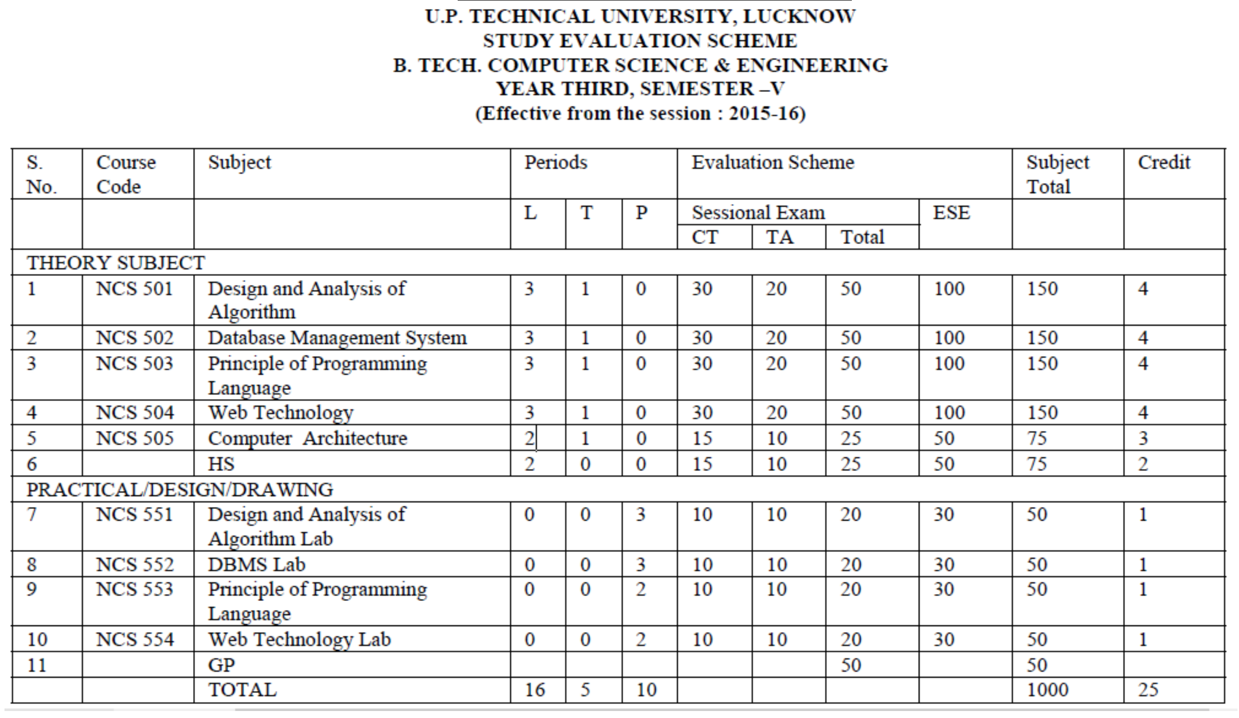
Database Management System (DBMS)
NCS-552
Lab Manual

Prepared by:
Dr. Avdhesh Gupta
Mr. Vivek Jain

Department of Computer Science & Engineering
IMS Engineering College, Ghaziabad

Evaluation Scheme



U.P. TECHNICAL UNIVERSITY, LUCKNOW
STUDY EVALUATION SCHEME
B. TECH. COMPUTER SCIENCE & ENGINEERING
YEAR THIRD, SEMESTER –V
(Effective from the session : 2015-16)

S. No.	Course Code	Subject	Periods			Evaluation Scheme				Subject Total	Credit
			L	T	P	Sessional Exam			ESE		
						CT	TA	Total			
THEORY SUBJECT											
1	NCS 501	Design and Analysis of Algorithm	3	1	0	30	20	50	100	150	4
2	NCS 502	Database Management System	3	1	0	30	20	50	100	150	4
3	NCS 503	Principle of Programming Language	3	1	0	30	20	50	100	150	4
4	NCS 504	Web Technology	3	1	0	30	20	50	100	150	4
5	NCS 505	Computer Architecture	2	1	0	15	10	25	50	75	3
6		HS	2	0	0	15	10	25	50	75	2
PRACTICAL/DESIGN/DRAWING											
7	NCS 551	Design and Analysis of Algorithm Lab	0	0	3	10	10	20	30	50	1
8	NCS 552	DBMS Lab	0	0	3	10	10	20	30	50	1
9	NCS 553	Principle of Programming Language	0	0	2	10	10	20	30	50	1
10	NCS 554	Web Technology Lab	0	0	2	10	10	20	30	50	1
11		GP						50		50	
		TOTAL	16	5	10					1000	25

NCS 552 DBMS Lab

Objectives:-

1. Installing oracle.
2. Creating Entity-Relationship Diagram using case tools.
3. Writing SQL statements Using ORACLE /MYSQL:
    a. Writing basic SQL SELECT statements.
    b. Restricting and sorting data.
    c. Displaying data from multiple tables.
    d. Aggregating data using group function.
    e. Manipulating data.
    f. Creating and managing tables.
4. Normalization in ORACLE.
5. Creating cursor in oracle.
6. Creating procedure and functions in oracle.
7. Creating packages & triggers in oracle

Content Beyond Syllabus:

1. Index / Sequences
2. PL/SQL
3. Few Syntaxes of Database Administration (ORACLE)
4. MS-Access

Applications:

1. A database application is a computer program whose primary purpose is entering and retrieving information from a computerized database. Relational database management systems (RDMS) will typically provide a series of tools for creating tables, conducting searches, producing printed reports, With a complicated database, however, it is usual for a database application to be written. A database application is a usually a program within a program, it is a program that runs inside the RDMS. Most, if not all RDMSs, provide an `application development language.' This will allow a computer programmer to create an application to perform specific tasks for a particular database, most commonly to provide a simpler and more efficient method of inputting data to the database, and for checking for errors. Often this will use a series of forms with menus and buttons.

IMS ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Faculty Name	:	Dr. Avdhesh Gupta, Mr. Vivek Jain
Subject with Code	:	NCS-552 (DBMS Lab)
Year & Semester	:	3rd year 5th Semester

Course Outcomes as per the University syllabus:

Course Name: DBMS Lab
Year of Study: 5th Semester (3rd Year)

NCS-552	**<Statement>**
NCS-552.1	Be familiar with Installation of oracle
NCS-552.2	Gain knowledge about how to prepare Entity-Relationship Diagram (Using Tool)
NCS-552.3	Gain knowledge about how to Write SQL statements Using ORACLE /MYSQL (DDL, DML and DCL)
NCS-552.4	Be familiar with Normalization in ORACLE (Integrity Constraints)
NCS-552.5	Gain knowledge about Creating cursor in oracle, Creating procedure and functions in oracle, Creating packages and triggers in oracle
NCS-552.6	Knowledge of PL/SQL

Mapping with POs

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1											1
CO2	2	2	3	3	2				1			3
CO3	3	3	3	3					1			3
CO4	3	3	3	3					1			3
CO5	2	2	2	2					1		1	3
CO6	2	2	2	2					1		1	3
Content Beyond Syllabus	2	2	1	1	2	2	2	2	2	2	2	2

Mapping with PSOs

Course Outcome	PSO1	PSO2	PSO3	PSO4
CO1	1			1
CO2	2	2		2
CO3	3	3		3
CO4	2	2	1	3
CO5	3	3		3
CO6	2	2	1	2
Content beyond	2	2	2	2

IMS Engineering College, Ghaziabad
Department of Computer Science & Engineering
Session 2016-17

Subject Name: DBMS Lab
Subject Code: NCS-552
Year and Branch: 3rd year

S. No.	Name	Outcomes	CO Mapped	CO-PO Mapping	CO-PSO Mapping
1	Introduction to DBMS: ORACLE, SQL, DB2 ORACLE/SQL Installation	Able to understand the ORACLE / SQL Installation	CO1	1,2,12	1,4
2	SQL Queries: Create, Insert, Select	Understanding of DDL syntaxes	CO3	1,2,3,12	1,2
3	Update, Delete, Modify, where clause with some aggregate functions like avg, sum, count, max, min	Understanding of DML syntaxes and able to understand use of aggregate functions	CO3	1,2,3,4,11 ,12	1,2
4	Primary key, Foreign Key constraints and other constraints	Able to understand the concept of RDBMS (Entity relationships)	CO4	1,2,3,4,6, 11,12	1,2
5	Alter, Drop table, Date Function	Able to understand to modify entity structure	CO3	1,2,3,12	1,2
6	ERD Tool like Dia	Learning of ERD tool	CO2	1,2,3,5,9, 11,12	1,2,3,4
7	Join Queries and its types	Able to understand to fetch the required data from more than one entity/table	CO4	1,2,3,4,5, 6,9,11,12	1,2,4
8	Sub Queries	Able to understand to execute nested queries	CO4	1,2,3,4,12	1,2
9	Index, Sequence	Able to understand to speedup data retrieval by use of index and Auto increment a field by use of sequences	CO5	1,2,3,12	1,2
10	Cursor and its types	Able to understand the memory area used by ORACLE	CO5	1,2,3,4,6, 12	1,2
11	Triggers and its types	Able to understand how to execute a stored program automatically when some event occur	CO5	1,2,3,4,6, 8,9,11,12	1,2,4
12	PL/SQL	Able to understand programming in SQL	CO6	1,5,11,12	1,2,4
13	**Admin and user privileges**	**Understanding of Administration of database handling and Admin privileges**		**1,2,3,5,6, 8,9,12**	**1,2,3,4**
14	**MS. Access**	**Learn to building automated queries**		**1,2,3,5,11 ,12**	**1,2,3,4**
15	**Mini Project**	**Learn to develop mini project in group**		**1,2,3,4,5, 6,7,8,9,10 ,11,12**	**1,2,3,4**

Program No: 1

TITLE: ORACLE Installation

1.1 Objective 1.2 Installation

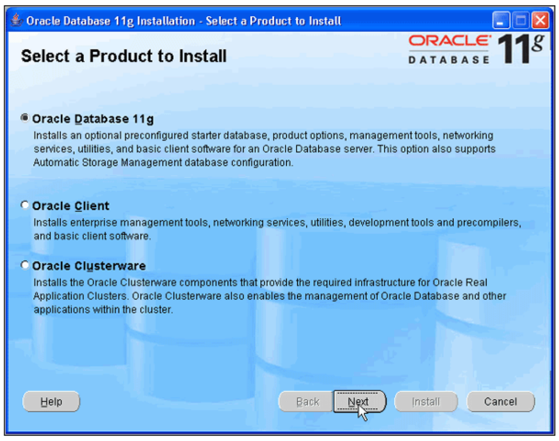
1.1 OBJECTIVE: learn the procedure of ORACLE installation

1.2 ORACLE INSTALLATION:

Installing Oracle Database 11*g* on Windows

To install the Oracle software, you must use the Oracle Universal installer.

1. For this installation, you need either the DVDs or a downloaded version of the DVDs. In this tutorial, you install from the downloaded version. From the directory where the DVD files were unzipped, open Windows Explorer and double-click on setup.exe from the \db\Disk1 directory.
2. The product you want to install is Oracle Database 11g. Make sure the product is selected and click Next.



Oracle Database 11g Installation - Select a Product to Install

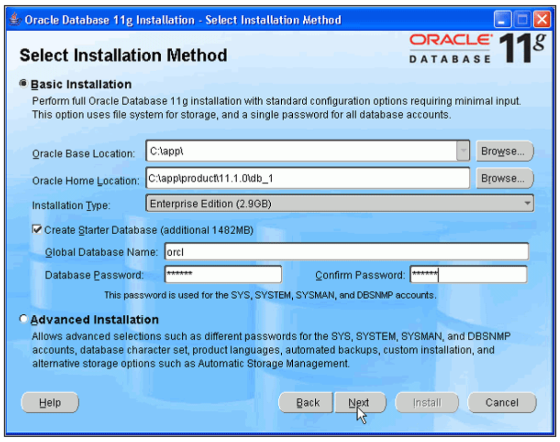
Select a Product to Install

[✓ Oracle Database 11g]
Installs an optional preconfigured starter database, product options, management tools, networking services, utilities, and basic client software for an Oracle Database server. This option also supports Automatic Storage Management database configuration.

Oracle Client
Installs enterprise management tools, networking services, utilities, development tools and precompilers, and basic client software.

Oracle Clusterware
Installs the Oracle Clusterware components that provide the required infrastructure for Oracle Real Application Clusters. Oracle Clusterware also enables the management of Oracle Database and other applications within the cluster.

3. You will perform a basic installation with a starter database. Enter **orcl** for the Global Database Name and **oracle** for Database Password and Confirm Password. Then, click **Next**



Oracle Database 11g Installation - Select Installation Method

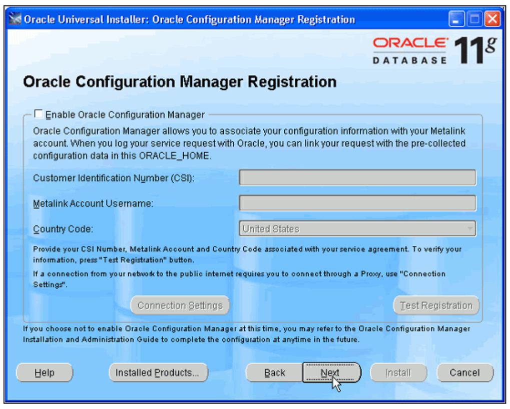
Select Installation Method

[✓ Basic Installation]
Perform full Oracle Database 11g installation with standard configuration options requiring minimal input. This option uses file system for storage, and a single password for all database accounts.

Oracle Base Location: C:\app\
Oracle Home Location: C:\app\product\11.1.0\db_1
Installation Type: Enterprise Edition (2.9GB)
[✓] Create Starter Database (additional 1482MB)
Global Database Name: orcl
Database Password: ****** Confirm Password: ******
This password is used for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts.

Advanced Installation
Allows advanced selections such as different passwords for the SYS, SYSTEM, SYSMAN, and DBSNMP accounts, database character set, product languages, automated backups, custom installation, and alternative storage options such as Automatic Storage Management.

4. Oracle Configuration Manager allows you to associate your configuration information with your Metalink account. You can choose to enable it on this window. Then, click Next.


Oracle Universal Installer: Oracle Configuration Manager Registration

Oracle Configuration Manager Registration

☐ Enable Oracle Configuration Manager

Oracle Configuration Manager allows you to associate your configuration information with your Metalink account. When you log your service request with Oracle, you can link your request with the pre-collected configuration data in this ORACLE_HOME.

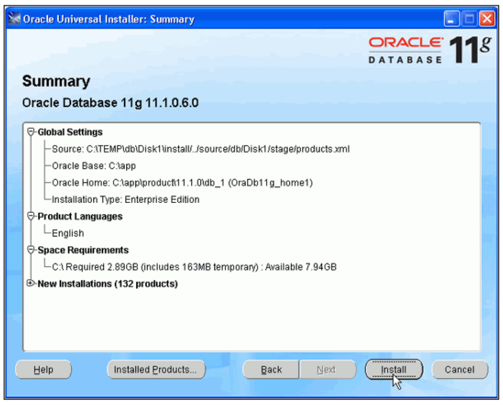
Customer Identification Number (CSI):	
Metalink Account Username:	
Country Code:	United States

Provide your CSI Number, Metalink Account and Country Code associated with your service agreement. To verify your information, press "Test Registration" button.

If a connection from your network to the public internet requires you to connect through a Proxy, use "Connection Settings".

If you choose not to enable Oracle Configuration Manager at this time, you may refer to the Oracle Configuration Manager Installation and Administration Guide to complete the configuration at anytime in the future.

5. Review the Summary window to verify what is to be installed. Then, click **Install**.

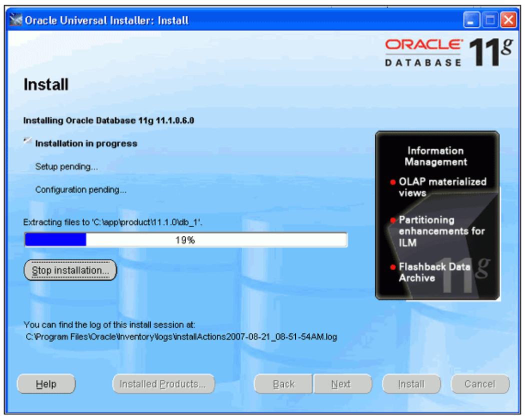

Oracle Universal Installer: Summary

Summary

Oracle Database 11g 11.1.0.6.0

```
Global Settings
  Source: C:\TEMP\db\Disk1\install/../source/db/Disk1/stage/products.xml
  Oracle Base: C:\app
  Oracle Home: C:\app\product\11.1.0\db_1 (OraDb11g_home1)
  Installation Type: Enterprise Edition
Product Languages
  English
Space Requirements
  C:\ Required 2.89GB (includes 163MB temporary) : Available 7.94GB
New Installations (132 products)
```

6. The progress window appears.


Oracle Universal Installer: Install

Install

Installing Oracle Database 11g 11.1.0.6.0

Installation in progress

Setup pending...

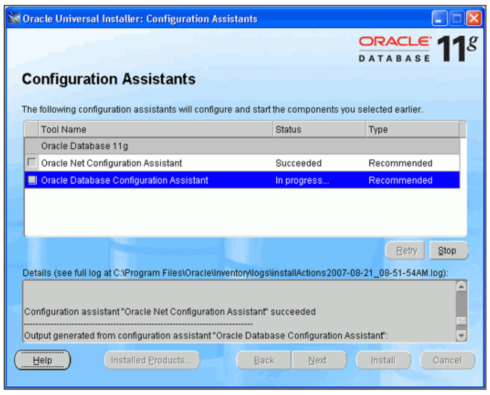
Configuration pending...

Extracting files to 'C:\app\product\11.1.0\db_1'.

19%

You can find the log of this install session at:
C:\Program Files\Oracle\Inventory\logs\installActions2007-08-21_08-51-54AM.log

7. The Configuration Assistants window appears.


Oracle Universal Installer: Configuration Assistants

Configuration Assistants

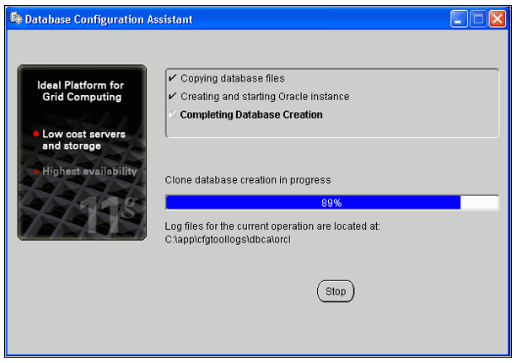
The following configuration assistants will configure and start the components you selected earlier.

Tool Name	Status	Type
Oracle Database 11g		
Oracle Net Configuration Assistant	Succeeded	Recommended
Oracle Database Configuration Assistant	In progress...	Recommended

Details (see full log at C:\Program Files\Oracle\Inventory\logs\installActions2007-08-21_08-51-54AM.log):

```
Configuration assistant "Oracle Net Configuration Assistant" succeeded
---------------------------------------------------------------------
Output generated from configuration assistant "Oracle Database Configuration Assistant":
```

8. Your database is now being created.


Database Configuration Assistant
Ideal Platform for Grid Computing
Low cost servers and storage
Highest availability
11g
Stop

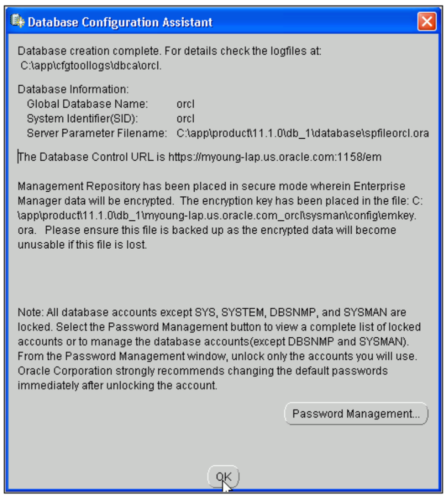
✔ Copying database files
✔ Creating and starting Oracle instance
Completing Database Creation

Clone database creation in progress

89%

Log files for the current operation are located at:
C:\app\cfgtoollogs\dbca\orcl

9. When the database has been created, you can unlock the users you want to use. Click **OK**.



Database creation complete. For details check the logfiles at:
C:\app\cfgtoollogs\dbca\orcl.

