Physical Database Design

“Modern Database Management”
McFadden/Hoffer
Chapter 7

“Database Management Systems”
Ramakrishnan
Chapter 16

Overview

Physical Database Design

What is physical database design?

Issues

- Database Size Determination
- Database Usage Determination

How is database size determined?

How is database usage determined?

Calculating Database Size

Guidelines for Indexing

Physical Database Design

- What issues should be considered before physical design?
- What techniques are available for physical design?

How should a database be indexed?

Indexing

(c) Stephen Mc Kearney, 2003.
The Database Design Process

- The database design process involves three major stages:
  
  1. **Conceptual Design** During the conceptual design stage the database designer builds an entity-relationship model (ERM) of the data. The purpose of the conceptual design stage is to describe the structure of the data independently of a particular database management system.

  2. **Logical Design** During the logical design stage the database designer translates the conceptual design (ERM) into a logical data model. The relational data model (RDM) is a logical data model. When an RDM is used, the ERM is translated into a set of relations.

  3. **Physical Design** During the physical design stage the database designer implements the logical data model in a specific database management system (DBMS). When the logical data model is a relational model, the designer must create a set of relations in the DBMS.

Physical Database Design

“Physical database design is concerned with transforming the logical database structures into an internal model consisting of stored records, files, indexes and other physical structures”

*McFadden & Hoffer*

- Physical database design is the process of implementing the logical data model in a DBMS.
- *We shall assume that the logical data model is the relational data model.*
- The process of implementing a relational data model in a DBMS involves creating a set of relations. Creating relations in a DBMS involves:
  
  - Selecting the files in which to store the relations.
  - Ensuring that the set of relations are efficient to access.
  - Implementing additional database structures the improve the efficiency of the database. For example, an index improves the speed of some queries and the database design must decide which indexes should be created.

- The decisions made during the physical database design stage affect the speed of the database, the accessibility of the database, the security implemented on the database and the user-friendliness of the database.
Overview

What issues affect physical design?

- Database Size
  - Number of relations
  - Number of tuples
  - Size of each tuple
- Database Usage
  - Number of updates
  - Number of inserts
  - Number of deletes
  - Number of queries
- DBMS Manufacturer

Before the physical database design can be performed it is important to understand:

1. The maximum size of the database. The size of a database is determined by how many relations are in the database and how much data is held in each relation. The amount of data in the database will determine how long operations take to perform.

2. The use that will be made of the database. If a database is queried often then it will have to be designed to answer a large number of queries quickly. If data is added to the database often then it will have to be designed to handle many inserts quickly.
What does database size determine?

- **Storage Space**
  - The amount of hard space required to store the data.
- **Processing Time**
  - Time taken to execute queries and perform updates.
- **Communication Time**
  - Time taken to move data between systems.

- The amount of data in the database will determine the type of DBMS that must be used.
  - For example, a small database may be implemented on a simple DBMS but a large database will require a complex DBMS.
- The amount of data also determines the amount of disc space required to store the data.
  - For example, a small database will require a small amount of disc space while a large database will require large amounts of disc storage.
- The time it takes to execute queries or updates in the database is determined by the amount of data.
  - For example, an update that requires all the tuples in a large relation to be changed will take longer than an update that changes all the tuples in a small relation.
- The time it takes to transfer the data between systems.
  - For example, transferring a large amount of data between the database server and the user’s machine will take longer than transferring a small amount of data.

Calculating the database’s size

- The size of the database is determined by:
  1. The number of relations or entities in the database.
  2. The number of tuples in each relation.
  3. The size of each tuple in the database.
  4. Additional storage space required by the DBMS to information describing the structure of the database.

- In the ER model above, the number of entities has been estimated for each entity type. For example, there are estimated to be 100 locations in this database.
- The number of occurrences of each entity in each relationship has also been estimated. For example, there are estimated to be 10 patients for each location entity.
- The number of entities taking part in each relationship has been calculated by using the average. For example, the 10000 charges and 500 items have been averaged at 20 charges per item (10000/500=20).
What does database usage determine?

- The importance of data
  - Popular data items
    - Accessed quickly
  - Unpopular data items
    - Accessed less quickly
- The implementation of the database
  - Methods selected to implement the database.

The use of the database determines which parts of the database are important and will be accessed often.

A data item that is accessed every hour should normally be quicker to access than a data item that is accessed once a year.

Once the database designer knows the size of database and how it will be used, he or she must decide the most efficient method of implementing the database.

Identifying Database Usage - What?

