Online Query Processing  
A Tutorial

Peter J. Haas  
IBM Almaden Research Center  
Joseph M. Hellerstein  
UC Berkeley

Goals for Today

- Exposure to online query processing algorithms and fundamentals
- Usage examples
- Basic sampling techniques and estimators
- Preferential data delivery
- Online join algorithms
- Relation to OLAP, etc.
- Some thoughts on research directions
- More resources to appear on the web
- Annotated bibliography
- Extended slide set
- Survey paper

Road Map

- Background and motivation
  - Human-computer interaction
  - Tech trends and prognostications
  - Goals for online processing
  - Examples of online techniques
  - Underlying technology
  - Related work
  - Looking forward

Human-Computer Interaction

- Iterative querying with progressive refinement
- Real-time interaction (impatience!)
- Spreadsheets, WYSIWYG editors
- Modern statistics packages
- Netscape STOP button
- Visually-oriented interface
- Approximate results are usually OK

Disk Appetite

- Greg Papadopoulos, CTO Sun:
  - “Moore’s Law Ain’t Good Enough” (Hot Chips ’98)

Source: J. Porter, Disk/Trend, Inc.  

The Latest Commercial Technology
Drawbacks of Current Technology

- Only exact answers are available
- A losing proposition as data volume grows
- Hardware improvements not sufficient
- Interactive systems fail on massive data
- E.g., spreadsheet programs (64K row limit)
- DBMS not interactive
- No user feedback or control ("back to the 60's")
- Long processing times
- Fundamental mismatch with preferred modes of HCI
- OLAP: a partial solution
- Can't handle ad hoc queries or data sets

Goals for Online Processing

- New "greedy" performance regime
- Maximize 1st derivative of the "mirth index"
- Mirth defined on-the-fly
- Therefore need FEEDBACK and CONTROL

Road Map

- Background and Motivation
- Examples of Online Techniques
  - Aggregation, visualization, cleaning/browsing
  - Underlying technology
  - Related work
  - Looking Forward

Online Aggregation

- SELECT AVG(temp) FROM t GROUP BY site
- 330K rows in table
- the exact answer:

Online Aggregation, cont'd

- A simple online aggregation interface (after 74 rows)

Online Aggregation, cont’d

- After 834 rows:
Online Data Visualization

In Tioga DataSplash

Online Enumeration

- Potter's Wheel [VLDB 2001]
- Scalable spreadsheet
  - A fraction of data is materialized in GUI widget
  - Scrolling = preference for data delivery in a quantile
- Interactive data cleaning
- Online structure and discrepancy detection
- Online aggregation

Online Aggregation

Example: Online Aggregation

Additional Features:
- Speed up
- Slow down
- Terminate
Road Map
- Background and motivation
- Examples of online techniques
- Underlying technology
  - Building blocks: sampling, estimation
  - Preferential data delivery
  - Pipelined adaptive processing algorithms
- Related work
- Looking forward

Sampling – Design Issues
- Granularity of sample
  - Instance-level (row-level): high I/O cost
  - Block-level (page-level): high variability from clustering
- Type of sample
  - Often simple random sample (SRS)
  - Especially for on-the-fly
  - With/without replacement usually not critical
- Data structure from which to sample
  - Files or relational tables
  - Indexes (B+ trees, etc)

Row-level Sampling Techniques
- Maintain file in random order
  - Sampling = scan
  - Is file initially in random order?
    - Statistical tests needed: e.g., Runs test, Smirnov test
  - In DB systems: cluster via RAND function
    - Must “freshen” ordering (online reorg)
- On-the-fly sampling
  - Via index on “random” column
  - Else get random page, then row within page
    - Ex: extent-map sampling
    - Problem: variable number of records on page

Acceptance/Rejection Sampling
- Accept row on page i with probability = \( \frac{i}{n_{\text{MAX}}} \)
- Commonly used in other settings
  - E.g. sampling from joins
  - E.g. sampling from indexes

Cost of Row-Level Sampling

Estimation for Aggregates
- Point estimates
  - Easy: SUM, COUNT, AVERAGE
  - Hard: MAX, MIN, quantiles, distinct values
- Confidence intervals – a measure of precision

Two cases: single-table and joins
Confidence Intervals

The Good and Bad News

Sampling Deployed in Industry

Precomputation Techniques

Road Map

Preferential Data Delivery

- "Simulated" Bernoulli sampling
  - SQL: SELECT * WHERE RAND() <= 0.01
  - Similar capability in SAS
- Bernoulli Sampling with pre-specified rate
  - Informix, Oracle 8i, (DB2)
  - Ex: SELECT * FROM T1 SAMPLE ROW(10%), T2
  - Ex: SELECT * FROM T1 SAMPLE BLOCK(10%), T2
- Not for novices
  - Need to pre-specify precision
  - recall the "multiresolution" patterns from example
  - No estimators provided in current systems