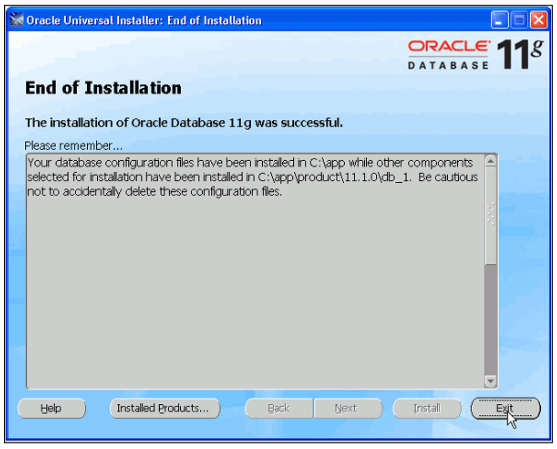
Database Information:
Global Database Name: orcl
System Identifier(SID): orcl
Server Parameter Filename: C:\app\product\11.1.0\db_1\database\spfileorcl.ora

The Database Control URL is https://myoung-lap.us.oracle.com:1158/em

Management Repository has been placed in secure mode wherein Enterprise Manager data will be encrypted. The encryption key has been placed in the file: C:\app\product\11.1.0\db_1\myoung-lap.us.oracle.com_orcl\sysman\config\emkey.ora. Please ensure this file is backed up as the encrypted data will become unusable if this file is lost.

Note: All database accounts except SYS, SYSTEM, DBSNMP, and SYSMAN are locked. Select the Password Management button to view a complete list of locked accounts or to manage the database accounts(except DBSNMP and SYSMAN). From the Password Management window, unlock only the accounts you will use. Oracle Corporation strongly recommends changing the default passwords immediately after unlocking the account.

10. Click Exit. Click **Yes** to confirm exit.



End of Installation

The installation of Oracle Database 11g was successful.

Please remember...

Your database configuration files have been installed in C:\app while other components selected for installation have been installed in C:\app\product\11.1.0\db_1. Be cautious not to accidentally delete these configuration files.

Testing Your Installation

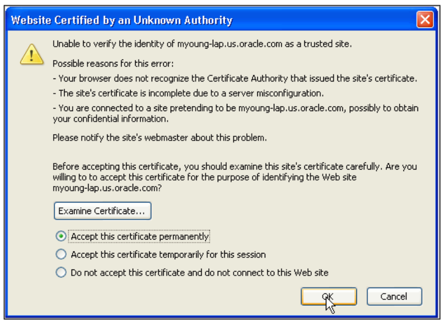
To test that your installation completed successfully, perform the following steps:

1. Open a browser and enter the following URL:

   https://<hostname>:1158/em

where <hostname> should be changed to your machine name, IP address, or localhost.

Because Enterprise Manager Database Control is a secure site, you need a certificate. Select the Accept this certificate permanently option, and then click OK.


Website Certified by an Unknown Authority

Unable to verify the identity of myoung-lap.us.oracle.com as a trusted site.

Possible reasons for this error:

- Your browser does not recognize the Certificate Authority that issued the site's certificate.
- The site's certificate is incomplete due to a server misconfiguration.
- You are connected to a site pretending to be myoung-lap.us.oracle.com, possibly to obtain your confidential information.

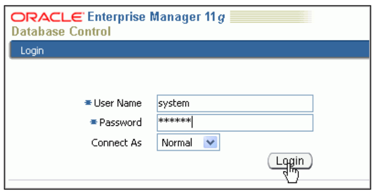
Please notify the site's webmaster about this problem.

Before accepting this certificate, you should examine this site's certificate carefully. Are you willing to to accept this certificate for the purpose of identifying the Web site myoung-lap.us.oracle.com?

Examine Certificate...

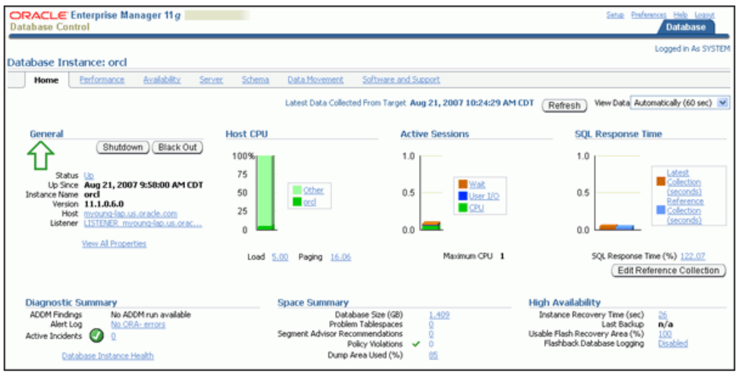
[✓ Accept this certificate permanently]
Accept this certificate temporarily for this session
Do not accept this certificate and do not connect to this Web site

2. Enter **system** as the User Name and **oracle** as the Password, and then click **Login**



* User Name: system
* Password: ******
Connect As: Normal

3. The Database Control Home Page appears. Your installation was successful.



Database Instance: orcl

Latest Data Collected From Target Aug 21, 2007 10:24:29 AM CDT

View Data Automatically (60 sec)

General

Status Up
Up Since Aug 21, 2007 9:58:00 AM CDT
Instance Name orcl
Version 11.1.0.6.0
Host myoung-lap.us.oracle.com
Listener LISTENER_myoung-lap.us.orac...

View All Properties

Host CPU

Load 5.00 Paging 16.06

Active Sessions

Maximum CPU 1

SQL Response Time

SQL Response Time (%) 122.07

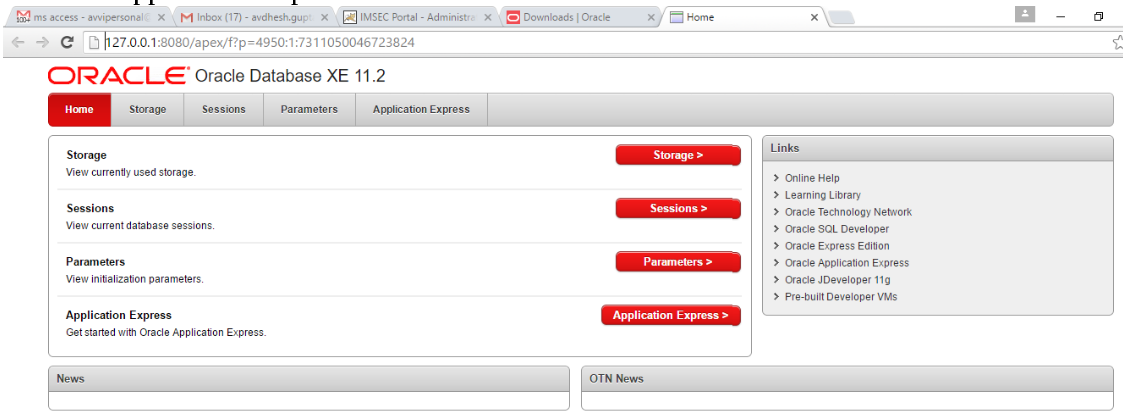
Diagnostic Summary
ADDM Findings No ADDM run available
Alert Log No ORA- errors
Active Incidents ✓ 0
Database Instance Health

Space Summary
Database Size (GB) 1.409
Problem Tablespaces 0
Segment Advisor Recommendations 0
Policy Violations ✓ 0
Dump Area Used (%) 85

High Availability
Instance Recovery Time (sec) 26
Last Backup n/a
Usable Flash Recovery Area (%) 100
Flashback Database Logging Disabled

Another way to work on ORACLE:

1. Goto https://www.oracle.com/downloads/index.html
2. Create your login and download latest Express Edition of ORACE and install in simple way
3. Run the ORACLE, It will open in Web Browser
4. Click on Application Express


ms access - avvipersonal@
Inbox (17) - avdhesh.gupt
IMSEC Portal - Administra
Downloads | Oracle
Home

127.0.0.1:8080/apex/f?p=4950:1:7311050046723824

Storage
View currently used storage.

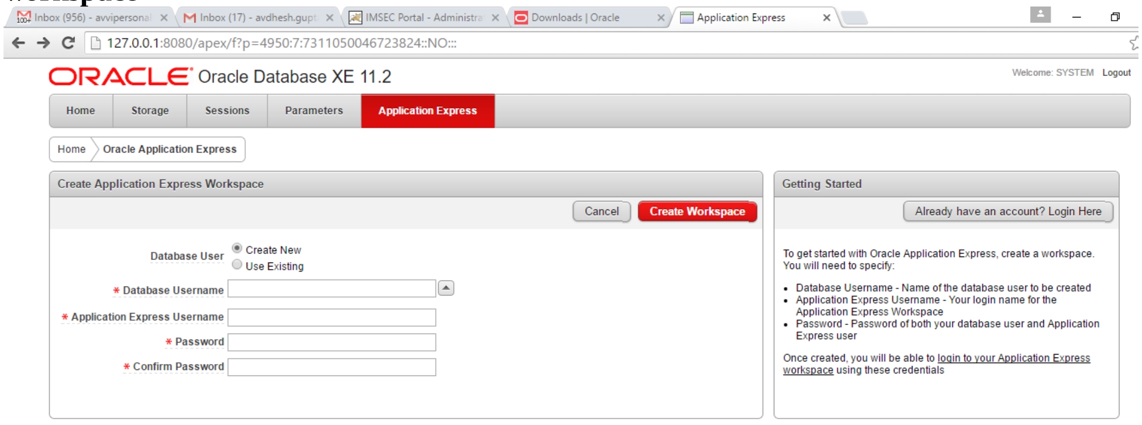
Sessions
View current database sessions.

Parameters
View initialization parameters.

Application Express
Get started with Oracle Application Express.

Links
- Online Help
- Learning Library
- Oracle Technology Network
- Oracle SQL Developer
- Oracle Express Edition
- Oracle Application Express
- Oracle JDeveloper 11g
- Pre-built Developer VMs

5. Login again and then create a new database user by filling the below entries and click on **creating workspace**


Inbox (956) - avvipersonal
Inbox (17) - avdhesh.gupt
IMSEC Portal - Administra
Downloads | Oracle
Application Express
127.0.0.1:8080/apex/f?p=4950:7:7311050046723824::NO:::
ORACLE Oracle Database XE 11.2
Welcome: SYSTEM Logout
Home
Storage
Sessions
Parameters
Application Express
Home
Oracle Application Express
Cancel
Create Workspace
Getting Started
Already have an account? Login Here

Create Application Express Workspace

Database User	(● Create New) Use Existing
* Database Username	
* Application Express Username	
* Password	
* Confirm Password	

To get started with Oracle Application Express, create a workspace. You will need to specify:

- Database Username - Name of the database user to be created
- Application Express Username - Your login name for the Application Express Workspace
- Password - Password of both your database user and Application Express user

Once created, you will be able to login to your Application Express workspace using these credentials

6. Login by using your created username and password and start using ORACLE

Program No: 2

TITLE : CREATING TABLES, INSERT INTO TABLES & SELECT FROM TABLES

2.1 Objective 2.2 Theory 2.3 Assignment 2.4. Solutions

2.1 OBJECTIVE: Create tables and specify the queries in SQL

2.2 THEORY & CONCEPTS:

Introduction about SQL-

SQL (Structured Query Language) is a nonprocedural language, you specify what you want, not how to get it. A block structured format of English key words is used in this query language. It has the following components.

DDL (Data Definition Language)-

The SQL DDL provides command for defining relation schemas, deleting relations and modifying relation schema.

DML (DATA Manipulation Language)-

It includes commands to insert tuples into, delete tuples from and modify tuples in the database.

View definition-

The SQL DDL includes commands for defining views.
Transaction Control- SQL includes for specifying the beginning and ending of transactions.

Embedded SQL and Dynamic SQL-

Embedded and Dynamic SQL define how SQL statements can be embedded with in general purpose programming languages, such as C, C++, JAVA, COBOL, Pascal and Fortran.

Integrity-

The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must specify. Updates that violate integrity constraints are allowed.

Authorization-

The SQL DDL includes commands for specifying access rights to relations and views.

Data Definition Language-

The SQL DDL allows specification of not only a set of relations but also information about each relation, including-

- Schema for each relation
- The domain of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk.

<u>Domain types in SQL-</u>

The SQL standard supports a variety of built in domain types, including-

- Char (n)- A fixed length character length string with user specified length .
- Varchar (n)- A variable character length string with user specified maximum length n.
- Int- An integer.
- Small integer- A small integer.
- Numeric (p, d)-A Fixed point number with user defined precision.
- Real, double precision- Floating point and double precision floating point numbers with machine dependent precision.
- Float (n)- A floating point number, with precision of at least n digits.
- Date- A calendar date containing a (four digit) year, month and day of the month.
- Time- The time of day, in hours, minutes and seconds Eg. Time '09:30:00'.
- Number- Number is used to store numbers (fixed or floating point).

<u>DDL statement for creating a table-</u>

<u>Syntax-</u>

```
 Create table tablename
(columnname datatype(size),  columnname datatype(size));
```

<u>Creating a table from a table-</u>

<u>Syntax-</u>

```
 CREATE TABLE TABLENAME
[(columnname, columnname, ………)]
AS SELECT columnname, columnname……..FROM tablename;
```

<u>Insertion of data into tables-</u>

<u>Syntax-</u>

```
 INSERT INTO tablename
[(columnname, columnname, ………)]
Values(expression, expression);
```

<u>Inserting data into a table from another table:</u>

<u>Syntax-</u>

```
INSERT INTO tablename
SELECT columnname, columnname, …….
FROM tablename;
```

<u>Insertion of selected data into a table from another table:</u>

<u>Syntax-</u>

```
 INSERT INTO tablename
SELECT columnname, columnname……..
FROM tablename
WHERE columnname= expression;
```

Retrieving of data from the tables-

Syntax-
```
 SELECT * FROM tablename;
```

The retrieving of specific columns from a table-

Syntax-
```
 SELECT  columnname, columnname, ….
FROM tablename;
```

Elimination of duplicates from the select statement-

Syntax-
```
SELECT DISTINCT columnname, columnname
FROM tablename;
```

Selecting a data set from table data-

Syntax-
```
 SELECT columnname, columnname
FROM tablename
WHERE searchcondition;
```

2.3 ASSIGNMENT

Q1. Create the following tables:

i) client_master

columnname	datatype	size
client_no	varchar2	6
name	varchar2	20
address1	varchar2	30
address2	varchar2	30
city	varchar2	15
state	varchar2	15
pincode	number 6	
bal_due	number 10,2	

ii) Product_master

Columnname	datatype	size
Product_no	varchar2	6
Description	varchar2	20
Profit_percent	number	10,2
Unit_measure	varchar2	10
Qty_on_hand	number	10
Reoder_lvl	number	10
Sell_price	number	10
Cost_price	number	10

Q2- Insert the following data into their respective tables:

Clientno	Name	city	pincode	state	bal.due
0001	Ivan	Bombay	400054	Maharashtra	15000
0002	Vandana	Madras	780001	Tamilnadu	0
0003	Pramada	Bombay	400057	Maharashtra	5000
0004	Basu	Bombay	400056	Maharashtra	0
0005	Ravi	Delhi	100001	Delhi	2000
0006	Rukmini	Bombay	400050	Maharashtra	0

Data for Product Master:

Product No.	Desciption	Profit % Percent	Unit measured	Qty on hand	Reorder lvl	Sell price	Cost price
P00001	1.44floppies	5	piece	100	20	525	500
P03453	Monitors	6	piece	10	3	12000	11200
P06734	Mouse	5	piece	20	5	1050	500
P07865	1.22 floppies	5	piece	100	20	525	500
P07868	Keyboards	2	piece	10	3	3150	3050
P07885	CD Drive	2.5	piece	10	3	5250	5100
P07965	540 HDD	4	piece	10	3	8400	8000
P07975	1.44 Drive	5	piece	10	3	1050	1000
P08865	1.22 Drive	5	piece	2	3	1050	1000

Q3:- On the basis of above two tables answer the following queries:

i) Find out the names of all the clients.
ii) Retrieve the list of names and cities of all the clients.
iii) List the various products available from the product_master table.
iv) List all the clients who are located in Bombay.
v) Display the information for client no 0001 and 0002.
vi) Find the products with description as ‘1.44 drive’ and ‘1.22 Drive’.
vii) Find all the products whose sell price is greater then 5000.
viii) Find the list of all clients who stay in in city ‘Bombay’ or city ‘Delhi’ or ‘Madras’.
ix) Find the product whose selling price is greater than 2000 and less than or equal to 5000.
x) List the name, city and state of clients not in the state of ‘Maharashtra’.

2.4. SOLUTION TO ASSIGNMENT

Q.1. Create the following tables:

(i) create table client_master
(client_no varchar2(6),
name varchar2(20),
address1 varchar2(30),
address2 varchar2(30),
city varchar2(15),
state varchar2(15),
pincode number(6),
bal_due number(10,2));

(ii) create table product_master
(product_no varchar2(6),
Description varchar2(20),
Profit_percent number(10,2),
Unit_measure varchar2(10),
Qty_on_hand number(10),
Reorder_level_number number(10),
Sell_price number(10),
Cost_price_number number(10));

Q.2. Insert the following data into their respective tables:

(i) Insert into client_master
Values('0001','ivan','bombay','maharastra','201001','15000');

(ii) Insert into product_master
Values('P00001','1.44floppies','5','piece','100','20','525','500');

Q.3. Solutions

i. select name from client_master;
ii. select name, city from client_master;
iii. select description from product_master;
iv. select * from client_master where city = 'bombay';
v. select * from client_master where client_no = '0001' or client_no = ' '0002';
vi. select * from product_master where description='1.44 drive' or '1.22 drive';
vii. select * from product_master where sell_price > 5000;
viii. select * from client_master where city = 'bombay' or city = 'madras';
ix. select * from product_master where sell_price > 2000 and sell_price <= 5000;
x. select name, city , state from client_master where state not in ('maharastra');

Program No: 3

TITLE : DATA MANIPULATION LANGUAGE

3.1 Objective 3.2 Theory 3.3 Assignment 3.4. Solution to Assignment

3.1 OBJECTIVE: Perform manipulation operations on created tables.

3.2 THEORY AND CONCEPTS: DML (Data Manipulation Language) Data manipulation is

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of information stored by the appropriate data model. There are basically two types.

(i) **Procedural DML**:- require a user to specify what data are needed and how to get those data.

(ii) **Non Procedural DML** : require a user to specify what data are needed without specifying how to get those data.

Updating the content of a table:
In creation situation we may wish to change a value in table without changing all values in the tuple . For this purpose the update statement can be used.

Update table name
Set columnname = experision, columnname =expression……
Where columnname = expression;

Deletion Operation:-
A delete request is expressed in much the same way as query. We can delete whole tuple (rows) we can delete values on only particulars attributes.

<u>Deletion of all rows</u>

Syntax:
Delete from tablename :

<u>Deletion of specified number of rows</u>
Syntax:

Delete from table name
Where search condition ;

Computation in expression lists used to select data

+	Addition	- Subtraction
*	multiplication	** exponentiation
/	Division	() Enclosed operation

Renaming columns used with Expression Lists: - The default output column names can be renamed by the user if required

Syntax:

```
Select column name          result_columnname,
       Columnname           result_columnname,
```

From table name;

Logical Operators:
The logical operators that can be used in SQL sentenced are

AND	all of must be included
OR	any of may be included
NOT	none of could be included

Range Searching: Between operation is used for range searching.

