Access path selection in a relational database management system

Jonathan Lam

09/25/21

1 Overview

by Selinger et. al.

- About the System R implementation at IBM
- SQL is a declarative language and doesn't specify the **access path** for a query

2 Processing of a SQL statement

- Four phases:
  - Parsing
    - Check for correct syntax
    - **Query block** comprises a `SELECT` list, `FROM` list, and `WHERE` tree
    - Each SQL query may have multiple query blocks
  - Optimization
    - Checks for the existence of the names from the query
    - Retrieves access paths for names from the query and performs access path selection
    - This is represented by a structural modification of the parse tree
    - The end result is an execution in the **Access Specification Language** (ASL)
  - Code generation

    - Uses a small number of code templates, one for each type of join
    - Nested queries are treated as subroutines
  - Execution
    - Calls the System **R storage system** (RSS) via the **storage system interface** (RSI)

3 RSS

- Manages physical storage of relations, access paths on these relations, locking, and logging/recovery facilities
- Presents a tuple-oriented interface (RSI) to users
- May be used independently of System R
- "Relations are stored in the RSS as a collection of tuples whose columns are physically contiguous. These tuples are stored on 4K byte pages; no tuple spans a page. Pages are organized into logical units called segments. Segments may contain one or more relations, but no relation may span a segment. Tiples from two or more relations may occur on the same page. Each tuple is tagged with the identification of the relation to which it belongs."
- `OPEN`, `NEXT`, and `CLOSE` are the principal commands on a scan.
- Two types of scans:
  - **Segment scan**:
    - Find all tuples of the given relation
  - **Index scan**:
    - Indices are implemented as B-trees
    - Indices are on separate pages than the tuples
    - Indices are chained together so sequential access is fast
    - If tuples in segment pages are clustered like they are in the index pages, then the index is **clustered**: "a clustered index has the property that no only each index page, but also each data page containing a tuple from that relation will be touched only once in a scan on that index"

    * Index scan need not scan the entire index; **search arguments** (sargs) may be given as a boolean expression of **sargable predicates** in disjunctive normal form (DNF)

4 Costs for single relation access paths

- Cost measure: `COST = PAGE FETCHES + W * (RSI CALLS)`
  - `RSI CALLS` is the predicted number of tuples returned by the RSS
- We say that a predicate or set of predicates **matches** an index access path when the predicates are sargable and the columns mentioned in the predicate(s) are an initial substring of the set of columns of the index key.
- Optimizer retrieves statistics on the relations in the query and on the access paths available on each relation:
  - For each relation `T`:
    * `NCARD(T)`, the cardinality of `T`
    * `TCARD(T)`, the number of pages in the segment that hold tuples of relation `T`
    * `P(T)`, the fraction of data pages in the segment that hold tuples of relation `T`
  - For each index `I` on relation `T`:
    * `ICARD(I)`, number of distinct keys in index `I`
    * `NINDX(I)`, number of pages in index `I`
  - These statistics are maintained at startup, and on the `UPDATE STATISTICS` command. They are not updated at every `UPDATE`, `INSERT`, or `DELETE` command.
- **Selectivity factor** (`F`) is chosen for each boolean factor in the predicate list, which is the expected fraction of tuples which will satisfy the predicate.
  - The paper lists a number of calculations for selectivity factors
- **Query cardinality** (`QCARD`) is the product of the cardinalities of every relation in the query block's `FROM` list times the product of all the selectivity factors of that query block's boolean factors.

- **Expected RSI calls** (`RSICARD`) is the product of the relation cardinalities times the selectivity factors of the sargable boolean factors
- **Interesting order**: Using an index access path or sorting tuples produces tuples in the index value or sort key order. We say that a tuple order is an interesting order if that order is one specified by the query block's `GROUP BY` or `ORDER BY` clauses.
- The cost of each interesting order is considered, as well as the cost of the cheapest unordered access path.
  - If there is no `GROUP BY` or `ORDER BY` clauses, then the cheapest path is chosen.
  - If there is one, the cheapest of the interesting orders and the (unordered path + cost to sort) is chosen.

5 Access path selection for joins

- Blasgen and Eswaran (1976) showed that there are two join methods is always optimal or near optimal
- For joins involving two relations, the two relations are called the **outer relation**, from which a tuple will be retrieved first, and hte **inner relation**, from which tuples will be retrieved, possibly depending on the values obtained in the outer relation tuple.
- A predicate which relates columns of two tables to be joined is called a **join** predicate.
- The columns referenced in a join predicate are called **join columns**.
- One method: **nested loops**: very straightforward
- Other method: **merging scans**: requires outer and inner relations to be scanned in join column order.
  - Columns of equi-join predicates also define interesting orders.
  - If more than one join predicate, one is used as the join predicate and the others are regular predicates.
  - This is only applied to equi-join predicates because of its increased complexity.

- N-way joins are repeated 2-way joins.
  - We can interleave joins.
  - Intermediate relations are not stored unless they need to be sorted.
- All joins requiring Cartesian products are performed as late in the join sequence as possible.

6 Nested queries

- If the nested query result is used in a relation, then it evaluates to a single expression.
- The nested query may be computed beforehand and incorporated into the outer expression.
- If the nested query can return a set of values, then they are returned in a temporary list.
- If the nested query contains a reference to a value obtained from a candidate tuple of a higher level query block, then it is called a **correlation subquery** and must be re-evaluated for each candidate tuple.