CSC 308

Database Design and Management

Table of Contents

Chapter 1 Introduction to Databases

1.1 Introduction

The database is such an integral part of our day-to-day life that often we are not aware that we are using one. The database is now the underlying framework of the information system and has fundamentally changed the way many organizations operate. In particular, the developments in this technology over the last few years have produced systems that are more powerful and more intuitive to use. This development has resulted in increasing availability of database systems for a wider variety of users. Unfortunately, the apparent simplicity of these systems has led to users creating databases and applications without the necessary knowledge to produce an effective and efficient system.

1.2 Traditional File-based Systems

Before we actually get into database, there is a need to introduce the database by reviewing its predecessors: the manual filing system and the file-based system. A few good reasons for studying the history (database), is that by the end of this section, we will learn that there are better ways to handle data, and that understanding the problems inherent in old systems may prevent us from repeating these problems in new (database) systems.

1.2.1 Manual Filing System

Physical filing cabinet is an example of manual filing system. The manual filing system works well as long as the number of items to be stored is small. It even works adequately when there are large numbers of items, and we only have to store and retrieve them. However, the manual filing system breaks down when we have to cross-reference or process the information in the files.

1.2.2 File-based Approach

File-based systems were an early attempt to computerize the manual filing system that we are all familiar with. File-based system is a collection of application programs that perform services for the end-users, such as the production of reports. Each program defines and manages its own data.

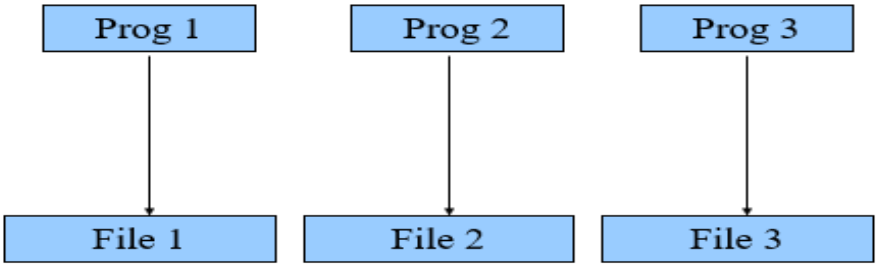


Figure 1.1 File-based Approach

1.2.3 Advantages and Disadvantages of the File-Based Approach

Advantages

1. Data private to a particular program
2. If new data items get added, it only has an effect on the local program

Disadvantages

1. No data sharing
2. Separation and isolation of data
3. Program-data dependence
4. Duplication of data
5. Potentially inconsistent data

1.3 Database Approach

The database approach tends to resolve the problems with the file-based approach. What emerged were the database and database management system (DBMS).

1.3.1 The Database

A database is a collection of non-redundant data shareable between different application systems. It can also be defined as a shared collection of logically related data and its description, designed to meet the information needs of an organization.

1.3.2 The Database Management System (DBMS)

The DBMS is a software system that enables users to define, create, maintain, and control access to the database. All access to the database is through the DBMS. Typically, a DBMS provides the following facilities:

- It allows users to define the database, usually through a **Data Definition Language (DDL)**.
- It allows users to insert, update, delete, and retrieve data from the database, usually through a **Data Manipulation Language (DML)**. Example through SQL.
- It provides controlled access to the database. For example, it may provide a security system, an integrity, a concurrency control system, a recovery control system, a user-accessible catalog, etc.

1.3.3 Functions of a DBMS

Functions and services of a multi-user DBMS include data storage, retrieval, and update; a user-accessible catalog; transaction support; concurrency control and recovery services; authorization services; support for data communication; integrity services; services to promote data independence; and utility services. We would discuss this further in later chapters.

