Chapter 7
Designing Physical Files and Databases

Contents

7. DESIGNING PHYSICAL FILES AND DATABASES

a. Physical File and Database Design
b. Designing Fields
c. Designing Physical Records
d. Designing Physical Files
e. Designing databases

Physical database design

- Describes the data in terms of collection of files, indices and other storage structure such as records ,formats ,record ordering and access paths.
- Specifies how the database will be executed in a particular DBMS such as oracle, Sybase ,etc. by taking into account the facilities and constraints of a given DBMS(Database Management System).
- After designing a logical database ,now its time to design an actual physical database.

- Followings are the things that need to be considered while designing the physical database :
  - Normalized relations obtained from the logical database including volume estimates.
  - Definitions of each attribute .
  - Descriptions of where and when data are used: entered, retrieved, deleted, and updated.
  - Expectations or requirements for response time and data integrity
  - Descriptions of the technologies used for implementing the files and database so that the range of required decisions and choices for each is known.

- Bottom-up approach is taken for reviewing physical file and database design. Thus, analysts begin the physical design phase by addressing the design of physical fields for each attribute in a logical data model.

- There are two basic steps followed during designing physical database:
  - Designing fields
  - Designing tables

Designing Fields

- A field is the smallest unit of application data recognized by system software, such as a programming language or database management system.
- An attribute from a logical database model may be represented by several fields.
- Each field requires a separate definition when the application system is implemented.
- For example, a student_name attribute in a normalized student relation might be represented as three fields: last name, first name, and middle name.
- While designing fields, TWO things need to be considered:
  A. Choosing right data types
  B. Controlling data integrity.

A. Choosing Data Type

- A data type is a coding scheme recognized by system software for representing organizational data.
- Selecting a data type balances four objectives that will vary in degree of importance for different applications:
  1. Minimize storage space
  2. Represent all possible values of the field
  3. Improve data integrity for the field
  4. Support all data manipulations desired on the field
- E.g. what kind of data type is best suited to represent "Name" field?[Ans: char type]

- Inappropriate choices of data type may result in need for proper maintenance and difficulty in extension .
  - **Calculated fields** also called derived or computed fields ; whose values can be obtained from other database fields.
    - Example: Amount field can be calculated multiplying Rate and Quantity Field.
  - **Coding and Compression Techniques** helps to limit possible range of values for the given field .
    - Example “Month ” field can have maximum of 12 values like JAN , FEB,……..,DEC.
  - The data type must be suitable for the life of the application; otherwise, maintenance will be required. E.g. use of int values for population of world.
  - Choose data types for future needs by anticipating growth. E.g. static array types
  - Make sure that date arithmetic can be done so that dates can be subtracted, or time periods can be added to or subtracted from a date.

B. Controlling Data Integrity

- Controlling data integrity means maintaining uniformity of data throughout the data system.
  - E.g. we can write Kathmandu as KTM, Kath, or Kth. But if all these short forms are used, confusions and conflicts may arise.
- There are FOUR data integrity control methods. They are:

1. Default value [unless an explicit value is entered for the field, field will assume a pre-determined value]
2. Range Control [allowing a limited set of permissible values for numeric or alphabetic data]
3. Referential Integrity [if a value of one attribute of a relation references a value of another attribute, then the referenced value must exist.]
4. Null Value Control [null values usage must be limited as much as possible]

Designing physical tables

- A physical table is a named set of rows and columns that specifies the fields in each row of the table.
- A physical table may or may not correspond to one relation.
- Whereas normalized relations possess properties of well-structured relations, the design of a physical table has two goals different from those of normalization: efficient use of secondary storage(disk space) and efficient data processing speed.
  - Disk Space is used most efficiently when the physical length of a table row divides close to evenly with storage unit.
  - Data processing is most efficient when stored next to each other in secondary memory.

De-normalization

- The opposite of Normalization is De-normalization.
- The undoing of relations into forms like 1 NF,2 NF ,3 NF ,etc. is said to be De-normalization.
- De-normalization is the process of splitting or combining normalized relations into physical tables based on how commonly rows and columns are used.
- The goal of normalization is to manage tables; the goal of de-normalization is to create tables that contain only the data used together in programs.
- By placing data used together close to one another on disk, the number of disk I/O operations to retrieve all the data needed by a program is minimized.
- The result of de-normalization is the definition of one or more physical files, which in OS sense, is a named set of table rows stored in a contiguous section of secondary memory.

When is de-normalization required?

1. When two entities exist with one-to-one relationship.
   - Two entities can have co-existence within a table.
   - If required, one of the entities can have a null value attribute.
   - E.g.

     STUDENT(Student_ID, Campus_Address, Application_ID)

     APPLICATION(Application_ID, Application_Date, Qualifications, Student_ID)

   - Based on 1-1 relationship, the two relations can be de-normalized into one as

     STUDENT(Student_ID, Campus_Address, ,Application_Date, Qualifications)

   - Here “application_date” and “quaifications” can have null values if required, as one student may or may not apply for a particular process.

2. When many-to-many relationship exist with non-key attributes.

- If associative entities exist among relations having many-to-many relationship, then a relation can be combined with the associative entity to avoid combining of all 3 relations.
- E.g.

  VENDOR (Vendor_ID, Address, Contact_Details)

  ITEM (Item_ID, Description)

  PRICE_QUOTE (Vendor_ID, Item_ID, Descriptions, Price)

- Here, the two relations ITEMS & PRICE_QUOTE can be de-normalized into one as

  VENDOR (Vendor_ID, Address, Contact_Details)

  ITEM_QUOTE (Vendor_ID, Item_ID, Description)

3. When more than one reference of an entity corresponds to exact one reference of another entity (two entities exist with many-to-one relationship).