Pattern Searching:
The most commonly used operation on string is pattern matching using the operation 'like' we describe patterns by using two special characters.

- Percent (%) ; the % character matches any substring we consider the following examples.
- 'Perry %' matches any string beginning with perry
- '% idge % matches any string containing' idge as substring.
- ' - - - ' matches any string exactly three characters.
- ' - - - % matches any string of at least of three characters.

Oracle functions:
Functions are used to manipulate data items and return result. function follow the format of function _name (argument1, argument2 ..) .An arrangement is user defined variable or constant. The structure of function is such that it accepts zero or more arguments.
Examples:
Avg return average value of n

Syntax:
Avg ([distinct/all]n)
Min return minimum value of expr.

Syntax:
MIN((distict/all)expr)
Count Returns the no of rows where expr is not null

Syntax:

Count ([distinct/all)expr]
Count (*) Returns the no rows in the table, including duplicates and those with nulls.
Max Return max value of expr

Syntax:

Max ([distinct/all]expr)
Sum Returns sum of values of n

Syntax:
Sum ([distinct/all]n)

<u>Sorting of data in table</u>
Syntax:

Select columnname, columnname
From table
Order by columnname;

3.3 ASSIGNMENT

Que.1 Using the table client master and product master answer the following queries.

i. Change the selling price of '1.44 floppy drive to Rs.1150.00
ii. Delete the record with client 0001 from the client master table.
iii. Change the city of client_no'0005' to Bombay.
iv. Change the bal_due of client_no '0001, to 1000.
v. Find the products whose selling price is more than 1500 and also find the new selling price as original selling price *15.
vi. Find out the clients who stay in a city whose second letter is a.
vii. Find out the name of all clients having 'a' as the second letter in their names.
viii. List the products in sorted order of their description.
ix. Count the total number of orders
x. Calculate the average price of all the products.
xi. Calculate the minimum price of products.
xii. Determine the maximum and minimum prices . Rename the tittle as 'max_price' and min_price respectively.
xiii. Count the number of products having price greater than or equal to 1500.

3.4 SOLUTION TO ASSIGNMENT:

i. update table client_master set sell_price = '1150' where description='1.44 drive';
ii. Delete from client_master where client_no = '0001';
iii. update table client_master set city='bombay' where client_no='0001';
iv. update table client_master set bal_due=1000 where client_no='0001';
v. select description from product_master where sell_price > 1500;
vi. select sell_price*15 "New Selling Price" from product_master;
vii. select * from client_master where city like '_a%';
viii. select name from client_master where name='_a%';
ix. select * from product_master order by description;
x. select count(*) from product_master;
xi. select avg(sell_price) from product_master;
xii. select min(sell_price) from product_master;
xiii. select max(sell_price) "Max_price" , min(sell_price) "Min_price";
xiv. select count(*) from product_master where sell_price >= 1500;

Program No: 4

TITLE: TO IMPLEMENT CONSTRAINTS ON TABLES.

4.1 Objective 4.2 Theory 4.3 Assignment 4.4. Solution to Assignment

4.1 OBJECTIVE: applying constraints on the columns

4.2 THEORY AND CONCEPTS:

Data constraints: Besides the cell name, cell length and cell data type there are other parameters i.e. other data constrains that can be passed to the DBA at check creation time. The constraints can either be placed at column level or at the table level.

- **i. Column Level Constraints:** If the constraints are defined along with the column definition, it is called a column level constraint.
- **ii. Table Level Constraints:** If the data constraint attached to a specify cell in a table reference the contents of another cell in the table then the user will have to use table level constraints.

Null Value Concepts:- while creating tables if a row locks a data value for particular column that value is said to be null . Column of any data types may contain null values unless the column was defined as not null when the table was created

Syntax:

Create table tablename
(columnname data type (size) not null ……)

Primary Key: primary key is one or more columns is a table used to uniquickly identity each row in the table. Primary key values must not be null and must be unique across the column. A multicolumn primary key is called composite primary key.

Syntax: <u>primary key as a column constraint</u>
Create table tablename
(columnname datatype (size) primary key,….)

<u>Primary key as a table constraint</u>
Create table tablename
(columnname datatype (size), columnname datatype(size)…
Primary key (columnname,columnname));

Unique key concept:-A unique is similar to a primary key except that the purpose of a unique key is to ensure that information in the column for each record is unique as with telephone or devices license numbers. A table may have many unique keys.

Syntax: <u>Unique as a column constraint</u>.
Create table table name
(columnname datatype (size) unique);

<u>Unique as table constraint:</u>
Create table tablename
(columnname datatype (size),columnname datatype (size)…unique (columnname,columnname));

Default value concept: At the line of cell creation a default value can be assigned to it. When the user is loading a record with values and leaves this cell empty, the DBA wil automatically load this cell with the default value specified. The data type of the default value should match the data type of the column

Syntax:

Create table tablename
(columnname datatype (size) default value,….);

Foreign Key Concept : Foreign key represents relationship between tables. A foreign key is column whose values are derived from the primary key of the same of some other table . the existence of foreign key implies that the table with foreign key is related to the primary key table from which the foreign key is derived .A foreign key must have corresponding primary key value in the primary key table to have meaning.
Foreign key as a column constraint

Syntax :
Create table table name
(columnname datatype (size) references another table name);

<u>Foreign key as a table constraint:</u>

Syntax :
Create table name
(columnname datatype (size)….
primary key (columnname);
foreign key (columnname)references table name);

Check Integrity Constraints: Use the check constraints when you need to enforce intergrity rules that can be evaluated based on a logical expression following are a few examples of appropriate check constraints.

- A check constraints name column of the coient_master so that the name is entered in upper case.
- A check constraint on the client_no column of the client _master so that no client_no value starts with 'c'

Syntax:
Create table tablename
(columnname datatype (size) CONSTRAINT constraintname)
Check (expression));

4.3 ASSIGNMENTS:
Que.1 Create the following tables:
i. Sales_master

Columnname	**Datatype**	**Size**	**Attributes**
Salesman_no	varchar2	6	Primary key/first letter must start with 'S'
Sal_name	varchar2	20	Not null
Address	varchar2		Not null
City	varchar2	20	
State	varchar2	20	
Pincode	Number	6	
Sal_amt	Number	8,2	Not null, cannot be 0
Tgt_to_get	Number	6,2	Not null, cannot be 0
Ytd_sales	Number	6,2	Not null, cannot be 0
Remarks	Varchar2	30	

ii. Sales_order

Columnname	**Datatype**	**Size**	**Attributes**
S_order_no	varchar2	6	Primary/first letter must be 0
S_order_date	Date	6	Primary key reference clientno of client_master table
Client_no	Varchar2	25	
Dely_add	Varchar2	6	
Salesman_no	Varchar2	6	Foreign key references salesman_no of salesman_master table
Dely_type	Char	1	Delivery part(p)/full(f),default f
Billed_yn	Char	1	
Dely_date	Date		Can not be lessthan s_order_date
Order_status	Varchar2	10	Values (‘in process’;’fulfilled’;back order’;’canceled

I. Sales_order_details

Column	**Datatype**	**Size**	**Attributes**
S_order_no	Varchar2	6	Primary key/foreign key references s_order_no of sales_order
Product_no	Varchar2	6	Primary key/foreign key references product_no of product_master
Qty_order	Number	8	
Qty_disp	Number	8	
Product_rate	Number	10,2	

Insert the following data into their respective tables using insert statement:
Data for sales_man master table

Salesman_no	**Salesman name**	**Address**	**City**	**Pin code**	**State**	**Salamt**	**Tgt_to_get**	**Ytd Sales**	**Remark**
S00001	Kiran	A/14 worli	Bom bay	400002	Mah	3000	100	50	Good
S00002	Manish	65,nariman	Bom bay	400001	Mah	3000	200	100	Good
S00003	Ravi	P-7 Bandra	Bom bay	400032	Mah	3000	200	100	Good
S00004	Ashish	A/5 Juhu	Bom bay	400044	Mah	3500	200	150	Good

(ii)
Data for salesorder table:

S_orderno	S_orderdate	Client no	Dely type	Bill yn	Salesman no	Delay date	Orderstatus
019001	12-jan-96	0001	F	N	50001	20-jan-96	Ip

019002	25-jan-96	0002	P	N	50002	27-jan-96	C
016865	18-feb-96	0003	F	Y	500003	20-feb-96	F
019003	03-apr-96	0001	F	Y	500001	07-apr-96	F
046866	20-may-96	0004	P	N	500002	22-may-96	C
010008	24-may-96	0005	F	N	500004	26-may-96	Ip

(iii)
Data for sales_order_details table:

S_order no	Product no	Qty ordered	Qty disp	Product_rate
019001	P00001	4	4	525
019001	P07965	2	1	8400
019001	P07885	2	1	5250
019002	P00001	10	0	525
046865	P07868	3	3	3150
046865	P07885	10	10	5250
019003	P00001	4	4	1050
019003	P03453	2	2	1050
046866	P06734	1	1	12000
046866	P07965	1	0	8400
010008	P07975	1	0	1050
010008	P00001	10	5	525

4.4 SOLUTION TO ASSIGNMENT:

Q.1. create the following tables:
(i) Create table sales_master
(salesman_no varchar2(6) primary key check salesman_no in ‘S%’,
sal_name varchar2(20) not null,
address varchar2(20) not null,
city varchar2(20),
state varchar2(20),
pincode number(8),
sal_amt number(8,2) not null check(sal_amt >0),
tgt_to get number(6,2) not null check(sal_amt >0),
ytd_sales number(6,2) not null check(sal_amt >0),
remarks varchar2(30));

(ii) create table sales_order
s_order_no varchar2(6) primary key check s_order_no in ‘0%’,
s_order_date date,
client_no varchar2(25),
dely_add varchar2(6),
salesman_no varchar2(6) foreign key references salesman_no(sales_master),

dely_type char(1) check delt_type in ‘p’ or ‘f’ default ‘f’,
billed_yn char(1),
dely_date date check not in(dely_date<order_date),
order_status varchar2(10) check order_status in (‘in_process’ or ‘fulfilled’ or ‘back_order’ or ‘canceled’));

Q.2. Insert the following data into their respective tables using insert statement:
(i) Insert into sales_master
Values(‘S0001’,’kiran’,’a/14 worli’,’bombay ’,’400002 ’,’mah ’,’3000 ’,’100 ’,’50 ’,’good’);

(ii) Insert into sales_order
Values(‘019001’,’12-jan-96’,’0001 ’,’F ’,’N ’,’50001 ’,’20-jan-96’,’IP’);

Program No: 5

TITLE : MODIFYING THE STRUCTURES OF THE TABLE.

5.1 Objective 5.2 Theory 5.3 Assignment 5.4. Solution to Assignment

5.1 OBJECTIVE: Use of Alter table statements

5.2 THEORY AND CONCEPTS: **Modifying the Structure of Tables**- Alter table command is used to changing the structure of a table. Using the alter table clause you cannot perform the following tasks:

(i) change the name of table
(ii) change the name of column
(iii) drop a column
(iv) decrease the size of a table if table data exists.

The following tasks you can perform through alter table command.

(i) **Adding new columns**:
Syntax
ALTER TABLE tablename
ADD (newcolumnname newdatatype (size));

(ii) **Modifying existing table**
Syntax:
ALTER TABLE tablename
MODIFY (newcolumnname newdatatype (size));

NOTE**:** Oracle not allow constraints defined using the alter table, if the data in the table, violates such constraints.

Removing/Deleting Tables- Following command is used for removing or deleting a table.

Syntax:
DROP TABLE tabename:

Defining Integrity constraints in the ALTER TABLE command-

You can also define integrity constraints using the constraint clause in the ALTER TABLE command. The following examples show the definitions of several integrity constraints.

(1) **Add PRIMARY KEY**-
Syntax:
ALTER TABLE tablename
ADD PRIMARY KEY(columnname);

(2) **Add FOREIGN KEY**-
Syntax:
ALTER TABLE tablename
ADD CONSTRAINT constraintname
FOREIGN KEY(columnname) REFERENCES tablename;

Droping integrity constraints in the ALTER TABLE command:

You can drop an integrity constraint if the rule that if enforces is no longer true or if the constraint is no longer needed. Drop the constraint using the ALTER TABLE command with the DROP clause. The following examples illustrate the droping of integrity constraints.

(1) **DROP the PRIMARY KEY**-
Syntax:
ALTER TABLE tablename
DROP PRIMARY KEY

(2) **DROP FOREIGN KEY**-
Syntax:
ALTER TABLE tablename
DROP CONSTRAINT constraintname;

5.3 ASSIGNMENT:

Que 1. Create the following tables:
Challan_Header

Column name	**data type**	**size**	**Attributes**
Challan_no	varchar2	6	Primary key
s_order_no	varchar2	6	Foreign key references s_order_no of sales_order table
challan_date	date		not null
billed_yn	char	1	values (‘Y’,’N’). Default ‘N’

Table Name : Challan_Details

Column name	**data type**	**size**	**Attributes**
Challan_no	varchar2	6	Primary key/Foreign key references Product_no of product_master
Qty_disp	number	4,2	not null

Q2. Insert the following values into the challan header and challan_details tables:

(i)

Challan No	**S_order No**	**Challan Date**	**Billed**
CH9001	019001	12-DEC-95	Y
CH865	046865	12-NOV-95	Y
CH3965	010008	12-OCT-95	Y

Data for challan_details table

Challan No	**Product No**	**Qty_Disp**
CH9001	P00001	4
CH9001	P07965	1
CH9001	P07885	1
CH6865	P07868	3
CH6865	P03453	4
CH6865	P00001	10
CH3965	P00001	5
CH3965	P07975	2

Answer the following queries

Q1. Make the primary key to client_no in client_master.
Q2. Add a new column phone_no in the client_master table.
Q3. Add the not null constraint in the product_master table with the columns description, profit percent , sell price and cost price.
Q4. Change the size of client_no field in the client_master table.
Q5. Select product_no, description where profit percent is between 20 and 30 both inclusive.

5.4 SOLUTION TO ASSIGNMENT:

Q.1. Create the following tables:
Create table challan_header (
Chalan_no varchar2(6) primary key,
S_order_no varchar2(6) foreign key references s_order_no(sales_order),
Challan_date date not null,
Billed_yn char(i) check billed_yn in ('Y' or 'N') default 'N');

create table challan_details(
challan_no varchar2(6) primary key foreign key references product_no(product_master),
qty_disp number(4,2) not null);

Q.2. Insert the following values into the challan header and challan_details tables:
Insert into challan_header
Values('ch9001','019001','12-dec-95','Y');

Insert into challan_details
Values('ch9001','p00001','4');

Program No: 6

TITLE: ER Diagram using Dia Tool

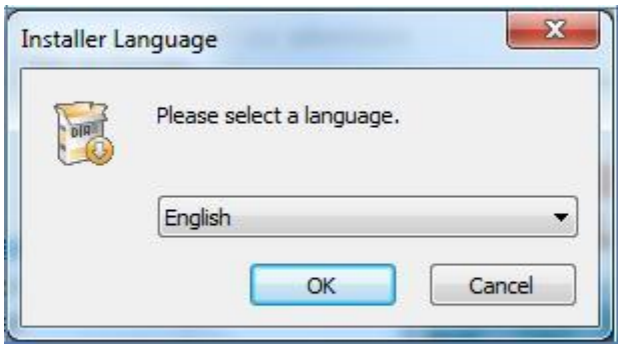
6.1 Objective 6.2 Theory (Installation & Dia Guide) 6.3 Assignment 6.4 Solution to Assignment

6.1 OBJECTIVE: applying tool to create ER model

6.2 THEORY AND CONCEPTS:
Introduction to Dia Diagram Editor:

a. Installation Guide

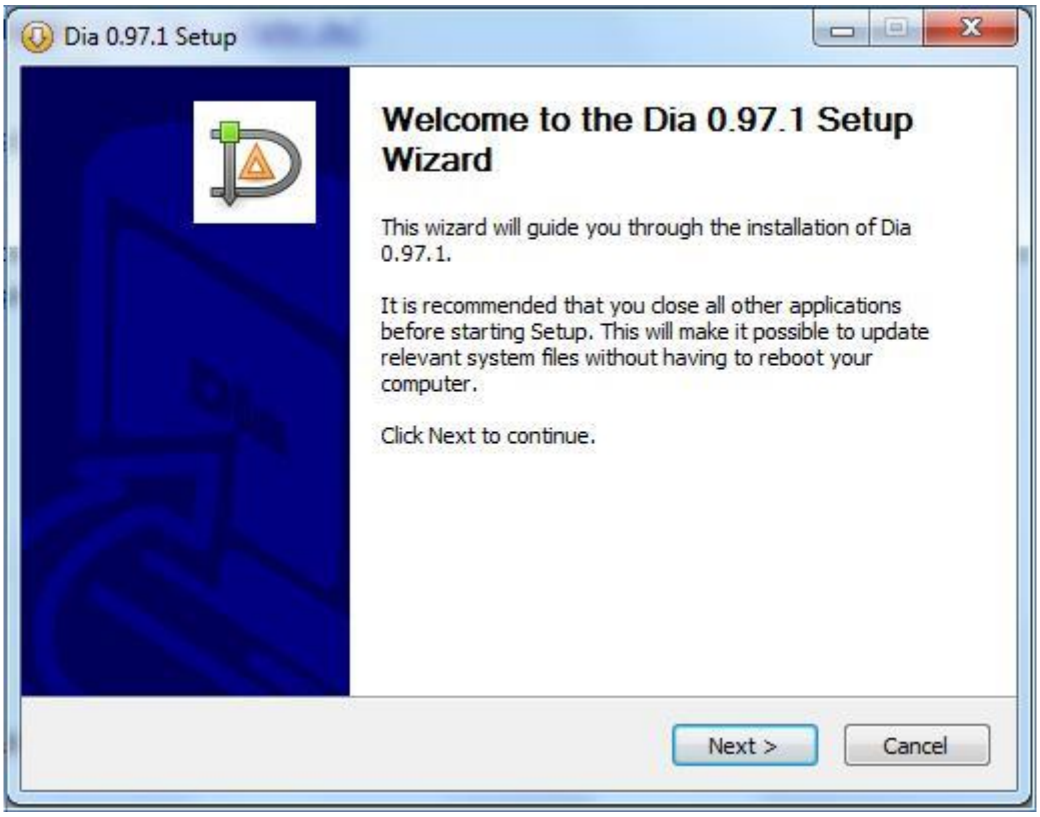
1. Download the installation file for your platform from http://dia-installer.de/.
2. Open the downloaded file, select preferred installation language, and press "OK".



Please select a language.

