

11

Database Concepts

In This Chapter

- 11.1 Introduction
- 11.2 Purpose of Databases
- 11.3 Database Abstraction
- 11.4 Different Data Models
- 11.5 The Relational Model
- 11.6 Comparison of Data Models

11.1 INTRODUCTION

A database system is basically a computer based record keeping system. The collection of data, usually referred to as the *database*, contains information about one particular enterprise. It maintains any information that may be necessary to the decision-making processes involved in the management of that organization.

A database may also be defined as a collection of inter-related data stored together to serve multiple applications ; the data is such stored so that it is independent of programs which use the data ; a common and controlled approach is used in adding

↪ **Database** A database is a collection of interrelated data and a database system is basically a computer based record keeping system.

new data and in modifying and retrieving existing data within the database. The data is structured so as to provide a foundation for future application development.

The intention of a database is that the same collection of data should serve as many applications as possible. Hence, a database is often conceived of as the repository of information needed for running certain functions in a corporation or organization. Such a database would permit not only the retrieval of data but also the continuous modification of data needed for control of operations. It may be possible to search the database to obtain answers to queries or information for planning purposes.

11.2 PURPOSE OF DATABASES

A database system should be a repository of the data needed for an organization's data processing. That data should be accurate, private, and protected from damage. It should be organized so that diverse applications with different data requirements can employ the data. Different application programmers and different end users will have different views of the data which must be derived from a common overall data structure. Their methods of accessing or searching the data will differ.

The ways in which end users want to utilize existing data will constantly change, and in some cases demands for new uses of the data will arise rapidly and urgently. The extent to which these demands can be satisfied determines the overall value of the database system.

In a typical file-processing system, permanent records are stored in various files. A number of different application programs are written to extract records from and add records to the appropriate files. But this scheme has a number of major limitations and disadvantages, such as data redundancy (duplication of data), data inconsistency, unsharable data, unstandardized data, insecure data, incorrect data etc. A *database management system* is answer to all these problems as it provides a centralized control of the data.

Let us consider some of the advantages provided by the database systems and see how a database system overcomes the above mentioned problems.

1. Databases reduce the data redundancy to a large extent

Data redundancy means duplication of data. Non-database systems maintain separate copy of data for each application. For example, in a college, student records are maintained and the hostel also maintains records of all those students who live in hostel.

Though the records for hosteler students are already being maintained by the college, the hostel keeps a separate copy of it. As far as everything goes well, the above mentioned file processing system works well. But duplication of data may lead to inconsistency or incorrect data at times.

Let us see how.

Suppose the permanent address of a hosteler gets changed. The hosteler student informs about it to the hostel authorities and the changed address is reflected in the student's record. But see there is a huge bug here. The data for the same student has not been changed in the college records. If the student forgets to inform the college authorities or college people themselves forget to update the student's record, the same student's record differs in two different files. Now, which record of the two would be considered accurate? Say, the college has to send certain information by post. Then, what will happen? The wrong data will get processed. The only way out is that every file that is maintaining the same record should be updated. But because of some human errors, certain files may

be left out and this will lead to inconsistent data. The other problem of redundancy is unnecessary wastage of storage space.

The database systems do not maintain separate copies of the same data. Rather, all the data are kept at one place and all the applications that require data refer to the centrally maintained database (collection of data). (See Fig. 11.1). Now, if any change is to be made to data, it will be made at just one place and the same changed information will be available to all the applications referring to it. Thus redundancy gets controlled and so are the problems associated with it.

➤ **Data Redundancy** Duplication of data is known as *Data Redundancy*.

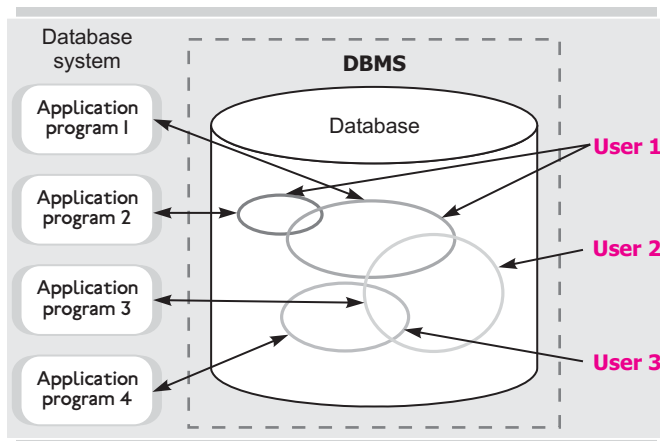


Fig. 11.1 Centrally Controlled Database System.

We do not mean to suggest that all redundancy should necessarily be eliminated. Sometimes there are sound business or technical reasons for maintaining multiple copies of the same data. In a data base system, however, redundancy can be controlled upto the desired extent and the system is aware of the redundancy, if any and assumes the responsibility for propagating updates.

2. Databases can control data inconsistency to a large extent

This is really a corollary of the previous point. When the redundancy is not controlled, there may be occasions on which the two entries about the same data do not agree (that is, when one of them stores the updated information and the other does not). At such times, database is said to be *inconsistent*. Obviously, an inconsistent database will provide incorrect or conflicting information.

By controlling redundancy, the inconsistency is also controlled. Even if there is some redundancy retained in the database due to some technical reasons, the database management system ensures that any change made to either of the two entries is automatically made to the other. This process is known as *propagating updates*.

3. Databases facilitate sharing of data

Sharing of data means that individual pieces of data in the database may be shared among several different users, in the sense that each of those users may have access to the same piece of data and each of them may use it for different purposes.

The database management system makes sure that not only the existing applications can share the data in the database, but also that new applications can be developed to operate against that same stored data. Or it can be said that the data requirements of new applications may be satisfied without having to create any new stored files.

4. Databases enforce standards

The database management systems can ensure that all the data (that is stored centrally) follow the applicable standards. There may be certain standards laid by the company or organization using the database. Or there may be certain industry standards that must be satisfied by the data. Similarly, there may be national or international standards. Standardizing stored data formats is particularly desirable as an aid to data *interchange* or migration between systems.

Data Security Data Security refers to protection of data against accidental or intentional disclosure to unauthorized persons, or unauthorized modification or destruction.

Privacy of Data Privacy of Data refers to the rights of individuals and organizations to determine for themselves when, how, and to what extent information about them is to be transmitted to others.

5. Databases can ensure data security

The information stored inside a database is sometimes of great value to a corporation. Therefore, it must be kept secure and private.

A database management system ensures data security and privacy by ensuring that the only means of access to the database is through the proper channel and also by carrying out authorization checks whenever access to sensitive data is attempted.

6. Integrity can be maintained through databases

By *integrated database* we mean unification of several otherwise distinct data files, with any redundancy among those files partially or wholly eliminated. When a database contains data employed by many different users it is important that the data items and

associations between data items not be destroyed. Hardware failures and various types of accidents will occur occasionally. The storage of data and its updation, and insertion procedures, defined by the database, are such that the system can (easily) recover from these circumstances without harm to the data.

In addition to protecting data from systems problems, the database management system designs certain integrity checks to ensure that data values confirm to certain specified rules. For example, a date cannot be like 25/25/12 ; it is invalid date. Or say the number of days-worked for an employee cannot exceed the number of working days in a month. Therefore, a database management system defines integrity checks like *range checks* to check for the values that must lie within certain range of values ; and *value matching* to check for the values that must be present already. (e.g., while preparing pay slips, the *employee-number* must be a valid one i.e., it must be present in the *employee-master*).

With the above mentioned advantages, the purpose of databases is very obvious. That is, databases are preferred so as to ensure *consistent, sharable, standardized, integrated* and *secure data*, managed effectively so as to serve the requirements of the organization owning it.

11.3 DATABASE ABSTRACTION

As we know that a collection of interrelated files and a set of programs that allow users to access and modify these files is known as a *database management system*. A major purpose of a database system is to provide the users only that much information that is required by them. This means that the system does not disclose all the details of data, rather it hides certain details of how the data is stored and maintained. A good database system ensures easy, smooth and efficient data structures in such a way so that every type of database user : *end user, application system analyst, and physical storage system analyst*, is able to access its desired information efficiently. (An *end user* is a person who is not a computer-trained person but uses the database to retrieve some information. For example, in a bank database, a customer, who wants to know how much balance remains in his account, is an *end-user*. An *application system analyst* is the one who is concerned about all of

the database of logical level i.e., what all data constitute the database, what are the relationships between the data-entities etc. without considering the physical implementation details. The third type of user, the *physical storage system analyst* is concerned with the physical implementation details of the database i.e., how would the database be stored on which storage device ? What will be the starting address of the database ? What will be the storage technique ? etc. etc.)

Since the requirements of different users differ from one another, the complexity of the database is hidden from them, if needed, through several levels of abstraction is order to simplify their interaction with the system. The various levels of database implementation are being discussed in the following section.



1. What is the function of a database management system ?
2. What is data redundancy ? What are the problems associated with it ?
3. How do database management systems overcome the problems associated with data redundancy ?
4. How do database management systems ensure data security and privacy ?
5. Define the following terms :
 - (i) integrated database
 - (ii) shared database
 - (iii) view
 - (iv) database system
 - (v) data security
 - (vi) data integrity.