1.3.4 Components of the DBMS Environment

We can identify five major components in the DBMS environment: hardware, software, data, procedures, and people, as illustrated in the figure below:

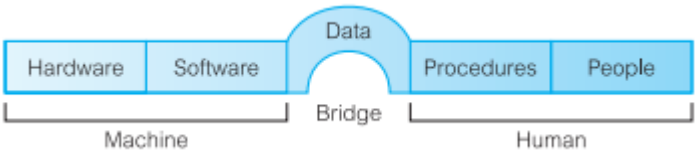


Figure 1.2 The DBMS Environment

- Hardware are the physical devices in the DBMS environment.
- The software component comprises the DBMS software itself, the application programs, and the operating system.
- The database contains both the operational data and the metadata, the "data about data."
- The users of the system and the staff who manage the database require documented procedures on how to use or run the system. Procedures refer to the instructions and rules that govern the design and use of the database. These procedures may consist of instructions on how to:
    - log on to the DBMS
    - start and stop the DBMS
    - make backup copies of the database, etc.
- The final component is the people involved with the system. They consist of:
    - Data and Database Administrators
    - Database Designers
    - Application Developers
    - End-Users

  We discuss more on people in Section 1.4.

1.3.5 (Database) Application Programs

Application programs are computer programs that interact with the database by issuing an appropriate request (typically and SQL statement) to the DBMS.

Users interact with the database through a number of application programs that are used to create and maintain the database and to generate information. These programs can be conventional batch applications or, more typically nowadays, online applications. The application programs may be written in a programming language or in higher-level fourth-generation language.

1.3.6 Database System

A database system is a database supported by a DBMS plus any applications programmes to support a specific data need (organisational/personal/functional, etc.)

1.4 Roles in the Database Environment

In this section, we examine what we listed in the section 1.3 as the fifth component of the DBMS environment: the **people**. We can identify four distinct types of people who participate in the DBMS environment: data and database administrators, database designers, application developers, and end-users.

1.4.1 Data and Database Administrators

Data Administrator (DA) is responsible for the management of the data resource, including database planning; development and maintenance of standards, policies, and procedures; and conceptual/logical database design.

The Database Administrator (DBA) is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users.

1.4.2 Database Designers

In large database design projects, we can distinguish between two types of designers: logical database designers and physical database designers.

The **logical database designer** is concerned with identifying the data (that is, the entities and attributes), the relationships between the data, and the constraints on the data that is to be stored in the database.

The **physical database designer** decides how the logical database design is to be physically realized. This involves:

- mapping the logical database design into a set of tables and integrity constraints.
- selecting specific storage structures and access methods for the data to achieve good performance.
- designing any security measures required on the data.

1.4.3 Application Developers

An application developer is a person who writes computer programs to meet specific requirements. They are usually responsible for translating software requirements into workable programming code. Each program contains statements that request the DBMS to perform some operation on the database, which includes retrieving data, inserting, updating, and deleting data.

1.4.4 End-Users

The end-users are the "clients" of the database. End-users can be classified according to the way they use the system:

- Naïve users are typically unaware of the DBMS. This means that they do not need to know anything about the database or the DBMS. For example, the checkout assistant at the local supermarket uses a bar code reader to find out the price of the item. However, there is an application program present that reads the bar code, looks up the price of the item in the database, reduces the database field containing the number of such items in stock, and displays the price on the till.
- Sophisticated users are familiar with the structure of the database and the facilities offered by the DBMS. Sophisticated users may use a high-level query language such as SQL to perform required operations on the database.

1.5 Advantages and Disadvantages of DBMS

Advantages

1. Control of data redundancy
2. Data consistency
3. Sharing of data
4. Improved data integrity
5. Improved security
6. Economy of scale
7. Increased concurrency
8. Improved backup and recovery services

Disadvantages

1. Complexity
2. Size
3. Cost of DBMS
4. Additional hardware costs
5. Cost of conversion
6. Performance
7. Greater impact of a failure

Chapter 2 Database Environment

A major aim of a database system is to provide users with an abstract view of data, hiding certain details of how data is stored and manipulated. Therefore, the starting point for the design of a database must be an abstract and general description of the information requirements of the organization that is to be represented in the database.

Furthermore, because a database is a shared resource, each user may require a different view of the data held in the database. To satisfy these needs, the architecture of most commercial DBMSs available today is based to some extent on the so-called ANSI-SPARC architecture. In this chapter, we discuss various architectural and functional characteristics of DBMSs.

