

## **B.E.**

Fourth Semester Examination, May-2008

# **DATA BASE MANAGEMENT SYSTEM**

**Note :** Attempt any five questions.

**Q. 1. (a) What are data models? Explain their different types.**

**Ans. Data Models :**

**Hierarchical Model:** The hierarchical data model organizes data in a tree structured. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1:N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parentchild relationship is one to many. This restricts a child segment to having only one parent segment

**Network Model :**

The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them. Thus, the complete network of relationships is represented by several pairwise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow). Usually, a set defines a 1:M relationship, although 1: 1 is permitted. The CODASYL network model is based on mathematical set theory.

**Relational Model :**

(RDBMS - relational database management system) A database based on the relational model developed by E.F. Codd. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organised in tables. A table is a collection of records and each record in a table contains the same fields.

### **Properties of Relational Tables :**

- Values Are Atomic
- Each Row is Unique
- Column Values Are of the Same Kind
- The Sequence of Columns is Insignificant The Sequence of Rows is Insignificant
- Each Column Has a Unique Name

Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables. For example, an "orders" table might contain (customer-ID, product-code) pairs and a "products" table might contain (product-code, price) pairs so to calculate a given customer's bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables. This can be extended to joining multiple tables on multiple fields. Because these relationships are only specified at retrieval time, relational databases are classed as dynamic database management system. The RELATIONAL database model is based on the Relational Algebra.

### **Object/relational Model :**

Object/relational database management systems (ORDBMSs) add new object storage capabilities to the relational systems at the core of modern information systems. These new facilities integrate management of traditional fielded data, complex objects such as time-series and geospatial data and diverse binary media such as audio, video, images, and applets. By encapsulating methods with data structures, an ORDBMS server can execute complex analytical and data manipulation operations to search and transform multimedia and other complex objects.

As an evolutionary technology, the object/relational (OR) approach has inherited the robust transcend performance management features of its relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and data definition languages (DDLs) while assimilating new object-management possibilities. Query and procedural languages and call interfaces in ORDBMSs are familiar: SQL3, vendor procedural languages, and ODBC, JDBC, and proprietary call interfaces are all extensions of RDBMS languages and interfaces. And the leading vendors are, of course, quite well known: IBM, Informix, and Oracle.

### **Q. 1. (b) What are advantages of DBMS over file processing system?**

#### **Ans. Advantages of DBMS over file processing system :**

The DBMS has a number of advantages as compared to traditional computer file-based processing approach. The DBA must keep in mind these benefits or capabilities during designing databases, coordinating and monitoring the DBMS. The main advantages of DBMS are described below.

**Data Consistency:** By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value (new value of item) is immediately available to all users. If the DBMS has controlled redundancy, the database system enforces consistency. It means that when data item that appears more than once. In the database is updated, the DBMS automatically updates each occurrence of a data item in the database. However some database systems

do not support enforce data consistency.

**Integration of data :**

In DBMS, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

**Database access language :**

Many DBMSs provide SQL as database access language e.g., SQL to access data from multiple tables of a database. Every DBMS has its own version of SQL.

**Development of application :**

The cost and time for developing new applications is also reduced. The DBMS provides tools that can be used to develop application programs. For example, some wizards are available to generate Forms and Reports. Stored procedures (stored on server side) also reduce the size of application programs.

**More Advantages :**

Data sharing, Controlling Data Redundancy, Data Security, Data Atomicity, Creating Forms, Control Over Concurrency, Backup and Recovery Procedures, Data Independence.

**Q. 2. (a) What is client-server architecture? What is it used for?**

**Ans. Client server architecture :**

The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interpretability, and scalability as compared to centralized, main-frame, time sharing computing.

A client is defined as a requester of services and a server is defined as the provider of services. A single machine can be both a client and a server depending on the software configuration.

**\* Client machines :**

- Run own copy of an operating system.
- Run one or more applications using the client machine's CPU, memory.
- Application communicates with DBMS server running on server machine through a Database

**Driver :**

- Database driver (middleware) makes a connection to the DBMS server over a network.

**- Examples of clients :**

PCs with MS Windows operating system. Forms and reports developed e.g.

**\* Oracle Developer/2000, etc.**

- Server Machines:
  - Run own copy of an operating system.
  - Run a Database Management System that manages a database.
- Provides a Listening daemon that accepts connections from client machines and submits transactions to

DBMS on behalf of the client machines.

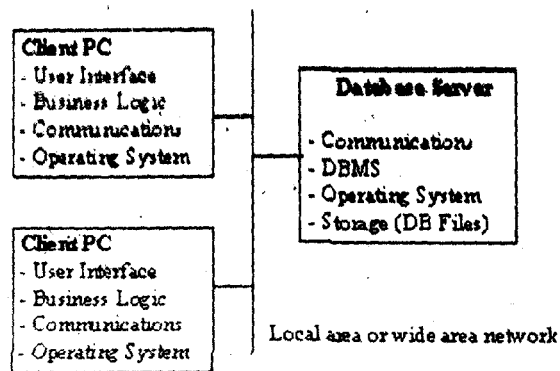
**Examples :**

Sun Sparc server running UNIX operating system. RDBMS such as Oracle Server, Sybase, Informix, DB2, etc. PC with Windows NT operating system.

**\* Middleware:**

- Small portion of software that sits between client and server.
- Establishes a connection from the client to the server and passes commands (e.g., SQL) between them.

**"Classic" Client/server Architecture :**



**Q. 2. (b) Explain three level architecture of database system.**

**Ans. Three tier architecture :**

Three-tier is a client-server architecture in which the user interface, functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.

The three-tier model is considered to be a software architecture and a software design pattern.

Apart from the usual advantages of modular software with well defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or technology change. For example, a change of operating system from Microsoft Windows to Unix in the presentation tier would only affect the user interface code.

Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic may consist of one or more separate modules running on a workstation or application server, and an RDBMS on a database server or mainframe contains the computer data storage logic. The middle tier may be multi-tiered itself (in which case the overall architecture is called an "n-tier architecture").

The 3-Tier architecture has the following three tiers:

### Presentation tier :

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

### Application Tier (Business Logic/Logic Tier)

The logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

### Data tier :

This tier consists of Database Servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

### Presentation tier :

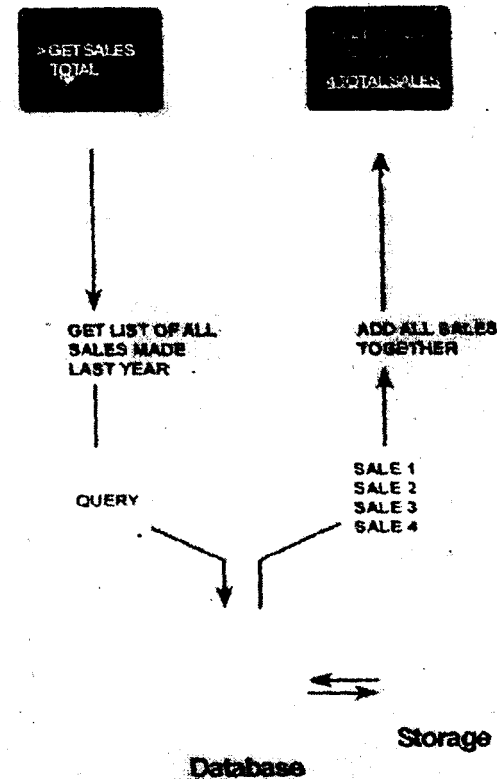
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and result to something the user can understand.

### Logic tier :

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

### Data tier :

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



### Q. 3. (a) What are direct files? What are different hashing techniques?

#### Ans. Direct files :

#### Random (or Direct) :

A randomly organised file contains records arranged physically without regard to the sequence of the primary key. Records are loaded to disk by establishing a direct relationship between the Key of the record and its address on the file, normally by use of a formula (or algorithm) that converts the primary Key to a physical disk address. This relationship is also used for retrieval.

The use of a formula (or algorithm) is known as 'Key Transformation' and there are several techniques that can be used : Division Taking Quotient

Division Taking Remainder

Truncation

Folding

Squaring

Radix Conversion

These methods are often mixed to produce a unique address (or location) for each record (key). The production of the same address for two different records is known as a synonym.

Random files show a distinct advantage where :

Hit Rate is low

Data cannot be batched or sorted

Fast response is required.

#### **Catering for expansion**

A normal tendency of master files is to expand. Records may be increased in size or may be added. Even if the total size or number of records does not increase, there will almost inevitably be changes.

Although it is usual to update files on disk by overlay there must be provision for additions and preferably some means of re-utilising storage arising from deletion.

#### **Overflow arises from :**

- (i) A record being assigned to a block that is already full.
- (ii) A record being expanded so that it can no longer be accommodated in the block.

There are a number of methods for catering for expansion :

- (i) Specifying less than 100% block packing density on initial load.
- (ii) Specifying less than 100% cylinder packing density.
- (iii) Specifying extension blocks (usually at the end of the file).

The first method is only effective and efficient if the expansion is regular. Where localised expansion occurs to any extent, even in one block, this system will fail.

The other two methods have the result of allowing space for first and second level overflow, respectively. Extension blocks are used when all other overflow facilities have been exhausted. Ideally, first level overflow is situated on the same cylinder as the overflowed block hence there is no penalty incurred in terms of head movement. Second level overflow normally consists of one or more blocks at the end of the file. If a significant number of accesses to second level overflow are made, run-time will increase considerably. There is the need to reorganise the file to bring record back from overflow.

Hashing Techniques: Key mod N

N is the table size.

- Key mod P

P is the smallest prime number  $\geq N$ .

- Truncation (Substringing)

Select any "appropriate" digits of the given key.

- Folding

Folding by boundary and folding by shifting.

- Squaring

... followed by truncating a portion of the result.

- Radix Conversion

The key is converted to base 10.

- Alphabetical Keys

Alphabetic or alphanumeric key values can be input to a hashing function if the values are interpreted as integers.

A hashing function that has a large number of collisions or synonyms is said to exhibit primary clustering.

**Aim :** reduce the number of collisions

**Solution 1:** Change the hashing functions.

**Solution 2 :** reduce the load factor (Le., # of stored records / total # storage locations)

- Load factor is a measure of storage utilization.
- As the load factor increases, the likelihood of a collision increases.
- e.g., compare with the number of cars in city traffic.
- Time-space Tradeoff:
- Increased load factor
- more space, decreased load factor more collisions
- Collisions typically increase rapidly when the packing factor goes beyond about 90 percent.

**Q. 3. (b) What are B-tree index files? Explain their structure.**

**Ans. Tree file organisation :**

B-tree indices are similar to B+-tree indices.

- Difference is that B-tree eliminates the redundant storage of search key values.
- In B+-tree of Figure 11.11, some search key values appear twice.
- A corresponding B-tree of Figure 11.18 allows search key values to appear only once.
- Thus we can store the index in less space.

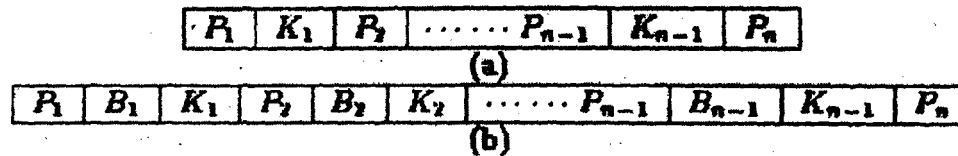


Figure 11.8 : Leaf and nonleaf node of a B-tree.