English

3. The Dia "Setup Wizard" window will appear. Click "Next".



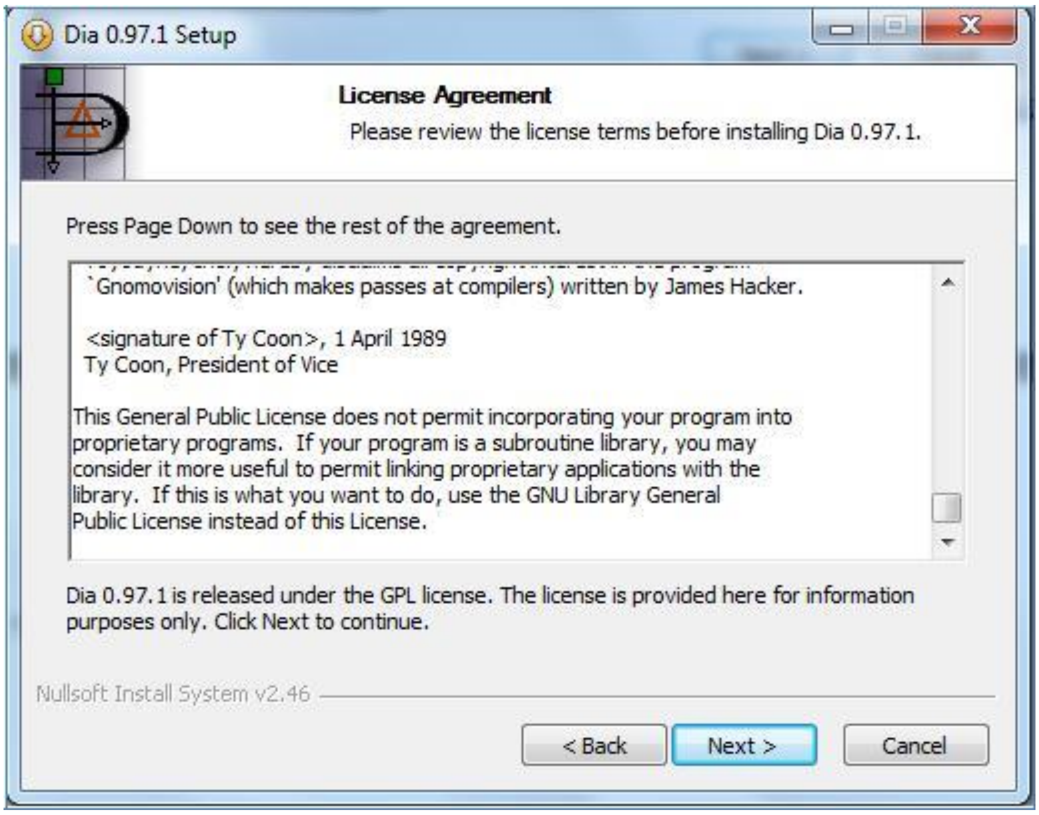
Welcome to the Dia 0.97.1 Setup Wizard

This wizard will guide you through the installation of Dia 0.97.1.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

4. In the “License Agreement” window click “Next” to continue installation.



Dia 0.97.1 Setup

License Agreement

Please review the license terms before installing Dia 0.97.1.

Press Page Down to see the rest of the agreement.

`Gnomovision' (which makes passes at compilers) written by James Hacker.

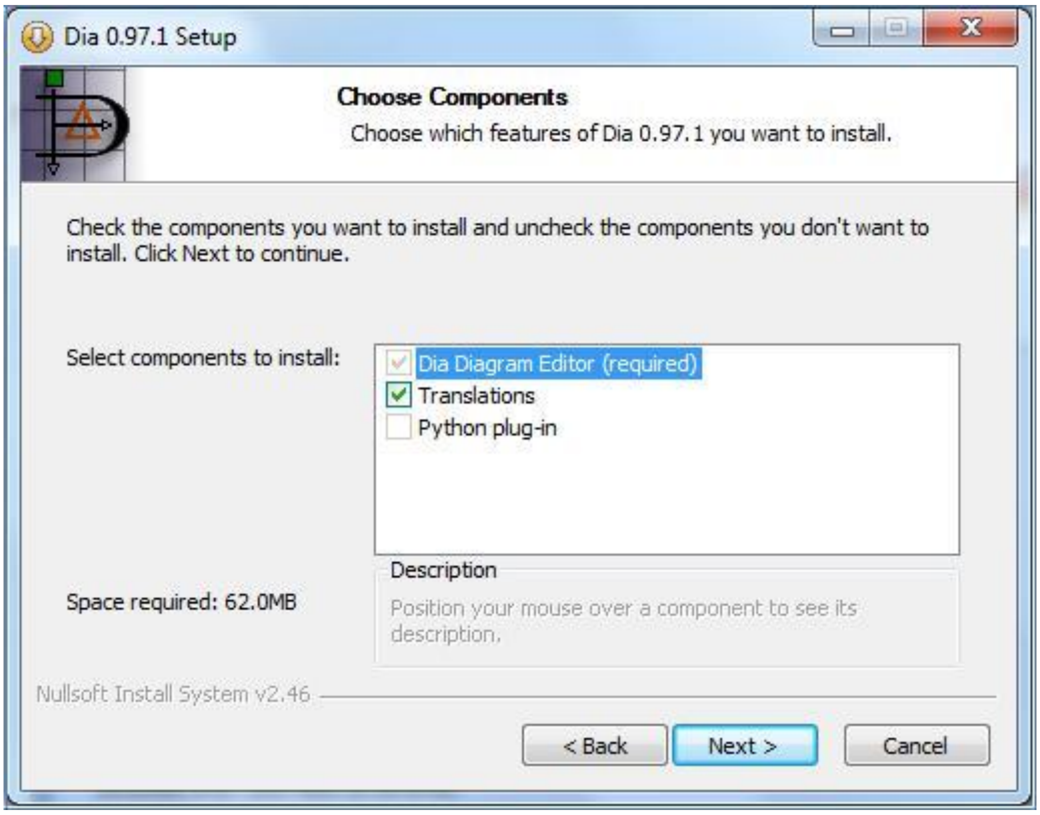
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Dia 0.97.1 is released under the GPL license. The license is provided here for information purposes only. Click Next to continue.

Nullsoft Install System v2.46

5. Choose the components you want to install and click “Next”.



Dia 0.97.1 Setup

Choose Components

Choose which features of Dia 0.97.1 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ✓ Dia Diagram Editor (required)
- ✓ Translations
- ☐ Python plug-in

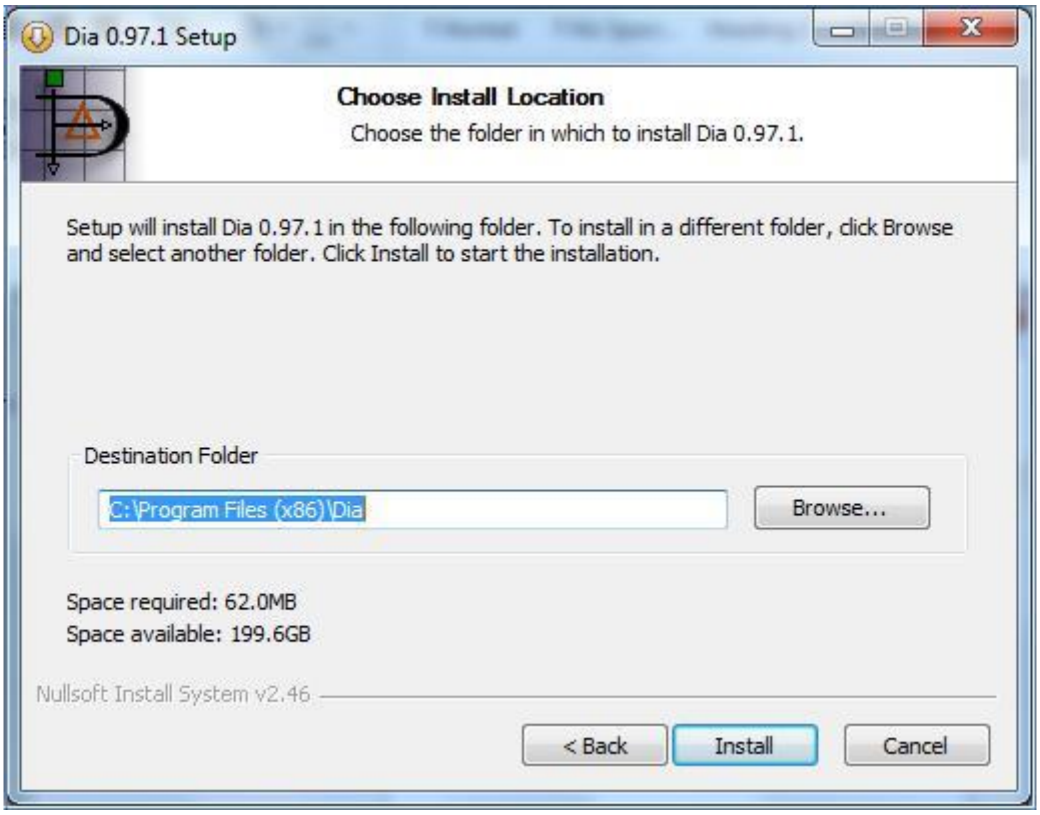
Space required: 62.0MB

Description

Position your mouse over a component to see its description.

Nullsoft Install System v2.46

6. Choose the installation location on your computer and click “Install”.


Dia 0.97.1 Setup

Choose Install Location

Choose the folder in which to install Dia 0.97.1.

Setup will install Dia 0.97.1 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files (x86)\Dia

Browse...

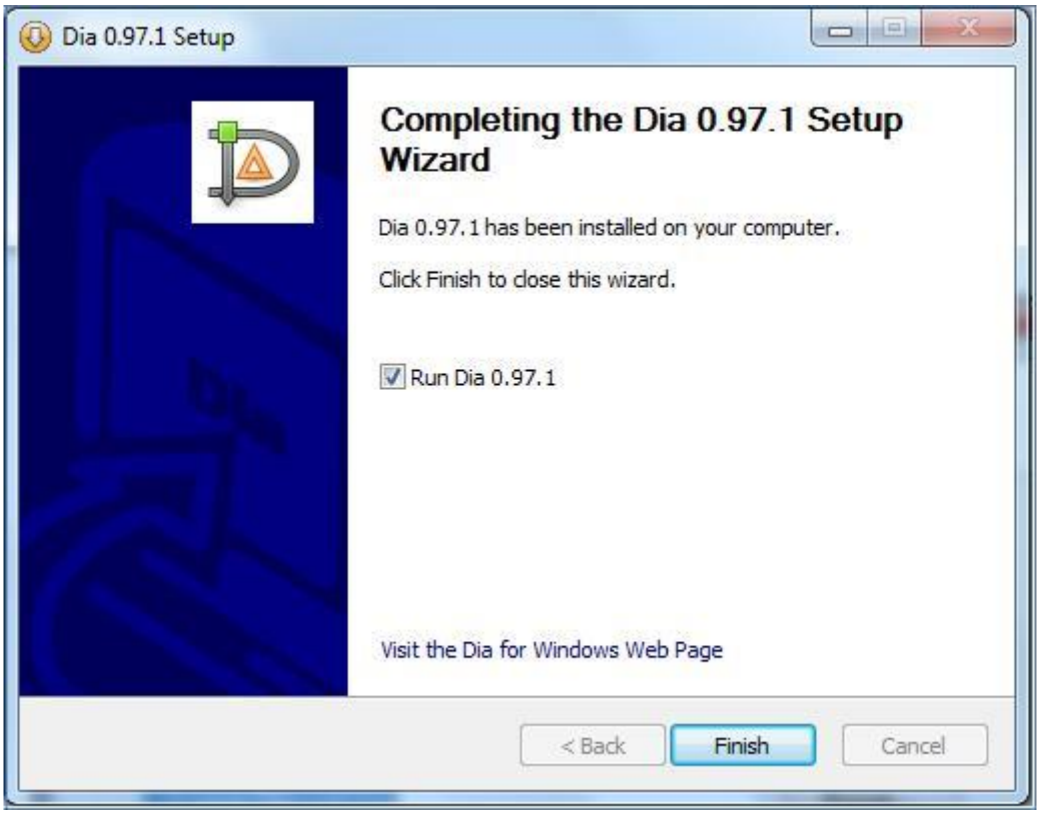
Space required: 62.0MB

Space available: 199.6GB

Nullsoft Install System v2.46

< Back | Install | Cancel

7. After the installation process is completed, click “Finish”.


Dia 0.97.1 Setup

Completing the Dia 0.97.1 Setup Wizard

Dia 0.97.1 has been installed on your computer.

Click Finish to close this wizard.

[✓] Run Dia 0.97.1

Visit the Dia for Windows Web Page

< Back | Finish | Cancel

b. Dia User Guide

Create an ER Diagram

In this section, basic guidelines are given on how to create ER database diagrams. An ER diagram consists of entity sets, attributes, and the relationship sets between entity sets. Let us create an ER diagram for a database called “Courses and Students”. The database will have two main entity sets, i.e., “Course” and “Student”. The relation between them defines which students belong to which course.

1. Start Dia Diagram Editor.

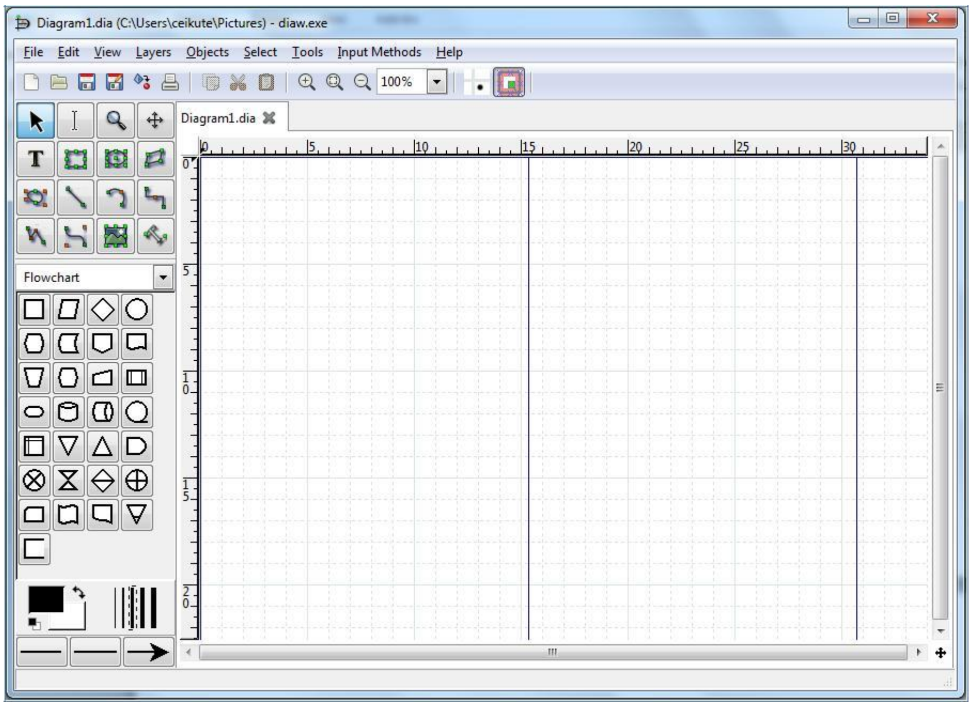
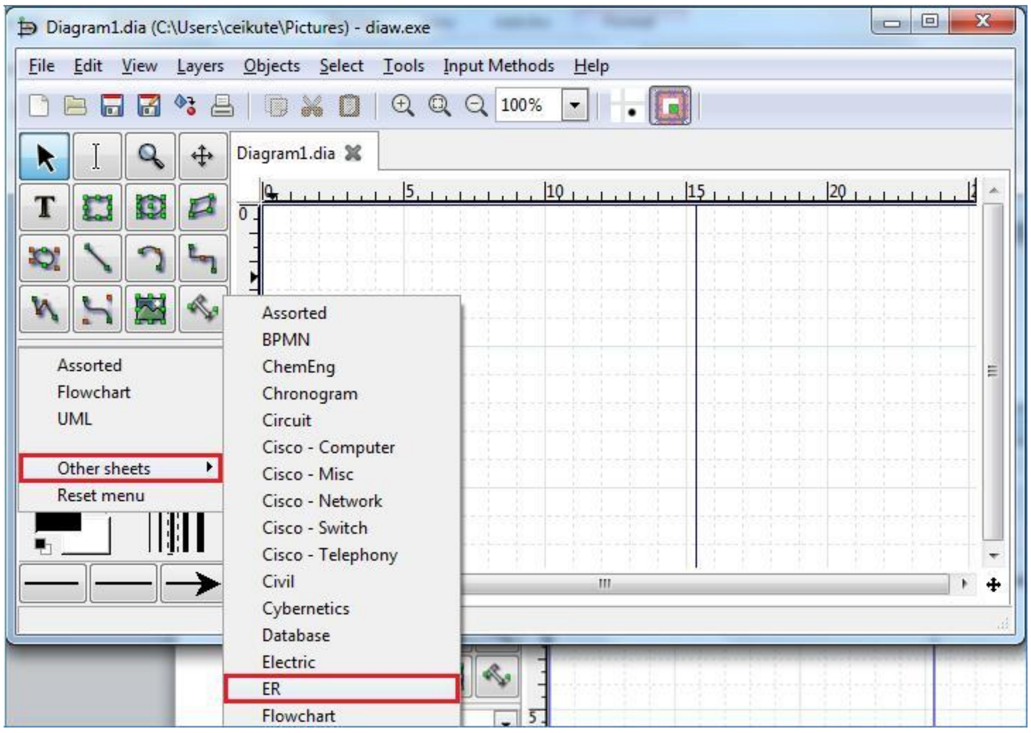


Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
Flowchart

2. On the left side of the menu click on the dropdown menu, select “Other sheets” and click on “ER”.



3. Now the menu consists only of shapes that are relevant to an ER diagram.

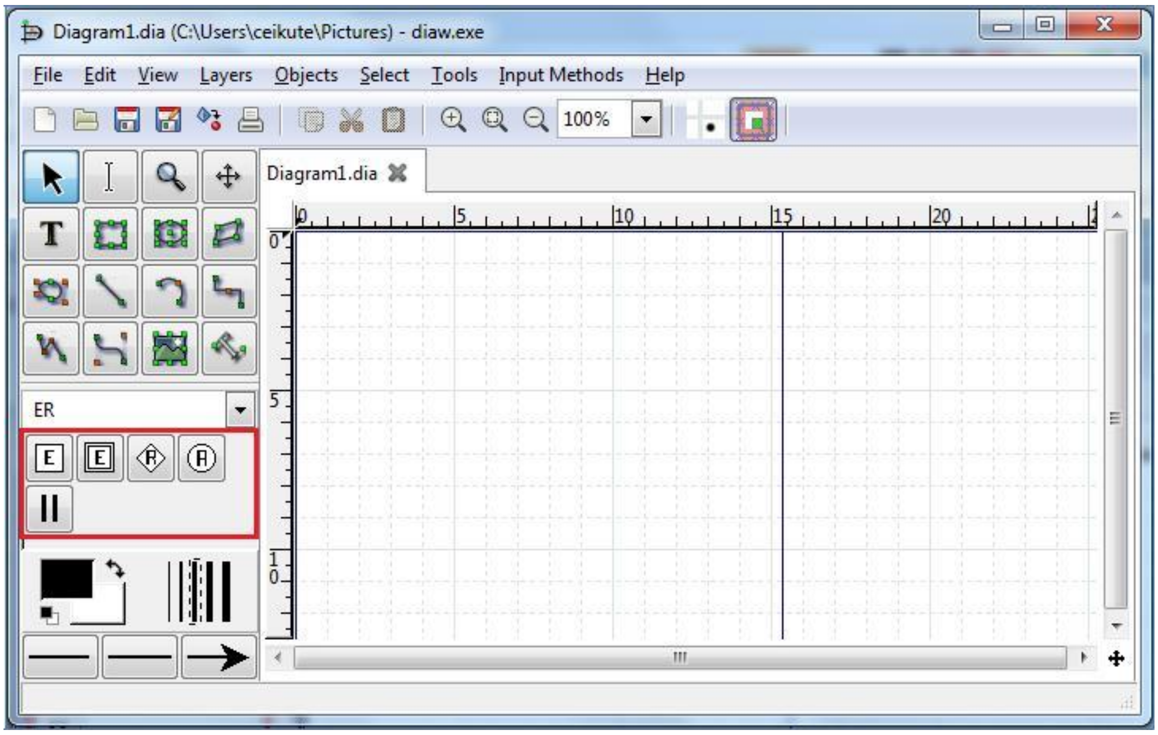


Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER

4. Let us create an entity called “Course”. Choose the “E” icon with a single frame in the shapes menu and click on the drawing space at the center. A rectangle with the name “Entity” will appear.