11.3.1 Various Levels of Database Implementation

A database is implemented through three general levels : *internal*, *conceptual* and *external* so as cater to the needs of its users.

1. Internal Level (Physical Level)

The lowest level of abstraction, the *internal level*, is the one closest to physical storage. This level is also sometimes termed as *physical level*. It describes how the data are actually stored on the storage medium. At this level, complex low-level data structures are described in details.

2. Conceptual Level

This level of abstraction describes *what* data are actually stored in the database. It also describes the relationships existing among data. At this level, the database is described logically in terms of simple data-structures. The users of this level are not concerned with how these logical data structures will be implemented at the physical level. Rather, they just are concerned about *what* information is to be kept in the database.

3. External Level (View Level)

This is the level closest to the users and is concerned with the way in which the data are viewed by individual users. Most of the users of the database are not concerned with all the information contained in the database. Instead, they need only a part of the database relevant to them. For example, even though the bank database stores a lot many information, an *account holder* (a user) is interested only in his account details and not with the rest of the information stored in the database. To simply such users' interaction with the system, this level of abstraction is defined. The system, thus provides many views for the same database. The following Fig. 11.2 illustrates the interrelationship among these *three* levels of abstraction.

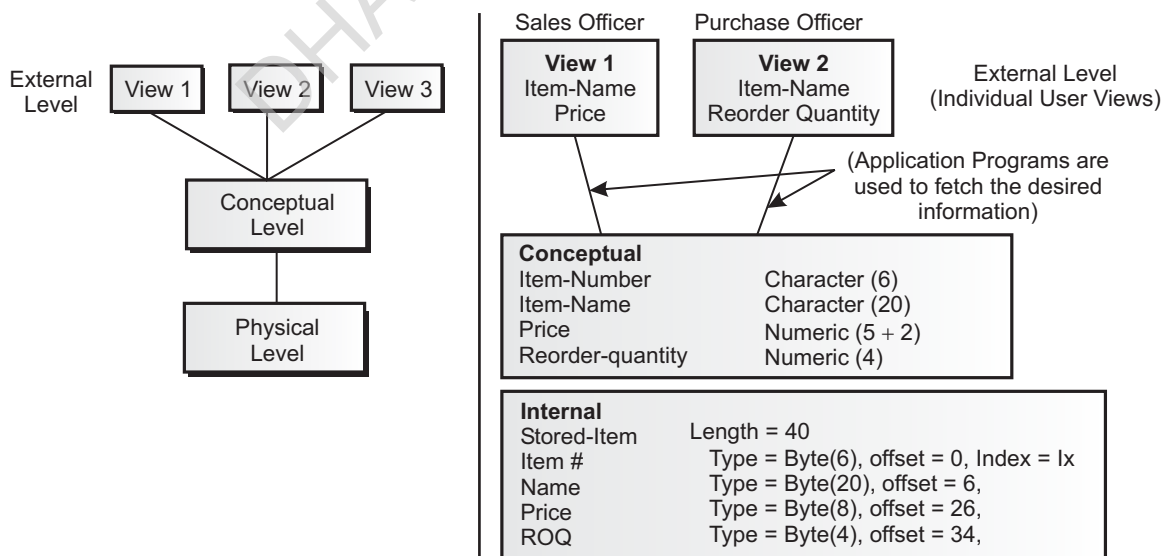


Fig. 11.2 The Three Levels of Data Abstraction.

11.3.2 Concept of Data Independence

Data Independence The ability to modify a scheme definition in one level without affecting a scheme definition in the next higher level is called *Data Independence*.

As a database may be viewed through three levels of abstraction, any change at a level may affect other levels' schemes. Since the databases keep on growing, there may be frequent changes at times. This should not lead to the redesigning and reimplementing of the database. The concept of data independence proves beneficial in such a context.

There are *two* levels of data independence : *physical* and *logical*.

1. Physical Data Independence

Physical Data Independence refers to the ability to modify the scheme followed at the *physical* level without affecting the scheme followed at the *conceptual* level. That is, the application programs remain the same even though the scheme at physical level gets modified. Modifications at the physical level are occasionally necessary in order to improve performance of the system.

2. Logical Data Independence

Logical Data Independence refers to the ability to modify the *conceptual* scheme without causing any changes in the schemes followed at *view* levels. The logical data independence ensures that the application programs remain the same. Modifications at the conceptual level are necessary whenever logical structures of the database get altered because of some unavoidable reasons. (For example, the introduction of paternity leave in the employee database for the first time.)

It is more difficult to achieve logical data independence than the physical data independence. The reason being that the application programs are heavily dependent on the logical structure of the database.

The *abstract data types* in modern programming languages implement the concept of *data independence* to a large extent. Both hide implementation details from the users. This allows users to concentrate on the general structure rather than low-level implementation details.

11.4 DIFFERENT DATA MODELS

The external level and conceptual level use certain data structures to help utilize the database efficiently. How are these data structures decided ? What data structures and associated operators should the system support ? The answer to this very crucial question depends upon the approach or model being used for database management. The *three* data models that are used for database management are :

- ❖ Relational data model
- ❖ Hierarchical data model
- ❖ Network data model



1. What are the three levels of data abstraction ?
2. What do you understand by data independence ?
3. What are two types of data independence ? How are they different ?

11.4.1 The Relational Data Model

In relational data model, the data is organized into tables (*i.e.*, rows and columns). These tables are called *relations*. A row in a table represents a *relationship* among a set of values. Since a table is a collection of such relationships, it is generally referred to using the mathematical term *relation*, from which the relational data model derives its name.

Let us see how a sample database can be represented in relational form. The sample database being shown here has three tables (*relations*) : *Suppliers*, *Items*, *Shipments*.

Suppliers (**Supp#**, Supp-Name, Status, City)
 Items (**Item#**, Item-Name, Price)
 Shipments (**Supp#**, **Item#**, Qty-supplied) [coloured fields represent *primary keys*¹]

Suppliers

Supp#	Supp-Name	Status	City
S1	Britannia	10	Delhi
S2	New Bakers	30	Mumbai
S3	Mother Dairy	10	Delhi
S4	Cookz	50	Bangalore
S5	Haldiram	40	Jaipur

Items

Item#	Item-Name	Price
I1	Milk	15.00
I2	Cake	5.00
I3	Bread	9.00
I4	Milk Bread	14.00
I5	Plain Biscuit	6.00
I6	Cream Biscuit	10.00
I7	Ice Cream	16.00
I8	Cold Drink	8.00
I9	Namkeen	15.00

Shipments

Supp#	Item#	Qty-supplied
S1	I2	10
S1	I3	20
S1	I6	20
S2	I4	20
S2	I5	10
S3	I1	10
S3	I7	10
S4	I8	30
S5	I9	30

Fig. 11.3 Sample Database in Relational Form.

Notice, that here each supplier has a unique supplier number, exactly one name, status value and location. Likewise, we assume that each item has a unique item number, name, price ; and also that, at any given time, no more than one shipment exists for a given supplier/item combination.

Each of these three tables closely resembles a conventional sequential file, with rows of the table corresponding to records of the file and columns corresponding to fields of the records. Each of these tables is actually a special case of the construct known in mathematics as a *relation*. (More about *relations* and their characteristics, we shall learn under section 11.5.1).

Rows of *relations* are generally referred to as *tuples* and the columns are usually referred to as *attributes*.

The relational data model is based on a collection of tables (*relations*). The user of the (relational) database system may query these tables, insert new tuples, delete tuples, and modify tuples. There are several languages for expressing these operations. One such language is relational query language. The relational algebra is a procedural language that

1. Primary-key is the key-field that can uniquely identify a row in a relation.

defines the basic operations used within the relational query language. The section 11.5, *The Relational Model* deals with relational data base management systems in details. But before that you must be introduced to two other data models : *network* and *hierarchical*.

11.4.2 The Network Data Model

In the relational data model, the relationships among data are represented by a collection of tables. The network model differs from the relational model in that data is represented by collections of *records* and relationships among data are represented by *links*.

In a network database, the collection of records are connected to one another by means of *links*. A *record* is a collection of *fields (attributes)*, each of which contains only one data value.

By a *link* we mean that it is an association between precisely two records. Now, let us see how the same sample database, shown earlier in relational form, can be defined in network model. (Fig. 11.4) The structure shown in Fig. 11.4 is known as *arbitrary graph*.