## 2. Advantages :

- Lack of redundant storage (but only marginally different).
- Some searches are faster (key may be in non-leaf node).

## 3 Disadvantages :

- Leaf and non-leaf nodes are of different size (complicates storage).
- Deletion may occur in a non-leaf node (more complicated)

Generally, the structural simplicity of B+-tree is preferred.

## 1. Insertions and deletions :

Insertion and deletion are more complicated, as they may require splitting or combining nodes to keep the tree balanced. If splitting or combining are not required, insertion works as follows :

- Find leaf node where search key value should appear.
- If value is present, add new record to the bucket.
- If value is not present, insert value in leaf node (so that search keys are still in order). Create a new bucket and insert the new record.

If splitting or combining are not required, deletion works as follows :

### Deletion :

Find record to be deleted, and remove it from the bucket.

- If bucket is now empty, remove search key value from leaf node.

## Q. 4. (a) What are various operators of relational algebra? Explain taking examples.

Ans. **Operations on relational algebra :** The relational algebra is a procedural query language.

### \* Six fundamental operations :

- select (unary)
- project (unary)
- rename (unary)
- Cartesian product (binary)
- union (binary)



- set-difference (binary)

\* **Several other operations, defined in terms of the fundamental operations :**

- set -intersection :
- natural join
- division
- assignment

\* **Operations produce a new relation as a result.**

**1. The select operation :**

Select selects tuples that satisfy a given predicate. Select is denoted by a lowercase Greek sigma ( $\sigma$ ), with the predicate appearing as a subscript. The argument relation is given in parentheses following the ( $\sigma$ ).

For example, to select tuples (rows) of the borrow relation where the branch is " SFU", we would write

$$\sigma_{bname = \text{"SFU"}}(\text{borrow})$$

Let Figure 3.3 be the borrow and branch relations in the banking example.

bname	Ican#	ename	amount	hname	assets	bcity
Downtown	17	Jones	1000	Downtown	9,000,000	Vanecover
Loughted.Mall	23	Smith	2000	Loughted Mall	21,000,000.	Burnaby
SPU	13	Fiayner	1600	SPU	17,000,000	Burnahy

**Fig. : The borrow and branch relations.**

The new relation created as the result of this operation consists of one tuple : (SFU, 15, Hayes, 1500).

We allow comparisons using  $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$  and  $\geq$  in the selection predicate.

We also allow the logical connectives  $\vee$  (or) and  $\wedge$  (and). For example :

$$\sigma_{bname = \text{"Down town"} \wedge \text{amount} > 1200}(\text{barrow})$$

ename	banker
Fayon	Jones
Johnson	Johnson

Suppose there is one more relation, client, shown in Figure 3.4, with the scheme

$$\text{Client\_scheme} = (\text{ename}, \text{banker})$$

we might write

$$\sigma_{bname = \text{ban ker}}(\text{client})$$

to find clients who have the same name as their banker.

## 2. The project operation :

Project copies its argument relation for the specified attributes only. Since a relation is a set, duplicate rows are eliminated.

Projection is denoted by the Greek capital letter pi ( $\Pi$ ). The attributes to be copied appear as subscripts.

For example, to obtain a relation showing customers and branches, but ignoring amount and loan#, we write

$$\Pi_{\text{bname, cname}}(\text{borrow})$$

We can perform these operations on the relations resulting from other operations.

To get the names of customers having the same as their bankers,

$$\Pi_{\text{bname}}(\sigma_{\text{cname}=\text{ban ker}}(\text{client}))$$

Think of select as taking rows of a relation, and project as taking columns of a relation.

## 3. The Cartesian product operation :

The Cartesian product of two relations is denoted by a cross ( $\times$ ), written

$$r_1 \times r_2 \text{ for relations } r_1 \text{ and } r_2$$

The result of  $r_1 \times r_2$  is a new relation with a tuple for each possible pairing of tuples from  $r_1$  and  $r_2$ .

In order to avoid ambiguity, the attribute names have attached to them the name of the relation from which they came. If no ambiguity will result, we drop the relation name.

The result  $\text{client} \times \text{customer}$  is a very large relation. If  $r_1$  has  $n_1$  tuples and  $r_2$  has  $n_2$  tuples,  $r = r_1 \times r_2$  will have  $n_1 \cdot n_2$  tuples.

The resulting scheme is the concatenation of the schemes of  $r_1$  and  $r_2$ , with relation names added as mentioned.

To find the clients of banker Johnson and the city in which they live, we need information in both  $\text{client}$  and  $\text{customer}$  relations. We can get this by writing

$$\Pi_{\text{balance}}(\text{deposit}) - \Pi_{\text{deposit.balance}}( )$$

However, the  $\text{customer.cname}$  column contains customers of bankers other than Johnson. (Why?)

We want rows where  $\text{client.cname} = \text{customer.cname}$ . So we can write

$$\sigma_{\text{client.cname}=\text{customer.cname}}(\sigma_{\text{ban ker} = \text{"Johnson"}}(\text{client} \times \text{customer}))$$

to get just these tuples.

Finally, to get just the customer's name and city, we need a projection :

$$\Pi_{\text{client.ename,ccity}}(\sigma_{\text{client.ename=customer.ename}}(\sigma_{\text{banker="Johnson"}}(\text{client} \times \text{customer})))$$

#### 4. The rename operation :

The rename operation solves the problems that occurs with naming when performing the Cartesian product of a relation with itself.

Suppose we want to find the names of all the customers who live on the same street and in the same city as Smith.