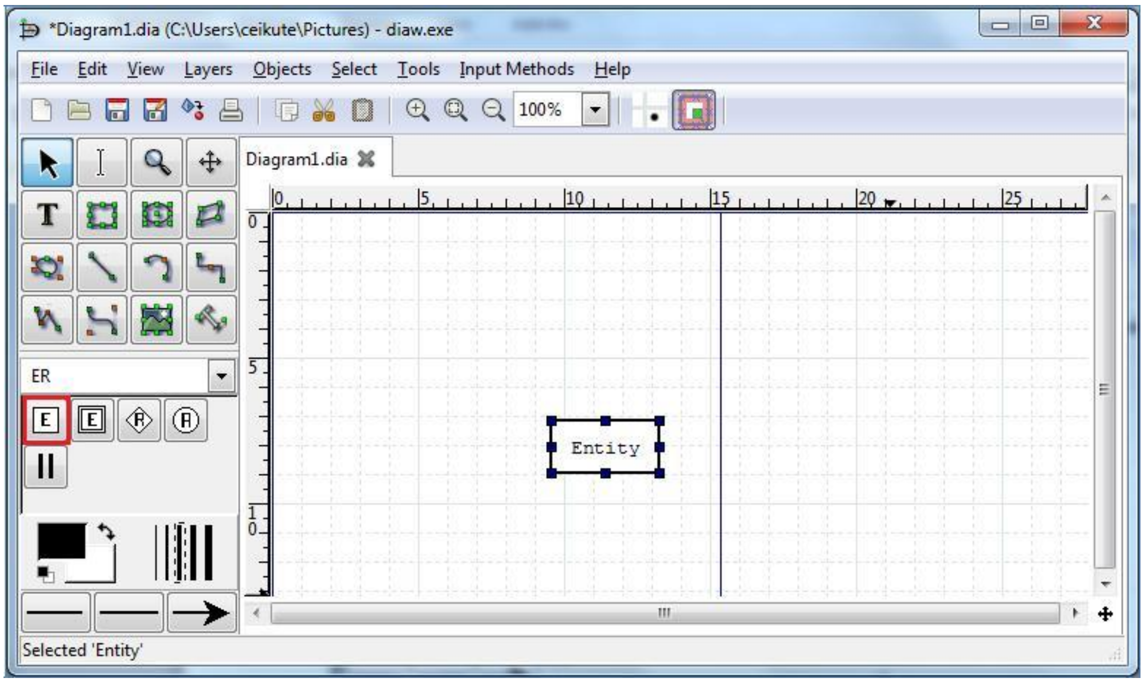


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
Entity
Selected 'Entity'

5. Double-click on the new entity set and the properties window will show up (or right-click and choose “Properties”). Change the name of the entity set to “Course” and click “OK”. The entity sets name will be changed.

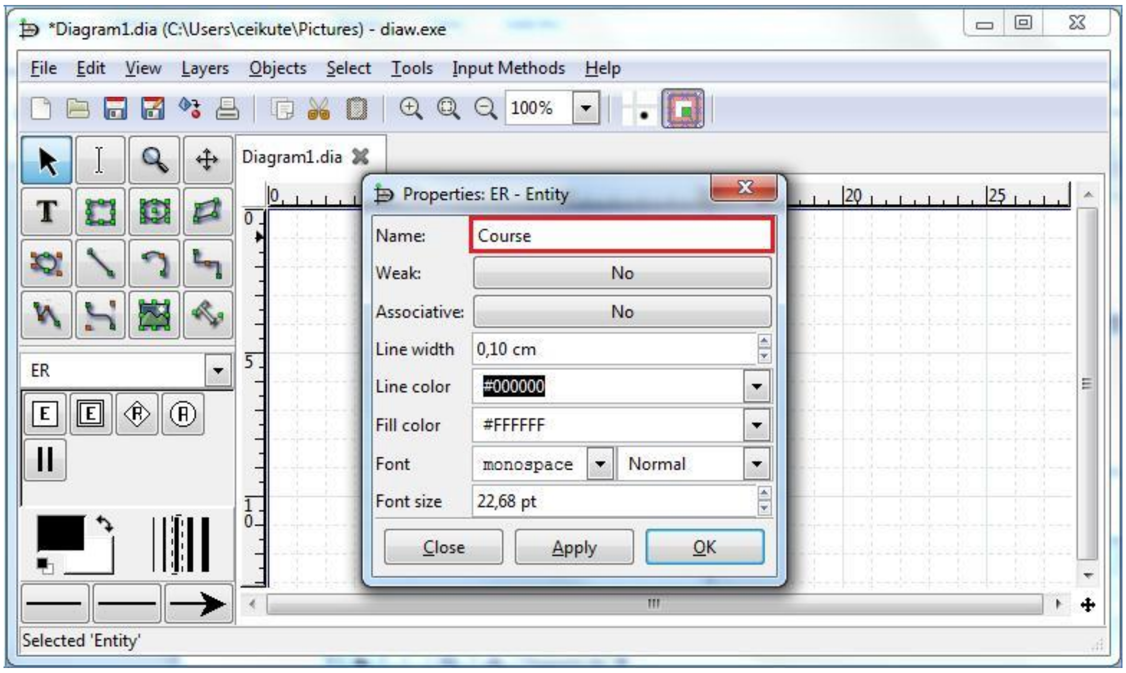


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER

Properties: ER - Entity

Name:	Course
Weak:	No
Associative:	No
Line width	0,10 cm
Line color	#000000
Fill color	#FFFFFF
Font	monospace Normal
Font size	22,68 pt

Selected 'Entity'

6. The “Course” entity set has several attributes: “courseId” (primary key), “title”, “ECTS”, “level”, “language”. In the shape menu select the “A” icon with an oval around it and click near the created “Course” entity set. An oval with the name “Attribute” will appear.

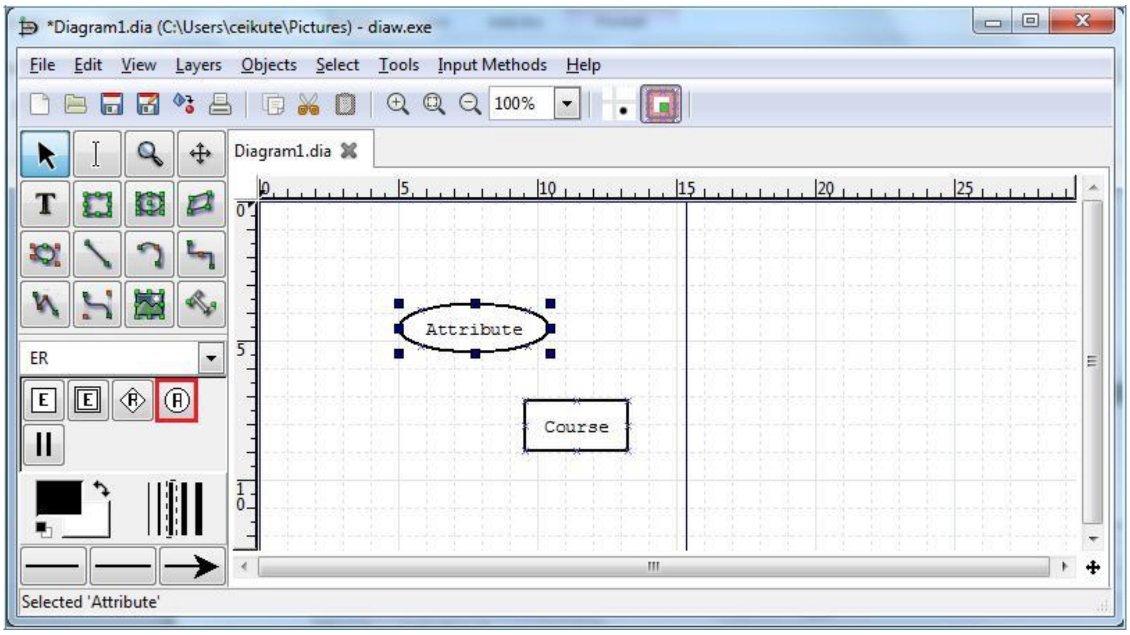


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
Attribute
Course

Selected 'Attribute'

7. Double-click on the attribute and in the properties window change the name to “courseId”. Since this attribute is also a primary key, select “Key” value “Yes”. Click “OK”.

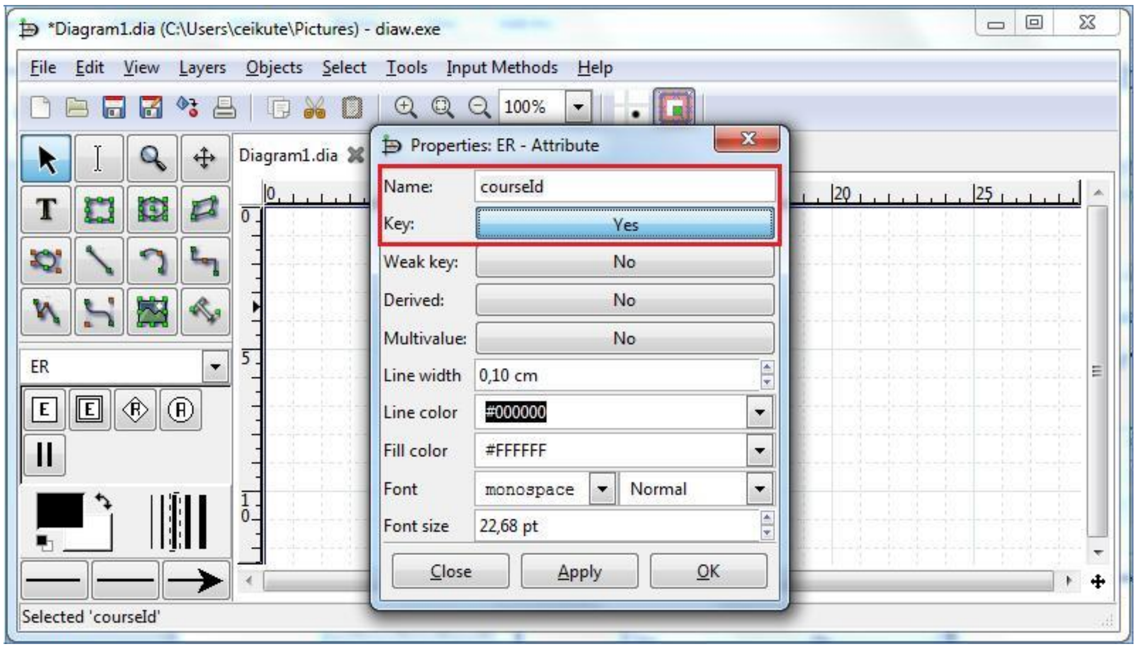


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
Close
Apply
OK
Selected 'courseId'

Properties: ER - Attribute

Name:	courseId
Key:	Yes
Weak key:	No
Derived:	No
Multivalue:	No
Line width	0,10 cm
Line color	#000000
Fill color	#FFFFFF
Font	monospace Normal
Font size	22,68 pt

8. In the shape menu click on the “Participation” icon and connect the entity set with the attribute.

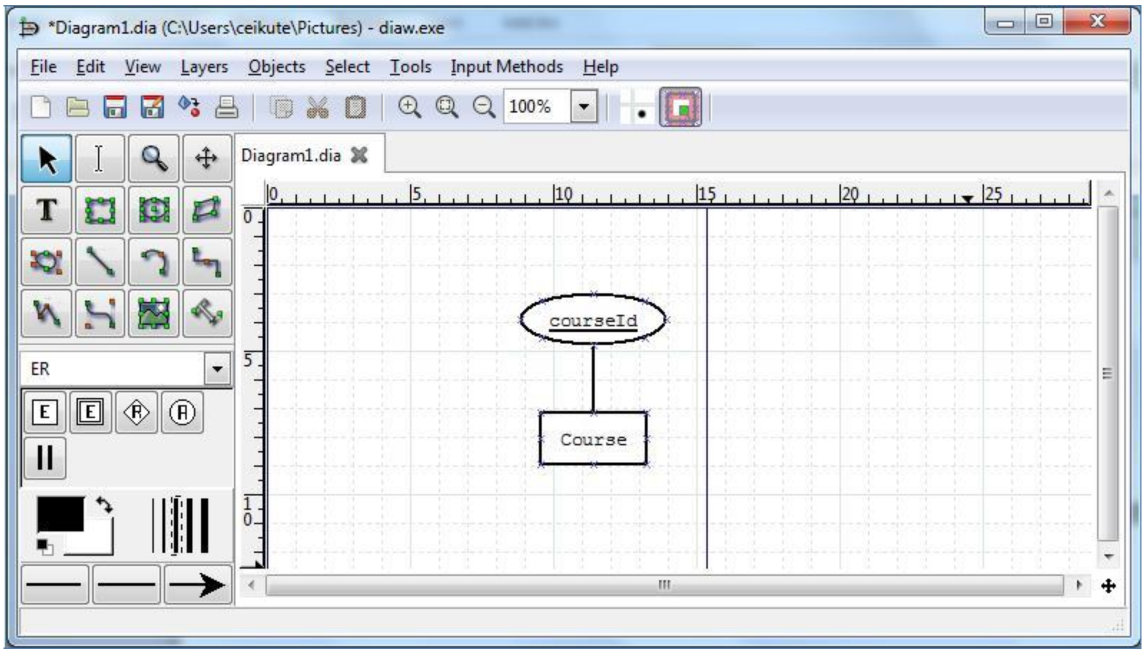


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
courseId
Course

9. It is also possible to connect an entity set with an attribute using different connectors. Choose the appropriate style at the end of a new connector by clicking on “arrow style at the end of the line new lines”. Select “line (L)” connector.

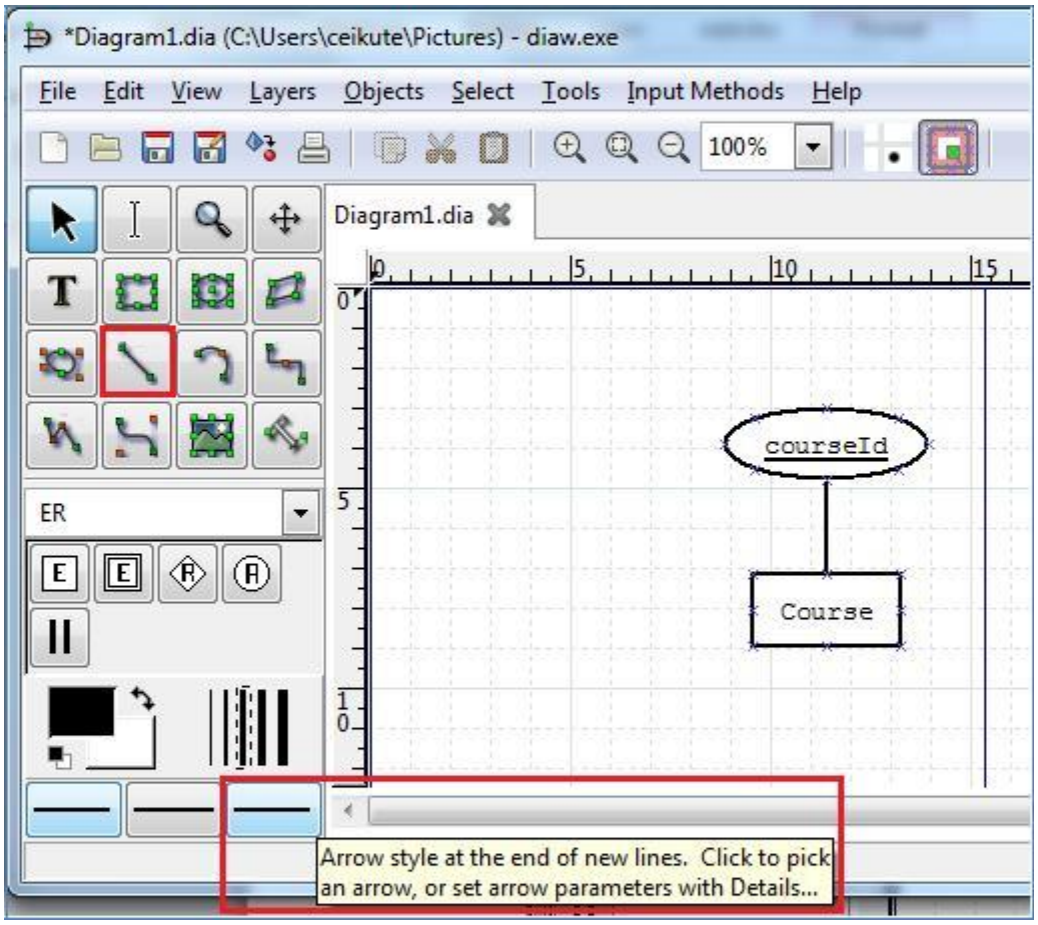


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
courseId
Course

Arrow style at the end of new lines. Click to pick an arrow, or set arrow parameters with Details...

10. Proceed with the rest of the attributes of the entity set “Course”.

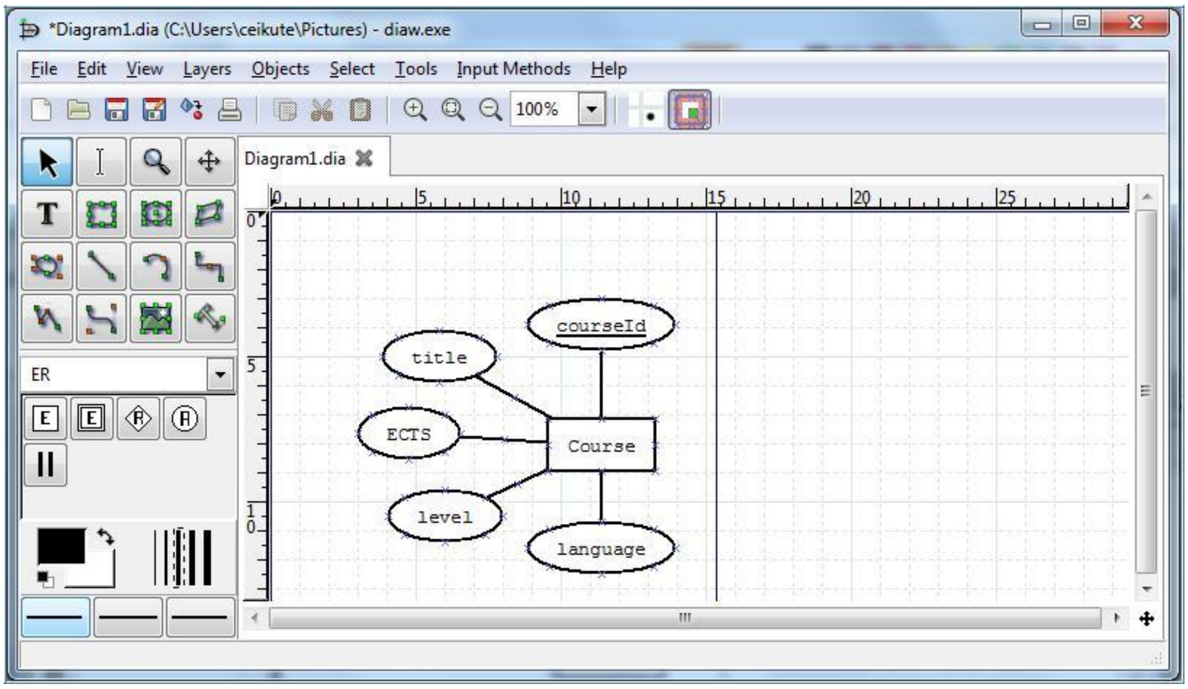


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
courseId
title
ECTS
Course
level
language

