

Chapter 18

Physical Database Design for the Relational Model

1

COMP211

Objectives

- Purpose of physical database 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.

2

COMP211

Comparison of Logical and Physical Database Design

- Sources of information for physical design process includes global logical data model and documentation that describes model.
- Logical database design is concerned with the what, physical database design is concerned with the how.

3

COMP211

Physical Database Design

- Process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

4

COMP211

Overview of Physical Database Design Methodology

- Step 3 Translate logical data model for target DBMS
 - Step 3.1 Design base relations
 - Step 3.2 Design representation of derived data
 - Step 3.3 Design general constraints
(e.g. DreamHome has a rule that prevents a staff from managing more than 100 properties at the same time)

5

COMP211

Overview of Physical Database Design Methodology (cont'd)

- Step 4 Design file organizations and indexes
 - Step 4.1 Analyze transactions
 - Step 4.2 Choose file organizations
 - E.g. Heap, Hash, Indexed Sequential Access Method (ISAM), B+-Tree, and Clusters.
 - This step can be omitted if DBMS does not allow the choice of file organizations
 - Step 4.3 Choose indexes
 - Step 4.4 Estimate disk space requirements

6

COMP211

Overview of Physical Database Design Methodology (cont'd)

- Step 5 Design user views
- Step 6 Design security mechanisms
- Step 7 Consider the introduction of controlled redundancy
- Step 8 Monitor and tune the operational system

7

COMP211

Step 3 Translate Logical Data Model for Target DBMS

- Objective: To produce a relational database schema from the logical data model that can be implemented in the target DBMS.
- Need to know functionality of target DBMS such as how to create base relations and whether the system supports the definition of:
 - PKs, FKs, and AKs;
 - required data – i.e. whether system supports NOT NULL;
 - domains;
 - relational integrity constraints – entity and referential integrity;
 - general constraints.

} Integrity constraints

8

COMP211

Step 3.2 Design Representation of Derived Data

- Examine logical data model and data dictionary, and produce list of all derived attributes.
- Derived attribute can be stored in database or calculated every time it is needed.
- 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.

9

COMP211

Step 4 Design File Organizations and Indexes

- Objective: To determine optimal file organizations to store the base relations and the indexes that are required to achieve acceptable performance; that is, the way in which relations and tuples will be held on secondary storage.

10

COMP211

Step 4 Design File Organizations and Indexes (cont'd)

- 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.
- However, no one factor is always correct. Typically, have to trade one factor off against another to achieve a reasonable balance.

11

COMP211

Step 4.1 Analyze Transactions

- To select appropriate file organizations and indexes:
 - Attempt to identify performance criteria, such as:
 - transactions that run frequently and will have a significant impact on performance;
 - transactions that are critical to the business;
 - times during the day/week when there will be a high demand made on the database (called the peak load).
- Use this information to identify the parts of the database that may cause performance problems.

12

COMP211

Example Transaction Analysis Form

Transaction Analysis Form 1-Sept-2001

Transaction (D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

Transaction volume
Average: 50 per hour
Peak: 100 per hour (between 17.00 and 19.00 Monday-Saturday)

SELECT propertyNo, p.street, p.postcode, type, rent
FROM Branch b INNER JOIN PropertyForRent p ON
b.branchNo = p.branchNo
WHERE p.city = 'Glasgow'
ORDER BY rent;

Predicate: p.city = 'Glasgow'
Join attributes: b.branchNo = p.branchNo
Ordering attribute: rent
Grouping attribute: none
Built-in functions: none
Attributes updated: none

Transaction usage map

Assume 4 Glasgow offices

avg = 1000
max = 3000

Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	Branch (entry)	R	100	5000	10000
2	PropertyForRent	R	4000-12000	200000-600000	400000-1200000
Total References			4100-12100	205000-605000	410000-1210000

13

COMP211

Step 4.3 Choose Indexes

- Objective: To determine whether adding indexes will improve the performance of the system.
- Secondary indexes provide a mechanism for specifying an additional key for a base relation that can be used to retrieve data more efficiently.
- However, there is an overhead involved in the maintenance and use of secondary indexes that has to be balanced against the performance improvement gained when retrieving data.

14

COMP211

Step 4.3 Choose Indexes (cont'd)

- Another approach is to order the tuples in a relation by specifying a primary or clustering index. In this case, choose the attribute for ordering or clustering the tuples as:
 - attribute that is used most often for join operations - this makes join operation more efficient, or
 - attribute that is used most often to access the tuples in a relation in order of that attribute.
- If the ordering attribute chosen is the key of the relation, the index is a **primary index**; otherwise, it is a **clustering index**.

15

COMP211

Step 4.3 Choose Indexes – Guidelines for Choosing 'Wish-List'

- (1) Do not index small relations.
- (2) Index PK of a relation if it is not a key of the file organization.
- (3) Add secondary index to a FK if it is frequently accessed.
- (4) Add secondary index to any attribute that is heavily used as a secondary key.
- (5) 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).

16

COMP211

Step 4.3 Choose Indexes – Guidelines for Choosing ‘Wish-List’

- (6) Add secondary index on attributes involved in built-in functions.
- (7) Add secondary index on attributes that could result in an index-only plan.
- (8) Avoid indexing an attribute or relation that is frequently updated.
- (9) Avoid indexing an attribute if the query will retrieve a significant proportion of the tuples in the relation.
- (10) Avoid indexing attributes that consist of long character strings.