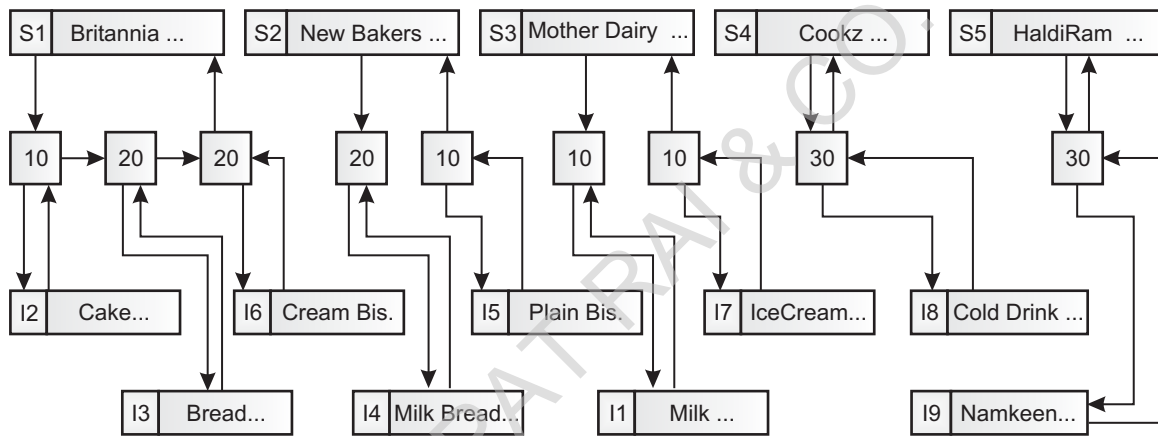


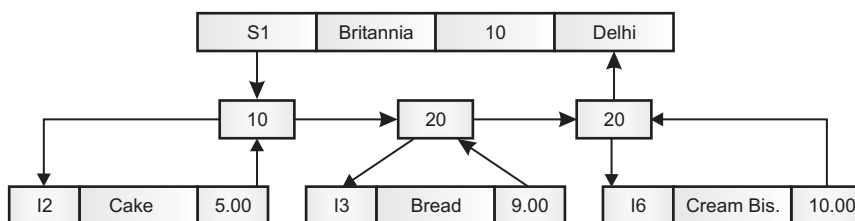
Fig. 11.4 Sample Database in Network Form.

Notice that, in the shipments table the supplier **S1** supplies three items **I2**, **I3** and **I6** in the quantities 10, 20 and 20 respectively. Now, this relationship has been shown in Fig. 11.5 as follows (following figure extracts a part of the Fig. 11.4).

See also, since only one quantity (10) is being supplied for item **I2**, **I2** joins with only one quantity. If there are more values for a same value (say X), the *links* form a chain and the last value in the chain points back to the same value X. See there is a chain of 10 → 20 → 20. The last value **20** in the chain for supplier **S1** points back to **S1**.

In network data model, while mapping to files, links are implemented by adding *pointer fields* to records that are associated via a link. Each record must have one pointer field for each link with which it is associated.

The operations on a network database are performed through a data manipulation language for network model. The operations that can be performed on a network database,



include *find*, *insert*, *delete*, *modify* etc. The inserting or removing records involve *connect*, *disconnect*, and *reconnect* operations.

Fig. 11.5 The Relationships are implemented through links in Network model.

11.4.3 The Hierarchical Data Model

In the network model, the data is represented by collections of records and relationships among data and are represented by *links*. This is true of hierarchical model as well. The only difference is that in the hierarchical model, records are organized as *trees* rather than arbitrary graphs. You can say that hierarchical model represents relationship

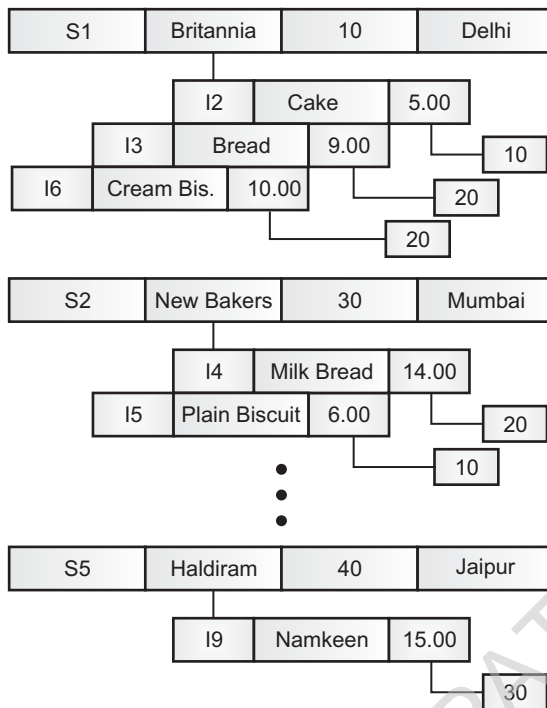


Fig. 11.6 Sample Database in Hierarchical Form (Suppliers superior to items).

among its records through *parent child relationships* that can be easily represented through *tree* like structures.

Therefore, a hierarchical database is also a collection of *records* connected to one another through *links*. Fig. 11.6 shows a possible *hierarchical view* for the suppliers-and-items database. In this view the data is represented by a simple tree structure, with *suppliers* superior to *items*.

The record type at the top of the tree – the supplier record type in our example – is usually known as the *root*. In general, the root may have any number of dependents, each of these dependents may have any number of lower-level dependents, and so on, to any number of levels.

The operations on a hierarchical database are performed through a *data manipulation language* for hierarchical data model. The operations that can be performed on hierarchical database include retrieval, insertions, deletion and modifications of records.

11.5 THE RELATIONAL MODEL

The relational model was propounded by *E.F. Codd* of the IBM and has since been acknowledged as a very important concept in DBMS (Data Base Management Systems) technology. The relational model has established itself as the primary data model for commercial data processing applications. Its success in this field has led to its application outside data processing in systems for computer-aided design and other environments. Let us explore this model in details.

11.5.1 Terminology

Different terms used in the relational model are being discussed here.



1. Name the different data models available for database systems. Which of them is the most preferred one ?
2. What are the similarities and differences between network and hierarchical data models ?
3. What do you mean by relational database ?

Relation

In general, a relation is a table *i.e.*, data is arranged in rows and columns.

A relation has the following properties :

1. In any given column of a table, all items are of the same kind whereas items in different columns may not be of the same kind.

2. For a row, each column must have an atomic (indivisible) value and also for a row, a column cannot have more than one value.
3. All rows of a relation are distinct. That is, a relation does not contain two rows which are identical in every column. That is, each row of the relation can be uniquely identified by its contents.
4. The ordering of rows within a relation is immaterial. That is, we cannot retrieve any thing by saying that from row number 5, column *name* is to be accessed. There is no order maintained for rows inside a relation.
5. The columns of a relation are assigned distinct names and the ordering of these columns is immaterial.

Domain

A domain is a pool of values from which the actual values appearing in a given column are drawn. For example, the values appearing in the **Supp#** column of both the *Suppliers* table and the *Shipments* table are drawn from the underlying domain of all valid supplier numbers. (See Fig. 11.7). A domain is said to be *atomic* if elements of the domain are considered to be indivisible units. For example, the set of integers is an atomic domain but the set of all sets of integers is a nonatomic domain.

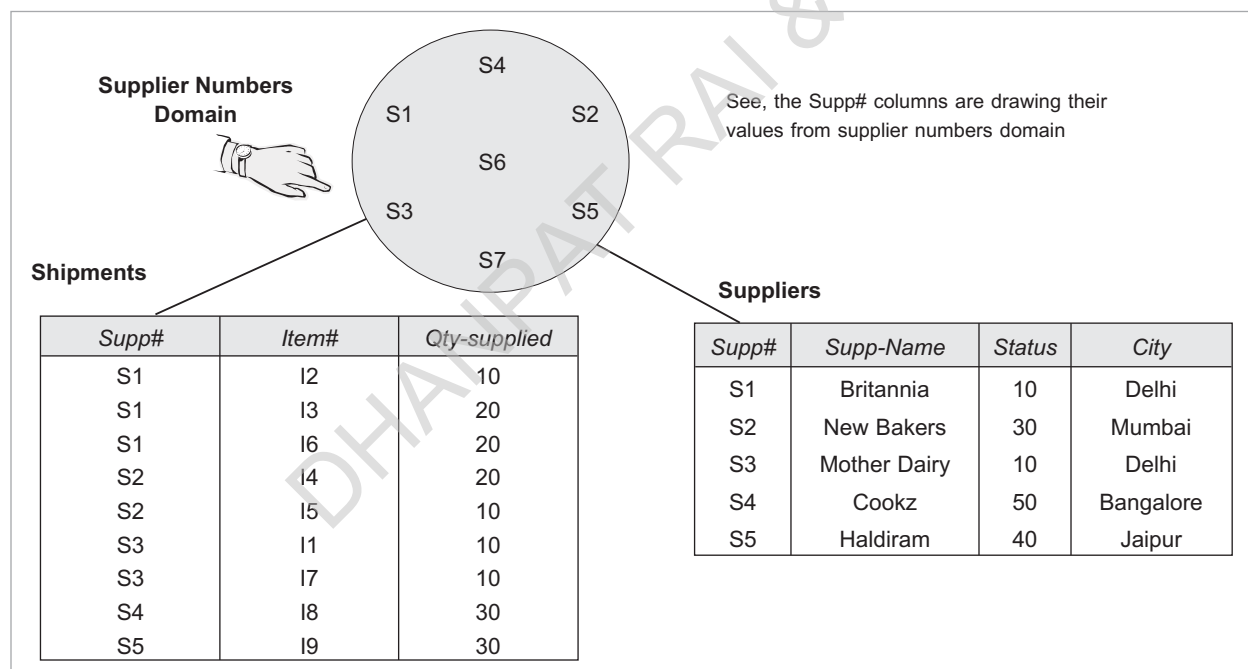


Fig. 11.7 Supp# in Suppliers and Shipments draw values from same domain.