- Database usage is:
  - The number of updates for each entity in the database.
  - The number of inserts for each entity in the database.
  - The number of deletes for each entity in the database.
  - The number of queries for each entity in the database.

In the ER model above, the estimated number of accesses (per hour) to each entity are marked on the diagram.

For example, there are 50 accesses by the user to the entity location. These accesses might represent queries requesting a list of all locations covered by the database.

Some accesses are internal to the database.

For example, there are 75 accesses from the patient entity to the treatment entity. These accesses may represent queries requesting details of the treatments administered to a particular patient.

Note that this diagram does not distinguish between entities that are updated and those that are queried. In many systems it may be necessary to make this distinction.
Identifying Database Usage - Access

- CRUD Matrix
  - CREATE
    - When is data created?
      - New tuples
  - READ
    - When is data read?
      - Queries
  - UPDATE
    - When is existing data changed?
  - DELETE
    - When is data deleted?

Identifying Data Usage - CRUD

- Linked to processes (eg DFDs)

<table>
<thead>
<tr>
<th>Process</th>
<th>CRUD Matrix</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill patient</td>
<td>x x</td>
<td>Creates, reads, updates, deletes in a process called ‘Bill patient’.</td>
</tr>
<tr>
<td>Charge</td>
<td>x</td>
<td>Process ‘Bill patient’ creates entity ‘Charge’.</td>
</tr>
<tr>
<td>Item</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The accesses to the database may be summarised in a CRUD matrix.
- The CRUD matrix shows the processes or transactions which occur in the system and matches them with the entities that they access.
- The CRUD matrix shows the creates, reads, updates and deletes that each process performs upon the entities. This allows the database designer to identify which processes create, read or change entities.
- The database designer can estimate the number and type of accesses made to the database by asking a series of questions about the data. For instance:
  - Which entities are searched/updated?
  - Which entities are deleted?
  - Which entities are affected by process ‘X’?
  - Which attributes are selected?
  - Which conditions are specified?
Overview

Physical Database Design

- **Objective**
  - Using knowledge of data usage and size improve the efficiency of the database

- **Techniques**
  - Indexing
  - Caching
  - Denormalisation

The objective of physical database design is to use the knowledge gained from analysing the use and size of the database to improve the efficiency of the database.

The efficiency of the database is determined by the requirements of the users and the database designer.

- For example, an efficient database may be one that executes queries quickly or uses the least amount of disc space.

There are many different tools that the database designer may use when implementing the physical database.

- **Indexing** increases the speed of accessing the data by storing special data structures which can be processed very quickly.
- **Caching** uses part of the computer’s main memory to store part of the database. Main memory is very quick to access.
- **Denormalisation** is a process of changing the structure of the relations to improve the performance of queries and updates.

*This lecture will discuss indexing.*
Indexing

One of the most common operations performed in a database is to locate a tuple (row) in a relation. For example, a query of an employee relation might ask the question “what is the salary of Mr. Smith?”. This query will return the row containing details about Mr. Smith.

When a database relation becomes very large, searching for information in it can be a very slow process.

Database management systems allow users to create indexes to speed up certain types of query.

McFadden et al define an index as “a table or other data structure that is used to determine the location of rows in a table [relation] that satisfy some condition”.

In the example above, an index has been created on the name attribute of the relation. The index is smaller than the relation. This means that it is faster to search the index than it is to search the relation.

The index speeds up queries that access individual tuples in a relation, for example, “show Smith’s salary”.

An index will not improve the speed of queries that access all the tuples in a relation, for example, “show all employee salaries”.

Indexing Guidelines

- Index the primary keys
- Index the foreign keys
- Index attributes that restrict queries
- Index attributes that are sorted
- Do not index attributes with few values
- Do not index every attribute

Guidelines for Indexing

- Index primary keys
  - Primary keys are unique and are often used to access individual tuples in a relation. Primary keys are used to join two relations.
- Index foreign keys
  - Foreign keys are used to join two relations together. An index will improve the speed of joins between relations.
- Index attributes that restrict queries
  - In the query “show the salary where name = ‘Smith’” the attribute name is restricting the query to all Smiths. To answer this query all the tuples containing ‘Smith’ must be retrieved. An index will improve the speed of queries that access individual names.
- Index attributes that are sorted
  - Sorting data is a very slow process. Indexes are sorted when they are built. Sorting data using an index simply means reading the index.
- Do not index attributes with few values
  - Indexes work best when accessing individual tuples. Attributes with few values will have many duplicates.
- Do not index every attribute
  - Indexes are stored on disc. Unnecessary indexes waste space.