

RELATIONAL DATABASE MANAGEMENT SYSTEM- II (RDBMS-II)



BACHELOR'S OF TECHNOLOGY (Computer Science Engineering)

SUBMITTED BY:-

Malaika

Roll no:-1803545

Semester:- 6th

CTIEMT

SUBMITTED TO:-

Mr Abhishiek Bhardwaj

(Assistant professor)

Table of Contents

Practical No.	Aim Of the Practical's	Page No.	Date	Signature
3.	Introduction to SQL (Structured Query Language)	1-3	16-03-20	
4.	Study and Usage of open source data mining tool: Weka	4-5	19-03-20	
5.	Study and usage of any object-oriented or object relational databases management software.	6-11	23-03-20	
6.	Study of web databases, Creating and use Web database in PHP.	12-14	26-03-20	
7.	Study and usage of backup and recovery features of database management, purpose of backup and Recovery	15-17	03-04-20	

Practical: 1

Normalisation

Case study 1: Health history report.

Health History Report

Pet-ID	Pet Name	Pet Type	Pet Age	Owner	Visit date	Procedure
246	ROVER	DOG	12	SAM COOK	JAN 13/2002 MAR 27/2002 APR 02/2002	01 - Rabies Vaccination 10 - Examine & Treat wound 05 - Heart Wurm Test
298	SPOT	DOG	2	TERRY KIM	JAN 21/2002 MAR 10/2002	08 - Tetanus Vaccination 05 - Heart Wurm Test
341	MORRIS	CAT	4	SAM COOK	JAN 28/2002 JAN 13/2002	01 - Rabies Vaccination 01 - Rabies Vaccination
519	THEEDY	BIRD	2	TERRY KIM	APR 30/2002 APR 30/2002	20 - Annual Checkup 12 - Eye Wash

Date..... Page No. 1

Normalization


Case Study 1 :-
Health History Report.

UNF:- Pet [Pet-ID, Pet-Name, Pet-type, Pet-age, Owner, (Visitdate, procedure-no, procedure-name)]

1NF:- Pet [Pet-ID, Pet-Name, Pet-type, Pet-age, Owner]
Pet-Visit [Pet-ID, Visitdate, procedure-no, procedure-name]

2NF:- Pet [Pet-ID, Pet-Name, Pet-type, Pet-age, Owner]
Pet-Visit [Pet-ID, Visitdate, procedure-no]
Procedure [Procedure-no, Procedure-Name]

3NF:- Pet [Pet-ID, Pet-Name, Pet-type, Pet-age, Owner]
Pet-Visit [Pet-ID, Visitdate, procedure-No]
Procedure [Procedure-no, Procedure-Name]

 Signature of Supervisor :

Case study 2: Invoice.

Case Study 2

Invoice

HILLTOP ANIMAL HOSPITAL
INVOICE # 987

DATE : JAN 13 - 2002

MR. RICHARD COOK
123 THIS STREET
MY CITY, ONTARIO
252666

PET	PROCEDURE	AMOUNT
ROYER	Rabbit Vaccination	30.00
MORRIS	Rabbit Vaccination	24.00
TOTAL		54.00
TAX (8%)		4.32
AMOUNT OWING		<u>58.32</u>

Date..... Page No..... 2

Case Study 2:-
Invoice

UNF:- invoice [invoice no, invoice date, cust name, cust address (pet name, procedure, amount)]

1NF:- invoice [invoice no, invoice date, cust name, cust address]
invoice-pet [invoice no, pet id, pet name, procedure, amount]

2NF:- invoice [invoice no, invoice date, cust name, cust address]
invoice-pet [invoice no, pet id, procedure, amount]
pet [pet id, pet name]

3NF:- invoice [invoice no, invoice date, cust no (FK)]
invoice-pet [invoice no (FK), pet id (FK), procedure, amount]
pet [pet id, pet name]
Customer [cust no, cust name, cust street, cust city, cust postal]

Signature of Supervisor :

Case study 3: ABC manufacturing customer order and product application.

Case Study 3

ABC manufacturing Customer Order and Product Application :-

ABC Manufacturing Order form

ORDER # 9932

MR. SD. KURTZ

123 THAT STREET

Toronto, Ontario

A9B 5C7

PHONE: (416) 879-0015 (416) 780-3241

DATE: Aug 30, 2002

SHIPPING ADDRESS

456 NO STREET

Hamilton, Ontario

L6K 5J4

Customer discount: 3%

ITEM #	Product Code	Description	Qty	Base Price	Filler	Price/unit	Amount
1	FR223	Half Size Refrigerator	2	0	2	750.99	1501.98
2	TB101	Patio Table	5	2	3	150.00	450.00
3	CH089	Patio Chair	20	0	20	35.00	700.00
TOTAL							2651.98
DISCOUNT AMT							76.56
AMOUNT OWING							2575.42

Date: Page No: 3

Case Study 3 :-

ABC Manufacturing Customer Order and Product Application :-

UNF:- Order [Order no, Order dt, Cust name, Cust address, ship address, phone 1, phone 2, Cust disc, Citem #, prod code, prod disc, Qty, filled, Price]

1NF:- Order [Order no, Order dt, Cust name, Cust address, ship address, Phone 1, Phone 2, Cust disc]
Order items [Order no, prod code, prod disc, Qty, filled, Price]

2NF:- Order [Order no, Order dt, Cust name, Cust address, ship address, Phone 1, Phone 2, Cust disc]
Order items [Order no, prod code, prod disc, Qty, filled, Price]
Product [prod code, prod disc, prod price]

3NF:- Order [Order no, Order dt, Cust no (FK)]
Order items [Order no, prod code, prod disc, Qty, filled]
Customer [Cust no, Cust name, Cust add, Cust ship, Ship add, Cust phone 1, Phone 2, Cust disc, Cust address]
Product [prod code, prod disc, prod price, prod desc]

Signature of Supervisor:

Case study 4: Gallery customer history form.