In the sample database, observe that relations *Suppliers* and *Shipments* have a common domain (Supplier numbers domain) and so do *Items* and *Shipments* (Item numbers domain).

NOTE

A crucial feature of relational data model is that associations between rows are represented solely by data values in columns drawn from a common domain.

See that in the table *Shipments* the supplier **S1** is supplying **10** units of item **I2** and from the table *Suppliers* it can be said that a **Delhi** based supplier **Britannia** is supplying **10** units of item **I2**. That is how the association between rows (**S1**, **I2**, **10** of *Shipments* and **S1**, **Britannia**, **10**, **Delhi** of *Suppliers*) can be represented.

→ **Tuples** The rows of a relation are known as Tuples.

→ **Attributes** The columns of a relation are known as Attributes.

→ **Degree** The number of attributes in a relation is called Degree.

→ **Cardinality** The number of rows in a relation is known as Cardinality.

Tuple

The rows of tables (relations) are generally referred to as *Tuples* (usually pronounced to rhyme with “couples”).

Attributes

The columns/fields of tables (relations) are generally referred to as *attributes*.

Degree

The number of attributes in a relation determine the *degree* of a relation. A relation having 3 attributes is said to be a relation of degree 3. Similarly, a relation having n attributes is said to be a relation of degree n . Relations of degree one are said to be *unary*, relations of degree two are *binary*, relations of degree three are *ternary*,, and relations of degree n are *n-ary*.

Cardinality

The number of tuples (rows) in a relation is called the *cardinality* of the relation ; e.g., the cardinality of *Shipments* relation is 9. Similarly, the cardinality of *Suppliers* relation is 5.


11.5.2 Views

A view is a kind of table whose contents are taken from other tables depending upon a condition. Views do not contain data of their own. The contents of a view are determined by carrying out the execution of the given query (the given condition). Let us elaborate the concept.

The kinds of tables (relations) that you have come across with until now are called *base tables*. These are the tables that actually contain data. There is another kind of tables, the views. *Views* are tables whose contents are derived from other tables (their *base tables*) everytime they are referred to. There is no stored file created for storing the contents of a view. Rather only the definition of a view is stored. Everytime a view is referred to, its contents are derived from its underlying base table(s) using its stored definition. Thus, a view is also referred to as a *virtual table*.

→ **View** A View is a (virtual) table that does not really exist in its own right but is instead derived from one or more underlying base table(s).

Following Fig. 11.8 explains the concept of a view. Views are like windows through which you view desired information that is actually stored in a base table.



Definition
of the view

```
CREATE VIEW GoodItems AS      (name of the view)
SELECT * FROM Items          (name of the base table)
WHERE Price > 12 ;            (the condition)
```

GoodItems (A Virtual Table based on **Items** table)

Item#	Item-Name	Price
11	Milk	15.00
14	Milk Bread	14.00
17	Ice Cream	16.00
19	Namkeen	15.00

Fig. 11.8
The concept
of Views.

A view can be used just like any other table. It can be queried, updated, inserted into, deleted from, and joined with other tables and views.

Views greatly extend the control you have over your data. They are an excellent way to give people access to some but not all of the information in a table. Even *Read-only* views can also be created which means that they can be queried, but they cannot be subjected to update commands.

(Though there is a lot that can be discussed about views yet we are not going further in explaining views as it is beyond the scope of this chapter and the book).

11.5.3 Structure of Relational Databases

Consider the *Items* table of our sample database. It has three attributes : *Item#*, *Item-Name*, and *Price*. For each attribute, there is a set of permitted values, called the *domain* of that attribute. For the attribute *Item-Name*, for example, the domain is the set of all item names. Let D_2 be this set (of item names). D_1 be the set of item numbers, and let D_3 be set of prices. Now any row of *Items* must be a 3-type (*i.e.*, must have values for 3 attributes), say (x_1, x_2, x_3) where x_1 is an item number, x_2 is the item name and x_3 is the *price* of the item. That means x_1 is in domain D_1 , x_2 is in domain D_2 , and x_3 is in domain D_3 . In general, the table *Items* will contain only a subset of the set of all possible rows. Therefore, *Items* is a subset of

$$D_1 \times D_2 \times D_3.$$

In general, a table having n attributes must be a subset of

$$D_1 \times D_2 \times \dots \times D_{n-1} \times D_n.$$

11.5.3A Keys

It is important to be able to specify how rows in a relation are distinguished conceptually, rows are distinct from one another, but from a database perspective the difference among them must be expressed in terms of their attributes. *Keys* come here for a rescue.

Primary Key

Within a given relation, a *set of one or more attributes* having values that are unique within the relation and thus are able to *uniquely identify that tuple*, is said to be the *primary key* of the relation.

Primary Key A primary key is a set of one or more attributes that can uniquely identify tuples within the relation.

Every relation does have a primary key. In our sample database, **Supp#** is the primary key for *Suppliers* as it contains unique value for each tuple in the relation. Similarly, **Item#** is the primary key for *Items* and the combination of **Supp#** and **Item#** is the primary key for the *Shipments* relation.

In some tables, combination of more than one attribute provides a unique value for each row. In such tables, the group of these attributes is declared as primary key. In such cases, the primary key consists of more than one attribute, it is called *composite-primary-key*.

The primary key is nonredundant *i.e.*, it does not have duplicate values in the same relation. The non-primary-key attributes of a table can be referred to as *non-key* attributes.

Candidate Keys All attribute combinations inside a relation that can serve as primary key are Candidate Keys as they are candidates for the primary key position.

Candidate Key

Occasionally we may encounter a relation in which there is more than one attribute possessing the unique identification property.

In our sample database, there are two candidate keys **Supp#** and **Supp-Name** in the *Suppliers* relation. Both of these attributes contain unique values for each tuple. Similarly, in *Items*, **Item#** and **Item-Name** are candidate keys.

In the case of two or more candidate keys, the database analyst decides one of them as the primary key for the relation.

Alternate Key

Alternate Key A candidate key that is not the primary key is called an Alternate Key.

In case of two or more candidate keys, only one of them serves as the primary key. The rest of them are *alternates* only.

In *Suppliers* table, **Supp-Name** is the alternate key and in *Items* table **Item-Name** is the alternate key.

Foreign Key

A foreign key is used to represent the relationship between two tables. A **foreign key** is a non-key attribute (or a group of non-key attributes) whose value is derived from the *primary-key* of another table. Or in other words, a non-key attribute of a table, which is the primary-key of some other table is known as **foreign-key**.

Foreign-Key A non-key attribute, whose values are derived from the primary key of some other table, is known as Foreign-Key in its current table.

The table in which this non-key attribute *i.e.*, the foreign-key attribute exists, is called a **Foreign table** or **Detail table**, and the table that defines the *Primary-key*, which the *foreign-key* of *detail-table* refers to, is called **Primary table** or **Master table**.

11.5.4 The Relational Algebra

The relational algebra is a collection of operations on relations. Each operation takes one or more relations as its operand(s) and produces another relation as its result. The operations defined in relational algebra include *select*, *project*, *cartesian product*, *union*, *set difference*, *set intersection*, *natural join*, *division* etc. The *select* and *project* are *unary* operations since they operate on one relation. The other operations are *binary* operations as they operate upon pairs of relations.

Let us discuss each of these operations individually.

11.5.4A The Select Operation

The *select* operation selects tuples (horizontal subset) from a relation that satisfy a given predicate (*i.e.*, a given condition) [see Fig. 11.9(a)]. The *selection* is denoted by lowercase Greek letter σ (sigma). To select those tuples from *Items* relation where the *price* is more than 14.00, we shall write

$$\sigma_{\text{price} > 14.00}(\text{Items})$$

That means from table *Items* (table name given in parenthesis), select (denoted by σ) the tuples satisfying the condition *price* > 14.00. The relation that results from the above query is as shown in Fig. 11.9(b). The *Items* relation is the same as in our sample database being used in this chapter.



1. Define the following terms :
 - (a) relation
 - (b) tuple
 - (c) attribute
 - (d) domain
 - (e) primary key
 - (f) candidate key
 - (g) cartesian product
 - (h) degree
2. What are views ? How are they useful ?
3. Define the following :
 - (i) primary key
 - (ii) candidate key
 - (iii) alternate key
 - (iv) foreign key.
4. What is an Alternate Key ?
5. What is the importance of a Primary Key in a table ? Explain with a suitable example.
6. What do you understand by the terms Primary Key and Degree of a relation in relational database ?
7. What do you understand by the terms Candidate Key and Cardinality of a relation in relational database ?