- In this case, one relation containing both the entities can be designed.
- E.g.

  ITEM (Item_ID, Description, Instruction_ID)

  STORAGE (Instruction_ID, Store, Container_Type)

- Based on Many-to-One relationship, the two relations can be de-normalized into one as

  ITEM (Item_ID, Description, Store, Container_Type)

Methods of De-Normalization

- Partitioning relations
- Adding redundant columns
- Adding derived columns
- Combining tables
- Repeating groups
- Creating extract tables

Partitioning relations

- Example

  Product(ProdID , Description , Color , Price ,Quantity, Prod_Manager)

- **Denormalization by columns**

  MarketingProduct(ProdID, Description , Color , Price , Prod_Manager)

  EngineeringProduct(ProdID , Description , DrawingNo , Weight , Color)

  AccountingProduct(ProdID , Unit_Cost)

Example

Customer

CustID	Name	Region	Salary
112	Ram	South	40000
113	Hari	North	50000
114	Gopal	South	60000

De-normalize by rows

SouthCustomer

CustID	Name	Salary
112	Ram	40000
114	Gopal	60000

NorthCustomer

CustID	Name	Salary
113	Hari	50000

Arrangement of physical files: File Organization

- A physical file contains rows and columns from one or more tables, as produced from de-normalization.
- Each table may be one file or the whole database may be in one file, depending on how the database technology and database designer organize data.
  - This also varies on the operating system used such as Linux, Windows
- The way the operating system arranges table rows in a file is called a file organization.
- With some database technologies, the systems designer can choose from among several organizations for a file.

Importance of File Organization

- Fast data processing and retrieval
- Efficient use of storage
- Avoid Data Loss
- Minimal Maintenance
- Extension
- Security from unauthorized people

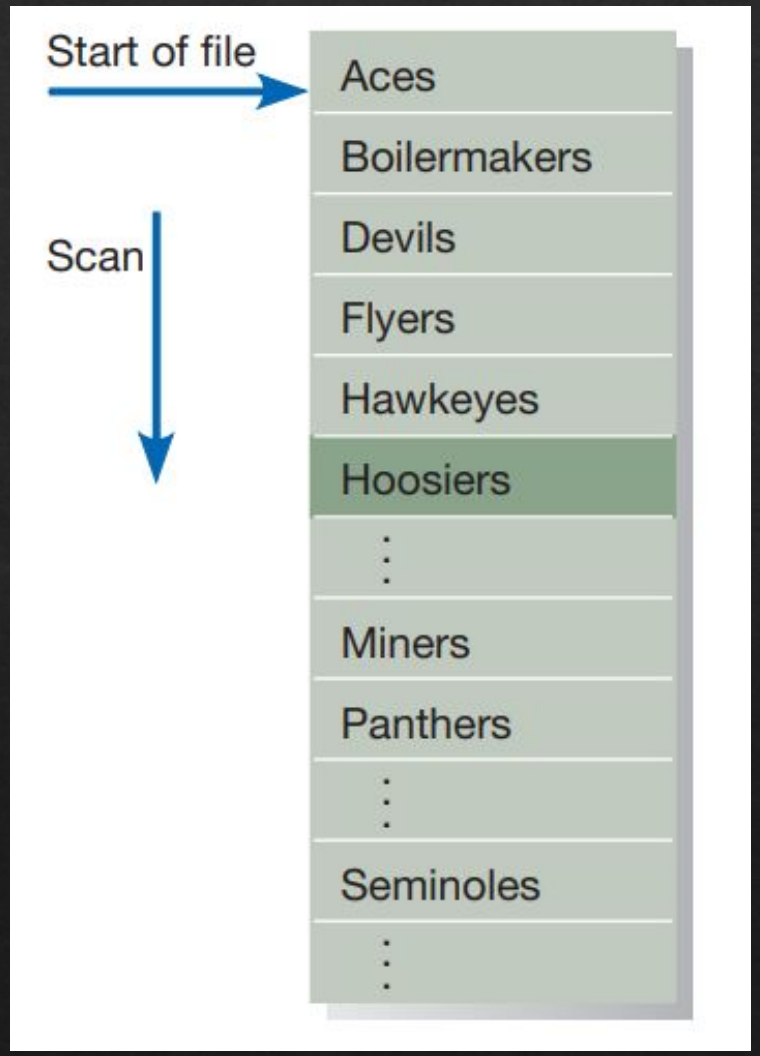
Types of File Organizations

1. Sequential File Organization
2. Indexed File Organization
3. Hashed File Organization

1. Sequential File Organization

- In a sequential file organization, the rows in the file are stored in sequence according to a primary key value.
- To locate a particular row, a program must normally scan the file from the beginning until the desired row is located.
- Sequential files are very fast if you want to process rows sequentially, but they are impractical for random row retrievals.
- Adding rows requires rewriting the file, at least from the point of insertion.
- Deleting rows can cause wasted space or the need to compress the file.

A classic example of a sequential file is the alphabetic list of persons in the white pages of a phone directory.



- **Pros and Cons of Sequential File Organization –**
  Pros –
  - Fast and efficient method for huge amount of data.
  - Simple design.
  - Files can be easily stored in magnetic tapes i.e cheaper storage mechanism.
- **Cons –**
  - Time wastage as we cannot jump on a particular record that is required, but we have to move in a sequential manner which takes our time.
  - Sorted file method is inefficient as it takes time and space for sorting records.