Gallery Customer History form.

Customer Name

Jackson, Elizabeth
123 - 4th Avenue
Fonthill, ON
L3T 4S4

Phone (206) 284-6783

Purchases Made

Artist	Title	Purchase date	Sales Price
03- Carol Channing	Laugh With Teeth	09/11/2000	7000.00
15- Dennis Hays	South towards Emerald Sea	05/11/2000	1800.00
03- Carol Channing	At the Movies	02/14/2002	5550.00
15- Dennis Hays	South towards Emerald Sea	01/15/2003	2200.00

Date..... Page No..... 4.....

Case Study 4:-
Gallery Customer History form.

UNF:- Customer [Customer no, Customer Name, Customer Address, Customer phone, (Artist ID, Artist Name, Art title, purchase date, Price)]

1NF:- Customer [Customer no, Customer Name, Customer Address, Customer phone]
Cust Art [Customer no, Art code, Purchase date, Artist ID, Artist Name, Art title, Price]

2NF:- Customer [Customer no, Customer Name, Customer Address, Customer Phone]
Cust Art [Customer no, Art Code, Purchase date, Price]
Art [Art Code, title, artist ID, Artist Name]

3NF:- Customer [Customer no, Customer name, Customer street, Customer city, Customer prov, Customer V postcd, Customer phone]
Cust Art [Customer no, Art Code, purchase date, Price]
Art [Art Code, Art title, Art ID (PK)]
Artist [Artist ID, Artist Name, Artist phone]

Signature of Supervisor :

Case study 5: Good news grocers.

Case Study 5

Good News Grocers

Department	Product Code	Aisle Number	Price	Unit of Measure
Produce	4081	1	0.35	lb
Produce	4027	1	0.90	ea
Produce	4108	1	1.99	lb
Butcher	331100	5	3.50	lb
Butcher	331105	5	2.40	lb
Butcher	332110	5	5.00	lb
Freezer	411100	6	1.00	ea
Freezer	521101	6	1.00	ea
Freezer	866503	6	5.00	ea

Date: Page No. 5

Case Study 5:- Good News Grocers.

UNF:- department [dept, Aisle Number (Product Code, Price, Measure-unit)]

1NF:- Department [dept No, dept Name, Aisle Number]
dept-product [department, Product Code, Price, Measure-unit]

2NF:- Department [dept No, dept Name, Aisle Number]
Product [Product Code, Price, Measure-unit, dept No]

3NF:- Department [dept No, dept Name, Aisle Number]
Product [Product Code, Price, Measure-unit, dept No]

Signature of Supervisor :

Practical: 3

Aim: - Introduction to SQL (Structured Query Language).

SQL is a standard language for accessing and manipulating databases.

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL Standard -

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

Using SQL in Your Web Site

To build a web site that shows data from a database, you will need:

- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- To use a server-side scripting language, like PHP or ASP
- To use SQL to get the data you want
- To use HTML / CSS to style the page

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Example

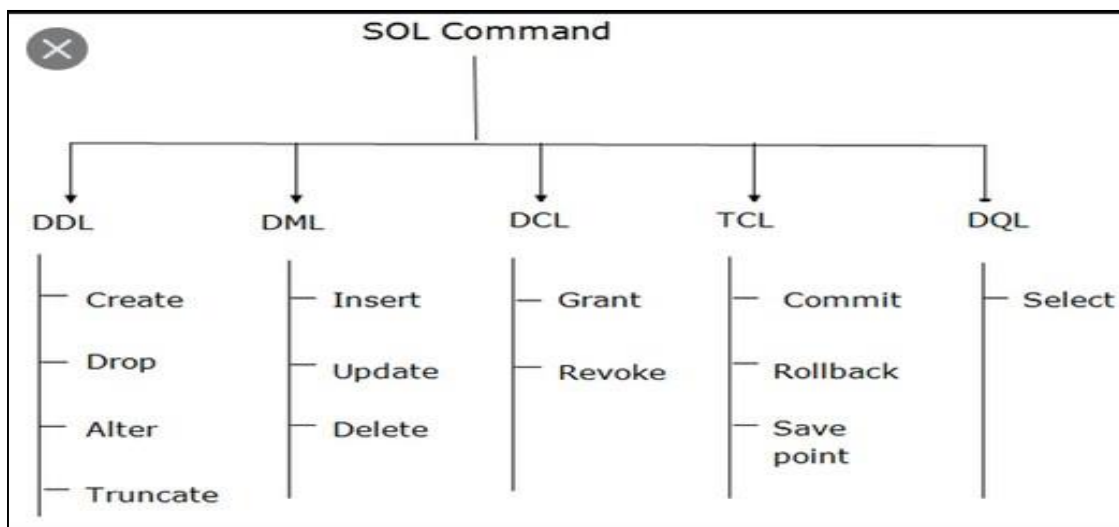
```
SELECT * FROM Customers;
```

Every table is broken up into smaller entities called fields. The fields in the Customers table consist of Customer ID, Customer Name, Contact Name, Address, City, Postal Code and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.