Item#	Item-Name	Price
I1	Milk	15.00
I2	Cake	5.00
I3	Bread	9.00
I4	Milk Bread	14.00
I5	Plain Biscuit	6.00
I6	Cream Biscuit	10.00
I7	Ice Cream	16.00
I8	Cold Drink	8.00
I9	Namkeen	15.00

(a)

Item#	Item-Name	Price
I1	Milk	15.00
I7	Ice cream	16.00
I9	Namkeen	15.00

(b)

Fig. 11.9 (a) SELECT operation ; (b) Result of $\sigma_{price > 14.00}$ (Items)

Similarly, tuples of the *Suppliers* where the city is "Delhi" can be selected as

$\sigma_{city = \text{"Delhi"}}$ (Suppliers)

Item#	Item-Name	Price
I5	Plain Biscuit	6.00
I8	Cold Drink	8.00

In general, in a selection predicate (condition), all relational operators ($=, \neq, <, \leq, >, \geq$) may be used. Also, more than one condition may be combined using the connectives *and* (denoted by \wedge) and *or* (denoted by \vee). Thus, to find those tuples pertaining to prices between 5.00 and 9.00 from relation *Items*, we shall write

Fig. 11.10 Result of $\sigma_{price > 5.00 \wedge price < 9.00}$ (Items).

$\sigma_{price > 5.00 \wedge price < 9.00}$ (Items)

The result of this query is shown in Fig. 11.10.

The selection predicate may include comparisons between two attributes. For example, if we have a relation *Customer* as shown below :

Cust#	Cust-Name	Banker-Name	Amount	Balance
C001	Anjuman	Rakesh	17510.00	17510.00
C002	Reva	Reva	21000.00	22300.00
C003	Aastha	Ravi	25199.00	24801.00

Fig. 11.11 The relation named Customer.

And we want to find out the names of those customers who have the same name as their banker, we may write

$\sigma_{cust-Name = banker-Name}$ (Customer)

Similarly, to find out the details of those customers who have their amount more than their balance, we'll write

$\sigma_{amount > balance}$ (Customer)

The relations resulting from above two queries are shown below in Fig. 11.12.

Cust#	Cust-Name	Banker-Name	Amount	Balance
C002	Reva	Reva	21000.00	22300.00

(a)

Cust#	Cust-Name	Banker-Name	Amount	Balance
C003	Astha	Ravi	25199.00	24801.00

(b)

Fig. 11.12 (a) Result of $\sigma_{cust-name = banker-name}$ (Customer), (b) Result of $\sigma_{amount > balance}$ (Customer).

11.5.4B The Project Operation

Supp#	Supp-Name	Status	City
S1	Britannia	10	Delhi
S2	New Bakers	30	Mumbai
S3	Mother Dairy	10	Delhi
S4	Cookz	50	Bangalore
S5	Haldiram	40	Jaipur

(a)

Supp-Name	City
Britannia	Delhi
New Bakers	Mumbai
Mother Dairy	Delhi
Cookz	Bangalore
Haldiram	Jaipur

(b)

Fig. 11.13 (a) The PROJECT operation will select the vertical subset of entire table. (b) Result of $\pi_{\text{Supp-Name, City}}(\text{Suppliers})$.

The *project* operation yields a “vertical” subset of a given relation in contrast to the “horizontal” subset returned by *select* operation. That is, the *projection* lets you select specified attributes in a specified order [see Fig. 11.13(a)]. As the result is also a relation, the duplicating tuples are automatically removed. Projection is denoted by Greek letter pi (π). To project Supplier names and their cities from the relation *Suppliers*, we shall write

$$\pi_{\text{Supp-Name, City}}(\text{Suppliers})$$

That means, from table *Suppliers* (table name is given in parenthesis), project (denoted by π) the attributes *Supp-Name* and *City*. The relation resulting from this query is as shown in Fig. 11.13(b).

However, if you give the project query as follows :

$$\pi_{\text{City, Supp-Name}}(\text{Suppliers})$$

the resulting relation will be as shown in Fig. 11.14.

City	Supp-Name
Delhi	Britannia
Mumbai	New Bakers
Delhi	Mother Dairy
Bangalore	Cockz
Jaipur	Haldiram

Fig. 11.14. Result of $\pi_{\text{City, Supp-Name}}(\text{Suppliers})$.

It is obvious from the above table that you can specify the order of attributes in the resulting relation. Since these operations result in a relation, *project* operation can also be applied on a resulting relation of a query. For example, if you are interested only in the names of those items that are costlier than ₹ 14.00, you may write it as

Item-Name
Milk
Ice cream
Namkeen

(a)

City
Delhi
Mumbai
Bangalore
Jaipur

(b)

Fig. 11.15 Result of (a) $\pi_{\text{Item-Name}}(\sigma_{\text{price} > 14.00}(\text{Items}))$ (b) $\pi_{\text{City}}(\text{Suppliers})$.

$$\pi_{\text{Item-Name}}(\sigma_{\text{price} > 14.00}(\text{Items}))$$

First the inner query is evaluated and then the outer query is evaluated using the relation that resulted from the inner query. The result of above query is as shown below in Fig. 11.15(a). Duplicating tuples are automatically removed in the resulting relation. For instance, if you write

$$\pi_{\text{City}}(\text{Suppliers})$$

the resulting relation will be as shown in Fig. 11.15(b).

11.5.4C The Cartesian Product Operation

The *cartesian product* is a binary operation and is denoted by a cross (\times). The Cartesian product of two relations **A** and **B** is written as $A \times B$. The cartesian product yields a new relation which has a degree (number of attributes) equal to the sum of the degrees of the two relations operated upon. The number of tuples (cardinality) of the new relation is the product of the number of tuples of the two relations operated upon. The *cartesian product* of two relations yields a relation with all possible combinations of the tuples of the two relations operated upon.

Student

Stud#	Stud-Name	Hosteler
S001	Meenakshi	Y
S002	Radhika	N

Instructor

Inst#	Inst-Name	Subject
I01	K. Lal	English
I02	R.L. Arora	Maths

All tuples of first relation are concatenated with all the tuples of second relation to form the tuples of the new relation. Let us consider how does it work. Let us assume there are two relations *Student* and *Instructor* as shown in Fig. 11.16.

Fig. 11.16 Student and Instructor Relations.

The cartesian product of these two relations, **Student** \times **Instructor**, will yield a relation that will have a degree of 6 ($3 + 3$: sum of degrees of Student and Instructor)

Student \times Instructor

Stud#	Stud-Name	Hosteler	Inst#	Inst-Name	Subject
S001	Meenakshi	Y	I01	K. Lal	English
S001	Meenakshi	Y	I02	R.L. Arora	Maths
S002	Radhika	N	I01	K. Lal	English
S002	Radhika	N	I02	R.L. Arora	Maths

and a cardinality 4 (2×2 : product of cardinalities of two relations). The resulting relation has been shown in Fig. 11.17(a).

See the resulting relation contains all possible combinations of tuples of the two relations.

The illustration of cartesian product of two tables has been given in Fig. 11.17(b).

Fig. 11.17 (a) The **Cartesian Product** of **Student** and **Instructor** tables.

Table A

1	ABC	15
2	DEF	30

Table B

101	M	Engl.
102	R	Maths

Table A \times B is

Combination of

1	ABC	15
---	-----	----

 with

101	M	Engl.
102	R	Maths

 is

1	ABC	15	101	M	Engl.
1	ABC	15	102	R	Maths

Combination of

2	DEF	30
---	-----	----

 with

101	M	Engl.
102	R	Maths

 is

2	DEF	30	101	M	Engl.
2	DEF	30	102	R	Maths

Thus Table storing cartesian product $A \times B$ is as shown below :

1	ABC	15	101	M	Engl.
1	ABC	15	102	R	Maths
2	DEF	30	101	M	Engl.
2	DEF	30	102	R	Maths

Fig. 11.17 (b) Cartesian Product of two tables.

11.5.4D The Union Operation

The *union* operation is a binary operation that requires two relations as its operands. It produces a third relation that contains tuples from both the operand relations [Fig. 11.18(a)]. The *union* operation is denoted by \cup . Thus, to denote the union of two relations X and Y , we will write as $X \cup Y$.

Before we consider and see how it works, one thing must be remembered about *union* that is, both the operand relations must be *union-compatible*. That means, for a union operation $A \cup B$ to be valid, the following two conditions must be satisfied by the two operands A and B :

1. The relations A and B must be of the same degree. That is, they must have the same number of attributes.
2. The domains of the i th attributes of A and the i th attribute of B must be the same.

In Fig. 11.12, two relations have been shown that display result from the queries

(a) $\sigma_{\text{Cust-name} = \text{banker-name}}$ (**Customer**) and (b) $\sigma_{\text{amount} > \text{balance}}$ (**Customer**).

Now, if you want to obtain *union* of these two relations (shown in Fig. 11.12), you may write

$(\sigma_{\text{cust-name} = \text{banker-name}}(\text{Customer})) \cup (\sigma_{\text{amount} > \text{balance}}(\text{Customer}))$

And the resultant relation will be as shown in Fig. 11.18(b).

