Maintenance of Cube Automatic Summary Tables

Wolfgang Lehner[i]
University of Erlangen-Nuremberg
Martensstr. 3, Erlangen, 91056, Germany
wolfgang@lehner.net

Richard Sidle, Hamid Pirahesh, Roberta Cochrane
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120, USA
{rsidle, pirahesh, bobbiec}@almaden.ibm.com

ABSTRACT

Materialized views (or Automatic Summary Tables—ASTs) are commonly used to improve the performance of aggregation queries by orders of magnitude. In contrast to regular tables, ASTs are synchronized by the database system. In this paper, we present techniques for maintaining cube ASTs. Our implementation is based on IBM DB2 UDB.

1. INTRODUCTION

ASTs are a well-known technique for improving the performance of aggregation queries that access a large amount of data while performing multiple joins in the context of a typical data warehouse star schema. Fully exploiting the power of the AST technique requires support from the database system in (a) picking the optimal set of ASTs for a specific application scenario and workload [1], (b) transparently rerouting user queries originally referencing base tables to those views [4], and (c) maintaining ASTs, i.e. synchronizing them with the base tables [2]. This paper focuses on techniques for maintaining cube ASTs.

2. DEFINITION OF ASTs

The example in Figure 1 defines a hierarchical data cube for location (city,state,country), product (group,lineitem), and time (month,year) dimensions further categorized according to marital status and income range of the customer. This example demonstrates that a complete OLAP scenario, providing data for 144 (4*3*3*4) grouping combinations at different levels of aggregation, can be specified as a single summary table.

Similar to a regular view, the content of an AST is defined by a SELECT expression. Additionally, an AST definition may contain an explicit specification of its physical layout similar to a regular base table, i.e. it can be partitioned, replicated, indexed, etc. Finally, each AST has a refresh mode. Declaring an AST as 'REFRESH IMMEDIATE' implies that all dependent ASTs are automatically synchronized when the underlying base data is modified. This is done optimally by applying the incremental maintenance strategy outlined in this abstract. If an AST is declared 'REFRESH DEFERRED' then no base table changes are propagated when a base table is modified. In lieu of sophisticated algorithms [3], refreshing a deferred AST implies full recomputation. While a deferred AST may be defined by any SELECT expression, the specification of an incrementally maintainable AST must adhere to the following rules:

- Grouping expression: The grouping expression may consist of single grouping columns or any valid combination of complex grouping expressions like CUBE(), ROLLUP(), or GROUPING SETS(). The evaluation of the grouping expression must not result in duplicate grouping combinations. For example, ROLLUP(a,b),a is not allowed, since it evaluates to ((a,b),a) = (a,b); ((a),a) = (a); and ((),a) = (a), resulting in the combination with (a) appearing twice.
- Aggregate functions: The set of aggregate functions is restricted to SUM and COUNT. Every AST must have a COUNT(*) column. If a column X is nullable and the AST computes SUM(X), a named COUNT(X) column is also required.
- Grouping functions: A GROUPING() function expression is required for any nullable grouping column that occurs in a complex grouping expression. This allows the system to differentiate between naturally occurring NULL-values and NULL-values that denote (sub-) totals. In the sample AST, the grouping columns “marital status” and “income range” are nullable. Since these columns may naturally produce NULL values, they require a GROUPING function column in the definition of the AST.

3. INCREMENTAL MAINTENANCE

The advantage of an incremental maintenance strategy is that the changes in the AST are computed directly from the changes of the base table. Consider an AST containing a join over several tables. Incremental maintenance can compute the changes to the AST using the joins of only the changes of the base tables (deltas) with all other tables of the AST definition. One unique feature of DB2 UDB’s AST maintenance strategies is that its infrastructure naturally supports incremental maintenance for complex ASTs like hierarchical data cubes over a set of tables.

Maintenance of Automatic Summary Tables in IBM DB2/UDB

STEP I: Building the Raw Delta. All local deltas, i.e. the inserted, updated or deleted rows of all base tables are combined to generate the global raw delta stream. Multiple local deltas might be caused within the context of a single statement while maintaining database semantics, such as enforcing referential integrity constraints using 'ON DELETE CASCADE’. To synchronize an AST with an underlying update operation, the delta consists of the rows before and after the update extended with a numeric tag.