A record, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table. A record is a horizontal entity in a table.

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

COMMANDS USED IN SQL:



CREATE COMMAND IN SQL:

Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

```
create table bloodbank(
blood_type varchar(10),
blood_bagno number(10)primary key,
plasma varchar(10),
cost number(10),
wbc varchar(10),
rbc varchar(10),
hemoglobin varchar(10));
```

Table created.

Name	Null?	Type
BLOOD_TYPE		VARCHAR2(10)
BLOOD_BAGNO	NOT NULL	NUMBER(10)
PLASMA		VARCHAR2(10)
COST		NUMBER(10)
WBC		VARCHAR2(10)
RBC		VARCHAR2(10)
HEMOGLOBIN		VARCHAR2(10)

INSERT COMMAND:

Workspace

Enter SQL, PL/SQL and SQL*Plus statements.

```
insert into donor_bank values('gagan',22,101,'male','no','2','02-june-17');
select * from donor_bank;
```

Execute Load Script Save Script Cancel

SELECT COMMAND:

```
select * from donor_bank;
```

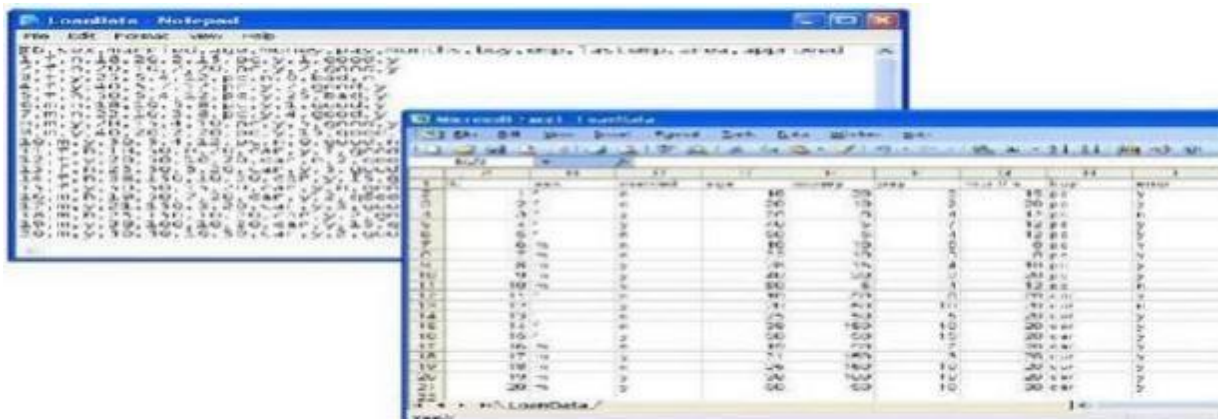
Execute Load Script Save Script Cancel

NAME	AGE	DID	SEX	DISEASE	QUANTITY	DATE_HONOUR
gagan	22	101	male	no	2	02-JUN-17

Practical: 4

Aim: Study and Usage of open source data mining tool: Weka.

WEKA:- Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection.



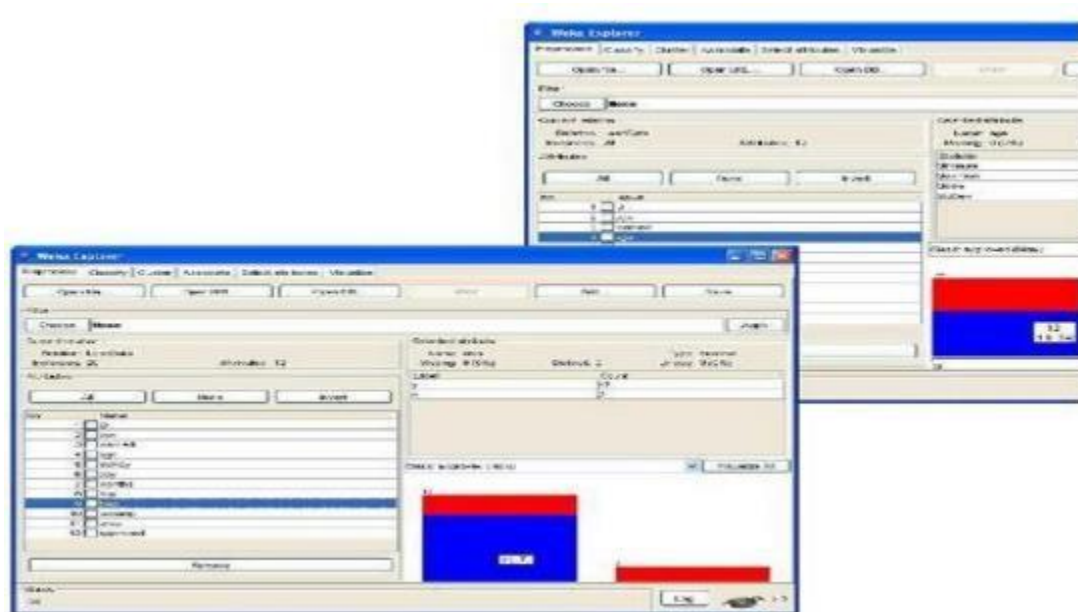
Data collection

Solution:

Step1 - Choose the file by Open File dialog:

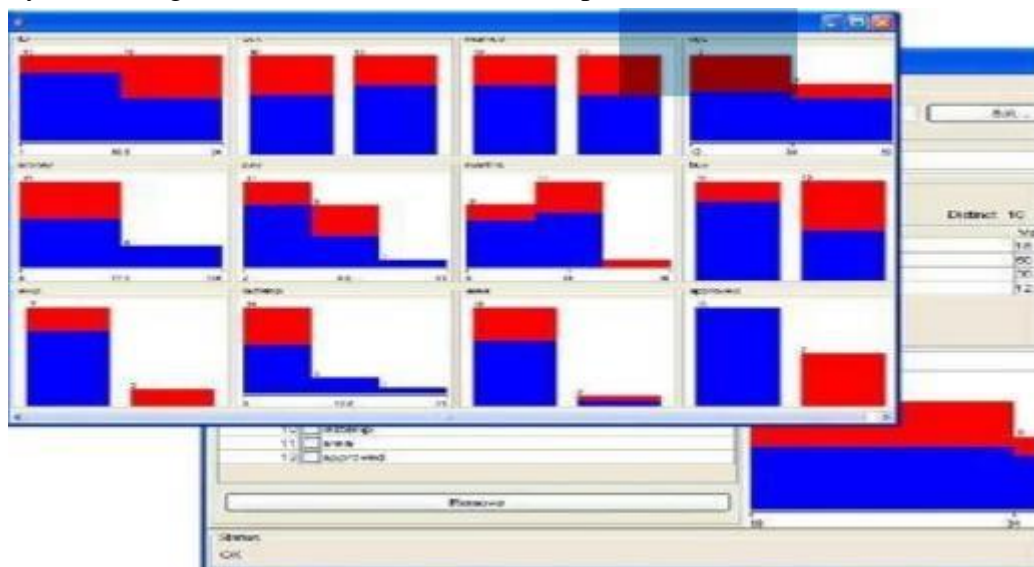


Step 2 - Choose the field to visualize the results. Example - emp as shown below:-



Visualize data collection

By clicking at all the complete result is as shown below as:



Results

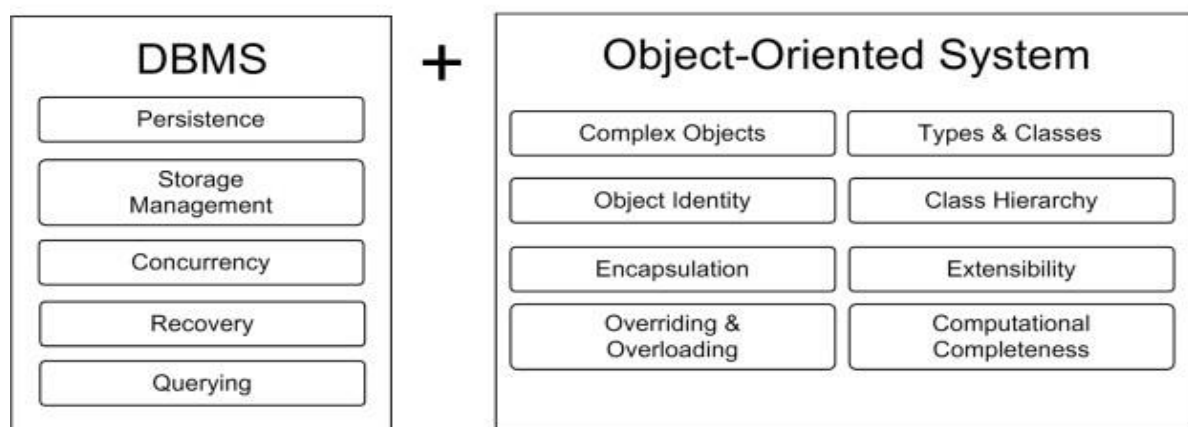
Practical: 5

Aim:-Study and usage of any object-oriented or object relational databases management software.

Motivation: The relational model is the basis of many commercial relational DBMS products (e.g., DB2, Informix, Oracle, Sybase) and the structured query language (SQL) is a widely accepted standard for both retrieving and updating data. The basic relational model is simple and mainly views data as tables of rows and columns. The types of data that can be stored in a table are basic types such as integer, string, and decimal. Relational DBMSs have been extremely successful in the market. However, the traditional RDBMSs are not suitable for applications with complex data structures or new data types for large, unstructured objects, such as CAD/CAM, Geographic information systems, multimedia databases, imaging and graphics. The RDBMSs typically do not allow users to extend the type system by adding new data types. They also only support first-normal-form relations in which the type of every column must be atomic, i.e., no sets, lists, or tables are allowed inside a column. Due to the new needs in database systems, a number of researches for OODBMS have begun in the early 80s.

Concept & Features : While a relational database system has a clear specification given by Codd, no such specification existed for object-oriented database systems even when there were already products in the market. A consideration of the features of both object-oriented systems and database management systems has led to a definition of an object-oriented database, which was presented at the First International Conference on Deductive, and Object-oriented Databases in the form of a manifesto in 1989. This "manifesto" distinguishes between the mandatory, optional and open features of an object-oriented database. The mandatory features, which must be present if the system is to be considered (in the opinion of the manifesto authors) to be an object-oriented database, are defined in the following two paragraphs. The first part describes features of object-oriented system, as the second part features of database system

Object-Oriented Database



Mandatory features of object-oriented systems Support for complex objects

A complex object mechanism allows an object to contain attributes that can themselves be objects. In other words, the schema of an object is not in first-normal -form. Examples of attributes that can comprise a complex object include lists, bags, and embedded objects.