11. Attributes of the “Student” entity set are: “studentId”, “firstName”, “lastName”, “startDate”.

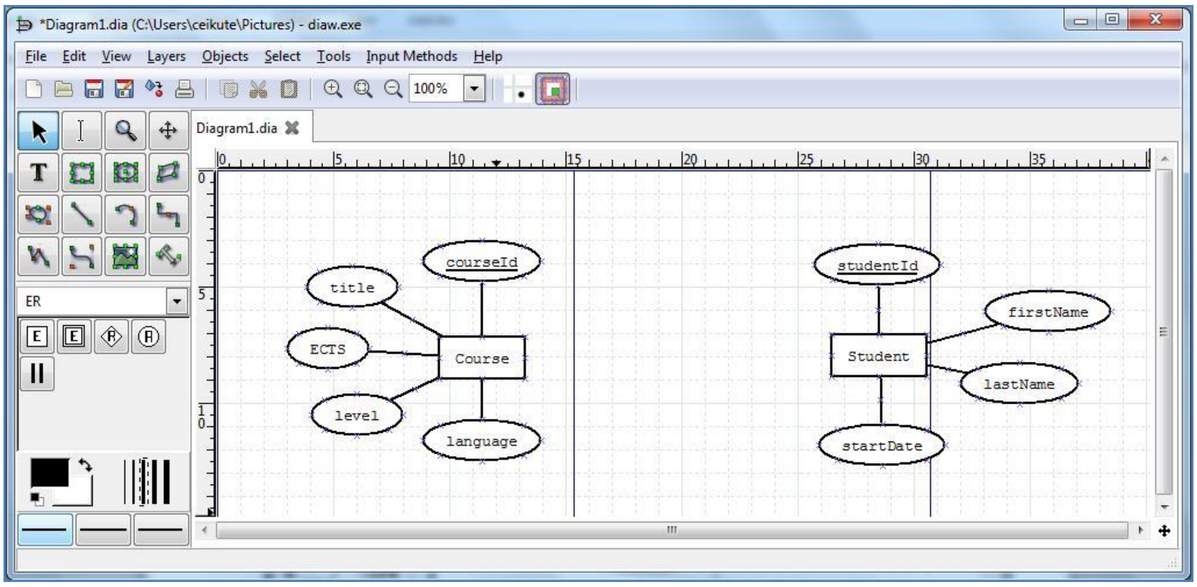
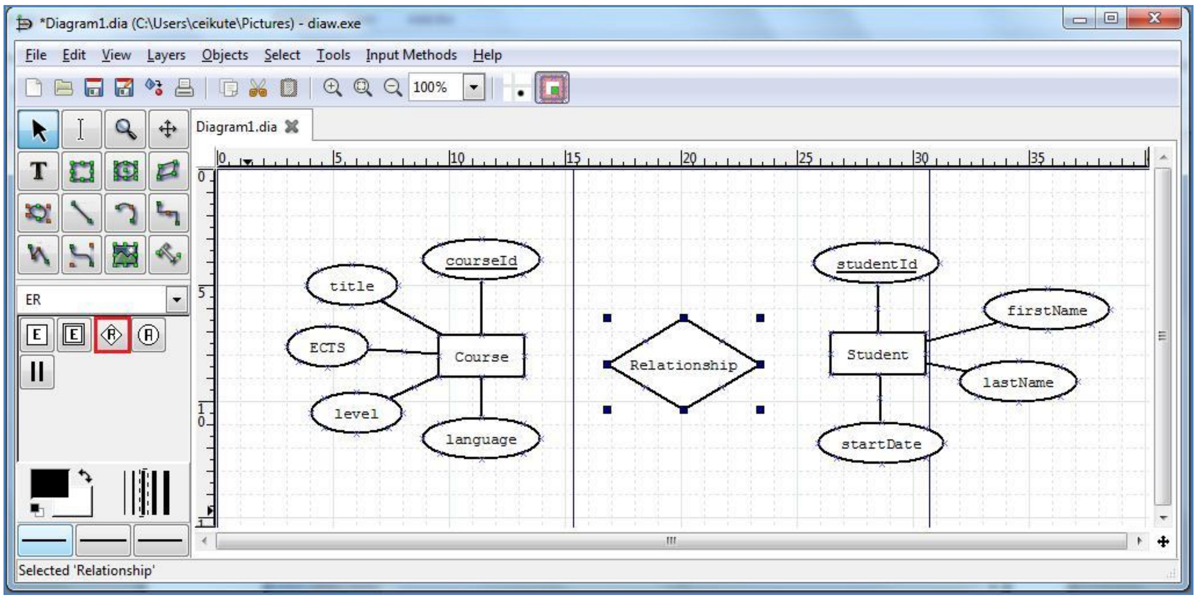


*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
courseId
title
ECTS
Course
level
language
studentId
firstName
Student
lastName
startDate

12. Create the relationship set between “Course” and “Student” with the name “Belongs” that also has an attribute “signUpDate”. In the shape menu select “R” with a diamond around it and click between the two entity sets in the drawing area.



13. Change the name to “Belongs” and assign attribute “signUpDate” to it.

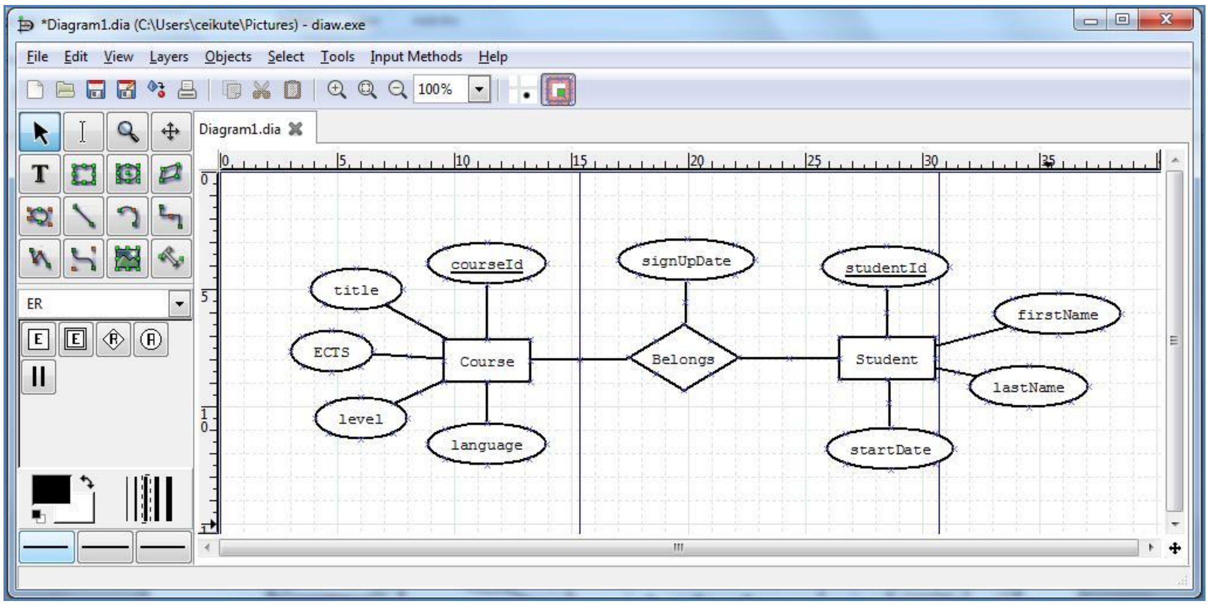


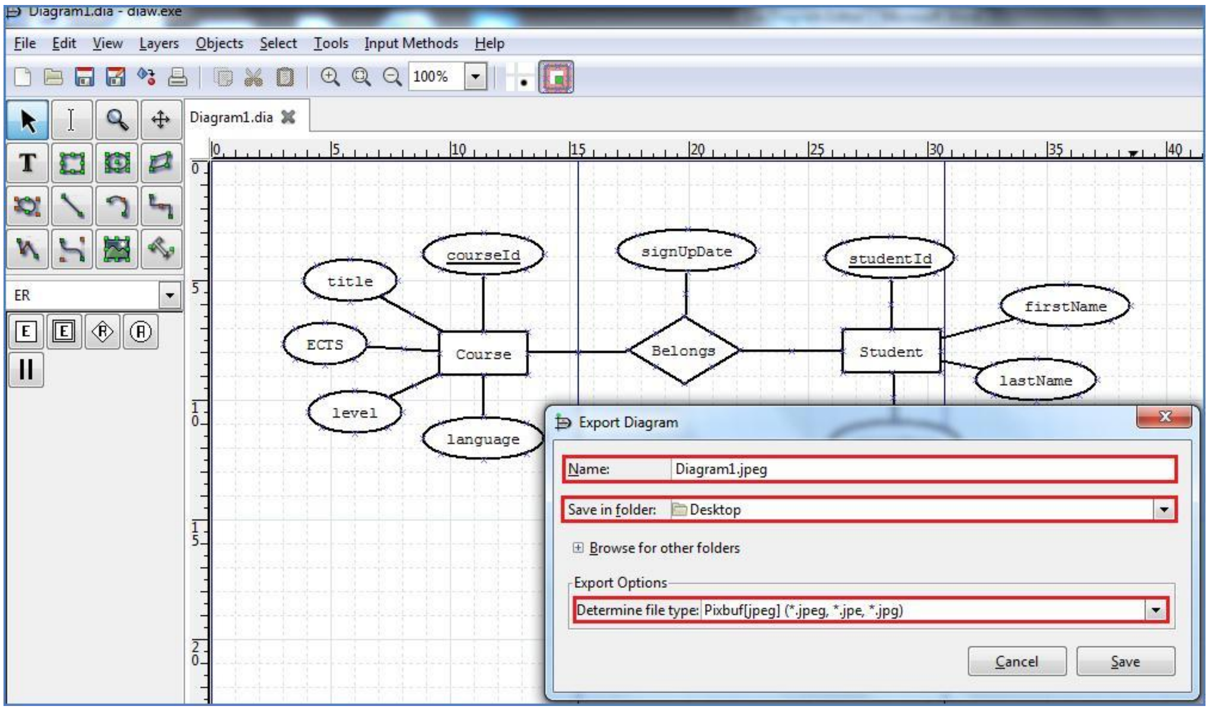
*Diagram1.dia (C:\Users\ceikute\Pictures) - diaw.exe
File Edit View Layers Objects Select Tools Input Methods Help
100%
Diagram1.dia
ER
courseId
title
ECTS
Course
level
language
signUpDate
Belongs
studentId
firstName
Student
lastName
startDate

The ER diagram for database “Courses and Students” was created successfully!

Note: save your diagram several times through all the creation process!

Export Created Diagram

1. Open the diagram you want to export.
2. Click on “File > Export“. Enter the name of the file, select the location you want the file to be saved, determine file type (e.g. JPG), and click “Save”.



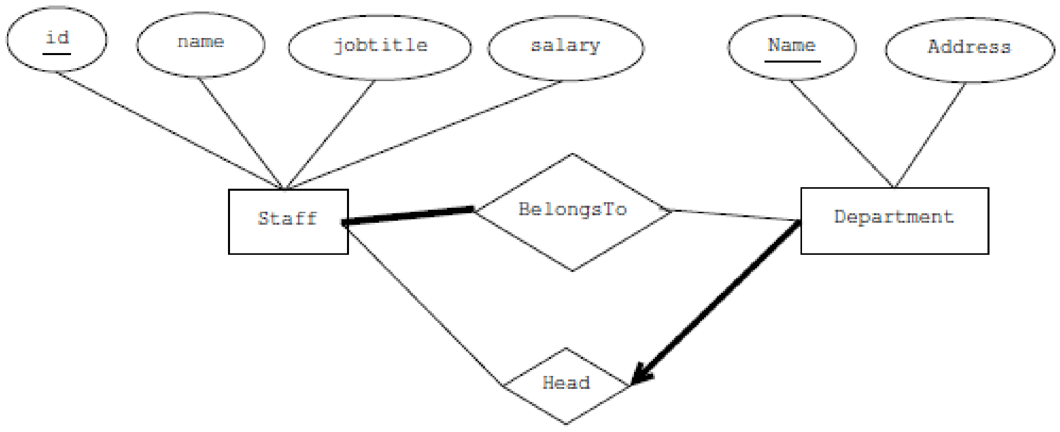
Name: Diagram1.jpeg

Save in folder: Desktop

Determine file type: Pixbuf[jpeg] (*.jpeg, *.jpe, *.jpg)

6.3 Assignment:
A university wants to set up a database to record details about its staff, and the departments they belong to. They intend to record the following information.
• For each member of staff, their staff identity number, name, job title, and salary.
• For each department, its name and address.
• For each member of staff, all departments that they belong to. It is required that every member of staff belongs to at least one department.
• For each department, the head of department. It is required that each department has exactly one head of department.
Draw an ER diagram that expresses the requirements for the database. Make sure that you capture all the constraints on the data mentioned above.

6.4 Solution to Assignment:
Here is one possible ER diagram:



Program No: 7

TITLE: USE OF JOINS.

7.1 Objective 7.2 Theory 7.3 Assignment 7.4. Solution to Assignment

7.1 OBJECTIVE: How different types of join is used.

7.2 THEORY AND CONCEPTS: **<u>Joint Multiple Table (Equi Join):</u>** Some times we require to treat more than one table as though manipulate data from all the tables as though the tables were not separate object but one single entity. To achieve this we have to join tables.Tables are joined on column that have dame data type and data with in tables.

The tables that have to be joined are specified in the FROM clause and the joining attributes in the WHERE clause.

Algorithm for JOIN in SQL:

1. Cartesian product of tables (specified in the FROM clause)
2. Selection of rows that match (predicate in the WHERE clause)
3. Project column specified in the SELECT clause.

1. Cartesian product:-

Consider two table student and course

```
Select B.*,P.*
FROM student B, course P;
```

2. INNER JOIN:

Cartesian product followed by selection

```
Select B.*,P.*
FROM student B, Course P
WHERE B.course # P.course # ;
```

3. LEFT OUTER JOIN:

LEFT OUTER JOIN = Cartesian product + selection but include rows from the left table which are unmatched pat nulls in the values of attributes belonging to th e second table

Exam:

```
Select  B.*,P*
FROM student B left join course p
ON B.course # P.course #;
```

4. RIGHT OUTER JOIN:

RIGHT OUTER JOIN = Cartesian product + selection but include rows from right table which are unmatched

Exam:

```
Select B.*,P.*
From student B RIGHT JOIN course P
B.course# = P course # ;
```

5. FULL OUTER JOIN

```
 Exam
Select B.*,P.*
From student B  FULL JOIN course P
 On B.course # = P course # ;
```

7.3 ASSIGNMENT

1. Find out the product which has been sold to 'Ivan Sayross.'
2. Find out the product and their quantities that will have do delivered.
3. Find the product_no and description of moving products.
4. Find out the names of clients who have purchased 'CD DRIVE'
5. List the product_no and s_order_no of customers haaving qty ordered less than 5 from the order details table for the product "1.44 floppies".
6. Find the products and their quantities for the orders placed by 'Vandan Saitwal ' and "Ivan Bayross".
7. Find the products and their quantities for the orders placed by client_no " C00001" and "C00002"
8. Find the order No,, Client No and salesman No. where a client has been received by more than one salesman.
9. Display the s_order_date in the format "dd-mm-yy" e.g. "12- feb-96"
10. Find the date , 15 days after date.

7.4 SOLUTION TO ASSIGNMENT

i.
```
Select product_name from product_master
where seller_name ='Ivan Bayross';
```

ii.
```
select product_name, product_qty from product_master
where dely_date > sysdate;
```

iii.
```
select product_no , description from product_master
where reorder_lvl <10;
```

iv.
```
select name from client_master c, product_master p
where p.client_no = c.client_no
and product_name = 'CD Drive';
```

v.
```
select product_no, order_no from product_master p, sales_order s
where p.product_no = s.product_no
and product_qty < 5
and product_name = '1.44 Floppies'
```

vi.
```
select product_name, product_qty from product_master p, sales_order s, client_master c
where p.product_no = s.product_no and
s.client_no  =  c.client_no
and client_name = 'Vandan Saitwal' and 'Ivan Bayross';
```

vii. select product_name, product_qty from product_master p, sales_order s, client_master c
where p.product_no = s.product_no and
s.client_no = c.client_no
and client_no = ‘C0001’ and ‘C00002’;

viii. select order_no, client_no , salesman_no from sales_order so, client_master cm, sales_master
sm where sm.salesman_no = so.salesman_no and
so.client_no = cm.client_no and
salesman>1;

ix. select s_order_date ‘dd-mm-yy’from sales_order;

x. select sysdate+15 “Date after 15 days” from sales_order;

Program No: 8

TITLE: GROUPING DATA FROM TABLES.

8.1 Objective 8.2 Theory 8.3 Assignment 8.4. Solution to Assignment

8.1 OBJECTIVE: use of group functions.

8.2 THEORY AND CONCEPTS: **Grouping Data From Tables:**

There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples, we specify this wish in SQL using the group by clause. The attribute or attributes given in the group by clause are used to form group. Tuples with the same value on all attributes in the group by clause are placed in one group.

Syntax:

```
SELECT columnname, columnname
FROM   tablename
GROUP BY columnname;
```

At times it is useful to state a condition that applies to groups rather than to tuples. For example we might be interested in only those branches where the average account balance is more than 1200. This condition does not apply to a single tuple, rather it applies to each group constructed by the GROUP BY clause. To express such query, we use the having clause of SQL. SQL applies predicates in the having may be used.

Syntax:

```
SELECT columnname, columnname
FROM   tablename
GROUP BY columnname;
HAVING searchcondition;
```

8.3. ASSIGNMENTS:

Answer the following queries:

Q1.- Print the description and total quantity sold for each product.

Q2.- Find the value of each product sold.

Q3.- Calculate the average quantity sold for each client that has a maximum order value of 15000.

Q4.- Find out the products which has been sold to Ivan.

Q5.- Find the names of clients who have ‘CD Drive’.

Q6.- Find the products and their quantities for the orders placed by ‘Vandana’ and ‘Ivan’.

Q7.- Select product_no, total qty_ordered for each product.

Q8.- Select product_no, product description and qty ordered for each product.

Q9.- Display the order number and day on which clients placed their order.

Q10.- Display the month and Date when the order must be delivered.

8.4 SOLUTION TO ASSIGNMENT

1. Select description, sum(product_qty) from product_master group by description;

2. Select distinct description from product_master;

3. select distinct client_name, avg(product_qty) from sales_order so,
   product_master pm where so.client_no = pm.client_no
   group by client_name
   having order_value = (Select max(order_value) from sales_order);

4. select product_name from product_master pm, sales_master sm
   where pm.client_no = sm.client_no and sal_name='Ivan';

5. select name from client_master where description = 'CD Drive';

6. select product_name, product_qty from product_master pm, sales_order so
   where pm.salesman_no= so.salesman_no and salesman_name = 'Vandana' and 'Ivan';

7. select distinct product_name, product_qty from product_master;

8. select distinct description , product_no, product_qty from product_master;

9. select order_no, date from sales_order;

10. select dely_date from sales_order;

Program No: 8

TITLE: SUBQUERIES.

8.1 Objective 8.2 Theory 8.3 Assignment 8.4. Solution to Assignment

8.1 OBJECTIVE: How subqueries is used.

8.2 THEORY AND CONCEPTS: a subquery is a form of an SQL statement that appears inside another SQL statement. It also termed as nested query. The statement containing a subquery called a parent statement. The rows returned bu the subquery are use by the following statement.

It can be used by the following commands:

1. To insert records in the target table.
2. To create tables and insert records in this table.
3. To update records in the target table.
4. To create view.
5. To provide values for the condition in the WHERE , HAVING IN , SELECT,UPDATE, and DELETE statements.

Exam:-
Creating clientmaster table from oldclient_master, table

```
Create table client_master
AS SELECT * FROM oldclient_master;
```

Using the Union, Intersect and Minus Clause:

Union Clause:

The user can put together multiple queries and combine their output using the union clause . The union clause merges the output of two or more queries into a single set of rows and column. The final output of union clause will be

Output: = Records only in query one + records only in query two + A single set of records with is common in the both queries.

