An Architecture for Using Tertiary Storage in a Data Warehouse

Theodore Johnson
Database Research Dept.
AT&T Labs - Research

johnsont@research.att.com
Motivation

• AT&T has huge data warehouses.
  – Data from billing, operations, network operations, etc.
  – In the form of billions of small objects (50 bytes)
  – Complex “decision support” and data mining queries.

• Very large data sets are difficult to store on-line
  – Cost of purchase
  – Cost of maintenance.

• Tape-resident data is difficult to query
  – Data is exported from databases - interface mismatch.
  – Tape management software does not support analyses
    • Aggregation queries over multi-terabyte data sets.
“Decision Support” Queries

• Summarize large collections of data (aggregation).
  – Multiple levels of aggregation
  – Multiple dimensions of aggregation
  – “Data cubes”

• Use the aggregates for comparisons
  – “What fraction of total sales are made in California, by week, during 1996?”
  – “What fraction of total 1996 sales made in California are made in Orange County on Mondays?”
  – “Show the yearly/monthly/hourly volume of telephone traffic between pairs of area codes.”
More complex queries

• “Complex” aggregation
  – “For each connection, count the number of packets exchanged between call setup and teardown”
  – “For each customer, what is the most common destination of calls with above-average length.”
  – “Which stores are favored by customers that make large purchases?”

• Data mining queries
  – Association rules (which objects tend to appear together)
  – Multidimensional distribution analysis (is it changing?)
  – Decision trees, etc.

• Needle-in-the-haystack
  – List all calls made to or from 212-555-4321 during the last 5 years.
Tape-resident data warehouses

- **Detail data**: Finest granularity information.
- **Summaries**: Precomputed aggregate tables.
- Summaries and the head of the detail data is on-line.
- Tail of detail data is on tape.
- Use disk-resident data whenever possible to answer ad-hoc queries.
- Query tape-resident data when necessary.
- The detail data sets are a few multi-terabyte detail data tables.
- Queries on tape resident data frequently range over terabytes.
Challenges

• Existing DBMSs do not integrate well with tertiary storage
  – Tape storage is inherently sequential.
  – Conventional DBMS query evaluation algorithms require random access.
    • “Cross-product” joins
• Management of very large collections of small objects presents storage problems
  – Indices may be too large to store on-line.
• Existing tape management software not appropriate
  – Terabyte scale, data intensive queries
  – Control over data layout.
  – Control over scheduling.
Pkt(Source, Dest, ts, StartCall, EndCall)

create view Temp as
  select R.Source, R.Dest, R.ts, EndTime=min(S.ts)
  from Pkt R, Pkt S
  where R.Source=S.Source AND R.Dest=S.Dest AND
    R.ts <= S.ts AND R.StartCall=1 and S.EndCall=1
  group by R.Source, R.Dest, R.ts

select T.Source, T.Dest, T.ts, count(*) from Pkt R, Temp T
where R.Source=T.Source AND R.Dest=T.Dest AND
  R.ts >= T.ts and R.ts <= T.EndTime
group by T.Source, T.Dest, T.ts
Decision support queries on tape data

• Our target is large-scale aggregation queries
  – Usually, we want to build a new summary table.
    • We can’t predict every query in advance.
    • Similar problems occur with scientific data sets.

• Long sequential scans
  – Aggregation queries over the entire data set.
    • Or dense subsets.
  – Small “dimension tables” stored on disk.
  – Multiple passes for complex aggregation.

• Temporal nature of warehouse data permits localization.
  – Many optimizations are possible to reduce memory use.
  – Joins are localized -- “cross product” is rare.
Requirements

• High sequential throughput
  – Parallel I/O
    • Partition data set, perform concurrent reads.
  – Parallel processing
    • CPU use is still a consideration
  – Lightweight architecture
    • Transactions are done somewhere else.
  – Tape to memory transfers
    • Caching does not help a sequential scan through a terabyte.

• Database features
  – Control over data layout
  – Indexing
  – Declarative access
    • Enable parallel access and scheduling
  – Access methods for decision support queries
Target System

- High end server
  - $O(1 \text{ Gbyte})$ of main memory
  - $O(1 \text{ Tbyte})$ on-line storage
  - $O(10)$ processors

- Moderate to large size robotic storage library
  - Tape storage
  - 10 to 100 Mbyte/sec aggregate transfer rate.
Ingest

- Move detail data from disk-resident representation to tape-resident representation.
- Create horizontal and (perhaps) vertical partitions.
- Multiple sort orders.
- Allocate to tapes.
Query

• Input:
  – List of tape partitions to access.
  – Executable for processing the data.
  – Hash function.

• Execution:
  – Start the processing executables.
  – Pick a schedule for accessing tape partitions.
  – Use the hash function to route tuples from tape to executable.
Access Methods

• Single pass, no correlated accesses.
• Single pass, correlated accesses
  – e.g., join a partitioned table.
• Multiple pass

• Joins to disk resident data done at processing executables.
• Build queries out of basic access methods.
Test Implementation

- Large Unix database server
- Robotic storage library with four DLT4000 tape drives.
- Network monitoring data

Ingest step:
- Extract a 60 Gbyte table from the database, reformat it.
- "Stripe" it across 3 tapes (stripe unit is 500 Mbytes).

Query
- Complex aggregation query
  - groups are defined by tuple sequences.
- Re-use a query written for disk-resident data.
- Proper handling of sequences that span horizontal partitions.
- Obtained 5.2 Mbyte/sec processing rate (maximum is 5.7 Mbyte/sec)
Related Research

- Tertiary storage system performance characterization
- Indexing tape-resident detail data
- Query languages
Performance Characterization

- Query optimization requires precise information about device performance.
  - Layout, scheduling, indexing.
- Precise data on tertiary storage system component performance is lacking
  - Disk drives relatively well understood.
  - Wide variety of tape drives, robotic storage libraries.
  - Many quirks.
- Related paper in this conference.
Indexing detail data

- **Motivation**: look up calls made to/from a phone number
  - Required for law enforcement
  - Terabytes of small records
  - 1 per billion selectivity, or less.

- **Dense indices do not scale.**
  - 10% of 10 terabytes is 1 terabyte.
  - In this case, index size is equal to data size.

- **Coarse indices**
  - Indicate regions on tape where a record will exist.
    - Short seeks are slow
    - Aggressive use of compressed bitmaps.
    - 20 to 1 index size reduction, or greater.

- **See** *Coarse Indices for a Tape-based Data Warehouse*, Int’l Conf. on Data Engineering 1998, pg. 231-240
Query Languages

• Conventional DBMSs rely on joins.
  – Complex aggregation queries frequently involve a self-join.
• Computing tape-to-tape joins is prohibitively expensive
  – Load part of one table into memory, scan the other table.
  – Many passes, the number of passes increases with table size.
• Most complex aggregation queries require only 1 pass
  – I.e., hand-written query plans make 1 pass.
  – More complex queries can require another pass.
• Multi-feature extension to SQL
  – Chatziantoniou and Ross
  – Join-free query plans that use long sequential scans.
• Optimize for tape-resident data
  – Reduce the number of passes over the data, often to 1 pass.
  – Reduce memory usage.
Conclusions

• Tape-resident decision support data warehouses are desirable.
  – Existing practice is to export the tail of the detail data from the database.
  – Difficulty of access discourages use.

• Tape-resident decision support data warehouses are feasible.
  – Restrict the universe of queries to those that can be answered efficiently.
  – This set of queries covers a surprisingly large class of interesting queries.

• Work on efficient access.
  – Ensure high-throughput sequential access.
  – Get the most out of sequential access.