Object identity

Every instance in the database has a unique identifier (OID), which is a property of an object that distinguishes it from all other objects and remains for the lifetime of the object. In object-oriented systems, an object has an existence (identity) independent of its value.

Encapsulation

Object-oriented models enforce encapsulation and information hiding. This means, the state of objects can be manipulated and read only by invoking operations that are specified within the type definition and made visible through the public clause. In an object-oriented database system encapsulation is achieved if only the operations are visible to the programmer and both the data and the implementation are hidden.

Support for types or classes

Type : in an object-oriented system, summarizes the common features of a set of objects with the same characteristics. In programming languages types can be used at compilation time to check the correctness of programs.

Class : The concept is similar to type but associated with run-time execution. The term class refers to a collection of all objects with the same internal structure (attributes) and methods. These objects are called instances of the class. Both of these two features can be used to group similar objects together, but it is normal for a system to support either classes or types and not both.

Class or type hierarchies

Any subclass or subtype will inherit attributes and methods from its superclass or supertype.

Overriding, Overloading and Late Binding Overloading:

A class modifies an existing method, by using the same name, but with a different list, or type, of parameters. Overriding: The implementation of the operation will depend on the type of the object it is applied to. Late binding: The implementation code cannot be referenced until run-time.

Computational Completeness

SQL does not have the full power of a conventional programming language. Languages such as Pascal or C are said to be computationally complete because they can exploit the full capabilities of a computer. SQL is only relationally complete, that is, it has the full power of relational algebra. Whilst any SQL code could be rewritten as a C++ program, not all C++ programs could be rewritten in SQL. For this reason most relational database applications involve the use of SQL embedded within a conventional programming language. The problem with this approach is that whilst SQL deals with sets of records, programming languages tend to work on a record at a time basis. This difficulty is known as the impedance mismatch. Object-oriented databases attempt to provide a seamless join between program and database and hence overcome the impedance mismatch. To make this possible the data manipulation language of an object-oriented database should be computationally complete

Mandatory features of database systems

A database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated.

Thus, a database system contains the five following features:

Persistence: As in a conventional database, data must remain after the process that created it has terminated. For this purpose data has to be stored permanently on secondary storage.

Secondary Storage Management: Traditional databases employ techniques, which manage secondary storage in order to improve the performance of the system. These are usually invisible to the user of the system.

Concurrency: The system should provide a concurrency mechanism, which is similar to the concurrency mechanisms in conventional databases.

Recovery: The system should provide a recovery mechanism similar to recovery mechanisms in conventional databases

Making OOPL a Database:

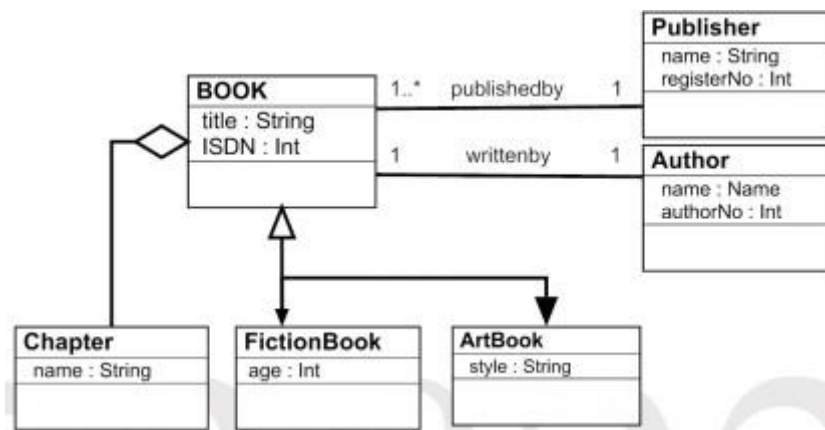
Basically, an OODBMS is an object database that provides DBMS capabilities to objects that have been created using an object-oriented programming language (OOPL). The basic principle is to add persistence to objects and to make objects persistent. Consequently application programmers who use OODBMSs typically write programs in a native OOPL such as Java, C++ or Smalltalk, and the language has some kind of Persistent class, Database class, Database Interface, or Database API that provides DBMS functionality as, effectively, an extension of the OOPL. Object-oriented DBMSs, however, go much beyond simply adding persistence to any one object-oriented programming language. This is because, historically, many object-oriented DBMSs were built to serve the market for computer-aided design/computer-aided manufacturing (CAD/CAM) applications in which features like fast navigational access.

DBMSs, therefore, support advanced object-oriented database applications with features like support for persistent objects from more than one programming language, distribution of data, advanced transaction models, versions, schema evolution, and dynamic generation of new types.

The following subsection describes object data modeling and the persistency concept in OODB.

Object data modeling

An object consists of three parts: structure (attribute, and relationship to other objects like aggregation, and association), behavior (a set of operations) and characteristic of types (generalization/serialization). An object is similar to an entity in ER model; therefore we begin with an example to demonstrate the structure and relationship



The structure of an object Book is defined as following:

```

Class Book{
Title: String;
ISDN: Int;
publishedBy :Publisher inverse publish;
writtenBy: Author inverse write;
chapterSet: Set<Chapter>;
}

Class Author{
name: String;
authorNo: Int;
write : Book inverse writteBy;

```

```
}
```

Attributes are like the fields in a relational model. However in the Book example we have, for attributes publishedBy and writtenBy, complex types Publisher and Author, which are also objects. Attributes with complex objects, in RDNS, are usually other tables linked by keys to the employee table.

