Overview and History of Databases and Transactions

CSE 305 – Principles of Database Systems
Paul Fodor
Stony Brook University

http://www.cs.stonybrook.edu/~cse305
Introduction

• What is a Database?
Introduction

What is a Database?
- Collection of data central to some enterprise
- Essential to operation of enterprise
  - Contains the only record of enterprise activity
- An asset in its own right
  - Historical data can guide enterprise strategy
  - Of interest to other enterprises
- State of database mirrors state of enterprise
  - Database is persistent
Introduction

- What is a Database Management System?
Introduction

• What is a Database Management System?
• DBMS is a program that manages a database:
  • Supports a high-level access language (e.g. SQL).
  • Application describes database accesses using that language.
  • DBMS interprets statements of language to perform requested database access.
Introduction

• What is a Transaction?
Introduction

• What is a Transaction?
  • When an event in the real world changes the state of the enterprise, a "transaction" is executed to cause the corresponding change in the database state
  • With an on-line database, the event causes the transaction to be executed in real time
  • A transaction is an application program with special properties (i.e., ACID=Atomicity, Consistency, Isolation, Durability) to guarantee it maintains database correctness
Introduction

• What is a Transaction Processing System?
  • Transaction execution is controlled by a Transaction Processing (TP) monitor
  • Creates the abstraction of a transaction, analogous to the way an operating system creates the abstraction of a process
  • TP monitor and DBMS together guarantee the special properties of transactions
  • A Transaction Processing System consists of TP monitor, possibly multiple databases, and transactions
Introduction

Transaction Processing System

TP Monitor

transactions

DBMS

database

database
Introduction

• Database Systems Requirements:
  • **High Availability**: on-line => must be operational while enterprise is functioning
  • **High Reliability**: correctly tracks state, does not lose data, controlled concurrency
  • **High Throughput**: many users => many transactions/sec
  • **Low Response Time**: on-line => users are waiting
Introduction

• Database Systems Requirements:
  • Long Lifetime: complex systems are not easily replaced
    • Must be designed so they can be easily extended as the needs of the enterprise change
  • Security: sensitive information must be carefully protected since system is accessible to many users
    • Authentication, authorization, encryption
Introduction

• Roles in Design, Implementation, and Maintenance of a TPS:
  • **System Analyst** - specifies system using input from customer; provides complete description of functionality from customer’s and user’s point of view
  • **Database Designer** - specifies structure of data that will be stored in database
  • **Application Programmer** - implements application programs (transactions) that access data and support enterprise rules
Introduction

- Roles in Design, Implementation, and Maintenance of a TPS:
  - **Database Administrator** - maintains database once system is operational: space allocation, performance optimization, database security
  - **System Administrator** - maintains transaction processing system: monitors interconnection of HW and SW modules, deals with failures and congestion
Introduction

- **On-line Transaction Processing (OLTP)**
  - Day-to-day handling of transactions that result from enterprise operation
  - Maintains correspondence between database state and enterprise state

- **On-line Analytic Processing (OLAP)**
  - Analysis of information in a database for the purpose of making management decisions
Introduction

• **On-line Analytic Processing (OLAP):**
  - Analyzes historical data (terabytes) using complex queries
  - **Summarizes the data and makes forecasts!**
  - Example: it answers operational questions like “What are the average sales of cars, by region and by year?”
  - Due to volume of data and complexity of queries, OLAP often uses a data warehouse and mining

• **Data Warehouse** - (offline) repository of historical data generated from OLTP or other sources

• **Data Mining** - use of warehouse data to *discover* relationships (discovers hidden patterns in data) that might influence enterprise strategy
Introduction

• Example: Supermarket:
  • OLTP
    • For the event of buying 1 milk and 1 box of diapers, the OLTP will **update** the database to reflect that event
  • OLAP
    • Last winter in all stores in northeast, how many customers bought milk and diapers together?
  • Data Mining
    • Are there any interesting combinations of products that customers frequently bought together?
A Brief History of Database Systems

• Pre-relational era (1970’s)
  • Hierarchical (IMS), Network (Codasyl)
  • Complex data structures and low-level query language