- Two components
  - Data reduction (often expensive)
  - Approximate reconstruction (quick)
- Pros and cons
  - Efficiency vs flexibility
  - Class of queries that can be handled
  - Degree of precision
  - Ease of implementation
  - How much of system must be modified
  - How sophisticated must developer be?
  - More widely deployed in industry
  - Will give overview later

Road Map

- Background and motivation
- Examples of online techniques
- Underlying technology
  - Building blocks: sampling, estimation
  - Preferential data delivery
  - Pipelined adaptive processing algorithms
- Related technology: precomputation
- Looking forward

Preferential Data Delivery

- Why needed
  - Speedup/slowdown arrows
  - Spreadsheet scrollbars
  - Pipeline quasi-sort
  - Continuous re-optimization (eddies)
- Index stride
  - High I/O costs, good for outliers
- Online Reordering ("juggle")
  - Excellent in most cases, no index required
  - [VLDB '99, VLDB '00]
Online Reordering

- Deliver "interesting" items first
- "Interesting" determined on the fly
- Exploit rate gap between produce and process/consume

Mechanism

- Two threads -- prefetch from input
- -- spool/enrich from auxiliary side disk
- Juggle data between buffer and side disk
  - keep buffer full of "interesting" items
  - getNext chooses best item currently on buffer
- Side disk management
  - hash index, populated in a way that postpones random I/O
  - play both sides of sort/hash duality

Policies

- "good" permutation of items \( t_1, \ldots, t_n \) to \( t_{P_1}, \ldots, t_{P_n} \)
- quality of feedback for a prefix \( t_{P_1}t_{P_2}\ldots t_{P_k} \):
  \[ \text{QOF}(UP(t_{P_1}), UP(t_{P_2}), \ldots, UP(t_{P_k})), \quad UP = \text{user preference} \]
- determined by application
- goodness of reordering: \( d\text{QOF} / dt \)
- implication for "juggle" mechanism
- process gets item from buffer that increases QOF the most
- juggle tries to maintain buffer with such items

QOF in Online Aggregation

- avg weighted confidence interval
  - preference acts as weight on confidence interval
    \[ \text{QOF} = \frac{\sum UP_i}{n_i^{1/2}}, \quad n_i = \text{number of tuples processed from group } i \]
  - process pulls items from group with max \( UP_i / n_i^{1/2} \)
    - desired ratio of group \( i \) in buffer = \( UP_i^{1/2} / \sum UP_j^{1/2} \)
    - juggle tries to maintain this by enrich/spool
- Similar derivations for other preferences
  - e.g. explicit rates, explicit ranking, etc.

Road Map

- Background and motivation
- Examples of online techniques
- Underlying technology
  - Building blocks: sampling, estimation, pre-computation
  - Preferential data delivery
  - Pipelined adaptive processing algorithms
- Related work
- Looking forward
Pipelined Data Processing

- Never, ever wait for anything to finish
- Selection: no problem
- Grouping: hash, don't sort
- Sorting: juggle if possible
- Joins?

Sample of joins vs. join of samples

```
SELECT AVG(R.a * S.b)
FROM R, S
WHERE R.c = S.c
```

Traditional Nested Loops

R

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

S

Ripple Joins [SIGMOD ’99]

- designed for online performance goals
- Completely pipelined
- Adapt to data characteristics
- designed for online performance goals
- simplest version
  - read new tuples \( s \) from \( S \) and \( r \) from \( R \)
  - join \( r \) and \( s \)
  - join \( r \) with old \( S \) tuples
  - join \( s \) with old \( R \) tuples

Basic Ripple Join

R

```
XXXX
XXX
XXX
```

S

Block Ripple Joins (Size = 2)

R

```
XXXXXX
XXXXXX
XXXXXX
```

S

Rectangular Ripple Join

R

```
XXXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXX
```

S
Ripple Joins, cont’d

- Variants:
  - Block: minimizes I/O in alternating nested loops
  - Index: coincides with index-nested loop
  - Hash: symmetric hash tables
  - Adaptive aspect ratio
    - User sets animation rate (via slider)
    - System solves optimization problem (approximately)
    - Samples from higher-variance relation faster

- System goal:
  - Minimize CI length
  - Subject to time constraint

- System solves optimization problem (approximately)
  - Samples from higher-variance relation faster

Prototypes in Informix, IBM DB2

- Ongoing work on scalability issues
  - Memory compaction technique
  - Parallelism
  - Graceful degradation to out-of-core hashing
    - A la Tukwila, Xjoin, but sensitive to statistical issues
  - Nested queries
  - Optimization issues
  - A number of API and other systems issues
    - DMKD journal paper on Informix implementation
    - Forthcoming paper on sampling in DB2