Relationships:

publish and writtenBy are associations with I:N and 1:1 relationship; composed_of is an aggregation (a Book is composed of chapters). The 1:N relationship is usually realized as attributes through complex types and at the behavioral level.

For example;

```
Class Publisher {
```

```
.....
```

```
Publish: Set<Book> inverse publishedBy;
```

```
.....
```

```
Method
```

```
Insert(Bo
```

```
Ok book){
```

```
Publish.a
```

```
Dd(book);
```

```
}
```

Generalization/Serialization: is the is_a relationship, which is supported in OODB through class hier- archy. An ArtBook is a Book, therefore the ArtBook class is a subclass of Book class. A subclass inherits all the attribute and method of its superclass.

```
Class
```

```
ArtBook
```

```
Extends
```

```
Book
```

```
{
```

```
Style:String;
```


Message: means by which objects communicate, and it is a request from one object to another to execute one of its methods. For example: `Publisher_object.insert ("Rose", 123,...)`
i.e. request to execute the insert method on a Publisher object) **Method:** defines the behavior of an object.

Methods can be used to change state by modifying its attribute values . to query the value of selected attributes The method that responds to the message example is the method insert defined in the Publisher class.

Persistence of objects Persistence, as mentioned before, means that certain program components survive the termination of the program. Thus these components have to be stored permanently on secondary storage. Typically, persistence or non-persistence is specified at object creation time. There are two possible ways to make an object persistent:

(1) explicitly call built-in function persistence . certain objects are persistent (2) automatically make object of persistent types persistent . all objects are persistent There are several object-oriented DBMSs in the market (e.g., Gemstone, Objectivity/DB, ObjectStore, Ontos, O2, Itasca, Matisse). These products all support an object-oriented data model. Specifically, they allow the user to create a new class with attributes and methods, have the class inherit attributes and methods from superclasses, create instances of the class each with a unique object identifier, retrieve the instances either individually or collectively and load and run methods. Most of these OODBs support a unified programming language and database language. That is, one language (e.g., C++ or Smalltalk) in which to do both general-purpose programming and database management

Practical: 6

Aim: Creating and use Web database in PHP.

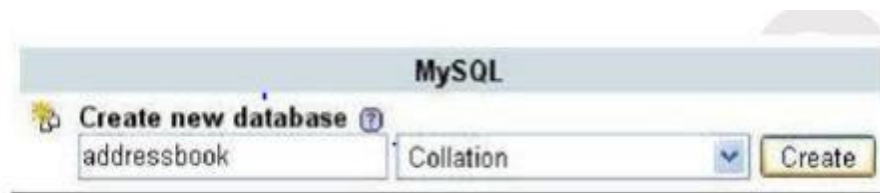
<u>ID</u>	<u>Title</u>	<u>First Name</u>	<u>Surname</u>
1	Mr	Test	Name
2	Mrs	Second	Test

TABLE

Solution:- Creating a database in PHP using PHPmyadmin:-



CREATE NEW DATABASE



This is where you type a name for your database. We're going to create a simple Address Book, so type that into the text box

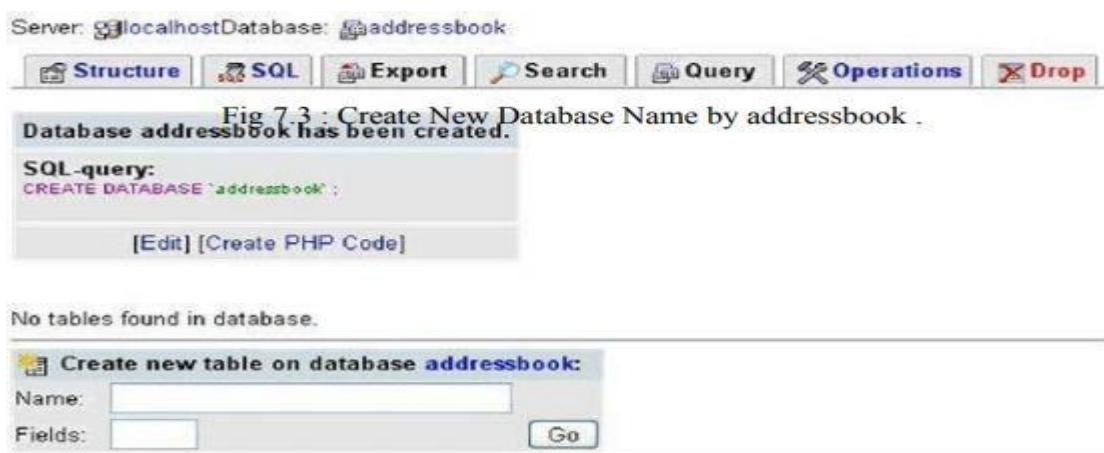


Fig 7.3 : Create New Database Name by addressbook .

After you have typed a name for your new database, click the “Create” button. You will be taken to a new area

Database has been Created .

In this new area, you can create a Table to go in your database. At the moment, as it says, there are Notables found in the database. But the database itself has been created.

To create a new table, type a name for it in the box at the bottom. You can also type a number for the Fields textbox. The fields are the columns, remember, and will be things like first_name, surname, address, etc. You can always add more later, but just type 4 in there. In fact, type it out exactly as it is below.