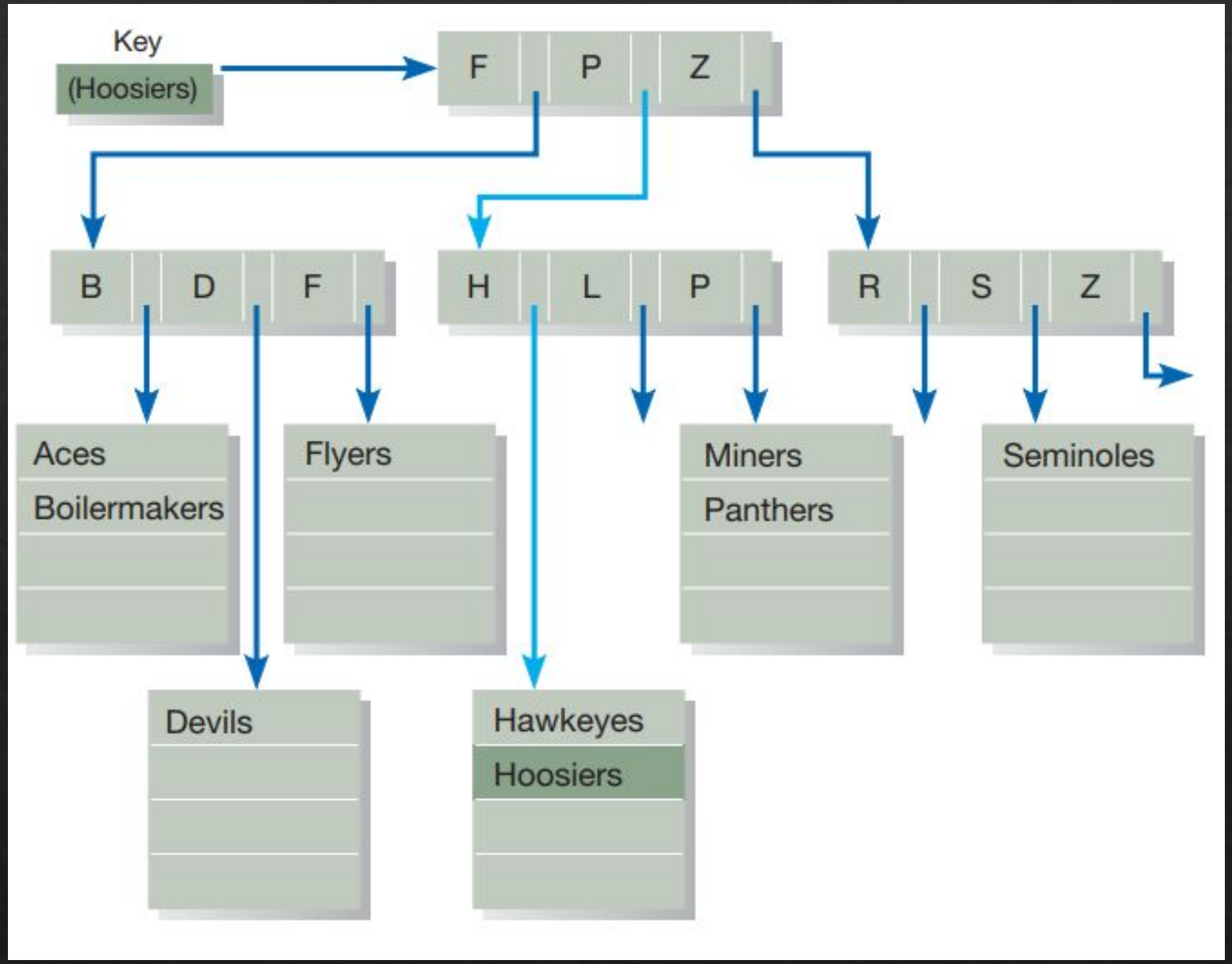
2. Indexed File Organization

- In an indexed file organization, the rows are stored either sequentially or non-sequentially, and an index is created that allows the application software to locate individual rows.

- An index can point to unique rows (a primary key index, such as on the Product_ID field of a product table) or to potentially more than one row.
  - An index that allows each entry to point to more than one record is called a secondary key index.

- One of the most powerful capabilities of indexed file organizations is the ability to create multiple indexes, like the title, author, and subject indexes in a library.
- Search results from the multiple indexes can be combined very quickly to find those records with precisely the combination of values sought.

A classic example of an indexed file is the card catalog in a library



Index	Data Position
A	1
B	2
C	3
D	4

Fig: An Index Table

Data Position	Data
1	Alex
2	Andrew
3	Bob
4	Catherine

Fig: A Data Table

- Types of index:
  - Primary index
    - Single index is used to point unique record.
    - Data file is ordered on a **key field.**
  - Secondary index
    - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
    - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

