CHAPTER 4
INTRODUCTION TO DATABASES

4.1 INTRODUCTION

A database is an organized repository of data. Nowadays the databases act as communication channels for applications that share data between themselves. A Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS provides utilities for defining, creating, and manipulating databases. The database not only contains the data but also a complete definition or description of the database. This definition is stored in the system data dictionary. The information stored in the data dictionary is sometimes referred to as meta-data. Since the database contains meta-data DBMS access programs can be written without the knowledge of how the data is stored.

There are several terms that need to be defined. These are common in the database literature, and would come up in our discussions frequently:

**Data Model** - A data model is a type of data abstraction that is used to provide the conceptual representation of data. It uses logical concepts, such as entities, their attributes, and their relationships, and hides the storage details. Hence it is a set of concepts that can be used to describe the structure of a database.

**Database Schema** - The description of a database is called the database schema.

**Database instance** - The data in the database at a particular moment in time is called a database instance.

The motivations for using a DBMS are:

**Data Independence**: It is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. This insulates the users of the database from changes in the database. This involves logical data independence as well as physical data independence. Logical data independence is the capacity to change the conceptual schema without having to change the external schemas or the application programs. Physical data independence is the ability to change the internal schema without having to change the conceptual or the external schema.
Data Shareability: The DBMS provides a mechanism for the sharing of data between several applications and users.

Non Redundancy: The DBMS provides a mechanism to ensure non-redundancy of the data stored in the database. [Redundancies can be added sometimes to improve performance].

Relatability: It allows expressing relationships between various data entities.

Integrity: The DBMS can make sure if the data is correct.

Access Flexibility: The DBMS provides higher level query language to ensure easy and flexible access to data.

Security: The DBMS provides facilities to limit access at various levels.

Performance and efficiency: Correct and consistent data is provided efficiently.

Discipline: The DBMS can enforce standards and discipline in the definition and use of data.

Backup and Recovery: Provides backup and recovery facilities for the data.

The DBMS provides a variety of languages and interfaces to support different categories of users. These are:

DDL - called the data definition language. It is used by the database administrator and the database designers to define conceptual and internal schemas.

VDL - called the view definition language. It is used to specify user views and their mappings to the conceptual schema.

DML - called the data manipulation language. It is used to perform manipulations on data. Typical operations are retrieval, insertion, deletion, and update of data.

4.1.1 THREE SCHEMA ARCHITECTURE

The computers do not understand and store data as the users see it in the real world. The computer stores data to allow efficient memory usage and faster access. The data structures and the methods used by the computer may change over time. In order to insulate the users and applications from these changes a conceptual layer is introduced between the application (external) layer and the machine (internal) layer. These levels are described in the ANSI/SPARC proposal for database architecture standardization as:

External level: The user's logical views of the enterprise without consideration for performance or storage issues.
**Conceptual level**: The information model, providing the mapping from the logical to the physical, or internal level, describing the semantics of the entities and relationships, including descriptions of connections and consistency constraints.

**Internal level**: An abstract model of the physical database concerned with the access paths to and the storage of data.

The three schema architecture helps achieve data independence as defined earlier.

### 4.2 INTERNAL LEVEL

The internal level of a database system is the level that is concerned with the way data is actually stored. An overriding performance objective of the DBMS is to minimize the number of disk accesses. Internal level is concerned with the techniques for achieving that objective. An arrangement of data on the disk is called a storage structure. There is no single structure that is optimal for all applications. The process of choosing an appropriate storage representation for a given database is referred to as physical database design. There are several ways of improving performance of the database system. One method of improving performance of the database is by **data clustering**. The basic idea behind clustering is to try and store records that are logically related (and therefore frequently used together) physically close together on the disk. The DBMS can support clustering by storing logically related records on the same page or adjacent pages.

Another way of improving performance is by **indexing**. An index is a special kind of storage file in which each entry consists of two values, a data value and a pointer. The data value is a value for some field of the indexed file, and the pointer identifies a record of that file that has that value for that field. Consider a table -

<table>
<thead>
<tr>
<th>BEAM#</th>
<th>LENGTH</th>
<th>CROSS_SECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>C</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>L</td>
</tr>
</tbody>
</table>

---

¹ This section is adapted from the book by C.J. Date.
If the cross-section is made the index to this table, then it exists as shown below.

![Cross-section diagram]

An index on a primary key field (e.g., index on the field BEAM#) is called a primary index. An index on any other field is called a secondary index. The disadvantage of using index is that it slows down the updates. A common kind of index is the B-tree. Most relational systems support B-trees as their principal form of storage structure. The reason for providing an index is to avoid physical sequential search on the indexed file. However, physical sequential search is still needed in the index, which can be time consuming if the index is large. Solution to the problem is building an index to the index. The idea can be extended to several levels if desired (usually three is the limit).

A B-tree is a particular type of tree-structured index. Various variations of B-trees exist. In Knuth's variation, the index consists of two parts, the sequence set and the index set. The sequence set consists of a single level index to the actual data. The entries in the index are grouped into pages, and the pages are chained together, such that the logical ordering represented by the index is obtained by taking the entries in physical order in the first page on the chain, followed by the entries in physical order in the second page on the chain, and so on. Thus the sequence set provides fast sequential access to the indexed data.