Create new table on database addressbook:

Name:

Fields:

Table on database

When you've finished, click the Go button. Another, more complex, area will appear:

Server: localhost Database: addressbook Table: tbl_address_book

Field	Type	Length/Values	Collation	Attributes	Null	Default	Extra
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		

Table comments:

Table type: Collation:

Add 1 field(s)

* If field type is "enum" or "set", please enter the values using this format: 'a','b','c'...

If you ever need to put a backslash ("\") or a single quote (") amongst those values, backslash it (for example 'xyz' or 'a\b').

** For default values, please enter just a single value, without backslash escaping or quotes, using this format: a

Tables columns

Fill the data as shown below:

Field	Type	Function	Null	Value
ID	smallint(6)	<input type="button" value="v"/>		1
First_Name	varchar(50)	<input type="button" value="v"/>		Test
Surname	varchar(50)	<input type="button" value="v"/>		Name
Address	tinytext	<input type="button" value="v"/>		12 Test Street

Insert values in database

Using a database in PHP:-

```
<?PHP

$user_name="root";

$password="";

$database="addressbook";

$server="127.0.0.1";

$db_handle=mysql_connect($server,$user_name,$password);

$db_found=mysql_select_db($database,$db_handle);

if($db_found)

{

    $SQL="SELECT* FROM tb-addres_book";

    $result=mysql_query($SQL);

    while($db_field=mysql_fetch_assoc($result))

    {

        print$db_field[„ID”].”<BR>”;

        print$db_field[„First_Name”].”<BR>”;

        print$db_field[„Surname”].”<BR>”;

    }

    Mysql_close($db_handle);

}

else

{

    Print”database NOT found”;mysql_close($db_handle);

}

?>
```


Practical: 7

Aim: Study and usage of backup and recovery features of database management, purpose of backup and recovery.

Backup and recovery refers to the process of **backing up** data in case of a loss and setting up systems that allow that data **recovery** due to data loss. **Backing up** data requires copying and archiving computer data, so that it is accessible in case of data deletion or corruption.

A **backup** is a copy of data. This copy includes important parts of your database such as the control file and data files. A backup is a safeguard against unexpected data loss and application errors; should you lose your original data, you can use the backup to make it available again.

Backups are divided into physical backups and logical backups. Physical backups, which are the primary concern of this guide, are copies of physical database files. In contrast, logical backups contain data that you extract using the Oracle Export utility and store in a binary file. You can use logical backups to supplement physical backups. You can make physical backups using either the Oracle8i Recovery Manager utility or O/S utilities.

To restore a physical backup is to reconstruct it and make it available to the Oracle server. To recover a restored data file is to update it using redo records, i.e., records of changes made to the database after the backup was taken. If you use Recovery Manager (RMAN), you can also recover restored data files with an incremental backup, which is a backup of a data file that contain only changed data blocks.

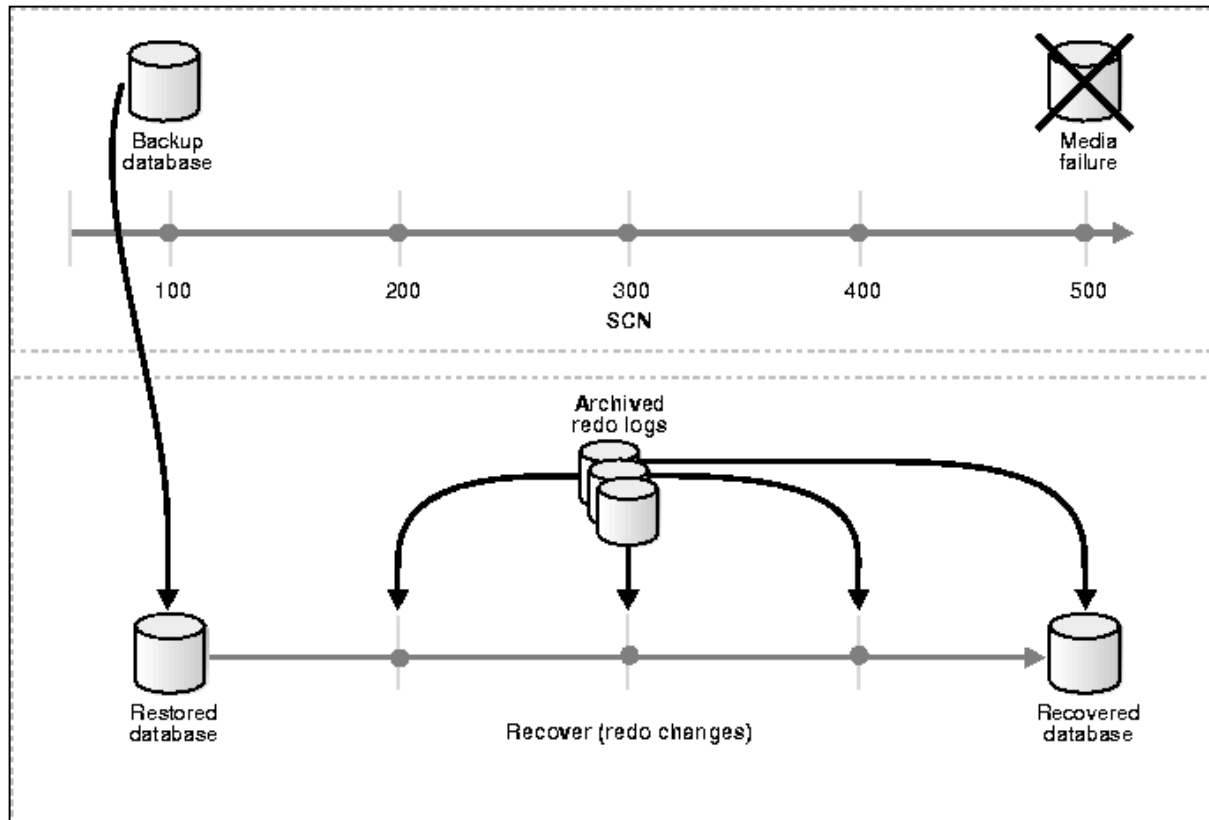
Oracle performs crash recovery and instance recovery automatically after an instance failure. Instance recovery is an automatic procedure that involves two distinct operations: rolling forward the backup to a more current time by applying online redo records and rolling back all changes made in uncommitted transactions to their original state.