• Relational DBMSs (1980s)
  • Edgar F. Codd's relational model in 1970
  • Set of tuples (i.e., tables) as data model
  • Powerful high-level query language

• Object-Oriented DBMSs (1990s)
  • Motivated by “impedance mismatch” between RDBMS and OO PL
  • Persistent types in C++, Java or Small Talk
  • Issues: Lack of high level QL, no standards, performance
A Brief History of Database Systems

- **Object-relational DBMS (OR-DBMS) (1990s)**
  - Relational DBMS vendors’ answer to OO
  - User-defined types, functions (spatial, multimedia)
  - Nested tables
- **XML/DBMS (2000s)**
  - Web and XML are merging
  - Native support of XML through ORDBMS extension or native XML DBMS
- **Decision support system (DSS) (2000s)**
  - Data warehousing and OLAP
A Brief History of Database Systems

• Data stream management systems (2000s)
  • Continuous query against data streams

• The era of big data (mid 2000-now):
  • Big data: datasets that grow so large (terabytes to petabytes) that they become awkward to work with traditional DBMS
  • Parallel DBMSs continue to push the scale of data
  • MapReduce dominates on Web data analysis
  • “NoSQL” (not only SQL) is fast growing
Stay updated

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.</td>
<td>1.</td>
<td>1.</td>
<td>Oracle</td>
<td>Relational DBMS</td>
<td>1427.72</td>
<td>-13.81</td>
<td>-25.30</td>
</tr>
<tr>
<td>2</td>
<td>2.</td>
<td>2.</td>
<td></td>
<td>MySQL</td>
<td>Relational DBMS</td>
<td>1357.03</td>
<td>-6.25</td>
<td>+65.00</td>
</tr>
<tr>
<td>3</td>
<td>3.</td>
<td>3.</td>
<td></td>
<td>Microsoft SQL Server</td>
<td>Relational DBMS</td>
<td>1205.04</td>
<td>+12.16</td>
<td>+96.39</td>
</tr>
<tr>
<td>4</td>
<td>4.</td>
<td>4.</td>
<td></td>
<td>MongoDB</td>
<td>Document store</td>
<td>318.49</td>
<td>+3.49</td>
<td>+23.84</td>
</tr>
<tr>
<td>5</td>
<td>5.</td>
<td>5.</td>
<td></td>
<td>PostgreSQL</td>
<td>Relational DBMS</td>
<td>315.25</td>
<td>+4.10</td>
<td>+33.39</td>
</tr>
<tr>
<td>6</td>
<td>6.</td>
<td>6.</td>
<td></td>
<td>DB2</td>
<td>Relational DBMS</td>
<td>185.89</td>
<td>+0.81</td>
<td>-15.35</td>
</tr>
<tr>
<td>7</td>
<td>7.</td>
<td></td>
<td>+8.</td>
<td>Cassandra</td>
<td>Wide column store</td>
<td>130.24</td>
<td>-0.47</td>
<td>+16.24</td>
</tr>
<tr>
<td>8</td>
<td>8.</td>
<td></td>
<td>-7.</td>
<td>Microsoft Access</td>
<td>Relational DBMS</td>
<td>124.05</td>
<td>-0.85</td>
<td>-20.15</td>
</tr>
<tr>
<td>9</td>
<td>9.</td>
<td></td>
<td></td>
<td>SQLite</td>
<td>Relational DBMS</td>
<td>109.86</td>
<td>+1.32</td>
<td>+4.04</td>
</tr>
<tr>
<td>10</td>
<td>10.</td>
<td></td>
<td></td>
<td>Redis</td>
<td>Key-value store</td>
<td>107.32</td>
<td>-0.71</td>
<td>+8.51</td>
</tr>
<tr>
<td>12</td>
<td>12.</td>
<td></td>
<td>+13.</td>
<td>Teradata</td>
<td>Relational DBMS</td>
<td>73.64</td>
<td>-0.29</td>
<td>+0.05</td>
</tr>
<tr>
<td>13</td>
<td>13.</td>
<td></td>
<td>-11.</td>
<td>SAP Adaptive Server</td>
<td>Relational DBMS</td>
<td>71.04</td>
<td>+0.31</td>
<td>-14.07</td>
</tr>
<tr>
<td>14</td>
<td>14.</td>
<td></td>
<td>-12.</td>
<td>Solr</td>
<td>Search engine</td>
<td>65.77</td>
<td>+1.08</td>
<td>-16.13</td>
</tr>
<tr>
<td>15</td>
<td>15.</td>
<td></td>
<td></td>
<td>HBase</td>
<td>Wide column store</td>
<td>55.51</td>
<td>+2.37</td>
<td>-4.43</td>
</tr>
<tr>
<td>16</td>
<td>16.</td>
<td></td>
<td>+17.</td>
<td>FileMaker</td>
<td>Relational DBMS</td>
<td>55.01</td>
<td>+3.45</td>
<td>+3.14</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>+18.</td>
<td>+18.</td>
<td>Splunk</td>
<td>Search engine</td>
<td>48.90</td>
<td>+2.26</td>
<td>+6.71</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>-17.</td>
<td>-16.</td>
<td>Hive</td>
<td>Relational DBMS</td>
<td>47.82</td>
<td>+0.27</td>
<td>-6.06</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>-19.</td>
<td>SAP HANA</td>
<td>Relational DBMS</td>
<td>42.73</td>
<td>+0.93</td>
<td>+4.48</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>+25.</td>
<td></td>
<td>MarlaDB</td>
<td>Relational DBMS</td>
<td>36.88</td>
<td>+1.08</td>
<td>+12.76</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>+22.</td>
<td></td>
<td>Neo4j</td>
<td>Graph DBMS</td>
<td>35.57</td>
<td>+1.88</td>
<td>+2.41</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>-20.</td>
<td></td>
<td>Informix</td>
<td>Relational DBMS</td>
<td>29.05</td>
<td>+0.49</td>
<td>-7.75</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>-21.</td>
<td></td>
<td>Memcached</td>
<td>Key-value store</td>
<td>27.69</td>
<td>+0.50</td>
<td>-5.69</td>
</tr>
</tbody>
</table>