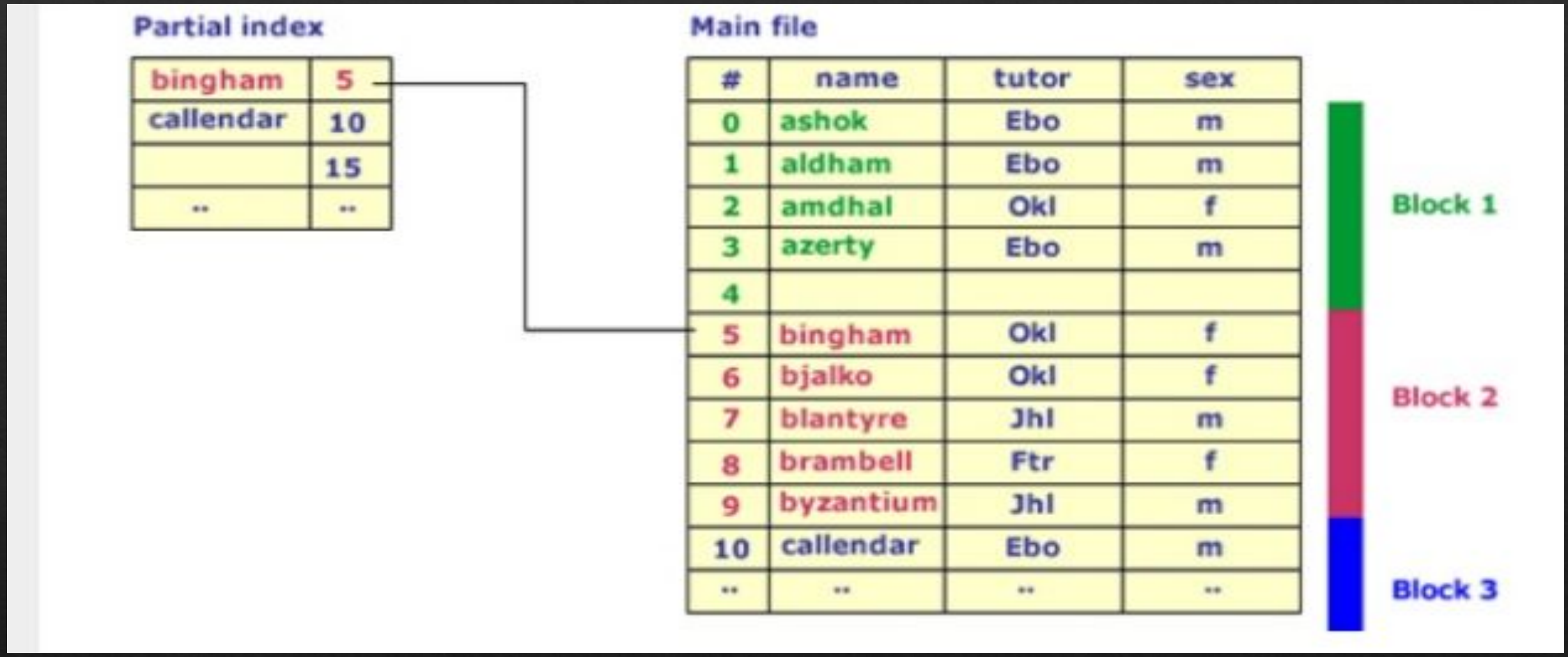


Fig: Primary Indexed Sequential File Organization

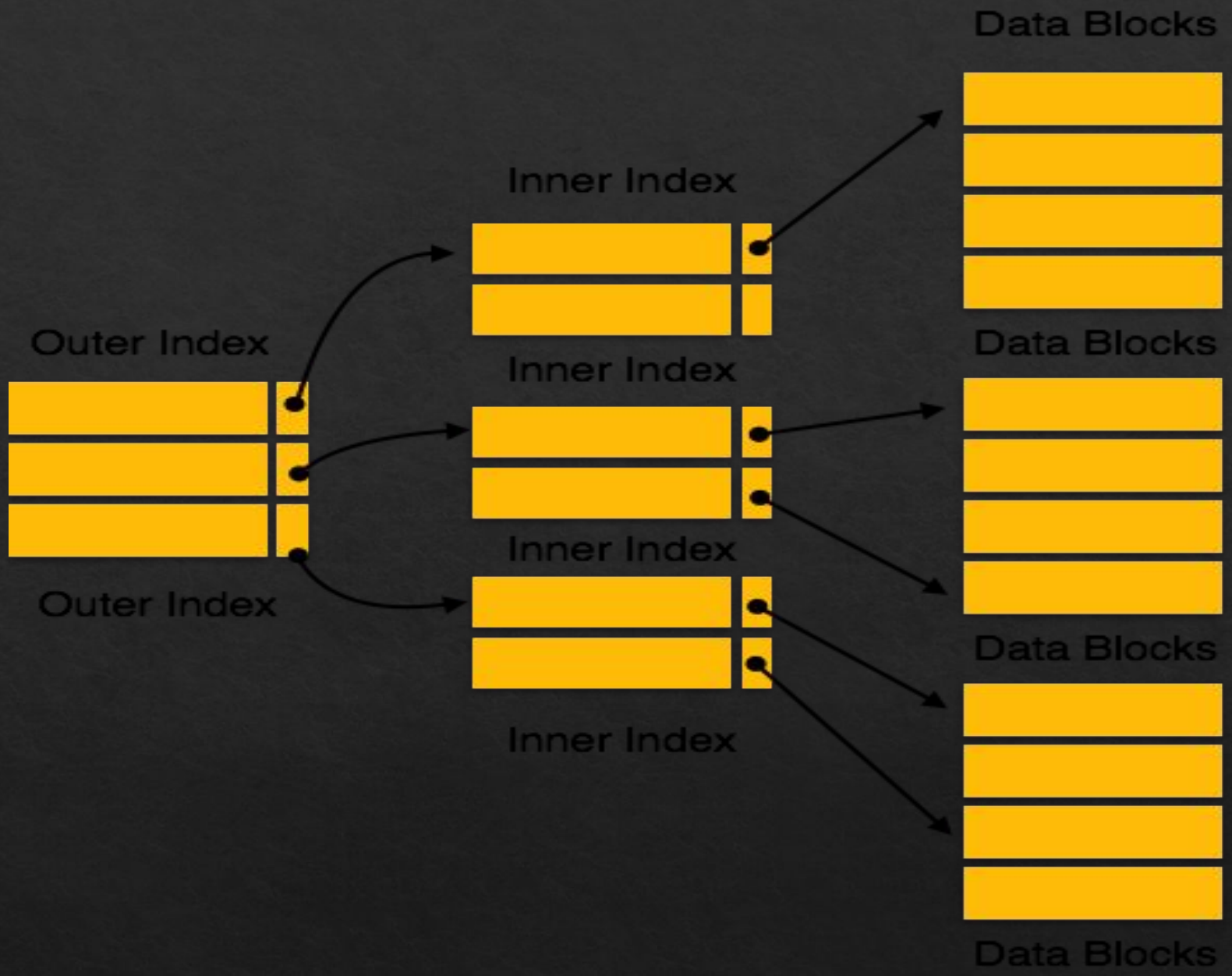


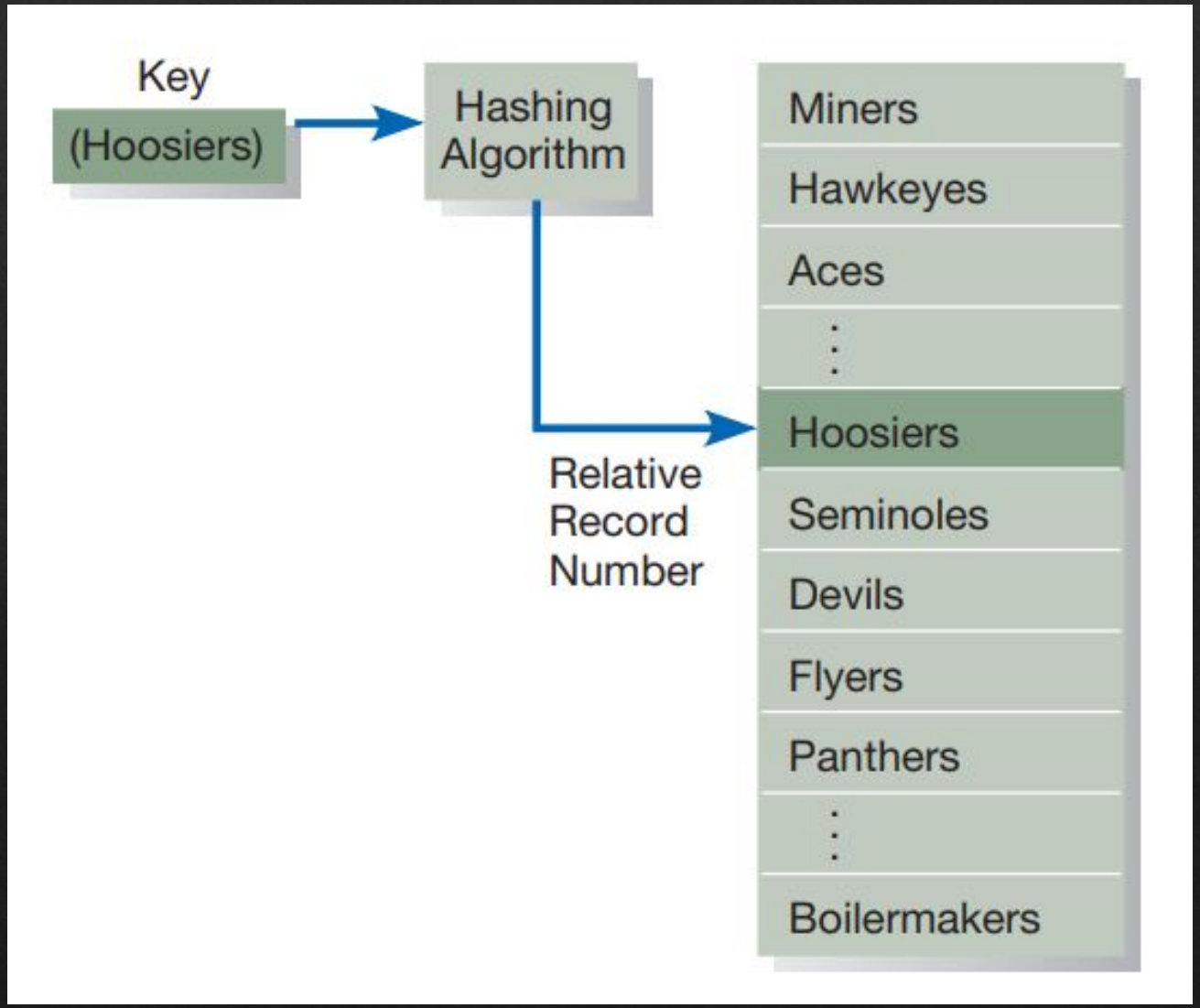
Fig: Secondary Indexed Sequential File Organization

3. Hashed File Organization

- In a hashed file organization, the location of each row is determined using an algorithm called the hashing algorithm that converts a primary key value into a row address.

- Although there are several variations of hashed files, in most cases the rows are located non-sequentially as dictated by the hashing algorithm.
  - Thus, sequential data processing is impractical.
  - On the other hand, retrieval of random rows is very fast.

- There are some issues in the design of hashing file organizations, such as how to handle two primary keys that translate into the same address.


Key
(Hoosiers)
Hashing Algorithm
Relative Record Number
Miners
Hawkeyes
Aces
Hoosiers
Seminoles
Devils
Flyers
Panthers
Boilermakers

Example:

Key Values(K)
1
3
8

H(K) = K

i.e. H(8)=8

Hash Table

	0
	1
	2
	3
	4
	5
	6
	7
	8
	9

Advantages

- Records need to be sorted after any transaction. Hence effort of sorting is reduced in this method.
- Since block address is known by Hash Function ,accessing any record is very fast.
- Similarly updating or deleting a record is also very quick.

Disadvantages

- Proper Handing Functions should be used in order to remove collision or there might be data loss.
- Since all data are stored randomly. They are scattered in memory .Hence memory is not efficiently used.
- Highly expensive one since complex programs need to be written to make this method.