We can get the street and city of Smith by writing

$$\Pi_{\text{strsst,ccity}}(\sigma_{\text{ename="smith"}}(\text{customer}))$$

To find other customers with the same information, we need to reference the customer relation again :

$$\sigma_P(\text{customer} \times (\Pi_{\text{strsst,ccity}}(\sigma_{\text{ename="smith"}}(\text{customer}))))$$

where P is a selection predicate requiring street and ccity values to be equal.

**Problem :** How do we distinguish between the two street values appearing in the Cartesian product, as both come from a customer relation?

**Solution :** Use the rename operator, denoted by the Greek letter rho ( $\rho$ ).

We write

$$\rho_x(r)$$

to get the relation under the name of x.

If we use this to rename one of the two customer relations we are using, the ambiguities will disappear.

$$\Pi_{\text{customer.ename}}(\sigma_{\text{cust2.straat=customer.street} \times \text{cust2.city=customer.ccity}}(\text{customer} \times (\Pi_{\text{strsst,ccity}}(\sigma_{\text{ename="smith"}}(\rho_{\text{cust2}}(\text{customer}))))))$$

#### 5. The union operation :

The union operation is denoted U as in set theory. It returns the union (set union) of two compatible relations.

For a union operation  $r \cup s$  to be legal, we require that

- r and s must have the same number of attributes.
- The domains of the corresponding attributes must be the same.

To find all customers of the SFU branch, we must find everyone who has a loan or an account or both at the branch.

We need both borrow and deposit relations for this :

$$\Pi_{\text{ename}}(\sigma_{\text{bname} = \text{"SFU"}}(\text{borrow})) \cup \Pi_{\text{ename}}(\sigma_{\text{bname} = \text{"SFU"}}(\text{deposit}))$$

As in all set operations, duplicates are eliminated, giving the relation of figure 3.5(a).

ename	ename
Hyer	Adams
Adams	

Figure 3.5: The union and set-difference operations

**6. The set difference operation :** Set difference is denoted by the minus sign (-). It finds tuples that are in one relation, but not in another.

Thus  $r - s$  results in a relation containing tuples that are in  $r$  but not in  $s$ .

To find customers of the SFU branch who have an account there but no loan, we write

$$\Pi_{\text{ename}}(\sigma_{\text{bname} = \text{"SFU"}}(\text{deposit})) - \Pi_{\text{ename}}(\sigma_{\text{bname} = \text{"SFU"}}(\text{borrow}))$$

The result is shown in Figure 3.5(b).

We can do more with this operation. Suppose we want to find the largest account balance in the bank.  
Strategy:

- Find a relation  $r$  containing the balances not the largest.
- Compute the set difference of  $r$  and the deposit relation.

To find  $r$ , we write

$$\Pi_{\text{deposit.balance}}(\sigma_{\text{deposit.balance} < d.\text{balance}}(\text{deposit} \times \rho_d(\text{deposit})))$$

This resulting relation contains all balances except the largest one. (See Figure 3.6(a)).

Now we can finish our query by taking the set difference:

$$\Pi_{\text{balance}}(\text{deposit}) - \Pi_{\text{deposit.balance}}(\sigma_{\text{deposit.balance} < d.\text{balance}}(\text{deposit} \times \rho_d(\text{deposit})))$$

Figure 3.6(b) shows the result.

Balance	Balance
400	1300
500	
700	

Figure 3.6: Find the largest account balance in the bank.

**Q. 4. (b) For what typical queries, do we use division operator?**

**Ans. Division operator :** Division, denoted  $\div$ , is suited to queries that include the phrase "for all".

Suppose we want to find all the customers who have an account at all branches located in Brooklyn.

Strategy: think of it as three steps.

We can obtain the names of all branches located in Brooklyn by

$$r_1 = \Pi_{\text{bname}}(\sigma_{\text{beity}=\text{"Brooklyn"}}(\text{branch}))$$

Figure 3.19 in the textbook shows the result.

We can also find all **ename, bname** pairs for which the customer has an account by

$$r_2 = \Pi_{\text{ename, bname}}(\text{deposit})$$

Figure 3.20 in the textbook shows the result.

Now we need to find all customers who appear in  $r_2$  with every branch name in  $r_1$ .

The divide operation provides exactly those customers :

$$\Pi_{\text{ename, bname}}(\text{deposit}) \div \Pi_{\text{bname}}(\sigma_{\text{beity}=\text{"Brooklyn"}}(\text{branch}))$$

which is simply  $r_2 \div r_1$

Formally,

- Let  $r(R)$  and  $s(S)$  be relations.
- Let  $S \subseteq R$
- The relation  $r \div s$  is a relation on scheme  $R-S$ .
- A tuple  $t$  is in  $r \div s$  if for every tuple  $t_s$  in  $s$  there is a tuple  $t_r$  in  $r$  satisfying both of the following :

$$t_r[S] = t_s[S] \quad (3.2.1)$$

$$t_r[R-S] = t[R-S] \quad (3.2.2)$$

- These conditions say that the  $R-S$  portion of a tuple  $t$  is in  $r \div s$  if and only if there are tuples with the portion and the  $S$  portion in  $r$  for every value of the  $S$  portion in relation  $S$ .

We will look at this explanation in class more closely.

The division operation can be defined in terms of the fundamental operations.

$$r \div s = \Pi_{n-s}(r) - \Pi_{n-s}((\Pi_{n-s}(r) \times s) - r)$$

**Q. 4. (c) How can expressions be made in tuple calculus?**

**Ans. Tuples calculus :**

The tuple relational calculus is a nonprocedural language. (The relational algebra was procedural.)

1. We must provide a formal description of the information desired.
2. A query in the tuple relational calculus is expressed as

$$\{t | P(t)\}$$

i.e. the set of tuples  $t$  for which predicate  $P$  is true.

3. We also use the notation

-  $t[a]$  to indicate the value of tuple  $t$  on attribute  $a$ .