Syntax:

```
SELECT columnname, columnname
FROM tablename 1
UNION
SELECT columnname, columnname
From tablename2;
```

Intersect Clause: The use can put together multiple queries and their output using the interest clause. The final output of the interest clause will be :

Output =A single set of records which are common in both queries

Syntax:

```
SELECT  columnname, columnname
FROM tablename 1
INTERSECT
SELECT columnname, columnname
FROM tablename 2;
```

MINUS CLAUSE:- The user can put together multiple queries and combine their output = records only in query one

Syntax:

```
SELECT columnname, columnname
FROM tablename ;
MINUS
SELECT columnname, columnname
FROM tablename ;
```

8.3 ASSIGNMENT

Answer the following queries:

i. Find the product_no and description of non- moving products.
ii. Find the customer name, address, city and pincode for the client who has placed order no "019001"
iii. Find the client names who have placed order before the month of may 96.
iv. Find out if product "1.44 Drive" is ordered by only client and print the client_no name to whom it was sold.
v. find the names of client who have placed orders worth Rs.10000 or more.
vi. Select the orders placed by 'Rahul Desai"
vii. Select the names of persons who are in Mr. Pradeep's department and who have also worked on an inventory control system.
viii. Select all the clients and the salesman in the city of Bombay.
ix. Select salesman name in "Bombay" who has atleast one client located at "Bombay"
x. Select the product_no, description, qty_on-hand,cost_price of non_moving items in the product_master table.

8.4. SOLUTION TO ASSIGNMENT

i.
```
select product_no, description  from product_master where reorder_lvl =3;
```

ii.
```
select name, address, city, pincode from client_master cm, sales_order so where
cm.client_no= so.client_no
and so.order_no = '019001';
```

iii.
```
select name from client_master cm, sales_order so
where cm.client_no = so.client_no
and order_date < 01-may-96;
```

iv.
```
select client_no, name from clien_master cm, product_master pm
where cm.client_no = pm.client_no
product_name='1.44 Drive';
```

v. select name from client_master cm, sales_order so
where cm.client_no = so.client_no
and order_amt >= 10000;

vi. select * from sales_order so, sales_master sm
where so.salesman_no= sm.salesman_no
and sal_name =’Rahul Desai’;

vii. select sal_person from sales_master where dept in (select dept from client_master
where name = ‘Pradeep’) and project_name=’Inventory Control System’;

viii. select * from client_master cm, sales_master sm
cm.client_no = sm.client_no and
cm.city =’bombay’;

ix. select sal_name from sales_master where city =’bombay’;

x. select product_no, description, qty_on-hand, cost_price from product_master;

Program No: 9

TITLE : INDEXES , SEQUENCES & VIEWS

9.1 Objective 9.2 Theory 9.3 Assignment 9.4. Solution to Assignment

9.1 OBJECTIVE: Use of different types of Indexes, uses of sequences and views.

9.2 THEORY AND CONCEPTS:

Indexes- An index is an ordered list of content of a column or group of columns in a table. An index created on the single column of the table is called simple index. When multiple table columns are included in the index it is called composite index.

Creating an Index for a table:-

Syntax (Simple)

```
CREATE INDEX index_name
ON tablename(column name);
```

Composite Index:-

```
CREATE INDEX index_name
ON tablename(columnname,columnname);
```

Creating an Unique Index:-

```
CREATE UNIQUE INDEX indexfilename
ON tablename(columnname);
```

Dropping Indexes:-

An index can be dropped by using DROP INDEX

SYNTAX:-

```
DROP INDEX indexfilename;
```

Views:-

Logical data is how we want to see the current data in our database. Physical data is how this data is actually placed in our database.

Views are masks placed upon tables. This allows the programmer to develop a method via which we can display predetermined data to users according to our desire.

Views may be created fore the following reasons:

1. The DBA stores the views as a definition only. Hence there is no duplication of data.
2. Simplifies queries.
3. Can be queried as a base table itself.
4. Provides data security.
5. Avoids data redundancy.

Sequences:

Use the CREATE SEQUENCE statement to create a sequence, which is a database object from which multiple users may generate unique integers. You can use sequences to automatically generate primary key values.

When a sequence number is generated, the sequence is incremented, independent of the transaction committing or rolling back. If two users concurrently increment the same sequence, then the sequence numbers each user acquires may have gaps, because sequence numbers are being generated by the other user. One user can never acquire the sequence number generated by another user. After a sequence value is generated by one user, that user can continue to access that value regardless of whether the sequence is incremented by another user.

Sequence numbers are generated independently of tables, so the same sequence can be used for one or for multiple tables. It is possible that individual sequence numbers will appear to be skipped, because they were generated and used in a transaction that ultimately rolled back. Additionally, a single user may not realize that other users are drawing from the same sequence.

After a sequence is created, you can access its values in SQL statements with the CURRVAL pseudocolumn, which returns the current value of the sequence, or the NEXTVAL pseudocolumn, which increments the sequence and returns the new value.

EXAMPLE:

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

Example:

```
create sequence noseq increment by 2 start with 1 minvalue 1 maxvalue 19 cycle nocache noorder;

select noseq.nextval from dual;

create table abcd (no number(2), name varchar2(25));

insert into abcd values (noseq.nextval/currval, 'RAM');
```

Views:-

Syntax:-

```
        CREATE VIEW viewname AS
        SELECT columnname,columnname
        FROM tablename
        WHERE columnname=expression_list;
```

Renaming the columns of a view:-

Syntax:-

```
CREATE VIEW viewname AS
SELECT newcolumnname….
FROM tablename
WHERE columnname=expression_list;
```

Selecting a data set from a view-

Syntax:-

```
SELECT columnname, columnname
FROM viewname
WHERE search condition;
```

Destroying a view-

Syntax:-

```
DROP VIEW viewname;
```

9.3 ASSIGNMENT:

Answer the following questions

i. Create an index on the table client_master, field client_no.
ii. Create an index on the sales_order, field s_order_no.
iii. Create an composite index on the sales_order_details table for the columns s_order_no and product_no.
iv. Create an composite index ch_index on challan_header table for the columns challan no and s_order_no.
v. Create an unique index on the table salesman_master, field salesman_no.
vi. Drop index ch_index on table challan_header.
vii. Create view on salesman_master whose sal_amt is less than 3500.
viii. Create a view client_view on client_master and rename the columns as name, add1,
ix. add2, city, pcode, state respectively.
x. Select the client names from client_view who lives in city 'Bombay'.
xi. Drop the view client_view.

9.4 SOLUTION TO ASSIGNMENT

i. create index indx_client on Client_master(client_no);

ii. create index indx_order on sales_order(s_order_no);

iii. create unique index indx_sales_order on sales_order_details(order_no, product_no);

iv. create unique index ch_index on challan_header(challan_no, s_order_no);

v. create unique index indx_sales on sales_master(salesman_no);

vi. drop index ch_indx on challan_header;

vii. create view sm_view on sales_master as select * from sales_master where sal_amt < 3500;

viii. create view client_view on client_master and alter table rename column name newname, add1 address……;

ix. select name from clinet_view where city = ‘bombay’;

x. drop view client_view;

Program No: 10

TITLE: CURSOR

10.1 Objective 10.2 Theory 10.3 Example for practice

10.1 OBJECTIVE: Use of cursor

10.2 THEORY AND CONCEPTS:
The central purpose of the Oracle PL/SQL language is to make it as easy and efficient as possible to query and change the contents of tables in a database. You must, of course, use the SQL language to access tables, and each time you do so, you use a *cursor* to get the job done. A cursor is a pointer to a private SQL area that stores information about the processing of a SELECT or data manipulation language (DML) statement (INSERT, UPDATE, DELETE, or MERGE). Cursor management of DML statements is handled by Oracle Database, but PL/SQL offers several ways to define and manipulate cursors to execute SELECT statements.

SELECT-INTO offers the fastest and simplest way to fetch a single row from a SELECT statement. The syntax of this statement is

SELECT select_list INTO variable_list FROM remainder_of_query;

EXAMPLE:

Get the last name for a specific employee ID (the primary key in the employees table):

```
DECLARE
  l_last_name  employees.last_name%TYPE;
BEGIN
  SELECT last_name
    INTO l_last_name
    FROM employees
   WHERE employee_id = 138;

  DBMS_OUTPUT.put_line (
     l_last_name);
END;
```

Fetch an entire row from the employees table for a specific employee ID:

```
DECLARE
   l_employee   employees%ROWTYPE;
BEGIN
   SELECT *
     INTO l_employee
     FROM employees
    WHERE employee_id = 138;

   DBMS_OUTPUT.put_line (
      l_employee.last_name);
END;
```

Using the Cursor FOR Loop
The cursor FOR loop is an elegant and natural extension of the numeric FOR loop in PL/SQL. With a numeric FOR loop, the body of the loop executes once for every integer value between the low and

high values specified in the range. With a cursor FOR loop, the body of the loop is executed for each row returned by the query.

The following block uses a cursor FOR loop to display the last names of all employees in department 10:

```
BEGIN
  FOR employee_rec IN (
      SELECT *
        FROM employees
       WHERE department_id = 10)
  LOOP
    DBMS_OUTPUT.put_line (
       employee_rec.last_name);
  END LOOP;
END;
```

10.3 Example for practice
EXAMPLE:

```
BEGIN
update abcd set no=22 where name='ramDA';
if SQL%FOUND then
dbms_output.put_line('success');
else
dbms_output.put_line('failed');
end if;
end;
```

```
DECLARE
ROWS_AFFECTED CHAR(4);
BEGIN
update abcd set no=23 where name='ram';
ROWS_AFFECTED :=TO_CHAR(SQL%ROWCOUNT);
if SQL%ROWCOUNT>0 then
dbms_output.put_line(ROWS_AFFECTED ||'success');
else
dbms_output.put_line('failed');
end if;
end;
```

Program No: 11

TITLE: TRIGGER

11.1 Objective 11.2 Theory 11.3 Example for practice

11.1 OBJECTIVE: understand the concept of trigger and their uses

11.2 THEORY AND CONCEPTS:
Triggers, which are procedures, stored in PL/SQL (fire) implicitly whenever a table or view is modified or when some user actions or database system actions occur. We can write triggers that fire whenever one of the following operations occurs:
DML statements (INSERT, UPDATE, DELETE) on a particular table or view, issued by any user DDL statements (CREATE or ALTER primarily) issued either by a particular schema/user or by any schema/user in the database. Database events, such as logon/logoff, errors, or startup/shutdown, also issued either by a particular schema/user or by any schema/user in the database. Triggers supplement the standard capabilities of Oracle to provide a highly customized database management system. For example, a trigger can restrict DML operations against a table to those issued during regular business hours.
A trigger has three basic parts:
1. A triggering event or statement
2. A trigger restriction
3. A trigger action

Types of triggers
Row Triggers and Statement Triggers
When you define a trigger, you can specify the number of times the trigger action is to be run:
Once for every row affected by the triggering statement, such as a trigger fired by an **UPDATE** statement that updates many rows
Once for the triggering statement, no matter how many rows it affects

1. Row Triggers
A **row trigger** is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement affects no rows, a row trigger is not run.
2. Statement Triggers
A **statement trigger** is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects, even if no rows are affected. For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only once.

BEFORE and AFTER Triggers

When defining a trigger, you can specify the **trigger timing**—whether the trigger action is to be run before or after the triggering statement. BEFORE and AFTER apply to both statement and row triggers.

BEFORE and AFTER triggers fired by DML statements can be defined only on tables, not on views. However, triggers on the base tables of a view are fired if an INSERT, UPDATE, or DELETE statement is issued against the view. BEFORE and AFTER triggers fired by DDL statements can be defined only on the database or a schema, not on particular tables.

3. Trigger Type Combinations

Using the options listed previously, you can create four types of row and statement triggers:

BEFORE *statement* trigger

Before executing the triggering statement, the trigger action is run.

BEFORE *row* **trigger**

Before modifying each row affected by the triggering statement and before checking appropriate integrity constraints, the trigger action is run, if the trigger restriction was not violated.

AFTER *statement* trigger

After executing the triggering statement and applying any deferred integrity constraints, the trigger action is run.

AFTER *row* trigger

After modifying each row affected by the triggering statement and possibly applying appropriate integrity constraints, the trigger action is run for the current row provided the trigger restriction was not violated. Unlike BEFORE row triggers, AFTER row triggers lock rows.

2. INSTEAD OF Triggers

INSTEAD OF triggers provide a transparent way of modifying views that cannot be modified directly through DML statements (INSERT ,UPDATE, and DELETE). These triggers are called INSTEAD OF triggers because, unlike other types of triggers, Oracle fires the trigger instead of executing the triggering statement.

11.3 Example for practice

```
CREATE TABLE product_price_history
(product_id number(5),
product_name varchar2(32),
supplier_name varchar2(32),
unit_price number(7,2) );

CREATE TABLE product
```

```
(product_id number(5),
product_name varchar2(32),
supplier_name varchar2(32),
unit_price number(7,2) );


CREATE or REPLACE TRIGGER price_history_trigger
BEFORE UPDATE OF unit_price
ON product
FOR EACH ROW
BEGIN
INSERT INTO product_price_history
VALUES
(:old.product_id,
 :old.product_name,
 :old.supplier_name,
 :old.unit_price);
END;


insert into PRODUCT (unit_price, product_id) values (1111,2222);


UPDATE PRODUCT SET unit_price = 800 WHERE product_id = 2222;


CREATE TABLE  tbl_Students

(
   Studentid number,
   Firstname varchar2(50),
   Lastname varchar2(50),
   Email varchar2(100)
)
```

Program No: 12

TITLE: PL/SQL

12.1 Objective 12.2 Theory 12.3 Example for practice

12.1 OBJECTIVE: understand the concept of PL/SQL

10.2 THEORY AND CONCEPTS:
PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are notable facts about PL/SQL:

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line SQL*Plus interface.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in TimesTen in-memory database and IBM DB2.

Features of PL/SQL

PL/SQL has the following features:

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports developing web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages:

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. Dynamic SQL is SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for Developing Web Applications and Server Pages.

12.3 Example for practice
Example 1:

```
DECLARE
```

```
   -- constant declaration
   pi constant number := 3.141592654;
   -- other declarations
   radius number(5,2);
   dia number(5,2);
   circumference number(7, 2);
   area number (10, 2);
BEGIN
   -- processing
   radius := 9.5;
   dia := radius * 2;
   circumference := 2.0 * pi * radius;
   area := pi * radius * radius;
   -- output
   dbms_output.put_line('Radius: ' || radius);
   dbms_output.put_line('Diameter: ' || dia);
   dbms_output.put_line('Circumference: ' || circumference);
   dbms_output.put_line('Area: ' || area);
END;
```

Program 2:

```
DECLARE
   i number(1);
   j number(1);
BEGIN
   << outer_loop >>
   FOR i IN 1..3 LOOP
      << inner_loop >>
      FOR j IN 1..3 LOOP
         dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
      END loop inner_loop;
   END loop outer_loop;
END;
/
```

Example 3:

```
DECLARE
   type namesarray IS VARRAY(5) OF VARCHAR2(10);
   type grades IS VARRAY(5) OF INTEGER;
   names namesarray;
   marks grades;
   total integer;
BEGIN
   names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
   marks:= grades(98, 97, 78, 87, 92);
   total := names.count;
   dbms_output.put_line('Total '|| total || ' Students');
   FOR i in 1 .. total LOOP
      dbms_output.put_line('Student: ' || names(i) || '
      Marks: ' || marks(i));
   END LOOP;
END;
/
```

Example 4:

```
DECLARE
   customer_rec customers%rowtype;
BEGIN
   SELECT * into customer_rec
   FROM customers
   WHERE id = 5;

   dbms_output.put_line('Customer ID: ' || customer_rec.id);
   dbms_output.put_line('Customer Name: ' || customer_rec.name);
   dbms_output.put_line('Customer Address: ' || customer_rec.address);
   dbms_output.put_line('Customer Salary: ' || customer_rec.salary);
END;
/
```

Program No: 13

TITLE: ADMIN & USER PRIVILEGES

13.1 OBJECTIVE: uses of admin & assign user privileges

13.2 THEORY AND CONCEPTS:

A user **privilege** is a right to execute a particular type of SQL statement, or a right to access another user's object. The types of privileges are defined by Oracle.

Roles, on the other hand, are created by users (usually administrators) and are used to group together privileges or other roles. They are a means of facilitating the granting of multiple privileges or roles to users.

System Privileges

There are over 100 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations.

Restricting System Privileges

Because system privileges are so powerful, Oracle recommends that you configure your database to prevent regular (non-DBA) users exercising ANY system privileges (such as UPDATE ANY TABLE) on the data dictionary. In order to secure the data dictionary, ensure that the O7_DICTIONARY_ACCESSIBILITY initialization parameter is set to FALSE. This feature is called the dictionary protection mechanism.

Accessing Objects in the SYS Schema

Users with explicit object privileges or those who connect with administrative privileges (SYSDBA) can access objects in the SYS schema. Another means of allowing access to objects in the SYS schema is by granting users any of the following roles:

- SELECT_CATALOG_ROLE

  This role can be granted to users to allow SELECT privileges on all data dictionary views.

- EXECUTE_CATALOG_ROLE

  This role can be granted to users to allow EXECUTE privileges for packages and procedures in the data dictionary.

- DELETE_CATALOG_ROLE

  This role can be granted to users to allow them to delete records from the system audit table (AUD$).

Additionally, the following system privilege can be granted to users who require access to tables created in the SYS schema:

- SELECT ANY DICTIONARY

This system privilege allows query access to any object in the SYS schema, including tables created in that schema. It must be granted individually to each user requiring the privilege. It is not included in GRANT ALL PRIVILEGES, nor can it be granted through a role.

Object Privileges

Each type of object has different privileges associated with it.

You can specify ALL [PRIVILEGES] to grant or revoke all available object privileges for an object. ALL is not a privilege; rather, it is a shortcut, or a way of granting or revoking all object privileges with one word in GRANT and REVOKE statements. Note that if all object privileges are granted using the ALL shortcut, individual privileges can still be revoked.

Likewise, all individually granted privileges can be revoked by specifying ALL. However, if you REVOKE ALL, and revoking causes integrity constraints to be deleted (because they depend on a REFERENCES privilege that you are revoking), you must include the CASCADE CONSTRAINTS option in the REVOKE statement.

User Roles

A **role** groups several privileges and roles, so that they can be granted to and revoked from users simultaneously. A role must be enabled for a user before it can be used by the user.

Oracle provides some predefined roles to help in database administration. These roles, listed in Table are automatically defined for Oracle databases when you run the standard scripts that are part of database creation. You can grant privileges and roles to, and revoke privileges and roles from, these predefined roles in the same way as you do with any role you define.

Role Name	**Created By (Script)**	**Description**
CONNECT	SQL.BSQ	Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE,CREATE VIEW
RESOURCE	SQL.BSQ	Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER,CREATE TYPE
DBA	SQL.BSQ	All system privileges WITH ADMIN OPTION
Note: The previous three roles are provided to maintain compatibility with previous versions of Oracle and may not be created automatically in future versions of Oracle. Oracle Corporation recommends that you design your own roles for database security, rather than relying on these roles.		
EXP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full and incremental database exports. Includes: SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, and INSERT, DELETE, and UPDATE on the tables SYS.INCVID, SYS.INCFIL,

Role Name	Created By (Script)	Description
		and SYS.INCEXP. Also the following roles:EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
IMP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full database imports. Includes an extensive list of system privileges (use view DBA_SYS_PRIVS to view privileges) and the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
DELETE_CATALOG_ROLE	SQL.BSQ	Provides DELETE privilege on the system audit table (AUD$)
EXECUTE_CATALOG_ROLE	SQL.BSQ	Provides EXECUTE privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
SELECT_CATALOG_ROLE	SQL.BSQ	Provides SELECT privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
RECOVERY_CATALOG_OWNER	CATALOG.SQL	Provides privileges for owner of the recovery catalog. Includes: CREATE SESSION, ALTER SESSION, CREATE SYNONYM, CREATE VIEW, CREATE DATABASE LINK, CREATE TABLE,CREATE CLUSTER, CREATE SEQUENCE, CREATE TRIGGER, and CREATE PROCEDURE
HS_ADMIN_ROLE	CATHS.SQL	Used to protect access to the HS (Heterogeneous Services) data dictionary tables (grants SELECT) and packages (grants EXECUTE). It is granted to SELECT_CATALOG_ROLEand EXECUTE_CATALOG_ROLE such that users with generic data dictionary access also can access the HS data dictionary.
AQ_USER_ROLE	CATQUEUE.SQL	Obsoleted, but kept mainly for release 8.0 compatibility. Provides execute privilege on DBMS_AQ and DBMS_AQIN.
AQ_ADMINISTRATOR_ROLE	CATQUEUE.SQL	Provides privileges to administer Advance Queuing. Includes ENQUEUE ANY QUEUE, DEQUEUE ANY QUEUE, and MANAGE ANY QUEUE, SELECT privileges on AQ tables andEXECUTE privileges on AQ packages.
SNMPAGENT	CATSNMP.SQL	This role is used by Enterprise Manager/Intelligent Agent. Includes ANALYZE ANY and grants SELECT on various views.