2.1 The Three-Level ANSI-SPARC Architecture

The American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC), ANSI-SPARC or three-level architecture is an abstract design standard for a Database Management System first proposed in 1975. It is a framework for managing access to data which involves three layers or schemas. The objective is to separate each user's view of the database from the way the database is physically represented. It also answers user needs for aspects such as data independence (which we'll come to momentarily).

The ANSI-SPARC database architecture uses three levels of abstraction, i.e., three levels at which data can be described: external, conceptual, and internal.

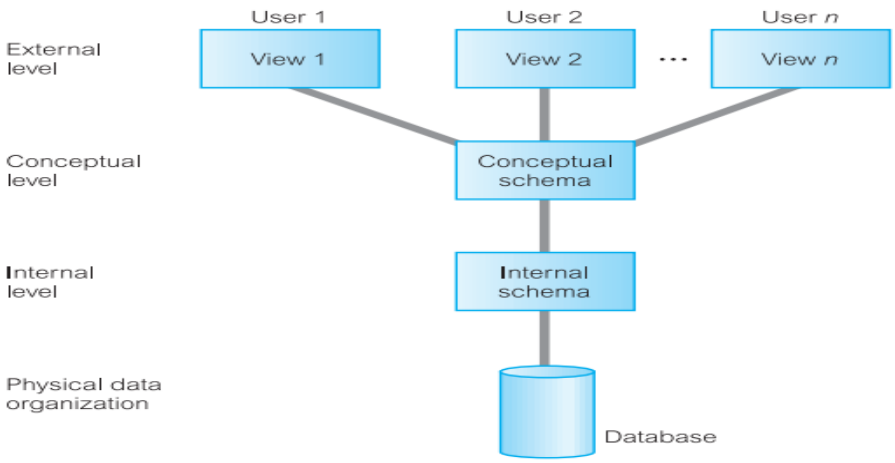


Figure 2.1 The ANSI-SPARC three-level architecture

2.1.1 The External Level

The external level consists of the users' views of the database. It is the way users perceive the data. This level describes that part of the database that is relevant to each user. An

example is that one user may have his WhatsApp wallpaper appearance set to dark mode, while another may like a white background.

In addition, different views may have different representations of the same data. For example, one user may view dates in the form (day, month, year), while another may view dates as (year, month, day).

2.1.2 The Conceptual Level

The conceptual level is the community view of the database. It is a way of describing **what** data is stored within the whole database and how the data is inter-related (i.e., the relationships among the data). This level contains the logical structure of the entire database as seen by the DBA. It specifies the information content of the entire database, independent of storage considerations.

The conceptual level represents all entities, their attributes, and their relationships, as well as the constraints on the data, and security and integrity information. However, this level must not contain any storage-dependent details. For instance, the description of an entity should contain only data types of attributes (for example, integer, real, character) and their length (such as the maximum number of digits or characters), but not any storage considerations, such as the number of bytes occupied.

2.1.3 The Internal Level

The internal level is the computer's view of the database. It is the way the DBMS and the OS perceive the data, where the data is actually stored using data structures and file organisations. This level is the physical representation of the database on the computer and specifies **how** the database is physically represented on the computer system.

The internal level is concerned with things such as: storage space allocation for data and indexes; record descriptions for storage (with stored sizes for data items); record placement; data compression and data encryption techniques.

2.2 Significance of the ANSI-SPARC Architecture

The ANSI-SPARC architecture is needed based on the following:

1. It has the aim of enabling users to access the same data but with a personalised view of it. So, each user gets to change the way he or she views the data, and this change should not affect other users.
2. Users should not have to deal directly with physical database storage details, such as indexing or hashing.
3. The DBA should be able to change the conceptual structure of the database without affecting all users.

2.3 Database Schemas

The overall description of the database is called the database schema. It is a description of the database structure and are written using a Data Definition Language (DDL). There are three different types of schemas in the database, and these schemas correspond to the three levels in the ANSI-SPARC architecture: the external schema, the conceptual schema, and the internal schema. The three levels are illustrated in the figure **2.2** below.