Advantages of Hash File Organization:

- Records need not be sorted after any of the transaction. Hence the effort of sorting is reduced in this method.
- Since block address is known by hash function, accessing any record is very faster. Similarly updating or deleting a record is also very quick.
- This method can handle multiple transactions as each record is independent of other. i.e.; since there is no dependency on storage location for each record**, multiple records can be accessed at the same time.**
- It is suitable for online transaction systems like online banking, ticket booking system etc.

Disadvantages of Hash File Organization:

- This method may accidentally delete the data.
- Since all the records are randomly stored, they are scattered in the memory. Hence memory is not efficiently used.
- If these hash columns are frequently updated, then the data block address is also changed accordingly. Each update will generate new address. This is also not acceptable.

Comparative Features of Sequential, Indexed, and Hashed File Organizations

Factor	File Organization		
	Sequential	Indexed	Hashed
Storage space	No wasted space	No wasted space for data, but extra space for index	Extra space may be needed to allow for addition and deletion of records
Sequential retrieval on primary key	Very fast	Moderately fast	Impractical
Random retrieval on primary key	Impractical	Moderately fast	Very fast
Multiple key retrieval	Possible, but requires scanning whole file	Very fast with multiple indexes	Not possible
Deleting rows	Can create wasted space or require reorganizing	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy
Adding rows	Requires rewriting file	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy, except multiple keys with same address require extra work
Updating rows	Usually requires rewriting file	Easy, but requires maintenance of indexes	Very easy

Extra Topics

Data models / Database models

Some types of data model :

- Hierarchical Model
- Network Models
- Relational Models
- Object-Oriented Model

a) Hierarchical data model

- It is a data model in which the data are organized into a tree-like structure.
- The data are stored as **records** which are connected to one another through **links**.
- A record is a collection of fields, with each field containing only one value.
- Each record is having one parent record and many children.
- The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- **Characteristics:**
  - Each parent can have many children.
  - Each child has only one parent.

Example of hierarchical data model

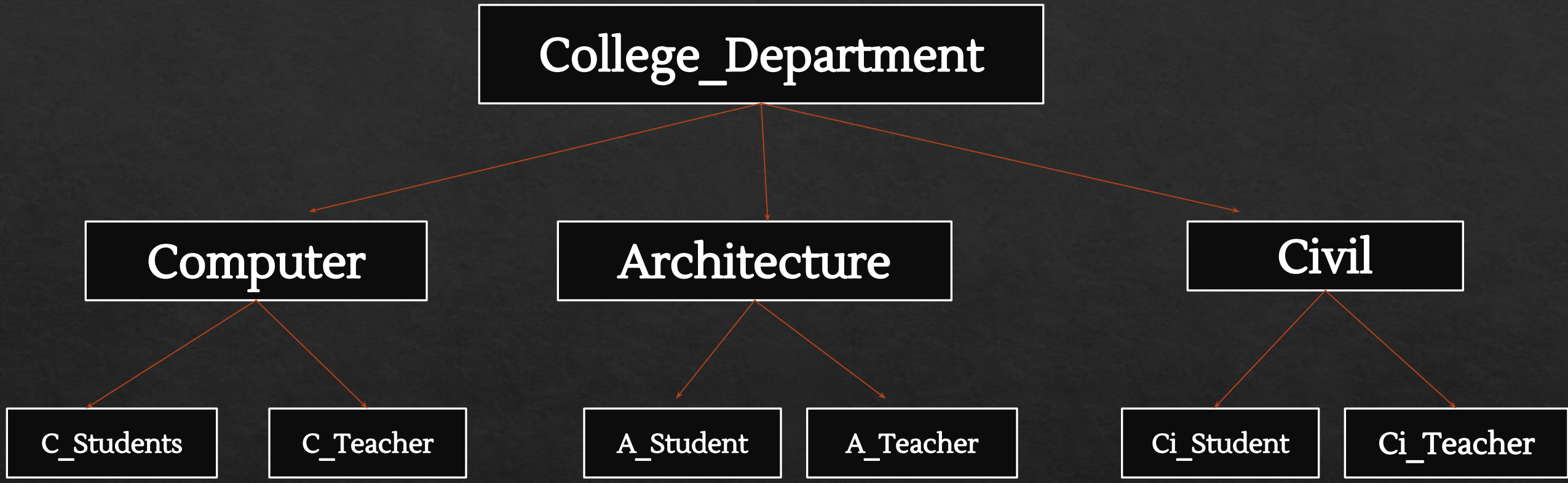


Figure: Hierarchical data model for college

Features of Hierarchical data model

- **One to many relationships**
- **Problem in Deletion**
- **Hierarchy of data**
- **Parent-child relationship**
- **Pointer:** Pointers are used for linking records that tell which is a parent and which child record is.
- **Predefined relationship:** All relations between root, parent and child nodes are predefined in the database schema.
- **Redundancy**

Advantages

- Promotes data sharing.
- It is conceptually simple due to the parent-child relationship.
- Database security is enforced.
- Efficient with 1: N relationships.
- A clear chain of command or authority.

Disadvantages

- Complex relationships are not supported.
- M: N relationship is not supported.
- Lack of standards.
- Poor flexibility
- Communication barriers

b) Network data model

- This is an extension of the Hierarchical model.
- In this model data is organized more like a graph, and are allowed to have more than one parent node.
- This data model is created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard.
- This database model was used to map many-to-many data relationships.
- In this type of model, a child can be linked to multiple parents, a feature that was not supported by the hierarchical data model.