-  $t \in r$  to show that tuple  $t$  is in relation  $r$ .

1. For example, to find the branch-name, loan number, customer name and amount for loans over \$1200:

$$\{t | t \in \text{borrow} \wedge t[\text{amount}] > 1200\}$$

This gives us all attributes, but suppose we only want the customer names. (We would use project in the algebra.)

We need to write an expression for a relation on scheme (ename).

$$\{t | \exists s \in \text{borrow} (t[\text{ename}] = s[\text{ename}] \wedge s[\text{amount}] > 1200)\}$$

In English, we may read this equation as "the set of all tuples  $t$  such that there exists a tuple  $s$  in the relation borrow for which the values of  $t$  and  $s$  on the ename attribute are equal, and the value of  $s$  on the amount attribute is greater than 1200."

The notation  $\exists t \in r (Q(t))$  means "there exists a tuple  $t$  in relation  $r$  such that predicate  $Q(t)$  is true".

How did we get the above expression? We needed tuples on scheme cname such that there were tuples in borrow pertaining to that customer name with amount attribute:  $>1200$ .

The tuples  $t$  get the scheme cname implicitly as that is the only attribute  $t$  is mentioned with.

Let's look at a more complex example.

Find all customers having a loan from the SFU branch, and the cities in which they live:

$$\{t | \exists s \in \text{borrow} (t[\text{ename}] = s[\text{ename}] \wedge s[\text{bname}] = \text{"SFU"})$$

$$\wedge \exists u \in \text{customer} (u[\text{ename}] = s[\text{ename}] \wedge t[\text{ccity}] = u[\text{ccity}])\}$$

**Q. 5. (a) Discuss entity integrity and referential integrity rules.**

**Ans. Entity Integrity :**

Entity Integrity ensures that there are no duplicate records within the table and that the field that identifies each record within the table is unique and never null.

The existence of the Primary Key is the core of the entity integrity. If you define a primary key for each entity, they follow the entity integrity rule.

Entity integrity specifies that the Primary Keys on every instance of an entity must be kept, must be unique and must have values other than NULL.

Although most relational databases do not specifically dictate that a table needs to have a Primary Key, it is good practice to design a Primary Key for each table in the relational model. This mandates no NULL content, so that every row in a table must have a value that denotes the row as a unique element of the entity.

Entity Integrity is the mechanism the system provides to maintain primary keys. The primary key serves as a unique identifier for rows in the table. Entity Integrity ensures two properties for primary keys:

- The primary key for a row is unique; it does not match the primary key of any other row in the table.
- The primary key is not null, no component of the primary key may be set to null.

The uniqueness property ensures that the primary key of each row uniquely identifies it; there are no duplicates. The second property ensures that the primary key has meaning, has a value; no component of the key is missing.

The system enforces Entity Integrity by not allowing operations (INSERT, UPDATE) to produce an invalid primary key. Any operation that creates a duplicate primary key or one containing nulls is rejected.

**Referential Integrity:** Referential integrity in a relational database is consistency between coupled tables. Referential integrity is usually enforced by the combination of a primary key or candidate key (alternate key) and a foreign key. For referential integrity to hold, any field in a table that is declared a foreign key can contain only values from a parent table's primary key or a candidate key. For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity. The relational database management system (RDBMS) enforces referential integrity, normally either by deleting the foreign key rows as well to maintain integrity, or by returning an error and not performing the delete. Which method is used would be determined by the referential integrity constraint, as defined in the data dictionary.

**Q. 5. (b) Taking an example relation, carry out the complete process of normalisation on it. Hence define various normal forms upto BCNF.**

**Ans. Normalization :**

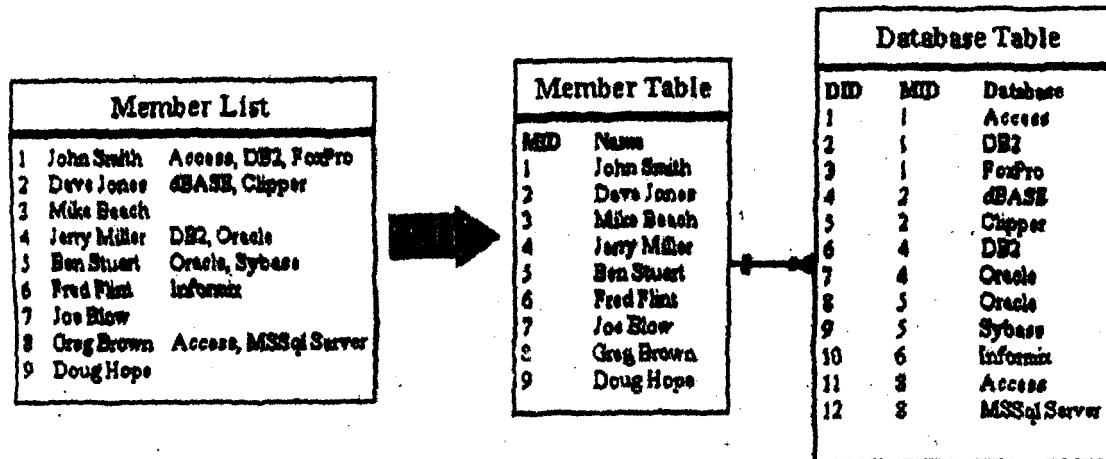
**The repeating groups :** Make a separate table for each set of related attributes, and give each table a key.

**The redundant data :** If an attribute depends on only part of a multi-valued key, remove it to a separate

**The columns not dependent on key :** If attributes do not contribute to a description of the key, remove a separate table.

**Codd normal form :** If there are non-trivial dependencies between candidate key attributes, separate it into distinct tables.