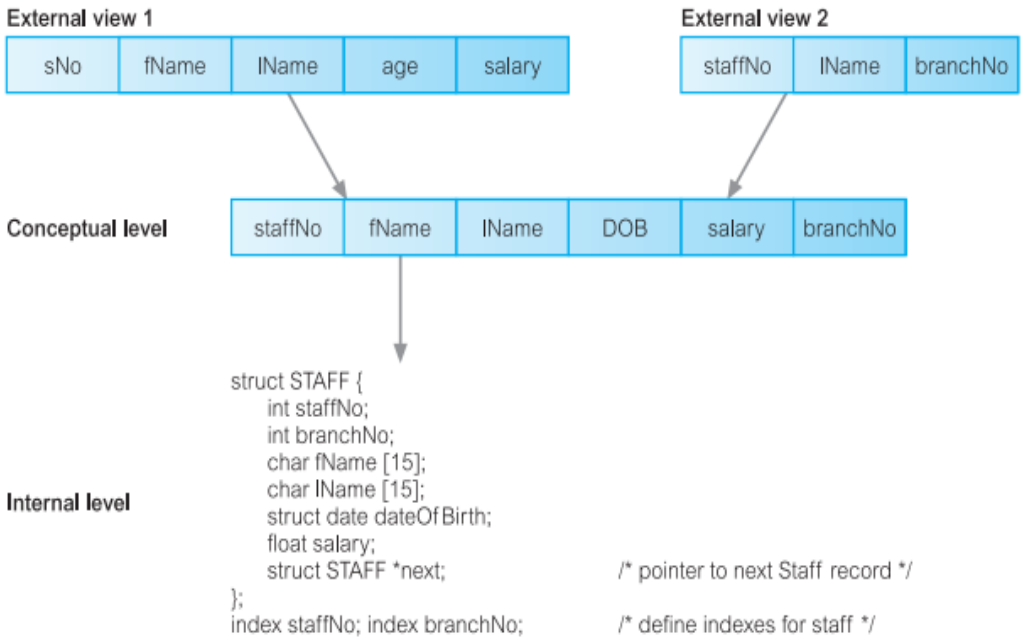


Figure 2.2 Differences between the three levels

2.3.1 External Schema

The external schema or user/applications view describe different external views of the data and there may be many external schemas for a given database.

2.3.2 Conceptual Schema

The conceptual schema or logical/global view describes all the data items (entities, attributes) and relationships between them, together with integrity constraints. There is only one conceptual schema per database.

2.3.3 Internal Schema

The internal schema or physical view defines the physical storage structure of the database. It contains definitions of stored records, the methods of representation, the data fields, etc. There is only one internal schema per database.

2.4 Data Independence

A major objective for the three-level architecture is to provide data independence, which means that upper levels are unaffected by changes to lower levels. Data independence makes each level immune to changes to lower levels. There are two kinds of data independence: logical and physical.

2.4.1 Logical Data Independence

This refers to the immunity of the external schemas to changes in the conceptual schema. Changes in terms of the global view of data do not affect applications in a non-productive way. For example, changes to the conceptual schema, such as the addition or removal of new entities, attributes, or relationships, should be possible without having to change existing external schemas or having to rewrite application programs. Clearly, the users for whom the changes have been made need to be aware of them, but what is important is that other users should not be.

2.4.2 Physical Data Independence

Physical data independence refers to the immunity of the conceptual schema to changes in the internal schema. Changes in the physical arrangement of data do not affect applications (except possibly in terms of speed). For example, changes to the internal schema, such as using different file organization or storage structures, using different storage devices, modifying indexes or hashing algorithms, should be possible without having to change the conceptual or external schemas. From the users' point of view, the only effect that may be noticed is a change in performance. In fact, deterioration in performance is the most common reason for internal schema changes. Figure 2.3 illustrates where each type of data independence occurs in relation to the three-level architecture.