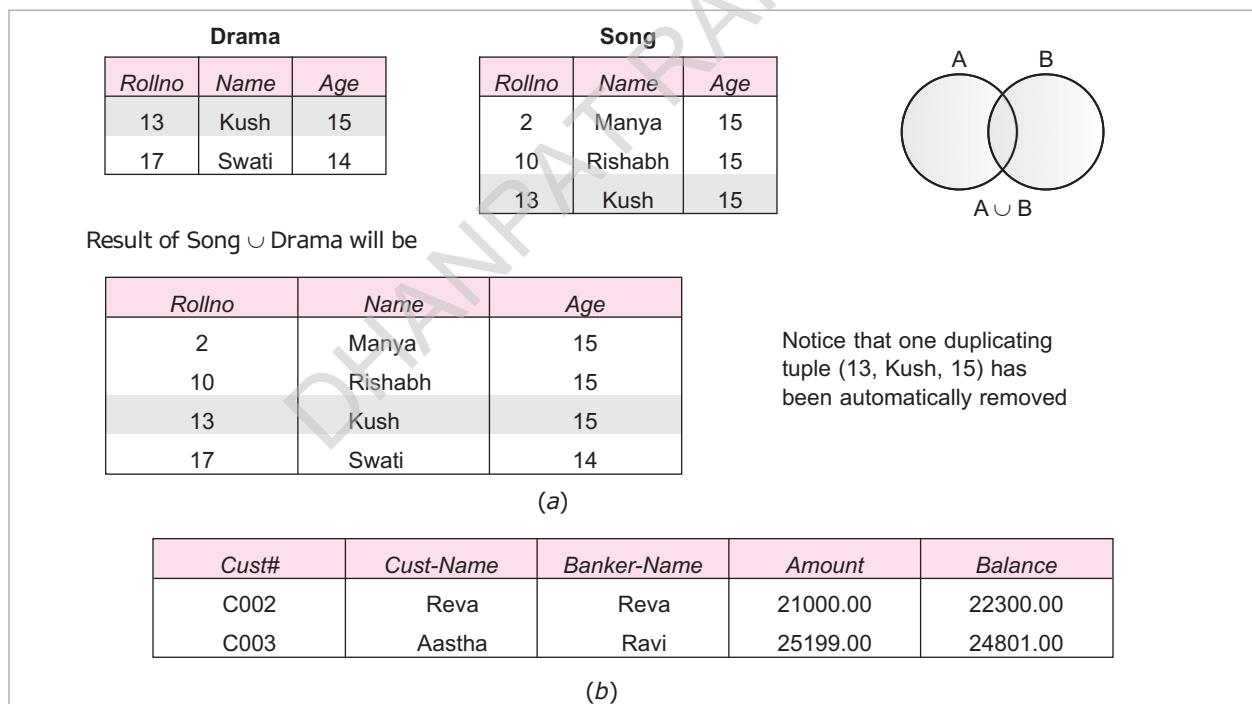


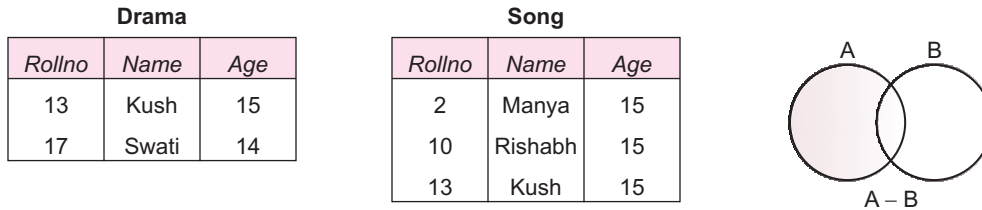
Fig. 11.18 (a) Union operation, (b) Result of $(\sigma_{\text{cust-name} = \text{banker-name}}(\text{Customer})) \cup (\sigma_{\text{amount} > \text{balance}}(\text{Customer}))$. (Named as **NewCust**)

Do remember that in the resultant relation, all the duplicating tuples will automatically be removed.

11.5.4E The Set Difference Operation

The *set difference* operation, denoted by $-$ (minus) allows us to find tuples that are in one relation but not in another [Fig. 11.19(a)]. The expression $A - B$ results in a relation

containing those tuples in **A** but not in **B**. Let us assume that the relation shown in Fig. 11.18 has been named as *NewCust*. Now the result of *difference* between *Customer* (shown in Fig. 11.11) and *NewCust* will be as shown in Fig. 11.19(b).



Result of Drama – Song will be

Rollno	Name	Age
17	Swati	14

(a) Result of *Drama – Song*

Cust#	Cust-Name	Banker-Name	Amount	Balance
C001	Anjuman	Rakesh	17510.00	17510.00

(b) Result of *Customer – NewCust*.

Fig. 11.19 Set Difference Operation

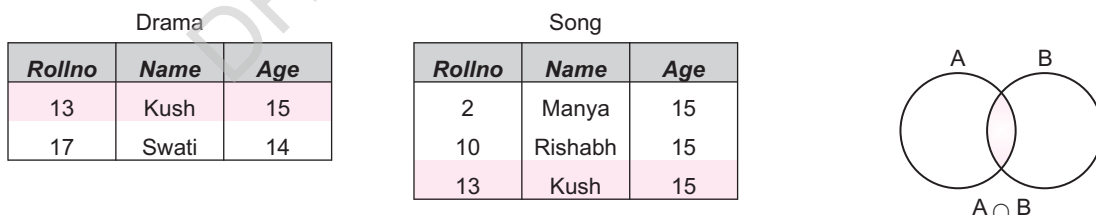
NOTE

Any relational algebra expression using set intersection can be rewritten by replacing the intersection operation with a pair of set difference operations as :

$$A \cap B = A - (A - B)$$

10.5.4F The Set Intersection Operation

The *set intersection* operation finds tuples that are common to the two operand relations [Fig. 11.20(a)]. The *set intersection* operation is denoted by \cap . That means $A \cap B$ will yield a relation having tuples common to **A** and **B**. The result produced by *Customer* \cap *NewCust* (*customer* shown in Fig. 11.11, *NewCust* shown in Fig. 11.18) has been shown in Fig. 11.20(b).



Result of Song \cap Drama will be

Rollno	Name	Age
13	Kush	15

(a)

Cust#	Cust-Name	Banker-Name	Amount	Balance
C002	Reva	Reva	21000.00	22300.00
C003	Aastha	Ravi	25199.00	24801.00

(b)

Fig. 11.20 (a) Set Intersection operation, (b) Result of *Customer* \cap *NewCust*.

11.6 COMPARISON OF DATA MODELS

Data models can be evaluated on the basis of usability, implementability and performance.

How easy are they for learning purpose and usage ?

How well they define and manipulate data structures ?

Data experts have been debating for some time which DBMS data model is the 'best'. The answer depends, in part, on philosophical orientation. The *hierarchical* model is the oldest model and has long been the most popular on mainframes. People who think the user (or programmer) should be able to have some control over the details of storage

allocation and search paths often prefer network systems. But rarely network models have gone commercial ; mostly they remain confined to development or testing stages. People who believe that users should be isolated from the mechanisms used to access data, and those who assign high priority to a model that is easy to understand and construct, frequently favour relational systems. However, few database systems are fully relational. Instead they graft relational features on a basic hierarchical/network structure.



1. What is the function of select and project operations ?
2. What is cartesian product ? How is it different from join operation ?
3. What does union operation do ? What are the conditions to carry out union operation ?
4. What are the set difference and set intersection operations ? How are they different ?

Let Us Revise

- ❖ A collection of data is referred to as database and a database (management) system is basically a computer based record keeping system.
- ❖ Database systems help reduce data redundancy, data inconsistency and facilitate sharing of data, standardization of data and data security.
- ❖ A database is implemented through three general levels : internal (*closest to physical storage*), conceptual (*describes entire data and its relationships*) and external (*closest to the users*).
- ❖ Data independence allows modification of a scheme definition without affecting other scheme definitions.
- ❖ There are two levels of data independence : logical and physical.
- ❖ There are three models available for database management : relational, network & hierarchical.
- ❖ A relational data model organizes the data into tables known as relations.
- ❖ A network data model represents data by collection of records and relationships among data are represented by links (pointers).
- ❖ A hierarchical data model represents data by records organized in form of trees and the relationships among data are represented by links.
- ❖ A view is a virtual table derived from one or more underlying base tables.
- ❖ There is no stored file created for storing a view's contents, rather, only the view definition is stored.
- ❖ The relational algebra is a collection of operations on relations. The various operations of relational algebra are select, project, cartesian product, union, set difference, set intersection, join etc.
- ❖ The selection extracts tuples from a relation depending upon a condition (symbol σ).
- ❖ The projection extracts columns from a relation. (Symbol π).
- ❖ The cartesian product yields a relation with all possible combinations of the tuples of the two relations operated upon. (Symbol \times).
- ❖ The union yields a relation containing tuples from both the operand relations. (Symbol \cup).
- ❖ The set difference finds tuples that are in one relation but not in another. (Symbol $-$).
- ❖ The set intersection finds tuples that are common to the two operand relations (Symbol \cap).

SOLVED PROBLEMS

- 1 ■ What is a database system ? What is its need ?

SOLUTION. A database is a collection of interrelated data and a database system is basically a computer based record keeping system.

A typical file processing system suffers from some major limitations like *data redundancy* (duplication of data), *data inconsistency*, *unsharable data*, *unstandardized data*, *insecure data*, *incorrect data* etc. On the other hand, a database system overcomes all these limitations and ensures continues efficiency.

The advantages provided by a database system are :

- | | | |
|-----------------------------|------------------------------------|-----------------------|
| (i) Reduced data redundancy | (ii) Controlled data inconsistency | (iii) Shared data |
| (iv) Standardized data | (v) Secured data | (vi) Integrated data. |

Therefore, to have the systems with increased performance and efficiency, the database systems are preferred.

