One Size Fits All

• Different applications have very different data management requirements
• DBMS vendors try to sell DBMS as DM solution to almost every application
• DBMSs are too bloated – “elephants” -- for many of today’s applications
How would this app be implemented using DBMS technology?
Outline

• Background
  – Data Warehousing
  – Indexing
  – Views
  – Transactions
  – Stream Processing
  – High Availability
  – Row vs Column storage

• Financial-Feed Streaming Application (revisited)

• Other Applications

• Software Architectures
Data Warehousing

• Multi-dimensional Data Model
• Collection of numeric measures, which depend on a set of dimensions.
  – E.g., measure Sales, dimensions Product (key: pid), Location (locid), and Time (timeid).

ETL process periodically (nightly, weekly) loads new data into data warehouse

Extract, Transform, Load, Refresh

Metadata

DATA MINING

OLAP

Data Warehouse

Operational DBs
External Data Sources

DB
DB
DB
Conceptual Design of Data Warehouses

- Fact table in BCNF; dimension tables un-normalized.
  - Dimension tables are small; updates/inserts/deletes are rare. So, anomalies less important than query performance.
- This kind of schema is very common in OLAP applications, and is called a **star schema**; computing the join of all these relations is called a **star join**.
OLAP Queries

• Influenced by SQL and by spreadsheets.

• A common operation is to aggregate a measure over one or more dimensions.
  – Find total sales.
  – Find total sales for each city, or for each state.
  – Find top five products ranked by total sales.

• Roll-up: Aggregating at different levels of a dimension hierarchy.
  – E.g., Given total sales by city, we can roll-up to get sales by state.
More OLAP Queries

• **Drill-down:** The inverse of roll-up.
  – E.g., Given total sales by state, can drill-down to get total sales by city.
  – E.g., Can also drill-down on different dimension to get total sales by product for each state.

• **Pivoting:** Aggregation on selected dimensions.
  – E.g., Pivoting on Location and Time yields this **cross-tabulation**:

• **Slicing and Dicing:** Equality and range selections on one or more dimensions.
Comparison with SQL Queries

- The cross-tabulation obtained by pivoting can also be computed using a collection of SQL queries:

<table>
<thead>
<tr>
<th>Year\State</th>
<th>WI</th>
<th>CA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>63</td>
<td>81</td>
<td>144</td>
</tr>
<tr>
<td>1996</td>
<td>38</td>
<td>107</td>
<td>145</td>
</tr>
<tr>
<td>1997</td>
<td>75</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>176</strong></td>
<td><strong>223</strong></td>
<td><strong>339</strong></td>
</tr>
</tbody>
</table>

**SELECT SUM(S.sales)**
**FROM** Sales S, Times T, Locations L
**WHERE** S.timeid=T.timeid **AND** S.locid=L.locid
**GROUP BY** T.year, L.state

**SELECT SUM(S.sales)**
**FROM** Sales S, Times T
**WHERE** S.timeid=T.timeid
**GROUP BY** T.year

**SELECT SUM(S.sales)**
**FROM** Sales S, Location L
**WHERE** S.locid=L.locid
**GROUP BY** L.state
• A **view** is just a relation, but we store a *definition*, rather than a set of tuples.
• Views can be dropped using the **DROP VIEW** command.
• What if table that the view is dependent on is dropped?
  • **DROP TABLE** command has options to let the user specify this.
Querying Views

CREATE VIEW YoungActiveStudents (name, grade) AS
SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age<21

SELECT name
FROM YoungActiveStudents
WHERE grade = ‘A’

Conceptually, you can think of rewriting using a subquery

SELECT name
FROM (SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age<21)
WHERE grade = ‘A’

Query views as with any table
Materialized Views

CREATE VIEW ParamountMovies AS
SELECT title, year
FROM movies
WHERE studioName='Paramount'

CREATE TABLE ParamountMovies AS
(SELECT title, year
FROM movies
WHERE studioName='Paramount')

• Views can be “materialized” for efficiency
• Updating the materialized view (materialized query table in DB2) : incremental or batch

Queries on base relation may be able to exploit materialized views!

SELECT title
FROM movies
WHERE studioName='Paramount' AND year=1990)
Indexes

• An **index** on a file speeds up selections on the **search key fields** for the index.
  – Any subset of the fields of a relation can be the search key for an index on the relation.
  – *Search key* is *not* the same as *key* (minimal set of fields that uniquely identify a record in a relation).

• An index contains a collection of **data entries**, and supports efficient retrieval of all data entries \( k^* \) with a given key value \( k \).
  – A data entry is usually in the form \( <\text{key}, \text{rid}> \)
  – Given data entry \( k^* \), we can find record with key \( k \) in at most one disk I/O. (Details soon ...)
Leaf pages contain **data entries**, and are chained (prev & next).