The index set, in turn, provides fast direct access to the sequence set (and thus the data too). The index set is actually a tree-structured index to the sequence set. The combination of index set and sequence set is sometimes called a B⁺ tree. The top level of the index set consists of a single node. The top node is called the root. An example is shown in figure 4.1. The values 3, 5, 9, ..., 99 are values of the indexed field. The algorithms for searching B-trees can be found in [Date].
Another technique for providing fast access is hashing. **Hashing** (or hash addressing) provides fast direct access to a specific stored record on the basis of a given value for some field. The field is usually the primary key. It works as follows:

Each stored record is placed in the database at a location whose address is computed as some function (the hash function) or some field of that record (the hash field). The computed address is called the hash address.

To store the record initially, the DBMS computes the hash address for the new record and instructs the file manager to place the record at that position.

To retrieve the record subsequently given the hash field value, the DBMS performs the same computation as before and fetches the record at the computed position.

Such hashing can cause address collision problems which are solved by using extended hashing, discussion on which can be found in [Date]. We now go on to describe different types of DBMSs.

### 4.3 HIERARCHICAL DATABASE SYSTEMS

A hierarchical database consists of an ordered set of multiple occurrences of a single type of tree. A tree type consists of a single “root” record type, together with an ordered set of zero or more dependent (lower level) subtree types. A subtree type in turn also consists of a single record type - the root of the subtree type - together with an ordered set of zero or more lower level dependent subtree types, and so on. The entire tree type thus consists of a hierarchic arrangement of record types. One of the first database systems commercially available was IMS from IBM. Consider an example -
Here the COURSE is the root record type and has two subtree types, rooted in PREREQ and OFFERING record types. COURSE is said to be the parent record type for record types PREREQ and OFFERING who are its child record types. The connection between a child and its corresponding parent is called a link. Here the connection between OFFERINGS and COURSEs is represented by the COURSE-OFFERING link instead of having a COURSE# field in the OFFERING record.

For each occurrence of the tree, there is one root record occurrence, together with an ordered set of zero or more occurrences of each of the subtree types immediately dependent on the root record type. All occurrences of a given child type that share a common parent occurrence are called twins. The hierarchical database can be manipulated using a set of operators, some of which are:

- Operator to locate a specific tree in the database.
- Operator to move from one such tree to the next.
- Operator to move from record to record within such a tree by moving up and down the various hierarchic paths.
- Operator to move from record to record in accordance with the hierarchic sequence of the database.
- Operator to insert a new record at a specific position within such a tree.
- Operator to delete a specific record.

The hierarchic structure has a built-in bias, it is good for some applications and bad for others. So a logical restructuring mechanism is required so that the data can be logically rearranged into whatever hierarchic form suitable for the application. This however is not sufficiently flexible because the number of possible hierarchies increases combinatorially (2 records - 2 hierarchies, 3 r - 12 h, 4 r - 88 h, ...) , there are applications for which no hierarchy is directly suitable, restructuring is cumbersome, etc. The hierarchic structure can handle one-many relationships, and many-many relationships using two hierarchies, but cannot represent ternary relationships. The insert/delete/replace operations are fairly complex. More details can be found in [Date].

4.4 NETWORK DATABASE SYSTEMS

The network data structure can be seen as an extended form of the hierarchic data structure. While in the hierarchic structure a child has exactly one parent, in the network
structure a child can have any number of parents. Example of a network DBMS is IDMS from Cullinet Software and was based on the proposals of the Data Base Task Group (DBTG) of the programming language committee.

A network database consists of two sets, a set of **records** and a set of **links**. Each link link type involves two record types, a parent record type and a child record type. Each occurrence of a given link type consists of a single occurrence of the parent, together with an ordered set of multiple occurrences of the child record type. Consider an example-

The database contains three record types: **S**, **P**, and **SP**. There are two link types **S-SP** and **P-SP**. _S_ represents the supplier, _P_ represents parts data.

Each occurrence of **S-SP** consists of a single occurrence of **S**, together with one occurrence of **SP** for each shipment by the supplier represented by that **S** occurrence. Each occurrence of **P-SP** consists of a single occurrence of **P**, together with one occurrence of **SP** for each shipment of the part represented by that **P** occurrence.

Each occurrence of a link represents a one-many relationship between the parent occurrence and the corresponding child occurrences.

A network data manipulation language consists of a set of operators for processing data, some of which are:

- Operator to locate a specific record, given a value for some field in that record.
- Operator to move from a parent to its first child in some link.
- Operator to move from one child to the next in some link.
- Operator move from a child to its parent within some link.
- Operator to create a new record.
- Operator to destroy an existing record.
- Operator to update an existing record.
- Operator to connect an existing child into a link.
- Operator to disconnect an existing child from a link.
These systems provide good performance but the optimization has to be done manually. The network structure and its operators are complex. It tends to fragment the information. This increase in complexity however does not offer additional functionality over relational systems which are discussed next. More details can be found in [Date].

4.5 RELATIONAL DATABASE SYSTEMS

The hierarchic and network databases are clearly not suitable for engineering applications as we will show in more detail. The relational databases offer much more flexibility than the previous two. Due to their simplicity and flexibility the relational systems have been highly popular. A relational system is one in which:

*The data is perceived by the user as tables; and
the operators at the user's disposal are operators that generate new tables from old.*

We will discuss relational databases in more detail in the following sections. Example of a relational DBMS is ORACLE from ORACLE Corporation.

NOTE: SEE APPENDIX ‘C’ FOR A COMPLETE DESCRIPTION OF THE RELATIONAL DATABASE MODEL
FIG. 4.1  Example of a B-tree