Example of network data model

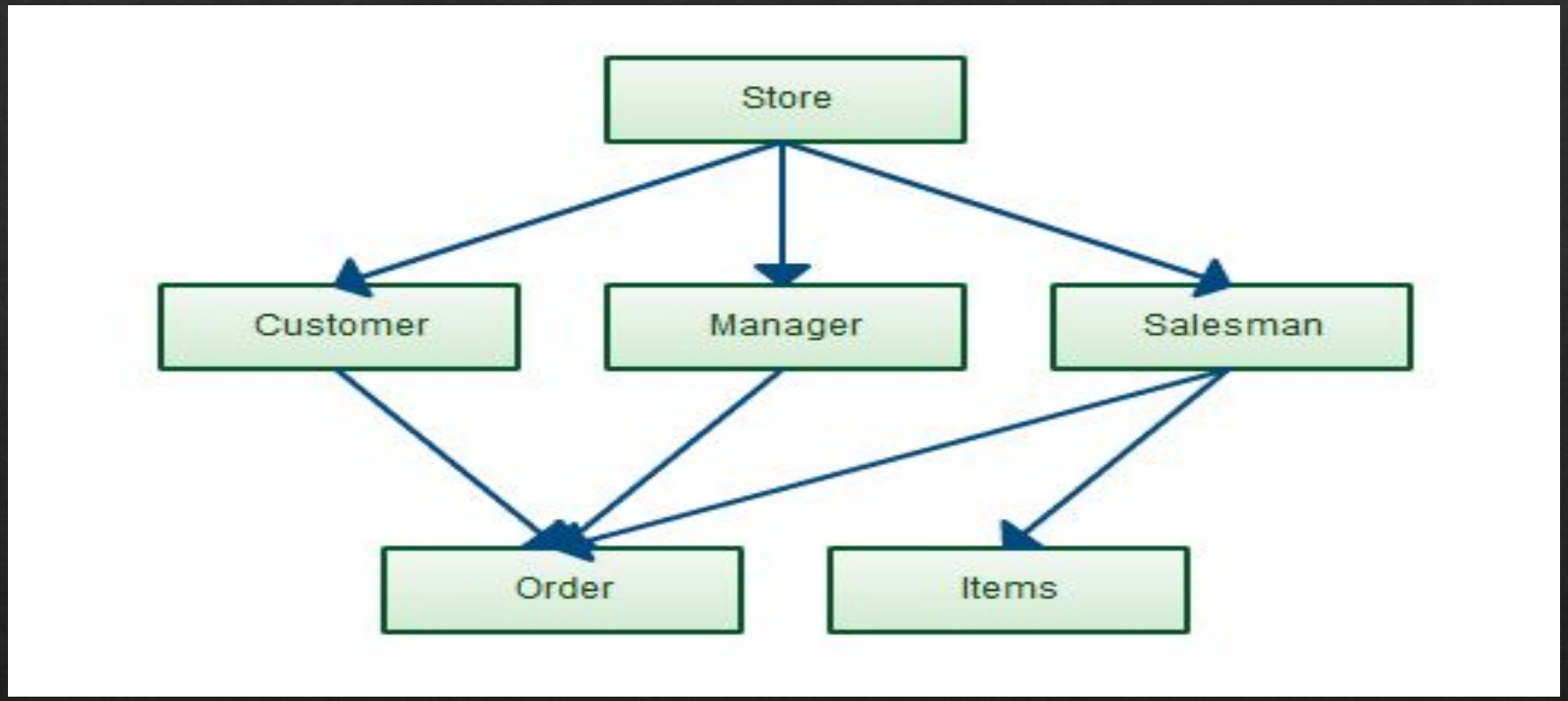


Figure: Network data model for store

- **Network Data Model Advantage**
  - Conceptual simplicity
  - Handles more relationship types
  - Data access flexibility
  - Promotes database integrity

- **Network Date Model Disadvantages**
  - System complexity
  - Detail structural knowledge is required
  - Lack of structural independence