A data entry typically contain a key value and a rid.

Non-leaf pages have **index entries**; only used to direct searches.
**Example B+ Tree**

- Find 28*? 29*? All > 15* and < 30*
- Insert/delete: Find data entry in leaf, then change it. Need to adjust parent sometimes.
  - And change sometimes bubbles up the tree.

Note how data entries in leaf level are sorted.
Bitmap Indexes

Boats Relation

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Interlake</td>
<td>Blue</td>
</tr>
<tr>
<td>102</td>
<td>Interlake</td>
<td>Red</td>
</tr>
<tr>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>104</td>
<td>Marine</td>
<td>Red</td>
</tr>
</tbody>
</table>

Bitmap index for color

<table>
<thead>
<tr>
<th>Blue</th>
<th>Red</th>
<th>Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- One bit vector for each distinct column value
- Length of bit vector is the cardinality of the relation instance
- Logical bitwise operations used to answer queries
- Bit vectors can be encoded and compressed
Stream Processing

• Continuous, unbounded, rapid, time-varying streams of data elements (tuples).

• Examples of streaming applications
  – Network monitoring and traffic engineering, Sensor networks, RFID tags, Telecom call records, Financial applications, Web logs and click-streams, Manufacturing processes

• DSMS = Data Stream Management System

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Streaming System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent relations</td>
<td>Transient Streams (&amp; relations)</td>
</tr>
<tr>
<td>One time queries</td>
<td>Continuous Queries</td>
</tr>
<tr>
<td>Random Access</td>
<td>Sequential Access</td>
</tr>
<tr>
<td>Access plan determined by DBMS</td>
<td>Unpredictable data characteristics</td>
</tr>
</tbody>
</table>
Pull vs Push

**Pull**
- updates ↓
- pull processing
- results ⇧
- storage

**Push**
- input streams
- push processing
- results
- optional storage
- optional archive access

---

1/9/2013
Lipyeow Lim -- University of Hawaii at Manoa
Process Model

• An **Operating System Process** combines an operating system (OS) program execution unit (a thread of control) with an address space private to the process. This single unit of program execution is scheduled by the OS kernel and each process has its own unique address space.
Transactions

- Atomicity
- Isolation
- Consistency
- Durability

- Write Ahead Log
- Locking Protocols
- User enforced
The Log

• The following actions are recorded in the log:
  – *Ti writes an object:* the old value and the new value.
    • Log record must go to disk *before* the changed page! (Write Ahead Log property)
  – *Ti commits/aborts:* a log record indicating this action.

• Log records are chained together by Xact id, so it’s easy to undo a specific Xact.

• Log is often *duplexed* and *archived* on stable storage.

• All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.
Recovering from a Crash

- There are 3 phases in the *Aries* recovery algorithm:
  - **Analysis:** Scan the log forward (from the most recent checkpoint) to identify all Xacts that were active, and all dirty pages in the buffer pool at the time of the crash.
  - **Redo:** Redoes all updates to dirty pages in the buffer pool, as needed, to ensure that all logged updates are in fact carried out and written to disk.
  - **Undo:** The writes of all Xacts that were active at the crash are undone (by restoring the *before value* of the update, which is in the log record for the update), working backwards in the log. (Some care must be taken to handle the case of a crash occurring during the recovery process!)
Lock-based Concurrency Control

• **Strict Two-phase Locking (Strict 2PL) Protocol:**
  – Each Xact must obtain a *S (shared)* lock on object before reading, and an *X (exclusive)* lock on object before writing.
  – All locks held by a transaction are released when the transaction completes.
  – If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.

• Strict 2PL allows only serializable schedules.
  – Additionally, it simplifies transaction aborts.
A database system is highly available (HA) if it remains accessible to users in the face of hardware failures.

Transaction logging techniques already provides crash recovery

HA requires close to zero down time.
Row vs Column Storage

**Column Store:**

- IBM: 60.25
- MSFT: 60.53

Used in: Sybase IQ, Vertica

**Row Store:**

- IBM: 60.25
- MSFT: 60.53

Used in: Oracle, SQL Server, DB2, Netezza, ...
Financial-Feed Streaming Application

How would this app be implemented using DBMS technology?
Other Applications

• Data Warehouse.
• **Sensor Networks**. eg. Medical monitoring, smart homes, etc
• **Text Search**. eg. Search engines
• **Scientific Databases**. Eg. Astronomy, Meteorology, Oceanography
• XML
Software Architectures

- Three basic services
  - Message Transport
  - Storage of state
  - Execution of application logic