**1 NF :** Moving the known databases into a separate table helps a lot. Separating the repeating groups of databases from the member information results in first normal form. The Member ID in the database table matches the primary key in the member table, providing a foreign key for relating the two tables with a join operation. Now we can answer the question by looking in the database table for "DB2" and getting the list of members.



## 2. Eliminate redundant data :

In the Database Table, the primary key is made up of the Member ID and the DatabaseID. This makes sense for other attributes like "Where Learned" and "Skill Level" attributes, since they will be different for every member/database combination. But the database name depends only on the DatabaseID. The same database name will appear redundantly every time its associated MID appears in the Database Table.

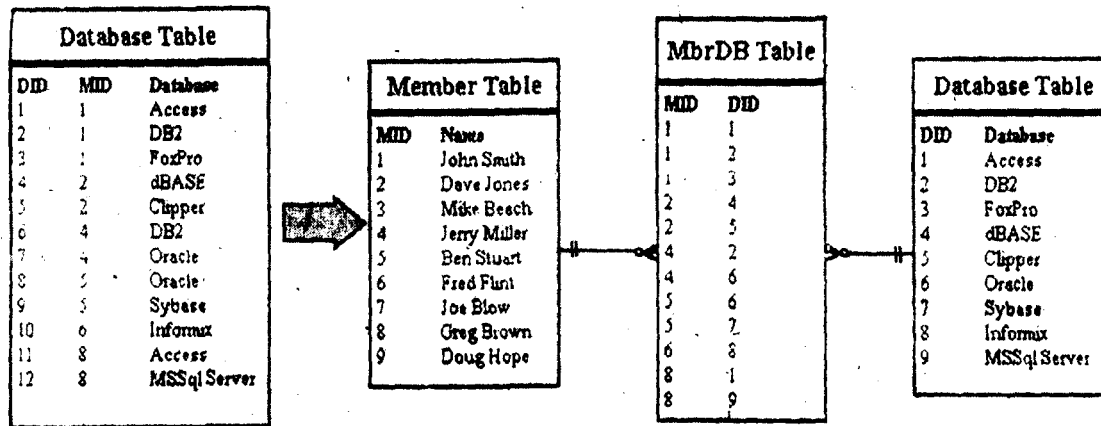
Suppose you want to reclassify a database - give it a different DatabaseID. The change has to be made for every member that lists that database! If you miss some, you'll have several members with the same database under different IDs. This is an update anomaly.

Or suppose the last member listing a particular database leaves the group. His records will be removed from the system, and the database will not be stored anywhere! This is a delete anomaly. To avoid these problems, we need second normal form.

To achieve this, separate the attributes depending on both parts of the key from those depending only on the DatabaseID. This results in two tables: "Database" which gives the name for each DatabaseID, and "MemberDatabase" which lists the databases for each member.

Now we can reclassify a database in a single operation: look up the DatabaseID in the "Database" table, and change its name. The result will instantly be available throughout the application.

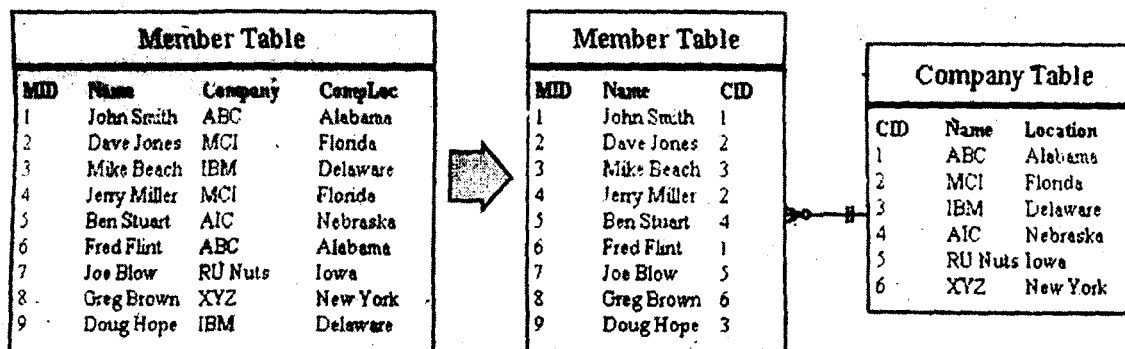




### 3. Eliminate columns not dependent on key :

The Member table satisfies first normal form - it contains no repeating groups. It satisfies second normal form since it doesn't have a multivalued key. But the key is MemberID, and the company name and location describe only a company, not a member. To achieve third normal form, they must be moved into a separate table. Since they describe a company, CompanyCode becomes the key of the new "Company" table.

The motivation for this is the same for second normal form: we want to avoid update and delete anomalies. For example, suppose no members from the IBM were currently stored in the database. With the previous design, there would be no record of its existence, even though 20 past members were from IBM!



### BCNF. boyce-codd normal form :

Boyce-Codd Normal Form states mathematically that : A relation R is said to be in BCNF if whenever  $X \rightarrow A$  holds in R, and A is not in X, then X is a candidate key for R. BCNF covers very specific situations where 3NF misses inter-dependencies between non-key (but candidate key) attributes. Typically, any relation that is

in 3NF is also in BCNF. However, a 3NF relation won't be in BCNF if (a) there are multiple candidate keys, (b) the keys are composed of multiple attributes, and (c) there are common attributes between the keys.

• **Q. 6. (a) What are distributed databases? What are ... advantages and disadvantages?**

**Ans. Distributed database:** A distributed database is a database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers.

Collections of data (eg. in a database) can be distributed across multiple physical locations. A distributed database is distributed into separate partitions/fragments. Each partition/fragment of a distributed database may be replicated (ie. redundant fail-overs, RAID like).

Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example, local autonomy, synchronous and asynchronous distributed database technologies. These technologies' implementation can and does depend on the needs of the business and the sensitivity/confidentiality of the data to be stored in the database, and hence the price the business is willing to spend on ensuring data security, consistency and integrity.

**Advantages of distributed databases :**

- \* **Reflects organizational structure** - database fragments are located in the departments they relate to.
- \* **Local autonomy** - a department can control the data about them (as they are the ones familiar with it.)
- \* **Improved availability** - a fault in one database system will only affect one fragment, instead of the entire database.
- \* **Improved performance** - data is located near the site of greatest demand, and the database systems themselves are parallelized, allowing load on the databases to be balanced among servers. (A high load on one module of the database won't affect other modules of the database in a distributed database.)
- \* **Economics** - it costs less to create a network of smaller computers with the power of a single large computer.
- \* **Modularity** - systems can be modified, added and removed from the distributed database without affecting other modules (systems).

**Disadvantages of distributed databases :**

- \* **Complexity** - extra work must be done by the DBAs to ensure that the distributed nature of the system is transparent. Extra work must also be done to maintain multiple disparate systems, instead of one big one. Extra database design work must also be done to account for the disconnected nature of the database - for example, joins become prohibitively expensive when performed across multiple systems.
- \* **Economics** - increased complexity and a more extensive infrastructure means extra labour costs.
- \* **Security** - remote database fragments must be secured, and they are not centralized so the remote sites must be secured as well. The infrastructure must also be secured (e.g., by encrypting the network links between remote sites).
- \* **Difficult to maintain integrity** - in a distributed database, enforcing integrity over a network may require too much of the network's resources to be feasible.

\* **Inexperience** - distributed databases are difficult to work with, and as a young field there is not much readily available experience on proper practice.

\* **Lack of standards** - there are no tools or methodologies yet to help users convert a centralized DBMS into a distributed DBMS.

\* **Database design more complex** - besides of the normal difficulties, the design of a distributed database has to consider fragmentation of data, allocation of fragments to specific sites and data replication..

**Q. 6. (b) Differentiate between horizontal and vertical fragmentation. Also define semi join operation.**

**Ans.** Two basic kinds of fragmentation : horizontal and vertical.

Horizontal refers to the cut between tuples, as vertical refers to the cut of the schema.

Horizontal fragmentation is achieved by a selection, as vertical by a projection.

The union of the horizontal fragments should be the original relation.

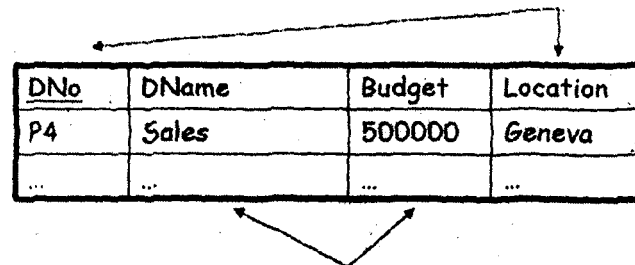
The join of the vertical fragments should be the original relation.

**Vertical Fragmentation :**

\* Vertical Fragmentation of a single relation .

\* Modeling the access to the relations

Site S3 issues 10 queries a day using these attributes



<u>DNo</u>	DName	Budget	Location
P4	Sales	500000	Geneva
...	...	...	...

Site S1 issues 5 queries a day using these attributes

Site S2 issues 20 queries a day using these attributes

Site S3 issues 10 queries a day using these attributes

Similarly as for tuples in the horizontal fragmentation we can also analyze for the vertical fragmentation of how attributes are accessed by applications running on different sites. Using this information then the goal would be to place attributes there were they are used most. In this example we see that two attributes are always accessed jointly, and that site S2 is the one that uses them most. So S2 might be a candidate to place the attributes. For similar reasons DNo and Location are best placed at S3.

### Semi-join Operations :

A semijoin from R to Ri to Rj on attribute A can be denoted as  $R_j \bowtie R_i$ . It is used to reduce the data transmission cost.

### Computing steps :

1. Project Ri on attribute A ( $R_i[A]$ ) and ship this projection ( a semijoin projection) from the site of Ri to the site of Rj;
2. Reduce Rj to Rj' by eliminating tuples where attribute A are not matching any value in  $R_i[A]$ .

**Q. 7. (a) Discuss various algorithms used for concurrency control in databases.**

**Ans. Concurrency Control :**

### Lock based protocol :

A lock is a mechanism to control concurrent access to a data item

Data items can be locked in two modes:

1. Exclusive (X) mode. Data item can be both read as well as written. X-lock is requested using lock-X instruction.
2. Shared (S) mode. Data item can only be read. S-lock is requested using lock-S instruction.

Lock requests are made to concurrency-control manager. Transaction can proceed only after request is granted. Lock-compatibility matrix

	S	X
S	true	false
X	false	false

A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions

Any number of transactions can hold shared locks on an item,

but if any transaction holds an exclusive on the item no other transaction may hold any lock on the item.

If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted. Example of a transaction performing locking:

T2 : lock-S(A);

read (A);

unlock(A);

lock-S(B);  
read (B);  
unlock(B);  
display(A+B)

Locking as above is not sufficient to guarantee serializability - if A and B get updated in-between the read of A and B, the displayed sum would be wrong.

A locking protocol is a set of rules followed by all transactions while requesting and releasing locks. Locking protocols restrict the set of possible schedules.

The potential for deadlock exists in most locking protocols. Deadlocks are a necessary evil.

Starvation is also possible if concurrency control manager is badly designed. For example :

A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item.

The same transaction is repeatedly rolled back due to deadlocks.

Concurrency control manager can be designed to prevent starvation. The Two-Phase Locking Protocol This is a protocol which ensures conflict-serializable schedules.