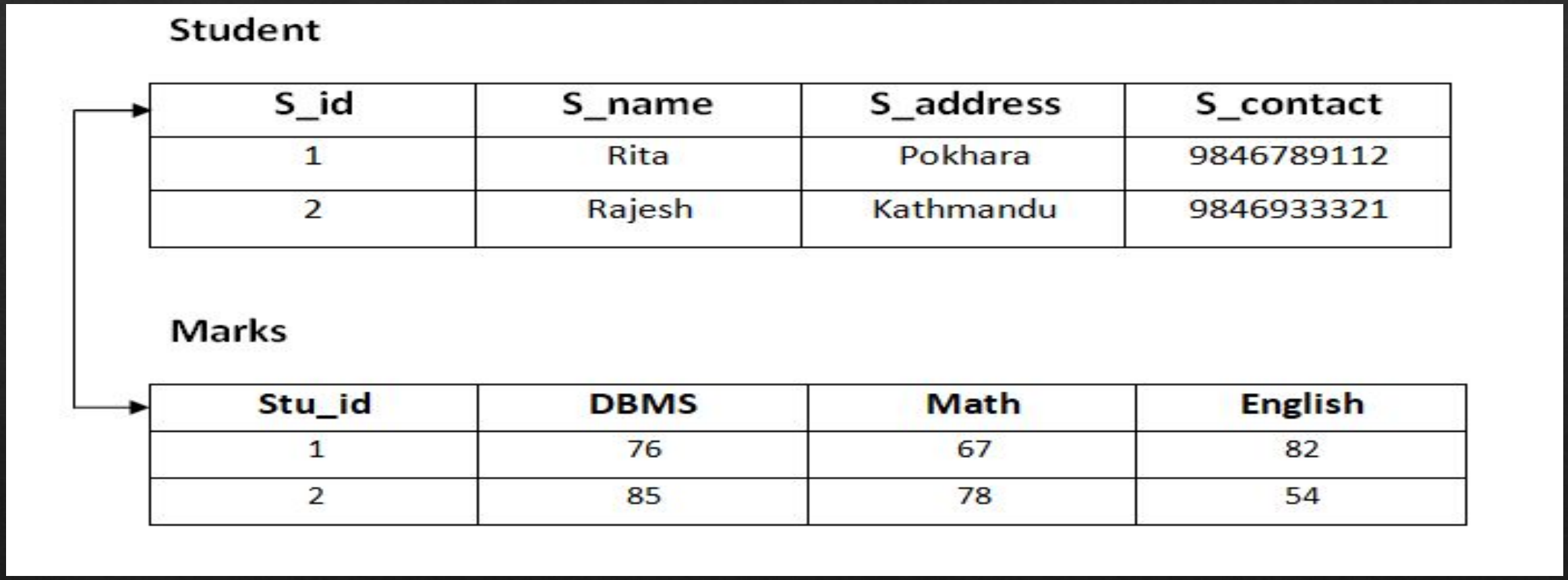
C) Relational data model

- In relational model, the data and relationships are represented by collection of inter-related tables.
- Each table is a group of column and rows, where column represents attribute of an entity and rows represents records.
- It represents the database as a collection of relations.
- It describes the data, relationship between that data, data semantic and constraints on the data in the relational database.
- A relation is table of values.
- Every row in the table represents a collection of related data values of particular person.
- Some terminologies:
  - Relation = table
  - Tuple= row
  - Attribute= column

Some popular Relational Database management systems (RDBMS) are:

- My-SQL
- MS-SQL/ MS-SQL Server
- MS_Access
- Oracle, etc.

Example of relational data model



Student

S_id	S_name	S_address	S_contact
1	Rita	Pokhara	9846789112
2	Rajesh	Kathmandu	9846933321

Marks

Stu_id	DBMS	Math	English
1	76	67	82
2	85	78	54

Figure: Relational model for student

Advantages of Relational model:

- **Simplicity**: A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence**: The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use**: It is easy as tables consisting of rows and columns are quite natural and simple to understand
- **Query capability**: It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence**: The Structure of Relational database can be changed without having to change any application.
- **Scalable**: Database could be enlarged easily to enhance as requirement.
- **Greater flexibility**
- **Easy to design**

Disadvantages of Relational model:

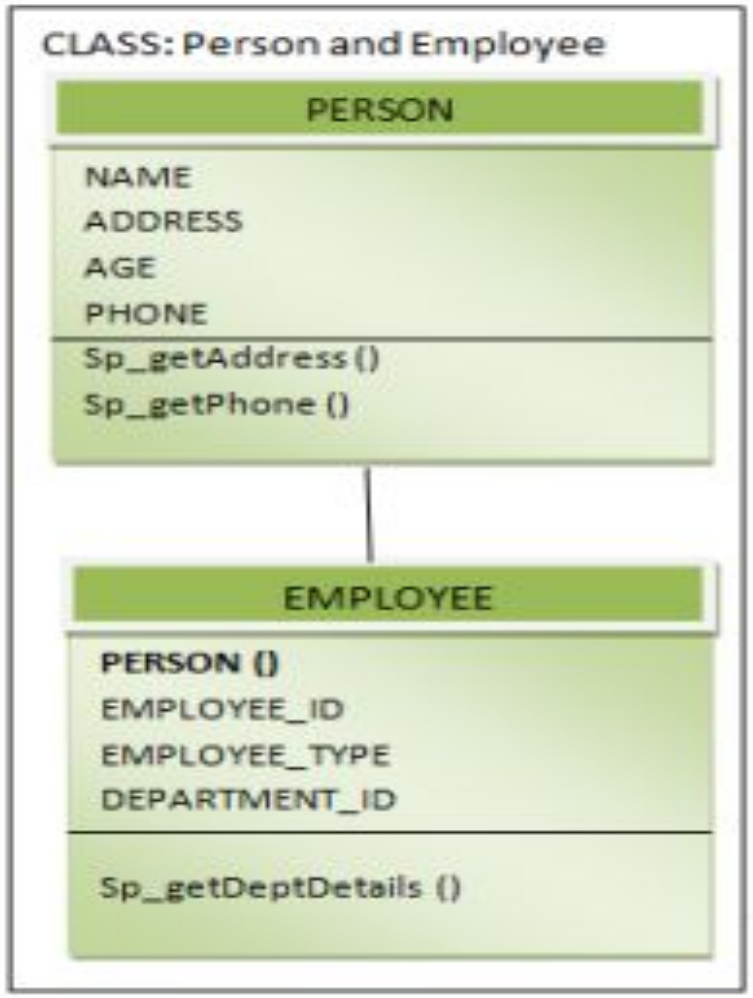
- Data anomalies (insertion, update, delete anomalies)
- Need of professional people having knowledge about relational model
- Need of additional software
- Complexity increases with increase in data in DB

d) Object-based data model

- This data model is represents real world objects.
- Increasingly complex real-world problems demonstrated a need for a data model that more closely represented the real world.
- It uses the concepts of object-oriented design.
- It considers each object in the world as objects and isolates it from each other.
- It groups its related functionalities together and allows inheriting its functionality to other related sub-groups.

Example of object-based data model

Class



Object 1

Object 2

- **Advantages**
  - Because of its inheritance property, we can re-use the attributes and functionalities.
    - It reduces the cost of maintaining the same data multiple times.
  - Due to encapsulated message, there is no fear being class misused by other objects.
  - If we need any new feature we can easily add new class inherited from parent class and adds new features.
    - Hence it reduces the overhead and maintenance costs.
  - Because of the above feature, it becomes **more flexible** in the case of any changes.
  - Codes are re-used because of inheritance.
  - Since each class binds its attributes and its functionality, it is same as representing the real world object. We can see each object as a real entity. Hence it is more **understandable.**
- **Disadvantages**
  - It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.
  - Lack of OODM standards
  - It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.