# Summary of Indexing and Hashing

## Introduction

Indexing and hashing are fundamental techniques in database management systems that improve the efficiency of data retrieval operations. Indexing organizes data in a way that minimizes the time needed to find specific information, while hashing maps data to locations in memory using a hash function, allowing faster access.

## Indexing Basics

Indexing is a data retrieval method that provides faster access to records in a database. It creates an index structure based on a specific search key, which is used to locate data. The two primary types of indexing are ordered indexing and hash indexing.

### Ordered Indexing

Ordered indices are based on the sorted order of search keys. Types include primary and secondary indices. Primary indices are used for sequentially ordered files, typically based on the primary key. Secondary indices, on the other hand, are used when data does not follow a sequential order.

### Types of Index Files

There are two main types of index files:  
- Dense Index Files: Each search key value in the file has an index record.  
- Sparse Index Files: Index records exist only for some search key values.

## Hashing Basics

Hashing is a technique used to map data to locations in memory using a hash function. It is primarily used for fast data retrieval and reduces the need for data sorting. Two main types of hashing techniques are static and dynamic hashing.

### Static Hashing

In static hashing, a hash function maps search key values to a fixed set of buckets. This technique can result in bucket overflow if too many records are mapped to the same bucket.

### Dynamic Hashing

Dynamic hashing allows the hash function to adapt as the database grows or shrinks. An example of dynamic hashing is extendable hashing, where the number of buckets can expand or contract based on the data size.

## B+-Trees and B-Trees

B+-Trees and B-Trees are widely used data structures for indexing. B+-Trees are particularly efficient for range queries, as they store all search keys in a sorted manner and maintain balanced levels. B-Trees, while similar, are less commonly used due to their higher complexity in operations.

## Comparison of Indexing and Hashing

Indexing is generally better suited for range queries, as it maintains an ordered structure. Hashing, on the other hand, is optimal for exact-match queries, providing faster access times. Choosing between the two depends on the type of queries expected in the database.

## Advantages and Limitations

Both indexing and hashing have unique benefits and drawbacks. Indexing improves data retrieval speed, especially for ordered data, but can be resource-intensive in terms of storage and maintenance. Hashing is efficient for random data access but requires careful handling of collisions to prevent bucket overflow.

## Practical Applications

In practice, databases may combine both indexing and hashing to optimize data access. For example, bank databases might use indexing for account numbers (ordered by name or date) and hashing for quick access to transaction records.