In contrast to instance recovery, media recovery requires you to issue recovery commands. If you use RMAN, then you issue the **recover** command to apply archived redo logs or incremental backups to the data files. RMAN automatically selects the appropriate incremental backups or redo logs and applies them. If you use SQL*Plus, you can issue the RECOVER or ALTER DATABASE RECOVER statements to apply the archived log.

FEATURES OF BACKUPS OF DATABASE MANAGEMENT SYSTEM:-

Physical **backup** extracts data from physical storage (usually from disk to tape). Operating system is an example of physical **backup**.

Logical **backup** extracts data using SQL from the **database** and store it in a binary file. Logical **backup** is used to restore the **database** objects into the **database**.



In every type of instance or media recovery (except media recovery using RMAN incremental backups), Oracle sequentially applies redo data to data blocks. Oracle uses information in the control file and data file headers to ascertain whether recovery is necessary.

Recovery has two parts: rolling forward and rolling back. When Oracle rolls forward, it applies redo records to the corresponding data blocks. Oracle systematically goes through the redo log to determine which changes it needs to apply to which blocks, and then changes the blocks. For example, if a user adds a row to a table, but the server crashes before it can save the change to disk, Oracle can use the redo record for this transaction to update the data block to reflect the new row.

During the rolling back phase, Oracle applies rollback segments to the datafiles. The rollback information is stored in *transaction tables*. Oracle searches through the table for uncommitted transactions, undoing any that it finds. For example, if the user never committed the SQL statement that added the row, then Oracle will discover this fact in a transaction table and undo the change.

If the database is mounted during recovery, rollback occurs only when the database is opened. If the database is open and a table space is offline during recovery, rollback for that table space occurs when the table spaces is brought online

Type of Recovery	Function
<u>time-based recovery</u>	Recovers the data up to a specified point in time.
<u>cancel-based recovery</u>	Recovers until you issue the CANCEL command.
<u>change-based recovery</u>	Recovers until the specified SCN.
<u>log sequence recovery</u>	Recovers until the specified log sequence number.

FEATURES OF RECOVERY OF DATABASE MANAGEMENT SYSTEM:-

The **DBMS** must not permit some operation of the transaction T to be applied to the **database** while other operations of T are not. This basically may happen if a transaction fails after executing some of its operations but before executing all of them.

PURPOSE OF BACKUPS AND RECOVERY:-

BACKUPS:-

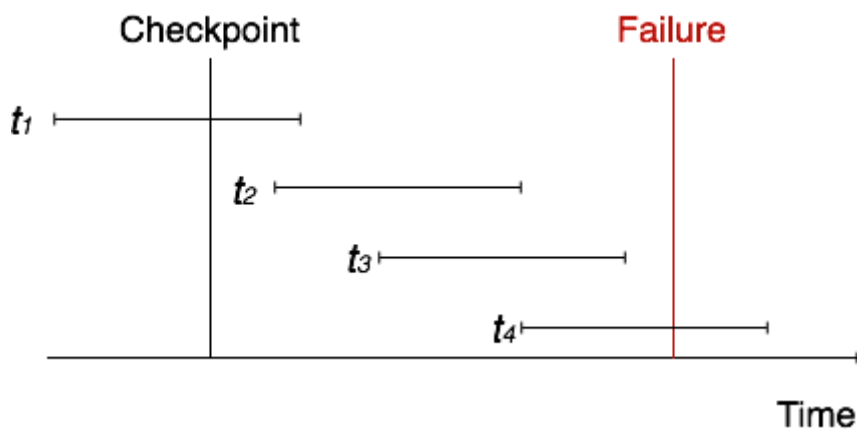
A data **backup** is the result of copying or archiving files and folders for the **purpose** of being able to restore them in case of data loss. Data loss can be caused by many things ranging from computer viruses to hardware failures to file corruption to fire, flood, or theft.

The main reason for data **backup** is to save **important** files if a system crash or hard drive failure occurs. There should be additional data **backups** if the original **backups** result in data corruption or hard drive failure. Additional **backups** are necessary if natural or man-made disasters occur.

RECOVERY:-

The **recovery** system reads the logs backwards from the end to the last checkpoint. It maintains two lists, an undo-list and a redo-list. If the **recovery** system sees a log with $\langle T_n, \text{Start} \rangle$ and $\langle T_n, \text{Commit} \rangle$ or just $\langle T_n, \text{Commit} \rangle$, it puts the transaction in the redo-list.

When a system with concurrent transactions crashes and recovers, it behaves in the following manner –



[illegible]