Course code : **CSE2007**
Course title : **Database Management System**
Module : **4**
Topic : **2**

Query Processing

Objectives

This session will give the knowledge about

- Algorithms for SELECT Operations

Schema for all Query Operation

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Algorithms for SELECT Operation

There are many algorithms for executing a SELECT operation, which is basically a search operation to locate the records in a disk file that satisfy a certain condition.

OP1: $\sigma_{Ssn = '123456789'}$ (EMPLOYEE)
OP2: $\sigma_{Dnumber > 5}$ (DEPARTMENT)
OP3: $\sigma_{Dno= 5}$ (EMPLOYEE)
OP4: $\sigma_{Dno= 5 \text{ AND } Salary > 30000 \text{ AND } Sex = 'F'}$ (EMPLOYEE)
OP5: $\sigma_{Essn = '123456789' \text{ AND } Pno =10}$(WORKS_ON)

Search Methods for Simple Selection

A number of search algorithms are possible for selecting records from a file. These are also known as file scans, because they scan the records of a file to search for and retrieve records that satisfy a selection

The following search methods (S1 through S6) are examples of some of the search algorithms that can be used to implement a select operation.

S1: Linear search (brute force):

Retrieve every record in the file, and test whether its attribute values satisfy the selection condition.

Search Methods for Simple Selection

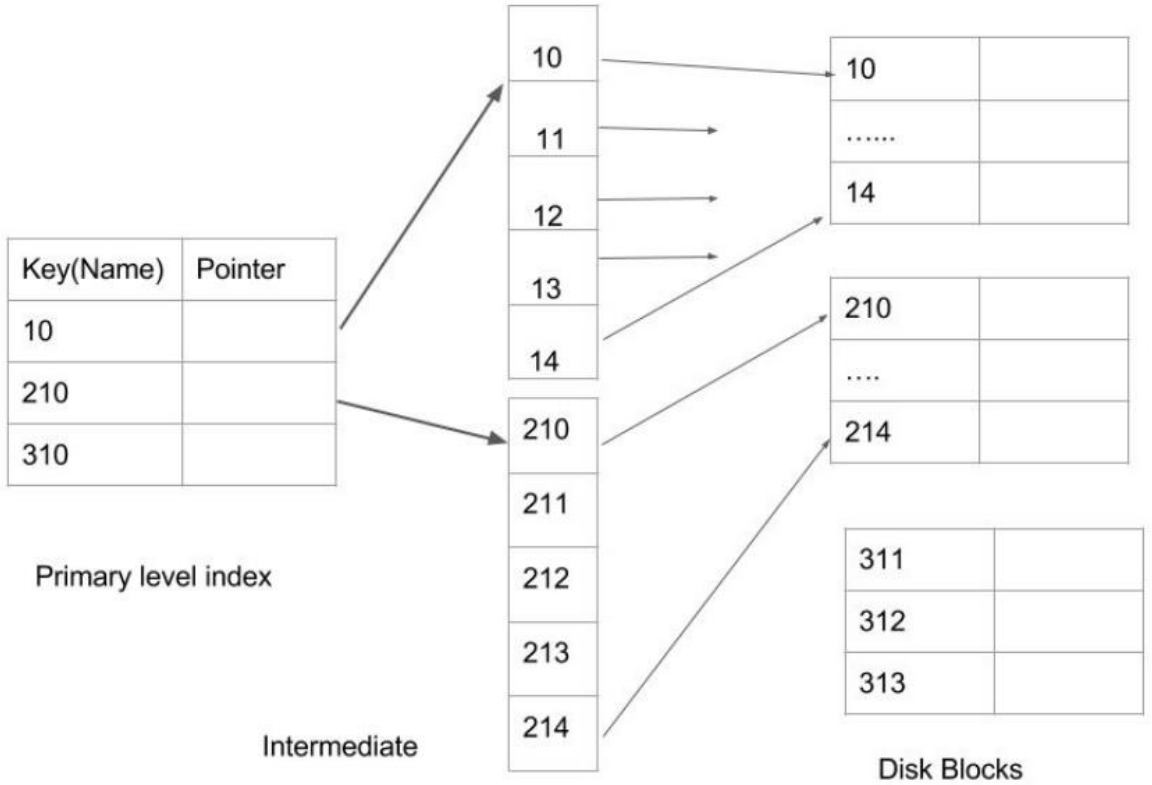
S2: Binary search:

If the selection condition involves an equality comparison on a key attribute on which the file is ordered, binary search (which is more efficient than linear search) can be used. (See OP1).

S3: Using a primary index or hash key to retrieve a single record:

If the selection condition involves an equality comparison on a key attribute with a primary index (or a hash key), use the primary index (or the hash key) to retrieve the record.