MOD 2000, Dallas, TX USA

```
CREATE SUMMARY TABLE ast_demo AS (
 SELET  loc.country, loc.state, loc.city,
        pg.lineid, pg.pgid,
        c.marital_status, c.income_range
        YEAR(t.pdate) AS year, MONTH(t.pdate) AS month
        SUM(ti.amount) AS amount,
        COUNT(*) AS count,
        GROUPING(c.marital_status) AS grp_mstatus,
        GROUPING(c.income_range) AS grp_income_range
 FROM   transitem AS ti, transaction AS t, location AS loc,
        pgroup AS pg, account AS a, customer AS c
 WHERE  ti.transid = t.transid AND ti.pgid = pg.pgid
   AND  t.locid = loc.locid AND t.acctid = a.acctid
   AND  a.custid = c.custid
 GROUP BY ROLLUP(loc.country, loc.state, loc.city),
          ROLLUP(pg.lineid, pg.pgid),
          ROLLUP(YEAR(t.pdate), MONTH(t.pdate)),
          CUBE(c.marital_status, c.income_range)
) DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
```

Figure 1: Sample AST Definition

STEP II: Aggregating the Delta. In this step, the delta stream is aggregated. If the underlying modification is an insertion or deletion, then the grouping specification contains all the combinations specified by the AST. For ASTs with complex grouping expressions, e.g. CUBE(), this step results in a complete delta cube with 'higher' delta aggregate values for all original delta changes. If the modification is an update, then the grouping specification contains all the combinations specified by the AST extended with the tag column. For updates, the resulting aggregate values are multiplied with the value of the tag, and a second delta aggregation step consisting of a simple aggregation over all grouping columns plus all grouping function columns is added to eliminate the tag column and compute the net aggregate changes (i.e. delta value) from the old to the new base table values.

STEP III: Pairing the Delta to the AST. After aggregation, the rows in the delta are paired with the current content of the AST using a left outer-join (the delta goes left) over the grouping and grouping function columns of the AST. Thus a delta group either matches with a single group of the summary table or no group at all. Delta groups that have matches cause the corresponding row in the AST to be modified; those that do not have matches are later added to the AST.

STEP IV: Aggregate Value Compensation. When a delta group has a corresponding group in the AST, then the new value for the group must be computed based on the value of the delta and the current value of the group. Since the AVG aggregation function can be mapped to an equivalent SUM/COUNT expression, '+' is the only aggregation value compensation function, required to support SUM, COUNT, and AVG.

For ASTs with complex grouping expressions (like CUBE(), ...), the overall summary value, or grand total, evaluates to NULL even if the number of contributing rows is zero and requires special treatment. The computation of the aggregate value SUM for non-nullable columns requires a COUNT(*) column to derive the new cardinality. If, however, the parameter column of the aggregate function is nullable, then the new cardinality is derived from the COUNT-values ranging over that nullable column.

STEP V: Applying the Delta to the AST-- Depending on the underlying base table operation, the delta stream is applied to the AST using the following operations:

- Base table insert: Already existing groups in the AST are updated, new groups are inserted into the AST.
- Base table delete: Groups of the delta with a new cardinality of zero are deleted, the remaining rows are updated with the new values of the delta stream. Note that in the case of ASTs with complex grouping expressions (like CUBE()), the grand total row is never be deleted.
- Base table update: This case may be considered a combination of base table insert and deletion resulting in a sequence of AST update, delete, and insert operation as described above.

4. FULL REFRESH

Although the incremental maintenance strategy provides an automatic synchronization for ASTs when the underlying base tables change, there are scenarios where 'DEFERRED' refresh is justified. For example, when incremental maintenance is becoming too expensive due to a high update frequency of the base tables and/or a high number of incrementally maintainable summary tables. In this case, ASTs can be fully refreshed.

5. SUMMARY AND FUTURE WORK

This paper outlines the current state-of-the-art in maintaining ASTs in the IBM DB2/UDB database system. The maintenance strategies provide a sound basis for a powerful data warehouse infrastructure within DB2.

6. REFERENCES

[1] Harinarayan, V.; Rajaraman, A.; Ullman, J.: Implementing Data Cubes Efficiently. In: SIGMOD'96, pp. 205

[2] Mumick, I.; Quass, D.; Mumick, B.: Maintenance of Data Cubes and Summary Tables in a Warehouse. In. SIGMOD' 97, pp. 100-111

[3] Beyer, K.; Cochrane, B. Lindsay, B.; Salem K.: How To Roll a Join. IBM research paper. In. SIGMOD'2000

[4] Zaharioudakis, M.; Cochrane, R.; Lapis, G.; Pirahesh, H.; Urata, M.: Answering Complex SQL Queries Using Automatic Summary Tables. In: SIGMOD'2000

[i] This work was done while author was visiting IBM Almaden Research Center.