- 2 ■ Can you think of disadvantages of using a database system ? What are these ?

SOLUTION. With the complex tasks to be performed by database systems, certain things may crop up that may be termed as disadvantages of using database system. These are :

1. Security may be compromised without good controls.
2. Integrity may be compromised without good controls.
3. Extra hardware may be required.
4. Performance overhead may be significant.
5. System is likely to be complex.

- 3 ■ How many types of users work on database systems ?

SOLUTION. A primary goal of a database system is to provide an environment for retrieving information from and storing new information into the database. There are three different types of database system users, differentiated by the way they expect to interact with the system.

1. **End User.** An end user is a person who is not a computer trained person but uses the database to retrieve some information. For example, in a railway reservation system, a customer retrieving the train-details is an end user.
2. **Application System Analyst.** This user is concerned about all of the database at logical level *i.e.*, what all data constitutes the database ? What are the relationships between the data-entities etc. without considering the physical implementation details.
3. **Physical Storage System Analyst.** This user is concerned with the physical implementation details of the database such as which storage device, which storage technique should be used ? etc.

- 4 ■ What are the various levels of data abstraction in a database system ?

SOLUTION. There are *three* levels of data abstraction :

1. **Internal Level (Physical Level).** This level describes how the data is actually stored on the storage medium. At this level, complex low-level data structures are described in details.
2. **Conceptual Level.** This level describes *what* data are actually stored in the database. It also describes the relationships existing among data. At this level, the database is described logically in terms of simple data-structures.
3. **External Level (View Level).** This level is concerned with the way the data is viewed by individual users. Only a part of the database relevant to the user(s) is provided to them through this level.

- 5 ■ What are the various data models available for database systems ?

SOLUTION. A data model is a collection of conceptual tools for describing data, data relationships, data semantics etc. There are generally *three* data models available : relational, network and hierarchical model.

1. **Relational Model.** The relational model represents data and relationships among data by a collection of tables known as *relations*, each of which has a number of columns with unique names.

2. **Network Model.** The network model represents data by collections of *records* and relationships among data are represented by *links* which can be viewed as pointers. The records in the database are organized as collection of *arbitrary graphs*.
3. **Hierarchical Model.** The hierarchical model is similar to the network model in the sense that data and relationships among data are represented by records and links respectively. It differs from the network model in that the records are organized as collections of *trees* rather than *arbitrary graphs*. The relational model differs from the network and hierarchical models in that it does not use pointers or links. Instead, the relational model relates records by the values they contain.

- 6 ■ What are views ? How are they useful ?

SOLUTION. A view is a virtual table that does not really exist in its own right but is instead derived from one or more underlying base table(s). The view is a kind of table whose contents are taken upon other tables depending upon a given query condition. No stored file is created to store the contents of a view rather its definition is stored only.

The usefulness of views lies in the fact that they provide an excellent way to give people access to some but not all of the information in a table.

- 7 ■ Differentiate between **Candidate Key** and **Primary Key** in context of RDBMS. (Delhi 2008)

SOLUTION. Candidate key. A candidate key is the one that is capable of becoming primary key *i.e.*, a field or attribute that has unique value for each row in the relation.

Primary key is a designated attribute or a group of attributes whose values can uniquely identify the tuples in the relation.

- 8 ■ Give a suitable example of a table with sample data and illustrate Primary and Candidate Keys in it.

(Delhi 2012)

SOLUTION. Example : **Table : Class 11**

AdmNo	RollNo	Name	Marks
1011	1	Rahat	85
1083	2	Irfan	75
2011	3	Maya	63
1000	4	Shaun	60
999	5	Sukhi	92
1200	6	Zoya	86

In the table, columns **AdmNo** and **RollNo** have unique values for each row, so both are candidates to become primary keys. Hence both of these columns are **Candidate Keys**. Out of these two, we can assign one as Primary key and the other one will become alternate key.

Candidate keys : **AdmNo, RollNo**

Primary key : **RollNo**

Alternate key : **AdmNo**

- 9 ■ Differentiate between Candidate Key and Alternate Key in context of RDBMS. (Outside Delhi 2008)

SOLUTION. Candidate key. A candidate key is the one that is capable of becoming primary key *i.e.*, a field or attribute that has unique value for each row in the relation.

A candidate key that is not a primary key is called an Alternate Key.

- 10 ■ Differentiate between **primary key** and **alternate key**. (Delhi 2007)

SOLUTION. Primary Key. It is the set of one or more attributes that can uniquely identify tuples within a relation.

Alternate Key. It is a candidate key which is not primary key.

- 11 ■ What is the importance of a Primary Key in a table ? Explain with a suitable example. (Outside Delhi 2007)

SOLUTION. A **Primary Key** is a set of one or more attributes that can uniquely identify tuples within the relation. For example, in the following table **Student**, the column **Rollno** can uniquely identify each row in the table, hence **Rollno** is the primary key of the following table.

Rollno	Name	Marks	Grade
1	.	.	.
2	.	.	.
3	.	.	.
4	.	.	.

- 12 ■ What is a primary key in a table ? (Outside Delhi 2003)

SOLUTION. A **Primary Key** is a set of one or more attributes that can uniquely identify tuples within the relation

- 13 ■ What is relation ? Define the relational data model. (Delhi 2002)

SOLUTION. A relation is a table having atomic values, unique rows and unordered rows and columns.

The relational model represents data and relationships among data by a collection of tables known as *relations*, each of which has a number of columns with unique names.

- 14 ■ What is an Alternate Key ? (Delhi 2006)

SOLUTION. A candidate key that is not a primary key is called an Alternate Key. In Suppliers table if there are two candidate keys – *Suppld* and *Supp_Name* and *Suppld* is the primary Key then *Supp_Name* is the alternate key.

- 15 ■ What do you understand by the terms Primary Key and Degree of a relation in relational database ? (Delhi 05)

SOLUTION. **Primary key** is an attribute or a group of attributes whose values can uniquely identify the tuples in the relation.

Degree. Number of attributes in a relation are called its *degree*.

- 16 ■ What do you understand by the terms Candidate Key and Cardinality of a relation in relational database ?

(Outside Delhi 2005)

SOLUTION. **Candidate key.** A candidate key is the one that is capable of becoming primary key *i.e.*, a field or attribute that has unique value for each row in the relation.

Cardinality of a relation represents number of rows in the relation.

- 17 ■ Differentiate between the terms **Degree** and **Cardinality** in context of RDBMS

(Delhi 2008C)

Or

What is the difference between degree and cardinality of a table ? What is the degree and cardinality of the following table ? (Delhi 2013)

Eno	Name	Salary
101	John Fedrick	45000
103	Raya Mazumdar	50600

SOLUTION. Refer to problem 14 and 15 for definition of degree and cardinality.

Degree : 3

Cardinality : 2

- 18 ■ What do you understand by Primary Key ? Give a suitable example of Primary Key from a table containing some meaningful data. (Outside Delhi 2010)

SOLUTION. A primary key is a designated attribute or group of attributes whose values can uniquely identify the tuples in the relation.

Example Table : ITEM

Ino	Item	Quantity
I01	Pen	560
I02	Pencil	340
I04	CD	540
I09	DVD	200
I10	Floppy	400

Primary Key

- 19 ■ What do you understand by Candidate Keys in a table ? Give a suitable example of Candidate keys from a table containing some meaningful data. Or (Delhi 2010)

What are candidate keys in a table ? Give a suitable example of candidate keys in a table. (Delhi 2009)

SOLUTION. A candidate key is the one that is capable of becoming primary key i.e., a field or attribute that has unique value for each row in the relation.

Example Table : ITEM

Ino	Item	Quantity
I01	Pen	560
I02	Pencil	340
I04	CD	540
I09	DVD	200
I10	Floppy	400

Candidate Keys

- 20 ■ What is the purpose of a key in a table ? Give an example of a key in a table. (Outside Delhi 2009)

SOLUTION. An attribute/group of attributes in a table that identifies each tuple uniquely is known as a Key (also called Primary Key).

Example Table : Item

Ino	Item	Qty
I01	Pen	560
I02	Pencil	780
I03	CD	450
I09	Floppy	700
I05	Eraser	300
I03	Duster	200

Key

- 21 ■ What do you understand by Union and Cartesian Product operations in relational algebra ? (Delhi 2011)

Or

Explain the concept of Cartesian Product between two tables, with the help of appropriate example.

(Outside Delhi 2014)

Or

Explain the concept of Union between two tables, with the help of appropriate example. (Delhi 2014)

SOLUTION. Cartesian Product. The Cartesian product of two relations **A** and **B** is written as $A \times B$. The cartesian product of two relations yields a relation with all possible combinations of the tuples of the two relations operated upon.