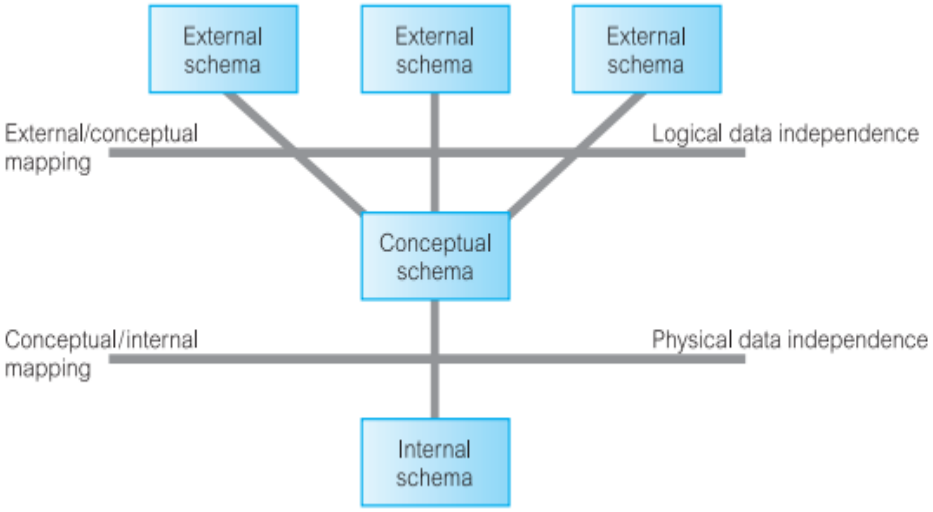


Figure 2.3 Data independence and the ANSI-SPARC three-level architecture

2.5 Database Languages

A data sublanguage consists of two parts: a Data Definition Language (DDL) and a Data Manipulation Language (DML). The DDL is used to specify the database schema and the DML is used to both read and update the database. The part of a DML that involves data retrieval is called a query language. These languages are called *data sublanguages* because they do not include constructs for all computing needs, such as conditional or iterative statements, which are provided by the high-level programming languages such as COBOL, C, C++, C#, Java, etc.

2.5.1 The Data Definition Language (DDL)

Data definition language (DDL) is a language that is used to create and modify the structure of database objects (such as tables, indices, and users) in a database. It is a language for describing data and its relationships in a database. It allows the DBA or user to describe and name the entities, attributes, and relationships required for the application, together with any associated integrity and security constraints. The DDL is used to define a schema or to modify an existing one. It cannot be used to manipulate data.

The result of the compilation of the DDL statements is a set of tables stored in special files collectively called the system catalog. Common examples of DDL statements include CREATE, ALTER, and DROP.

2.5.2 The Data Manipulation Language (DML)

Data manipulation language is a language which enables users to access and manipulate data in a database. The DML is used to both read and update the database. The part of a DML that involves data retrieval is called a query language. The goal is to provide efficient human interaction with the system. Data manipulation operations usually include the following:

- insertion of new data into the database
- modification of data stored in the database
- retrieval of data contained in the database
- deletion of data from the database

Chapter 3 Data Models: The Relational Model

A model is a representation of real-world objects and events, and their associations. A data model represents the organization itself. A data model is a collection of concepts that can be used to describe a set of data, the operations to manipulate the data, and a set of integrity constraints for the data. A data model can be thought of as comprising three components:

(1) a **structural part**, consisting of a set of rules according to which databases can be constructed

(2) a **manipulative part**, defining the types of operation that are allowed on the data (this includes the operations that are used for updating or retrieving data from the database and for changing the structure of the database)

(3) a **set of integrity constraints**, which ensures that the data is accurate

3.1 Data Modelling Approach

There are many data models that can be used but being in line with the course content, we are going to discuss only the relational data model. The relational data model is based on the concept of mathematical relations. In the relational model, data and relationships are represented as tables, each of which has a number of columns with a unique name.

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Figure 3.1 A sample instance of a relational schema

3.1.1 The Relational Database Management System (RDBMS)

A relational database is a type of database that stores data in tables made up of rows and columns. Each table represents a specific object in the database like users, products, orders, etc. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. The relational database management system is a software system that is used to maintain relational databases.

