Query Processing and Optimization

Implementing Join

CSCI-B 561

Adapted from Elmasri & Navathe
Basic Join Algorithms

- Common methods to join tables
  - Nested Loop Join
  - Sort Merge Join
  - Hash Join
General Overview

• A join of two tables occurs in two steps
  – where one table becomes the outer table
  – and the other becomes the inner table.

• During join processing
  – records from outer table are presented one-by-one to the inner table
  – and the inner table searched for records matching the one presented to it.
Implementing the Join Operation

- **TWO WAY JOIN:**
  - R JOIN \( A=B \) S

- **MULTI-WAY JOIN:**
  - R JOIN \( A=B \) S JOIN \( C=D \) T ......

- Example operations:
  - (OP6): employee JOIN dno=dnumber department
  - (OP7): department JOIN mgrssn=ssn employee
Nested Loop Join

• J1. **Nested (inner-outer) loop** approach (brute force):

  • For each record \( t \) in \( R \) (outer loop), retrieve every record \( s \) from \( S \) (inner loop) and test whether the two records satisfy the join condition \( t[A] = s[B] \).
Implementing Nested Loop Join

• Process:
  – Records of the outer table are retrieved (or presented) using a table scan (or indexing, if possible).
    • Only candidate records are retrieved.
    • Candidate records satisfy local predicates.
  – For each retrieved row from the outer table:
    • the inner table is searched for qualifying records
    • & the results merged into a new record in an output table.

```
SELECT CustomerID, Name, Street, City, State, Zip, TotalSales
FROM Customers, Sales
WHERE State = "IN"
AND Customers.CustomerID = Sales.CustomerID
```
Nested Loop Joins

• Appropriate:
  – When the outer table has only a few records (after applying predicates)
  – Where the inner table is small or has an index usable to access qualifying records

• Drawbacks when the inner and outer tables are not indexed or clustered on the same values:
  – The inner table (including indexes) may be scanned repetitively to find matching records.
  – The outer table is processed inefficiently for records with the same value in the join columns.
Using Indexes to Speed up Join Operation

• J2. Using an **access structure to retrieve the matching record(s):**

• If an index (or hash key) exists for one of the two join attributes say, B of S retrieve each record t in R, one at a time, and then use the access structure to retrieve directly all matching records s from S that satisfy s[B] = t[A].
Sort-Merge Join

• J3. Sort-merge join:
• If the records of R and S are *physically sorted* (ordered) by value of the join attributes A and B, respectively, we can implement the join in the most efficient way possible. Both files are scanned in order of the join attributes, matching the records that have the same values for A and B. In this method, the records of each file are scanned only once each for matching with the other file, unless both A and B are non-key attributes, in which case the method needs to be modified slightly.
Sort-Merge Joins

• Algorithm where the 2 tables are scanned only once:
  - For each table, a temporary table is created of qualifying candidate records.
  - Each temporary table is sorted on the same columns based on the joining predicates (conditions).
  - Finally the two temporary tables are merged and joined (as an inner and outer table) into a third table.

• If either of the original two tables has an index on the selection and joining predicates, it may be possible to skip the creation of the corresponding temporary table.
Hash-Join Operation

- **J4. Hash-join**: The records of files R and S are both hashed to the same hash file, using the same hashing function on the join attributes A of R and B of S as hash keys. A single pass through the file with fewer records (say, R) hashes its records to the hash file buckets. A single pass through the other file (S) then hashes each of its records to the appropriate bucket, where the record is combined with all matching records from R.
Hash-Join Operation

- **J4. Hash-join:** The records of files R and S are both hashed to the *same hash file*, using the *same hashing function* on the join attributes A of R and B of S as hash keys.
- **Step 1:** A single pass through the file with fewer records (say, R) hashes its records to the hash file buckets.
- **Step 2:** A single pass through the other file (S) then hashes each of its records to the appropriate bucket, where the record is combined with all matching records from R.