Creating a Role

You can create a role using the CREATE ROLE statement, but you must have the CREATE ROLE system privilege to do so. Typically, only security administrators have this system privilege.

You must give each role you create a unique name among existing usernames and role names of the database. Roles are not contained in the schema of any user. In a database that uses a multibyte character set, Oracle recommends that each role name contain at least one single-byte character. If a role name contains only multibyte characters, the encrypted role name/password combination is considerably less secure.

The following statement creates the clerk role, which is authorized by the database using the password bicentennial:

```
CREATE ROLE clerk IDENTIFIED BY bicentennial;
```

The IDENTIFIED BY clause specifies how the user must be authorized before the role can be enabled for use by a specific user to which it has been granted. If this clause is not specified, or NOT IDENTIFIED is specified, then no authorization is required when the role is enabled. Roles can be specified to be authorized by:

- The database using a password
- An application using a specified package
- Externally by the operating system, network, or other external source
- Globally by an enterprise directory service

These authorizations are discussed in following sections.

Later, you can set or change the authorization method for a role using the ALTER ROLE statement. The following statement alters the clerk role to specify that the user must have been authorized by an external source before enabling the role:

```
ALTER ROLE clerk IDENTIFIED EXTERNALLY;
```

Role Authorization by an Application

The INDENTIFIED USING *package_name* clause lets you create an application role, which is a role that can be enabled only by applications using an authorized package. Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role and specify which PL/SQL package is authorized to enable the role.

The following example indicates that the role admin_role is an application role and the role can only be enabled by any module defined inside the PL/SQL package hr.admin.

```
CREATE ROLE admin_role IDENTIFIED USING hr.admin;
```

When enabling the user's default roles at login as specified in the user's profile, no checking is performed for application roles.

Role Authorization by an External Source

The following statement creates a role named accts_rec and requires that the user be authorized by an external source before it can be enabled:

```
CREATE ROLE accts_rec IDENTIFIED EXTERNALLY;
```

Dropping Roles

In some cases, it may be appropriate to drop a role from the database. The security domains of all users and roles granted a dropped role are immediately changed to reflect the absence of the dropped role's privileges. All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all users' default role lists.

Because the creation of objects is not dependent on the privileges received through a role, tables and other objects are not dropped when a role is dropped.

You can drop a role using the SQL statement DROP ROLE. To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

The following statement drops the role CLERK:

DROP ROLE clerk;

Granting System Privileges and Roles

You can grant system privileges and roles to other users and roles using the GRANT statement. The following privileges are required:

- To grant a system privilege, you must have been granted the system privilege with the ADMIN OPTION or have been granted the GRANT ANY PRIVILEGE system privilege.
- To grant a role, you must have been granted the role with the ADMIN OPTION or have been granted the GRANT ANY ROLE system privilege.

> **Note:**
>
> You cannot grant a roll that is IDENTIFIED GLOBALLY to anything. The granting (and revoking) of global roles is controlled entirely by the enterprise directory service.

The following statement grants the system privilege CREATE SESSION and the accts_pay role to the user jward:

GRANT CREATE SESSION, accts_pay TO jward;

Granting the ADMIN OPTION

A user or role that is granted a privilege or role specifying the WITH ADMIN OPTION clause has several expanded capabilities:

- The grantee can grant or revoke the system privilege or role to or from *any* user or other role in the database. Users cannot revoke a role from themselves.
- The grantee can further grant the system privilege or role with the ADMIN OPTION.
- The grantee of a role can alter or drop the role.

In the following statement, the security administrator grants the new_dba role to michael:

GRANT new_dba TO michael WITH ADMIN OPTION;

The user michael cannot only use all of the privileges implicit in the new_dba role, but can grant, revoke, or drop the new_dba role as deemed necessary. Because of these powerful capabilities, exercise caution when granting system privileges or roles with the ADMIN OPTION. Such privileges are usually reserved for a security administrator and rarely granted to other administrators or users of the system.

When a user creates a role, the role is automatically granted to the creator with the ADMIN OPTION

Creating a New User with the GRANT Statement

Oracle allows you to create a new user with the GRANT statement. If you specify a password using the IDENTIFIED BY clause, and the username/password does not exist in the database, a new user with that username and password is created. The following example creates ssmith as a new user while granting ssmith the CONNECT system privilege:

```
GRANT CONNECT TO ssmith IDENTIFIED BY p1q2r3;
```

Granting Object Privileges

You also use the GRANT statement to grant object privileges to roles and users. To grant an object privilege, you must fulfill one of the following conditions:

- You own the object specified.
- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.
- The WITH GRANT OPTION clause was specified when you were granted the object privilege by its owner.

The following statement grants the SELECT, INSERT, and DELETE object privileges for all columns of the emp table to the users jfee and tsmith:

```
GRANT SELECT, INSERT, DELETE ON emp TO jfee, tsmith;
```

To grant all object privileges on the salary view to the user jfee, use the ALL keyword, as shown in the following example:

```
GRANT ALL ON salary TO jfee;
```

Specifying the GRANT OPTION

Specify WITH GRANT OPTION to enable the grantee to grant the object privileges to other users and roles. The user whose schema contains an object is automatically granted all associated object privileges with the GRANT OPTION. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any users in the database, with or without the GRANT OPTION, or to any role in the database.
- If both of the following are true, the grantee can create views on the table and grant the corresponding privileges on the views to any user or role in the database.

- o The grantee receives object privileges for the table with the GRANT OPTION.
- o The grantee has the CREATE VIEW or CREATE ANY VIEW system privilege.

The GRANT OPTION is not valid when granting an object privilege to a role. Oracle prevents the propagation of object privileges through roles so that grantees of a role cannot propagate object privileges received by means of roles.

Granting Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege allows users to grant and revoke any object privilege on behalf of the object owner. This provides a convenient means for database and application administrators to grant access to objects in any schema without requiring that they connect to the schema. This eliminates the need to maintain login credentials for schema owners so that they can grant access to objects, and it reduces the number of connections required during configuration.

This system privilege is part of the Oracle supplied DBA role and is thus granted (with the ADMIN OPTION) to any user connecting AS SYSDBA (user SYS). As with other system privileges, the GRANT ANY OBJECT PRIVILEGE system privilege can only be granted by a user who possesses the ADMIN OPTION.

When you exercise the GRANT ANY OBJECT PRIVILEGE system privilege to grant an object privilege to a user, if you already possess the object privilege with the GRANT OPTION, then the grant is performed in the usual way. In this case, you become the grantor of the grant. If you do not possess the object privilege, then the object owner is shown as the grantor, even though you, with the GRANT ANY OBJECTPRIVILEGE system privilege, actually performed the grant.

For example, consider the following. User adams possesses the GRANT ANY OBJECT PRIVILEGE system privilege. He does not possess any other grant privileges. He issues the following statement:

```
GRANT SELECT ON hr.employees TO blake WITH GRANT OPTION;
```

If you examine the DBA_TAB_PRIVS view, you will see that hr is shown as being the grantor of the privilege:

```
SQL> SELECT GRANTEE, OWNER, GRANTOR, PRIVILEGE, GRANTABLE
  2>    FROM DBA_TAB_PRIVS
  3>    WHERE TABLE_NAME = 'EMPLOYEES' and OWNER = 'HR';

GRANTEE  OWNER GRANTOR PRIVILEGE    GRANTABLE
--------         -----      -------       -----------         ----------
BLAKE          HR         HR           SELECT            YES
```

Now assume that blake also has the GRANT ANY OBJECT PRIVILEGE system. He, issues the following statement:

```
GRANT SELECT ON hr.employees TO clark;
```

In this case, when you again query the DBA_TAB_PRIVS view, you see that blake is shown as being the grantor of the privilege:

```
GRANTEE  OWNER GRANTOR  PRIVILEGE    GRANTABLE
--------      -----     --------        --------         ----------
BLAKE         HR        HR          SELECT          YES
CLARK         HR       BLAKE       SELECT           NO
```

This is because blake already possesses the SELECT privilege on hr.employees with the GRANT OPTION.

Revoking System Privileges and Roles

You can revoke system privileges and roles using the SQL statement REVOKE.

Any user with the ADMIN OPTION for a system privilege or role can revoke the privilege or role from any other database user or role. The revoker does not have to be the user that originally granted the privilege or role. Users with GRANT ANY ROLE can revoke *any* role.

The following statement revokes the CREATE TABLE system privilege and the accts_rec role from tsmith:

REVOKE CREATE TABLE, accts_rec FROM tsmith;

Revoking Object Privileges

The REVOKE statement is used to revoke object privileges. To revoke an object privilege, you must fulfill one of the following conditions:

- You previously granted the object privilege to the user or role.
- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.

You can only revoke the privileges that you, the grantor, directly authorized, not the grants made by other users to whom you granted the GRANT OPTION. However, there is a cascading effect. The object privilege grants propagated using the GRANT OPTION are revoked if a grantor's object privilege is revoked.

Assuming you are the original grantor, the following statement revokes the SELECT and INSERT privileges on the emp table from the users jfee and tsmith:

REVOKE SELECT, insert ON emp FROM jfee, tsmith;

Program No: 14

TITLE: MS ACCESS

14.1 Objective 14.2 Theory 14.3 Exercise

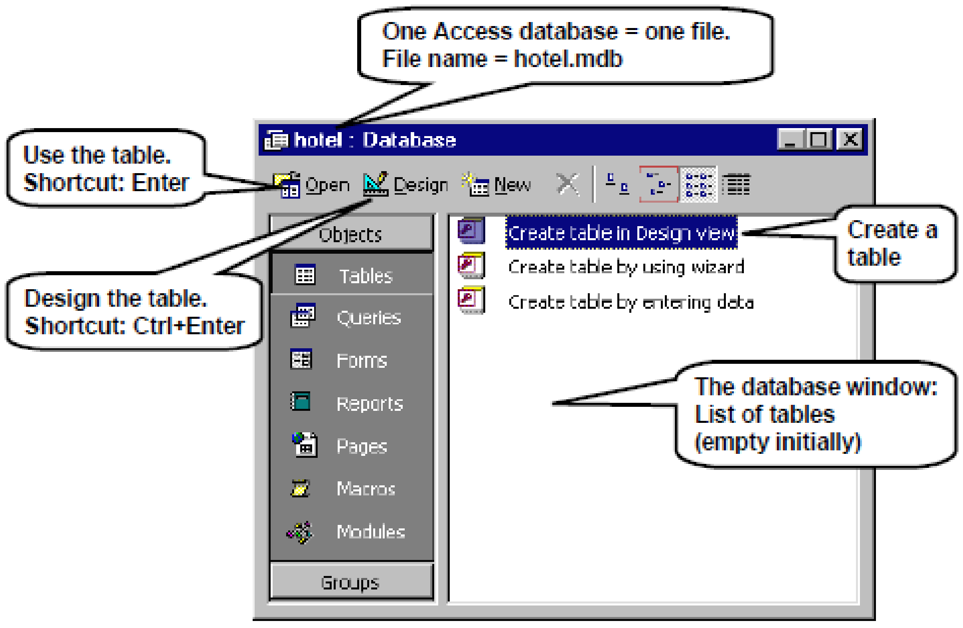
14.1 OBJECTIVE: learn to realize a data model as a relational database in Microsoft Access

14.2 THEORY AND CONCEPTS:
Microsoft Access is a database management system (DBMS) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately.
Microsoft Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.
In Microsoft Access a database consists of one single file. The file contains all the tables of the database, the relationships (the crow's feet), queries (computed tables), forms (user windows), and many other things.

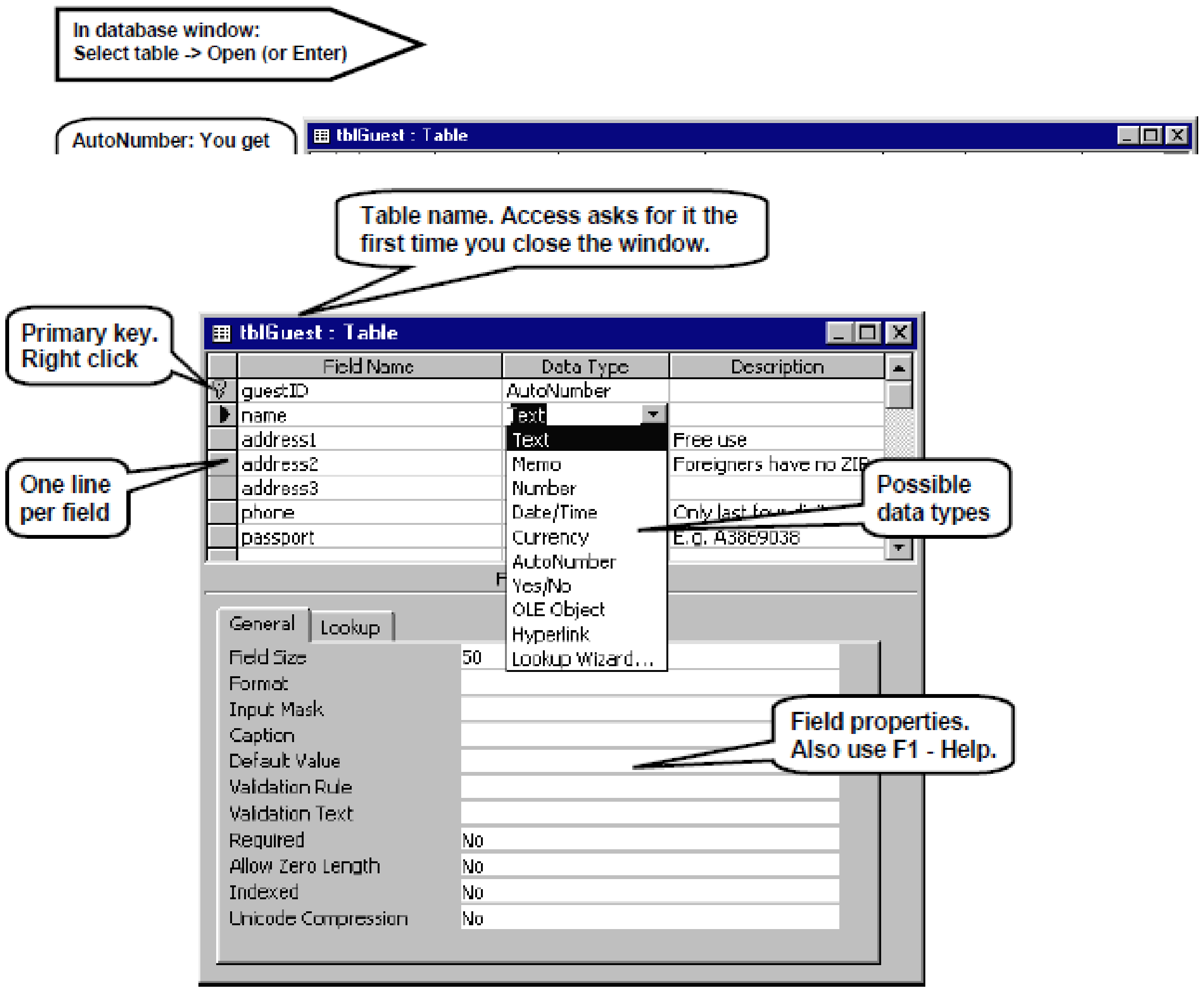
14.3 Exercise
Create the database:
1.Locate the Access program. Depending on the way the system is set up, you may find it under Pro-grams -> Microsoft Access or Programs -> Micro-soft Office -> Microsoft Access.
2.In Access 97 and 2000: Open Access and ask for a "blank" database. In Access 2003: Open Access and click the New icon (under the File menu). Then click Blank database in the help area to the far right.
3.Access now asks where to store the new database. Select the folder you want and give the database the name hotel (or hotel.mdb).
The screen now shows the database window.

Figure: The Access database window

Define a table:
2. Double click on Create table in Design view.
2. Fill in all the field lines according to the attributes in the guest table (see the figure). All the fields



are of type Text, except the guestID which is of type AutoNumber.



Field Name	Data Type	Description
guestID	AutoNumber	
name	Text	
address1	Text	Free use
address2	Memo	Foreigners have no ZIP
address3	Number	
phone	Date/Time	Only last fou...
passport	Currency	E.g. A3869038
	AutoNumber	
	Yes/No	
	OLE Object	
	Hyperlink	
	Lookup Wizard...	

Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	No
Unicode Compression	No

Key fields
3. Right-click somewhere in the guestID line. Then select Primary Key. Access now shows that the field is the key.
4. Close the window. Access asks you for the name of the table. Call it tblGuest.

Enter data:
1. Select the guest table in the database window. Click Open or just use Enter.
2. Enter the guests shown on the below Figure

Figure: Enter the Data in User mode
Close and reopen the database

1. Close the large Access window.

2. Find your database file (hotel.mdb) in the file fold-ers. Use Enter or double click to open it.

3. The file may not be safe. Do you want to open it? Your database is safe, so answer Open.

4. Unsafe expressions are not blocked. Do you want to block them? You want full freedom, so answer No.

5. Access warns you one more time whether you want to open. Say Open or Yes.

Shortcut keys for data entry:
F2: Toggles between selecting the entire field and selecting a data entry point.
Shift+F2: Opens a small window with space for the entire field. Useful for entering long texts into a field that is shown only partly in the table. How-ever, the text cannot be longer than you specified in the table definition.
Alt+ArrowDown: Opens a combo box. Choose with the arrows and Enter.

Experiment Mapping with CO, PO, PSO

EX-1	PO1							PO8	PO9	PO10	PO11	PO12
CO	1	1	1	1	1	1	1	1	1	1	1	1
PO	3	3										3
	PSO1	PSO2	PSO3	PSO4								
PSO	3			3								

EX-2	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	3	3	3	3	3	3	3	3	3	3	3	3
PO	3	3	3									3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-3	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	3	3	3	3	3	3	3	3	3	3	3	3
PO	3	3	3	3							3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-4	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	4	4	4	4	4	4	4	4	4	4	4	4
PO	3	3	3	3		3					3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-5	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	3	3	3	3	3	3	3	3	3	3	3	3
PO	3	3	3									3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-6	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	2	2	2	2	2	2	2	2	2	2	2	2
PO	3	3	3		3				3		3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3	3	3								

EX-7	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	4	4	4	4	4	4	4	4	4	4	4	4
PO	3	3	3	3	3	3			3		3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3		3								

EX-8	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	4	4	4	4	4	4	4	4	4	4	4	4
PO	3	3	3	3								3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-9	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	5	5	5	5	5	5	5	5	5	5	5	5
PO	3	3	3									3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-10	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	5	5	5	5	5	5	5	5	5	5	5	5
PO	3	3	3	3		3					3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3										

EX-11	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	5	5	5	5	5	5	5	5	5	5	5	5
PO	3	3	3	3		3		3	3		3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3		3								

EX-12	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO	6	6	6	6	6	6	6	6	6	6	6	6
PO	3				3						3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3		3								

EX-13	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PO	3	3	3		3	3		3	3			3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3	3	3								

EX-14	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PO	3	3	3		3						3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3	3	3								

EX-15	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PO	3	3	3	3	3	3	3	3	3	3	3	3
	PSO1	PSO2	PSO3	PSO4								
PSO	3	3	3	3								