price
                   FROM products
                   WHERE category_id = 80)
```

ANY

- The following Oracle example would retrieve all products whose price is lower than at least one of the prices of the products in category 80.
- When seeking to know which value is less than at least one of the values in a specific list, you actually seek to find the value that is less than the maximum (because the requested value must be less than at least one of the values in the list, no matter which value).

```
SELECT product_id , product_name , price
FROM products
WHERE price < ANY (SELECT price
FROM products
WHERE category_id = 80)
```

- The following Oracle example would retrieve all products whose price is equal to at least one of the prices of products in category 80. =ANY is equivalent to IN

```
SELECT product_id , product_name , price
FROM products
WHERE price = ANY (SELECT price
FROM products
WHERE category_id = 80)
```

ALL Operator

- The Oracle ALL operator allows comparing a column with all of the values returned from the subquery. When using this operator, it is possible to work with the following comparison methods: < ALL, >ALL.

ALL

- The followng Oracle example would retrieve all products whose price is greater than the prices of all products in category 80.

- When seeking to find the value that is greater than all values in a certain list, you actually look for a value that is greater than the maximum (for a value to be greater than all of the values, it must necessarily be greater than the maximum value in the list of values).

```
SELECT product_id , product_name , price
FROM products
WHERE price > ALL (SELECT price
FROM products
WHERE category_id = 80)
ALL
```

The following Oracle example would retrieve all products whose price is lower than all of the prices of all products in category 80.

When seeking to find the value that is less than all values in a certain list, you actually look for a value that is less than the minimum (for a value to be less than all of the values, it must necessarily be less than the minimum value in the list of values).

```
SELECT product_id , product_name , price
FROM products
WHERE price < ALL (SELECT price
FROM products
WHERE category_id = 80)
```

The Oracle = ALL operator attempts to retrieve a value that equals to all of the values returned from the subquery (the product whose price is worth 4300, and also 5200, and also 6700, and also 8200 and also 12500). This condition seeks to carry out an illogical operation, and will therefore usually not be used.

Check your Understanding:

Question 1.

When two or more Oracle Select statements are combined with the ____ operator, the result sets are combined.

a. UNION
b. UNITE
c. UNIQUE
d. None

Question 2.

Oracle lets you combine two ____ statements and remove duplicate rows between them using the UNION operator.

a. SELECT
b. DELETE
c. UPDATE
d. ALTER

Question 3.

As part of a UNION operation, individual SELECT statements must have ___ fields and similar types of data in their result sets.

a. Duplicate
b. Triplate
c. Identical
d. Multiple

Question 4.

In the case of UNION Operations, there must be the ___ number of expressions in both statements of the SELECT statement.

a. Different
b. Multiple
c. Same
d. None

Question 5.

After the UNION, the result set would only contain a ____ instance of it because Oracle removes duplicates.

a. Single
b. Double
c. Triple
d. Multiple

Question 6.

You can use the Oracle ____ operator if you don't want to remove duplicate data.

a. UNION NONE
b. UNION ALL
c. UNION ONE
d. UNION MULTIPLE

Question 7.

The ORDER BY clause can be used to order the results of a query using the Oracle ____ operator.

a. UNION
b. UNION NONE
c. UNION ONE
d. None

Question 8.

UNION ALL combines the results from multiple ____ statements in Oracle.

a. INSERT
b. SELECT
c. UPDATE
d. CREATE

Question 9.

By contrast, UNION ALL does not ____ duplicate rows between several SELECT statements, which is what sets it apart from the UNION operator.

a. Add
b. Update
c. Remove
d. None

Question 10.

In the results sets of the SELECT statements inside the UNION ALL, the result sets must contain fields of.

a. Similar
b. Different
c. Multiple
d. None

THE END

Great! Practice Regularly.