Physical DB Design & Tuning

Objectives

- What is Physical DB Design +
- Why Physical DB Design is Needed +
- I/O Model for DBs
- Issues in Physical DB Design +
- File Organization and access Methods +
- Physical DB Design Process +

- What is Physical DB Design?

- Physical DB design is the process of choosing specific storage structures and access paths for the DB files to achieve good performance for the various DB applications.
- Each DBMS offers a variety of options for file organization and access paths.
- Once a specific DBMS is chosen, the physical DB design process is restricted to choosing the most appropriate file organizations and access paths for the DB files from among the options available in the DBMS.

- Why Physical DB Design is Needed? ...

- We know that data in a DB systems is stored on a secondary storage using media like magnetic disk, optical disk, or the like. Magnetic tapes are normally used for keeping the backup of the DBs.
- The data in the DBs is stored using storage structure normally known as “file organization”. Some very common file organizations are:
  - Sequential (unordered or unsorted)
  - Sequential (ordered or sorted)
  - Indexed
  - Hashed

… - Why Physical DB Design is Needed?

- Each file organization has its own very specific characteristics or properties.
- For example, if data is stored in a sequential (unordered) file, we cannot perform binary search on the file. The data access will be sequential and the average retrieval time of a data item in the file will directly depend on the size (i.e., the number of records) of the file.
- If records are added to the file and few are deleted from the file, the retrieval time of a data item from such a file will go on increasing and ultimately the response time of the query using the data item will become worse and worse. If we don’t take care of the problem properly, the DB performance may become unacceptable.

- I/O models for DBs ...

- Important steps in writing data to DB or reading data from DB are:
  1. User submits a data request command to the DBMS
  2. DBMS analyses the command
  3. DBMS generates an operating system command that should be executed to fulfill the data request of the user
  4. As a part of the DB access, the relevant files of the database are opened and blocks of data are transferred from the secondary storage to a main memory storage area called “buffers”
  5. The data from the buffers are transferred to the program work area.
  6. The data from the program work area is transferred to the user if the processing is complete for retrieval or it is transferred to the buffers to continue the processing.
  7. In case of writing the data to the DB, the buffers are written back to the storage.

… - I/O models for DBs

- What is a persistent DB system?
  - Data stored on storage devices such that the existence of the data doesn't depend upon the existence of the program using it and the data can withstand power failures.

- A DB system handles a user query through three phases:

1. Logical processing of the user request
2. Transformation of user request to an access plan
3. Data transfer.

- Issues in Physical DB Design ...

- Criteria for choosing appropriate file organization and access paths are:
  - **Response time**: is the time from the submission of a DB query for execution to receiving a response. The response time of a query mainly depends on the access time for data items referenced in the query.
  - **Space utilization**: is the amount of storage space used by the DB files and their access path structures like index on disk
  - **Transaction throughput**: is the average number of transactions that can be processed per unit time.

… - Issues in Physical DB Design

- In a relational DB, each base relation in the database schema is stored in a physical DB file.
- During the physical DB design many design decisions need to be made.
- For each relation in the DB, it should be decided that:
  - What file organization should be used to store the relation.
  - Whether the file should be indexed or not.
  - Which attributes should be used for indexing
  - Should the index be clustered or not.
  - Should hashing or tree index be used
  - If it is hashing then should it be static or dynamic hashing

- File Organization and access Methods

- File Oraganizations +
- Index Structures for Files +

-- File Oraganizations

- Heap (unordered sequential) +
- Ordered Sequential +
- Hashed File +

-- Heap

- Heap file are file of unordered records. Records are placed in the file in the order in which the were inserted.
- Access: only linear search
- Insertion: new records are inserted at the end of the file.
- Deletion: in heap file deletion can be done in the following two methods:
  - Immediate physical deletion +
  - Delayed physical deletion +

--- Immediate Physical Deletion

1. Record to be deleted is searched
2. The disk block containing the record is transferred to the buffer
3. The record is deleted
4. The records following the deleted record are adjusted to use the freed space
5. Block is written back to disk.

--- Delayed Physical Deletion

- Also known as deletion marker method
- Steps in deleting a record:
  1. The record to be deleted is searched
  2. If found it is then marked as deleted.
  3. After some time during file reorganization the marked records are deleted.

-- Ordered Sequential Files

- In ordered sequential files, records are physically sorted based on the values of one field.
- **Access**: various fast serach algorithms like binary search can be performed
- **Insertion**: a new record is inserted at is proper position according to the value of the ordering field and the other records are adjusted accordingly.
- **Deletion**: whenever a record is deleted from a sequential ordered file the remaining records are adjusted to maintain the order.

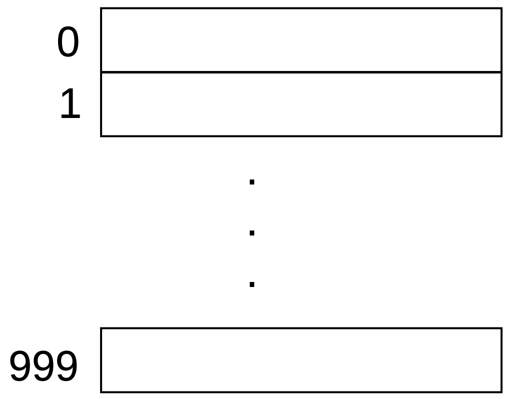
-- Hash File ...