3.2 Terminology

In this section we explain the terminology and structural concepts of the relational model and the RDBMS.

3.2.1 Relational Data Structure

Relation: a relation is a table with rows and columns. In the relational model, relations are used to hold information about the objects to be represented in the database. A relation is represented as a two-dimensional table in which the rows of the table correspond to individual records and the table columns correspond to attributes.

Attribute: an attribute is a named column of a relation. Attributes can appear in any order and the relation will still be the same relation, and therefore will convey the same meaning. A column header of a database is an attribute.

Tuple: a tuple is a row of a relation. Tuples can appear in any order and the relation will still be the same relation, and therefore convey the same meaning.

Degree: the degree of a relation is the number of attributes it contains.

Cardinality: the cardinality of a relation is the number of tuples it contains.

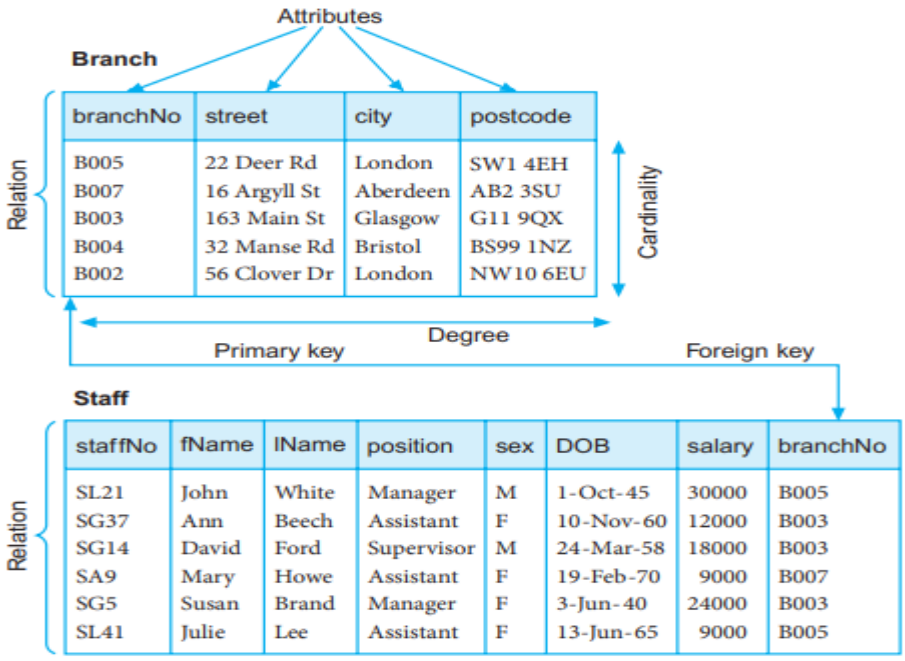


Figure 3.2 Instances of the Branch and Staff Relations.

Domain: a domain is the set of allowable values for one or more attributes. A domain defines the permitted range of values for an attribute of an entity. A simple definition of a database domain is the data type used by a column in a database. This data type can be a built-in type (such as an integer or a string) or a custom type that defines constraints on the data.

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Figure 3.3 Domains for some attributes of the branch and staff relations