Phase 1 : Growing Phase

Transaction may obtain locks

Transaction may not release locks

Phase 2 : Shrinking Phase

Transaction may release locks

Transaction may not obtain locks

The protocol assures serializability. It can be proved that the transactions can be serialized in the order of their lock points (i.e. the point where a transaction acquired its final lock).

Two-phase locking does not ensure freedom from deadlocks

Cascading roll-back is possible under two-phase locking. To avoid this, follow a modified protocol called strict two-phase locking. Here a transaction must hold all its exclusive locks till it commits/aborts.

Rigorous two-phase locking is even stricter: here all locks are held till commit/abort. In this protocol transactions can be serialized in the order in which they commit.

There can be conflict serializable schedules that cannot be obtained if two-phase locking is used.

However in the absence of extra information (e.g., ordering of access to data), two-phase locking is needed for conflict serializability in the following sense :

Given a transaction  $T_i$  that does not follow two-phase locking, we can find a transaction  $T_j$  that uses two-phase locking, and a schedule for  $T_i$  and  $T_j$  that is not conflict serializable.

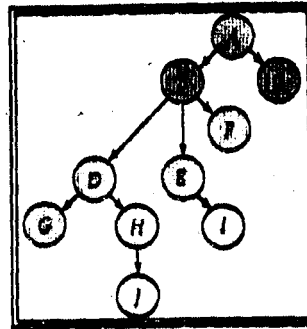
nGraph-based protocols are an alternative to two-phase locking

nImpose a partial ordering @ on the set  $D = \{d_1, d_2, \dots, d_h\}$  of all data items.

IIIf  $d_i \rightarrow d_j$  then any transaction accessing both  $d_i$  and  $d_j$  must access  $d_i$  before accessing  $d_j$

IIImplies that the set  $D$  may now be viewed as a directed acyclic graph, called a database graph.

nThe tree-protocol is a simple kind of graph protocol.



1. Only exclusive locks are allowed.
2. The first lock by  $T_i$  may be on any data item. Subsequently, a data  $Q$  can be locked by  $T_i$  only if the parent of  $Q$  is currently locked by  $T_i$ .
3. Data items may be unlocked at any time.
4. A data item that has been locked and unlocked by  $T_i$  cannot subsequently be relocked by  $T_i$ .

**Q. 8. Write short notes on :**

- (a) SQL
- (b) Hierarchical and network model
- (c) Responsibility of DBA
- (d) Data mining.

**Ans. (a) SQL :**

SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management. SQL is a standard interactive and programming language for querying and modifying data and managing databases. Although SQL is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. The core of SQL is formed by a command language that allows the retrieval, insertion, updating, and deletion of data, and performing management and administrative functions. SQL also includes a Call Level Interface (SOULI) for accessing and managing data and databases remotely.

**(b) Hierarchical and network model :**

### **Hierarchical model :**

The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1:N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment.

The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The data model is a simple network, and link and intersection record types (called junction records by IDMS) may exist, as well as sets between them. Thus, the complete network of relationships is represented by several pairwise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow). Usually, a set defines a 1:M relationship, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.

### **(c) Responsibility of DBA :**

1. Creates and maintains all databases required for development, testing, education and production usage.
2. Performs the capacity planning required to create and maintain the databases. The DBA works closely with system administration staff because computers often have applications or tools on them in addition to the Oracle Databases.
3. Performs ongoing tuning of the database instances.
4. Install new versions of the Oracle RDBMS and its tools and any other tools that access the Oracle database.
5. Plans and implements backup and recovery of the Oracle database.
6. Controls migrations of programs, database changes, reference data changes and menu changes through the development life cycle.
7. Implements and enforces security for all of the Oracle Databases.
8. Performs database re-organisations as required to assist performance and ensure maximum uptime of the database.
9. Puts standards in place to ensure that all application design and code is produced with proper integrity, security and performance. The DBA will perform reviews on the design and code fre-

quently to ensure the site standards are being adhered to.

10. Evaluates releases of Oracle and its tools, and third party products to ensure that the site is running the products that are most appropriate. Planning is also performed by the DBA, along with the application developers and System administrators, to ensure that any new product usage or release upgrade takes place with minimal impact.
11. Provides technical support to application development teams. This is usually in the form of a help desk. The DBA is usually the point of contact for Oracle Corporation.
12. Enforces and maintains database constraints to ensure integrity of the database.
13. Administers all database objects, including tables, clusters, indexes, views, sequences, packages and procedures.
14. Assists with impact analysis of any changes made to the database objects.
15. Troubleshoots with problems regarding the databases, applications and development tools.
16. Create new database users as required.
17. Manage sharing of resources amongst applications.
18. The DBA has ultimate responsibility for the physical database design.

**(d) Data mining :**

Data mining is the process of sorting through large amounts of data and picking out relevant information. It is usually used by business intelligence organizations, and financial analysts, but is increasingly being used in the sciences to extract information from the enormous data sets generated by modern experimental and observational methods. It has been described as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" and "the science of extracting useful information from large data sets or databases. Data mining in relation to enterprise resource planning is the statistical and logical analysis of large sets of transaction data, looking for patterns that can aid decision making.

Data mining is primarily used today by companies with a strong consumer focus - retail, financial, communication, and marketing organizations. It enables these companies to determine relationships among "internal" factors such as price, product positioning, or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And, it enables them to determine the impact on sales, customer satisfaction, and corporate profits. Finally, it enables them to "drill down" into summary information to view detail transactional data.

With data mining, a retailer could use point-of-sale records of customer purchases to send targeted promotions based on an individual's purchase history. By mining demographic data from comment or warranty cards, the retailer could develop products and promotions to appeal to specific customer segments.

For example, Blockbuster Entertainment mines its video rental history database to recommend rentals to individual customers. American Express can suggest products to its card holders based on analysis of their monthly expenditures.