Primary Indexing



Secondary level Indexing

Search Methods for Simple Selection

S4: Using a primary index to retrieve multiple records:

If the comparison condition is >, ≥, <, or ≤ on a key field with a primary index, use the index to find the record satisfying the corresponding equality condition, then retrieve all subsequent records in the (ordered) file.

Example, Dnumber > 5 in OP2—use the index to find the record satisfying the corresponding equality condition (Dnumber = 5); then retrieve all subsequent records in the (ordered) file. For the condition Dnumber < 5, retrieve all the preceding records.

Search Methods for Simple Selection

S5: Using a clustering index to retrieve multiple records:

If the selection condition involves an equality comparison on a nonkey attribute with a clustering index, use the clustering index to retrieve all the records satisfying the selection condition.

Example, Dno = 5 in OP3 - use the index to retrieve all the records satisfying the condition.

Search Methods for Simple Selection

S6: Using a secondary (B+-tree) index:

On an equality comparison, this search method can be used to retrieve a single record if the indexing field has unique values (is a key) or to retrieve multiple records if the indexing field is not a key.

In addition, it can be used to retrieve records on conditions involving >,>=, <, or <=. (FOR RANGE QUERIES).

Search Methods for Simple Selection

S7a: Using a bitmap index:

If the selection condition involves a set of values for an attribute (e.g., Dnumber in (3,27,49) in OP6), the corresponding bitmaps for each value can be OR-ed to give the set of record ids that qualify.

In this example, that amounts to OR-ing three bitmap vectors whose length is the same as the number of employees.

S7b: Using a functional index:

In OP7, the selection condition involves the expression ((Salary*Commission_pct) + Salary). then this index can be used to retrieve employee records that qualify.

Conjunctive and Disjuctive Selection

Conjunctions

This is the reference to the conditions in the WHERE clause combined with AND operator.

Example:

SELECT * FROM STUDENT WHERE CLASS_ID = ‘DESIGN_01’ AND AGE>=18;

Disjunction

This is the reference to the conditions in the WHERE clause combined with OR operator. This is even considered as queries with UNION operator.

Example:

SELECT * FROM STUDENT WHERE CLASS_ID = ‘DESIGN_01’ OR AGE>=18;

Search Methods for Conjunctive Selection

S8: Conjunctive selection using an individual index:

If an attribute involved in any single simple condition in the conjunctive select condition has an access path that permits the use of one of the methods S2 to S6, use that condition to retrieve the records and then

Check whether each retrieved record satisfies the remaining simple conditions in the conjunctive select condition.

Search Methods for Conjunctive Selection

S9: Conjunctive selection using a composite index:

If two or more attributes are involved in equality conditions in the conjunctive select condition and a composite index (or hash structure) exists on the combined fields.

Example, if an index has been created on the composite key (Essn, Pno) of the WORKS_ON file for OP5—we can use the index directly

Search Methods for Conjunctive Selection

S10: Conjunctive selection by intersection of record pointers:

This method is possible if secondary indexes are available on all (or some of) the fields involved in equality comparison conditions in the conjunctive condition and if the indexes include record pointers (rather than block pointers).

Each index can be used to retrieve the record pointers that satisfy the individual condition.

The intersection of these sets of record pointers gives the record pointers that satisfy the conjunctive condition, which are then used to retrieve those records directly.

If only some of the conditions have secondary indexes, each retrieved record is further tested to determine whether it satisfies the remaining conditions

Search Methods for Disjunctive Selection

Compared to a conjunctive selection condition, a disjunctive condition (where simple conditions are connected by the OR logical connective rather than by AND) is much harder to process and optimize. For example, consider OP4′:

$$OP4': \sigma_{Dno=5 \text{ OR } Salary > 30000 \text{ OR } Sex = 'F'}(EMPLOYEE)$$

With such a condition, the records satisfying the disjunctive condition are the union of the records satisfying the individual conditions.

Hence, if any one of the conditions does not have an access path, we are compelled to use the brute force, linear search approach.

Only if an access path exists on every simple condition in the disjunction can we optimize the selection by retrieving the records satisfying each condition

Implementing the SELECT Operation

Whenever a single condition specifies the selection, we can only check whether an access path exists on the attribute involved in that condition.

- If an access path exists, the method corresponding to that access path is used; otherwise, the “brute force” linear search approach of method S1 is used. (See OP1, OP2 and OP3)

For conjunctive selection conditions, whenever more than one of the attributes involved in the conditions have an access path, query optimization should be done to choose the access path that retrieves the fewest records in the most efficient way.

Summary

This session will give the knowledge about

- Algorithms for SELECT Operations