Chapter Six

Physical Database Design and Performance

Purpose of physical DB design

- How to map the logical database design to a physical database design.
- How to design base relations for target DBMS.
- How to design enterprise constraints for target DBMS.
- How to select appropriate file organizations based on analysis of transactions.
- When to use secondary indexes to improve performance.
- How to estimate the size of the database.
- How to design user views.
- How to design security mechanisms to satisfy user requirements.

What is physical database design ?

- The process of producing a description of the implementation of the database on secondary storage.
- Describes the base relation, file organization, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.
- Logical database design is concerned with the **what**; physical database design is concerned with the **how**
- Describes the storage structures and access methods used to achieve efficient access to the data

Steps in physical DB design(1)

- Translate logical data model for target DBMS
  - Design base relation
  - Design representation of derived data
  - Design enterprise constraint
- Design physical representation
  - Analyze transactions
  - Choose file organization
  - Choose indexes
  - Estimate disk space and system requirement

Steps in physical DB design(2)

- Design user view
- Design security mechanisms
- Consider controlled redundancy
- Monitor and tune the operational system

Translate logical data model for target DBMS

- To produce a relational database schema in the target DBMS.
- Includes
  - creating the data dictionary based on the logical model and information gathered.
  - Understanding the functionality of the target DBMS

Knowledge of the DBMS includes:

- How to create base relations
- Whether the system supports:
  - Definition of Primary key
  - Definition of Foreign key
  - Definition of Alternate key
  - Definition of Domains
  - Referential integrity constraints
  - Definition of enterprise level constraints

Design base relation

- Involves identification of all necessary requirements about a relation starting from the name up to the referential integrity constraints.
- For each relation, need to define:
  - The name of the relation;
  - A list of simple attributes in brackets;
  - The PK and, where appropriate, AKs and FKs.
  - A list of any derived attributes and how they should be computed;
  - Referential integrity constraints for any FKs identified.
- For each attribute, need to define:
  - Its domain, consisting of a data type, length, and any constraints on the domain;
  - An optional default value for the attribute;
  - Whether the attribute can hold nulls.

Design representation of derived data

- Examine logical data model and data dictionary, and produce list of all derived attributes.
- Whether to store derived attributes in a base relation or calculate them when required is a decision to be made by the designer considering the performance impact.
- Option selected is based on:
  - Additional cost to store the derived data and keep it consistent with operational data from which it is derived;
  - Cost to calculate it each time it is required.
- Less expensive option is chosen subject to performance constraints.
- The representation of derived attributes should be fully documented.

Cont….

- **Design enterprise constraint**
- **Design physical representation**
  - Number of factors that may be used to measure efficiency:
  - Transaction throughput: number of transactions processed in given time interval.
  - Response time: elapsed time for completion of a single transaction.
  - Disk storage: amount of disk space required to store database files.
- Typically, have to trade one factor off against another to achieve a reasonable balance.
  - **Analyze transactions**
  - **Choose file organization**
  - **Choose indexes**
  - **Estimate disk space and system requirement**

File Organizations

- Technique for physically arranging records of a file on secondary storage
- Factors for selecting file organization:
  - Fast data retrieval and throughput
  - Efficient storage space utilization
  - Protection from failure and data loss
  - Minimizing need for reorganization
  - Accommodating growth
  - Security from unauthorized use
- Types of file organizations
  - Sequential
  - Indexed
  - Hashed

Guidelines for choosing indexes

- Do not index small relations.
- Index PK of a relation if it is not a key of the file organization.
- Add secondary index to a FK if it is frequently accessed.
- Add secondary index to any attribute that is heavily used as a secondary key.
- Add secondary index on attributes that are involved in: selection or join criteria; ORDER BY; GROUP BY; and other operations involving sorting (such as UNION or DISTINCT).
- Add secondary index on attributes involved in built-in functions.
- Add secondary index on attributes that could result in an index-only plan.
- Avoid indexing an attribute or relation that is frequently updated.
- Avoid indexing an attribute if the query will retrieve a significant proportion of the tuples in the relation.
- Avoid indexing attributes that consist of long character strings.

Cont….

- Design user view
- Design security mechanisms (System security and Data security)
- Consider controlled redundancy
- Monitor and tune the operational system

Denormalization(1)

- sometimes a normalized database design does not provide maximum processing efficiency.
- Also consider that denormalization:
  - Makes implementation more complex;
  - Often sacrifices flexibility;
  - May speed up retrievals but it slows down updates.
- Denormalization refers to a refinement to relational schema such that the degree of normalization for a modified relation is less than the degree of at least one of the original relations.

Denormalization(2)

- Transforming **normalized** relations into **unnormalized** physical record specifications
- Benefits:
  - Can improve performance (speed) by reducing number of table lookups (i.e. reduce number of necessary join queries)
- Costs (due to data duplication)
  - Wasted storage space
  - Data integrity/consistency threats
- Common denormalization opportunities
  - One-to-one relationship
  - Many-to-many relationship with attributes
  - Reference data (1:N relationship where 1-side has data not used in any other relationship)

Denormalization(3)

Consider denormalization in following situations, specifically to speed up frequent or critical transactions:

Step 1 - Combining 1:1 relationships

Step 2 - Duplicating non-key attributes in 1:* relationships to reduce joins

Step 3 - Duplicating foreign key attributes in 1:* relationships to reduce joins

Step 4 - Introducing repeating groups

Step 5 - Merging lookup tables with base relations

Step 6 - Creating extract tables.

T h e E N D ! ! !