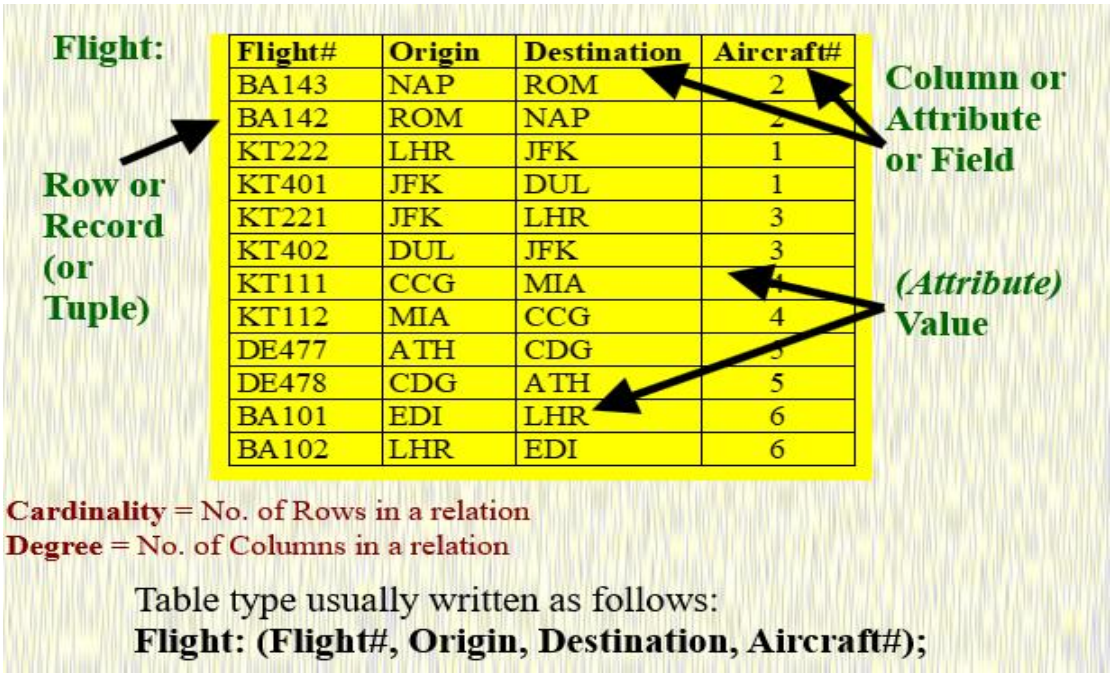
Alternative Terminology

The terminology for the relational model can be quite confusing. We have introduced two sets of terms. In fact, a third set of terms is sometimes used. Table 4.1 summarizes the different terms for the relational model.

Table 3.1 Instances of the Branch and Staff Relations.

FORMAL TERMS	ALTERNATIVE 1	ALTERNATIVE 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Here is another illustration of the terminologies:



Flight#	Origin	Destination	Aircraft#
BA143	NAP	ROM	2
BA142	ROM	NAP	2
KT222	LHR	JFK	1
KT401	JFK	DUL	1
KT221	JFK	LHR	3
KT402	DUL	JFK	3
KT111	CCG	MIA	4
KT112	MIA	CCG	4
DE477	ATH	CDG	5
DE478	CDG	ATH	5
BA101	EDI	LHR	6
BA102	LHR	EDI	6

Figure 3.3 Alternative terminology for relational model.

3.3 Relational Keys

Keys in DMMS is an attribute or set of attributes that help to uniquely identify a row (tuple) in a relation (table). Keys are also used to establish relationships between different tables and columns of a relational database. There are mainly eight different types of Keys in DBMS, and each key has its different functionality:

1. Primary Key
2. Foreign Key
3. Candidate Key
4. Alternate Key
5. Super Key
6. Composite Key
7. Compound Key
8. Surrogate Key

Let's explain some of the keys below:

3.3.1 Primary Key

Primary key is a column or group of columns in a table that uniquely identify a row in that table. Primary keys must contain UNIQUE values and cannot contain NULL values. A table can have only one primary key.

3.3.2 Foreign Key

Foreign key is a column that creates a relationship between two tables. It is a field (or collection of fields) in one table that refers to the primary key in another table. The purpose

of a foreign key is to maintain data integrity and allow navigation between two different instances of an entity. Foreign keys can contain NULL values.

3.3.3 Super Key

A super key is a group of single or multiple keys which identifies rows in a table. The word super denotes the superiority of a key. Thus, a super key is the superset of a key known as candidate key. All super keys can't be candidate keys, but the reverse is true.

3.3.4 Candidate Key

A candidate key is a set of attributes that uniquely identify tuples in a table. It is a key that may become a primary key. A table can have multiple candidate keys but only a single primary key. A candidate key is a minimal super key.