Road Map

- Background and motivation
- Examples of online techniques
- Underlying technology
- Related work
  - Online query processing
  - Precomputation
  - Looking forward

Related Work on Online QP

- Morgenstein’s PhD, Berkeley ’80
- Online Association Rules
  - Ng, et al’s CAP, SIGMOD ’88
  - Hidber’s CARMA, SIGMOD ’99
- Implications for deductive DB semantics
  - Monotone aggregation in LDL++, Zaroli and Weng
  - Online agg with subqueries
  - Yan, et al. VLDB ’99
  - Dynamic Pipeline Scheduling
  - Urban/Franklin VLDB ’01
  - Pipelining Hash joins
  - Raschid, Wijkstra/Opers, Tukwila, Xjoin
  - Relation to semi-naive evaluation
  - Anytime Algorithms
    - Zilberstein, Russell, et al.

Precomputation: Explicit

- OLAP Data Cubes (drill-down hierarchies)
  - MOLAP, ROLAP, HOLAP
- Semantic hierarchies
  - APPROXIMATE (Viensky, et al.)
  - Query Relaxation, e.g. CoBase
  - Multiresolution Data Models (Silberschatz/Reed/Fussell)
- More general materialized views
  - See Gupta/Mumick’s text

Precomputation: Stat. Summaries

- Histograms
  - Originally for aggregation queries, many flavors
  - Extended to enumeration queries recently
- Parametric estimation
  - Multi-dimensional histograms
- Wavelets and Fractals
- Discrete cosine transform
- Regression
- Curve fitting and splines
- Singular-Value Decomposition (aka LSI, PCA)
- Indexes: Hierarchical histograms
  - Ranking and pseudo-ranking
  - Aoki’s use of GISTs as estimators for ADTs
- Data Mining
  - Clustering, classification, other multidimensional models
Precomputed Samples

- Materialized sample views
- Olken’s original work
- Chaudhuri et al., join samples
- Statistical inferences complicated over “recycled” samples?
- Barbarà’s quasi-cubes
- AQUA “join synopses” on universal relation
- Maintenance issues
  - AQUA’s backing samples
- Can use fancier/more efficient sampling techniques
  - Stratified sampling or AQUA’s “congressional” samples
- Haas and Swami AVF statistics
  - Combine precomputed “outliers” with on-the-fly samples

Stratified Sampling

Road Map

- Background and motivation
- Examples of online techniques
- Underlying technology
- Related Work
- Looking forward
  - Adaptive systems
  - Human-centered systems

Looking Forward: Adaptive Systems

- Observation/Decision = Modeling/Prediction
  - usually statistical
- Already critically important in today’s systems
  - And imagine how important in ubiquitous computing!

A DBMS Tradition

- One instance: System R optimization
  - Observe: Runstats
  - Decide: Query Optimization
  - Act: Query Processing
- A powerful aspect of our technologies
  - Data independence & declarative languages
- Yet quite coarse-grained
  - Runstats once per day/week
  - Actions only per-query
    - Disk resource management: index and motive selection
    - Memory resource management: buffers and sort/hash space
    - Concurrency management: admission control

“Built-in” adaptivity

- Info systems should have adaptivity as a basic goal
  - Not just best-case performance
- Needs to pervade system
  - Core architectural work to be done here
    - E.g. pipelining required for multi-operator adaptivity
      - Observe more than one thing at a time
    - E.g. adaptive operators (a la ripple join)
    - E.g. adaptive optimization architectures (a la Eddies)
    - E.g. unify query processing with database design
- Adaptivity should be built-in, not “bolted-on”
  - Wizards to turn existing knobs
    - Less helpful
    - Certainly less elegant
    - Might be technically more difficult!
Looking Forward:
Human-Centered Systems

- Annual plea for UI work in DB Directions Workshops
  - UI’s perceived as “soft”, hard to measure/publish
- Yet people use our systems
  - And arguably we are trying to make them better for people
- Problem: our performance metrics
  - “Mirth index” vs. wall-clock time
  - One can find reasonable “hard” metrics for mirth
  - Many of these metrics may be statistical
  - Also consider “wise index”, e.g. in maintainability
  - Most of these indices have to do with user time
  - Not, e.g., resource utilization
- Good UI work need not require good UIs!
  - Can attack new metrics directly
  - We don’t have to go back to art school

Lessons Learned

- Dream about UIs, work on systems
  - User needs drive systems design!
- Systems and statistics intertwine
- All 3 go together naturally
  - User desires and behavior: 2 more things to model, predict
  - “Performance” metrics need to reflect key user needs

“What unlike things must meet and mate…”
-- Art, Herman Melville

More?

- Annotated bibliography & slides soon...
  http://control.cs.berkeley.edu/sigmod01/