- A hash file is a file in which a hash function uses one key field (also called hash field) value to generate the address of the record in the storage,

  h(hash-field-value) = address

  where *h* is the hash function

- Example: Suppose we have 1000 records of data about employees and each employee is assigned a unique 5-digit employee ID number. We can use *MOD(employeeID, 1000)* as a hash function. The function will give us value between 0 and 999 which can be used as an address of a record in the file.



... -- Hash File ...

- Operations:
  - **Insertion**: In order to insert a record into a hash file, a hash function is applied onto the key value of the record. The record is stored at the address generated by the hash function.
  - **Deletion**: To delete a record form a hash file, a hash function is applied on the key value and the record from the address generated by the hash function is deleted from the file.
  - **Accessing**: In order to access a record in a hash file, the hash file is applied on the key value and the record from the address generated by the hash function is retrieved from the file.

… -- Hash File

- The hash file organization works very fine if the hash function can generate a unique address for every possible key value in the file. If this is not the case, then hash functions may generate the same address for two or more key values.
- A **collision** occurs when the hash field value of a record that is being inserted hashes to an address that already contains a different record. The process of finding another position to store the record is called **collision resolution**.
- Some methods for collision resolution are:
  - Open addressing
  - Chaining
  - Multiple hashing
- Hashing when applied to store and retrieve records from a disk is called external hashing

-- Index Structures

- Definition +
- Types of Indexes +

--- Definition

- An index is an access structures created to make the access to the data in the data file faster.
- The field of the records in the file on which an index is created is called the index field
- The index structure provide secondary access path

--- Types of Indices ...

- **Single-level index**: If the size of the data file is such that an index structure of reasonable size can be created and is sufficient to provide an efficient a access to the data in the file.
- **Multi-level index**: If the file of the index gets very large and another index is created on to the first index.
- **Primary Index**: Is an ordered file whose records are of fixed length with two fields. The first field is the same type as the ordering key field, called the primary key of the data file, and the second field is a pointer to a disk block.
- **Secondary indexes**: Indexes created on the fields other than the primary key.

… --- Types of Indices …

- **Clustering index**: is an index created on a nonkey field used for physically ordering records of a file. (one entry in the index for each distinct value of the ordering field)

- **Dense Index**: an index in which there is one entry for every search key value in the data file.

- **Sparse or non-dense Index**: An index which has entries for only some of the search value.

- Physical DB Design Process

- Introduction to Physical DB Design Process+
- Inputs to the Physical DB process+
- Outputs of the Physical DB Process +
- Guidelines for Index Selection +

-- Introduction to Physical DB Design Process

- After EER design, by mapping it to the relational data model, we get a relational schema for the DB.
- The next phase in the DB design process is physical DB design.
- Physical DB design is the process in which the designer should come up with the appropriate structures for the data storage on secondary devices such that it guarantees good performance of the DB.
- During Physical DB design we:
  - Choose appropriate file structure for each relation in the DB schema
  - Decide which indexes need to be created
  - Ensure that the performance goals of the DB will be met

-- Inputs to the Physical DB process ...

- The two main inputs for the physical DB design process are:
  1. The DB workload
  2. The DB performance requirements
- The DB workload includes
  1. A list of queries and their frequencies
  2. A list of updates and their frequencies
  3. Performance goals for each type of query and update.

... -- Inputs to the Physical DB process

- For each query in the workload, we must identify:
  - Which relations are accessed
  - Which attributes are retained (the SELECT clause)
  - Which attributes have selection or join conditions expressed on them and how selective these conditions are likely to be. (the WHERE clause)

- For each update in the workload, we must identify:
  - Which attributes have selection or join condition expressed on them and how selective these conditions are likely to be.
  - The type of update (INSERT, DELETE, UPDATE)
  - Relations and attributes that are modified by each update.

-- Outputs From the Physical DB Design

- The outputs from the physical DB design process include:
  - For each relation in the DB schema:
    - The appropriate file organization
  - For each index
    - The attribute(s) to index on
    - The type of index
    - The index file structure

-- Guidelines for Index Selection ...

- **Guideline 1:**
  - Don’t build index unless some queries benefit from it. Whenever possible choose indexes that speed up more than one query.
- **Guideline 2:**
  - Attributes mentioned in the WHERE clause are candidates for indexing.
  - An exact-match selection condition suggests that the attribute be considered for indexing.
  - A range selection condition suggests that B+ tree index be considered for attribute(s) in the selection condition.

… -- Guidelines for Index Selection …

- **Guideline 3:**
  - Indexes with multiple-attribute search keys should be considered in the following two solutions:
    1. A WHERE clause includes conditions one or more than one attributes of a relation.
    2. The attributes enable index-only evaluation strategies for important queries, i.e. only the index will be accessed and accessing the relation can be avoided.
- **Guideline 4:**
  - At most one index on a given relation can be clustered.
  - Clustering affects performance greatly, so the careful choice of clustering index is important.
  - If several range queries are posed on a relation involving different set of attributes, then these attributes are good candidates for clustered indexes

… -- Guidelines for Index Selection

- **Guideline 5:**
  - A B+ index is preferable because it supports range queries as well as equality queries.
- **Guideline 6:**
  - After preparing the initial list of indexes to create, consider the impact of each index on the updates in the workload. If maintaining an index slows down frequent update operations, consider dropping the index.