For example, given two relations **Student** and **Instructor** as shown below :

Student

Stud#	Stud-Name	Hosteler
S001	Meenakshi	Y
S002	Radhika	N
S003	Abhinav	N

Instructor

Inst#	Inst-Name	Subject
I01	K. Lal	English
I02	R.L. Arora	Maths

The cartesian product of these two relations, **Student** \times **Instructor**, will yield a relation as :

Stud#	Stud-Name	Hosteler	Inst#	Inst-Name	Subject
S001	Meenakshi	Y	I01	K. Lal	English
S001	Meenakshi	Y	I02	R.L. Arora	Maths
S002	Radhika	N	I01	K. Lal	English
S002	Radhika	N	I02	R.L. Arora	Maths
S003	Abhinav	N	I01	K. Lal	English
S003	Abhinav	N	I02	R.L. Arora	Maths

Union. The union produces a third relation that contains tuples from both the operand relations which must be *union-compatible*. To denote the union of two relations X and Y, we write as $X \cup Y$.

Table X

No	Name
1	Toy
2	Drum

Table Y

No	Name
3	Pencil

SQL Statement

SELECT * FROM X UNION SELECT * FROM Y ;

will give table : X UNION Y

No	Name
1	Toy
2	Drum
3	Pencil

- 22 ■ What do you understand by Selection and Projection operations in relational algebra ? (O.D. 2011)

SOLUTION. Selection means selecting some rows (tuples) from a relation according to given condition

e.g., $\sigma_{\text{price} > 20.2}(\text{Item})$

Project operation yields a vertical subset of a given relation(i.e., select all tuples containing only given columns of a relation).

e.g., $\pi_{\text{NAME, DESIG}}(\text{Employee})$

- 23 ■ Observe the following table CANDIDATE carefully and write the name of the RDBMS operation out of (i) SELECTION (ii) PROJECTION (iii) UNION (iv) CARTESIAN PRODUCT, which has been used to produce the output as shown in RESULT. Also, find the Degree and Cardinality of the RESULT.

Table : CANDIDATE

NO	NAME	STREAM
C1	AJAY	LAW
C2	ADITI	MEDICAL
C3	ROHAN	EDUCATION
C4	RISHAV	ENGINEERING

Table : RESULT

NO	NAME
C3	ROHAN

(CBSE D 2017)

SOLUTION. SELECTION and PROJECTION, both operations have taken place.

Result : Degree = 2 ; Cardinality = 1

- 24 ■ Observe the following table MEMBER carefully and write the name of the RDBMS operation out of (i) SELECTION (ii) PROJECTION (iii) UNION (iv) CARTESIAN PRODUCT, which has been used to produce the output as shown in RESULT. Also, find the Degree and Cardinality of the RESULT.

Table : MEMBER

NO	MNAME	STREAM
M001	JAYA	SCIENCE
M002	ADITYA	HUMANITIES
M003	HANSRAJ	SCIENCE
M004	SHIVAK	COMMERCE

Table : RESULT

NO	MNAME	STREAM
M002	ADITYA	HUMANITIES

(CBSE OD 2017)

SOLUTION. SELECTION operation ; Degree = 3, Cardinality = 1

- 25 ■ A database is to contain information about persons and skills. At a particular time the following persons are represented in the database, and their skills are as indicated.

Person	Skills
Anuj	Programming
Bhawna	Operating and Programming
Chetan	Engineering and Programming
David	Operating and Engineering

For each person, the database contains various personal details, such as address. For each skill it contains an identification of the appropriate basic training course, an associated job grade code, and other information. The database also contains the date each person attended each course, where applicable (the assumption is that attendance at the course is essential before the skill can be said to be acquired). Sketch a relational view for this data.

SOLUTION. Relational view

Person		Skill		
Pname	Address	Skill-Name	Course	Jobcode
Anuj	Programming	-	-
Bhawna	Operating	-	-
Chetan	Engineering	-	-
David			

Perskill		
Pname	SkillName	Date
Anuj	Programming	-
Bhawna	Operating	-
Bhawna	Programming	-
Chetan	Engineering	-
Chetan	Programming	-
David	Operating	-
David	Engineering	-

SOLVED PROBLEMS

1. What is a relation ? What is the difference between a tuple and an attribute ?

(CBSE Delhi 1998)

Solution. A relation is a table having atomic values, unique rows and unordered rows and columns.

A row in a relation is known as **tuple** whereas a column of a table is known as an **attribute**.

2. What is data redundancy ? What are the problems associated with it ?

Solution. Duplication of data is data redundancy. It leads to the problems like wastage of space and data inconsistency.

3. How many types of users work on database systems ?

Solution. A primary goal of a database system is to provide an environment for retrieving information from and storing new information into the database. There are three different types of database system users, differentiated by the way they expect to interact with the system.

1. **End User.** An end user is a person who is not a computer trained person but uses the database to retrieve some information. For example, in a railway reservation system, a customer retrieving the train-details is an end user.
2. **Application System Analyst.** This user is concerned about all of the database at logical level *i.e.*, what all data constitutes the database ? What are the relationships between the data-entities etc. without considering the physical implementation details.
3. **Physical Storage System Analyst.** This user is concerned with the physical implementation details of the database such as which storage device ? Which storage technique should be used ?

4. What are the various data models available for database systems ?

Solution. A data model is a collection of conceptual tools for describing data, data relationships, data semantics etc. There are generally three data models available : relational, network and hierarchical model.

1. **Relational Model.** The relational model represents data and relationships among data by a collection of tables known as *relations*, each of which has a number of columns with unique names.
2. **Network Model.** The network model represents data by collections of *records* and relationships among data are represented by *links* which can be viewed as pointers. The records in the database are organized as collection of *arbitrary graphs*.
3. **Hierarchical Model.** The hierarchical model is similar to the network model in the sense that data and relationships among data are represented by records and links respectively. It differs from the network model in that the records are organized as collections of *trees* rather than arbitrary graphs.

The relational model differs from the network and hierarchical models in that it does not use pointers or links. Instead, the relational model relates records by the values they contain.

Glossary

Alternate Key	A candidate key which is not primary key.
Attribute	A column of a relation.
Base Table	A table having an independent existence ; represented in storage by a distinct stored file.
Candidate Keys	Attribute combinations in a relation that can serve as a primary key.
Cardinality	Number of tuples in a relation.
Data Independence	Ability to modify a scheme definition within a database without affecting other scheme definitions.
Data Redundancy	Duplication of data.
Database	Collection of interrelated data.
Database System	A computer based recordkeeping system.
Degree	Number of attributes in a relation.
Domain	A pool of values from which the actual values appearing in a given column are drawn.
Foreign Key	Non-key attribute, which is primary key of another table.
Logical Data Independence	The property of being able to change the overall logical structure of the database without changing the application program's view of data.
Physical Data Independence	The property of being able to change the physical structure of the data without changing the logical structure.
Physical Database	A database in the form in which it is stored on the storage media, including pointers or other means of interconnecting it.
Primary Key	Set of one or more attributes that can uniquely identify tuples within a relation.
Relation	Data arranged in rows and columns and having certain properties.
Relational Algebra	A language providing a set of operators for manipulating relations.
Tuple	A row of a relation.

ASSIGNMENTS

SHORT ANSWER QUESTIONS

1. Draw a diagram explaining various levels of data abstraction.
 2. What is meant by “data independence”? Make a list of data independence capabilities.
 3. What are the main differences between a file-processing system and a database management system?
 4. This chapter has described some of the major advantages of a database system. What are the disadvantages?
 5. Explain the difference between physical and logical data independence.
 6. Illustrate the difference between the three levels of data abstraction.
 7. What is a relation? What is the difference between a tuple and an attribute? (Delhi 1998)
 8. What is a relation? Define the properties of a relation.
 9. Summarise the major differences between a relation and a traditional file.
 10. Give a network data-structure diagram for the following relational database :
 - (a) lives (person-name, street, city)
works (person-name, company-name, salary)
located-in (company-name, city)
manages (person-name, manager-name)
 - (b) course (course-name, room, instructor)
enrolment (course-name, student-name, grade)
 11. For the relational databases mentioned in questions 10, give an hierarchical data-structure.

Give the following relations

EMPL	(FirstName, Lastname, Workdept, Empno, Edulevel, Jobcode, Salary, Bonus, Hiredate)
Dept	(Deptno, Deptname, Mgrno, Adurdept)
Supplier	(Sno, Pno)
Part	(Pno, colour)
- Write relational expressions for questions 12 – 25 using the above relations.*
12. List employees’ first and last names.
 13. Give all information on employees in manufacturing systems.
 14. Just give employee names in manufacturing systems.
 15. Give employee names outside manufacturing systems.
 16. List employee names and their departments for employee numbers between 100 and 300.
 17. List employees numbers, names and educational levels for three education levels 16, 18, and 20.
 18. List employee number and name in the manufacturing systems for job code 54.
 19. Same request as above with education level below 15.
 20. List those employees’ last name and monthly salary plus bonus, whose job code is at least three times their education level and whose annual basic salary plus bonus exceed 100000.
 21. List employee last name, work departments and monthly salary for those in Services, Planning and Information centre departments.
 22. List employee last name, work department, department number, and department name.
 23. List employee number, name and department number for employees who are managers.
 24. Which employees earn more than their managers?
 25. List suppliers who supply only red parts.
 26. Give a suitable example of a table with sample data and illustrate Primary and Alternate Keys in it. (Outside Delhi 2012)
 27. Explain the concept of candidate keys with the help of an appropriate example. (Outside Delhi 2013)