http://db-engines.com/en/ranking

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
Stay updated

August 2016

DB-Engines Ranking

http://db-engines.com/en/ranking

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
A Brief History of DBMS Products

- First hierarchy DBMS: IBM Information Management System (IMS)
  - starting in 1966 for the Apollo program
  - Still going strong over 40 years later
  - Mainframe only

- IDMS (Integrated Database Management System) is a network model based system
  - The roots of IDMS go back to Dr. Charles Bachman's IDS (Integrated Data Store) developed at GE
  - Since 1989 the product has been owned by Computer Associates, who renamed it as CA-IDMS
  - Mainframe only
A Brief History of DBMS Products

• Two early RDBMS projects started and were operational in late 1970s: INGRES and System R

• INGRES (INteractive Graphics REtrieval System) started at UC Berkeley, by Michael Stonebraker and Eugene Wong
  • In the early 1980s, Ingres competed head-to-head with Oracle, but lost market due to Oracle’s marketing and Ingres’ own proprietary QUEL
  • Since the mid-1980s, Ingres has spawned into: Sybase, Microsoft SQL Server, NonStop SQL, etc

• Postgres (Post Ingres) started in the mid-1980s, later evolved into PostgreSQL

• In the 1990s Stonebraker commercialized Postgres as Illustra, later sold to Informix (sold to IBM in 2001)
A Brief History of DBMS Products

- **IBM System R** was a research project at IBM San Jose Research (now IBM Almaden Research) in the 1970s.
- **SQL/DS** was IBM's first commercial DBMS for mainframe built around SQL in early 1980s.
- A little later, in 1983, IBM released **DB2** on its MVS mainframe platform.
- IBM brought DB2 to other platforms (LUW) in 90s. DB2 renamed as **DB2 UDB z/OS, DB2 UDB LUW**.
- Larry Ellison and his friends started **Software Development Laboratories (SDL)** in 1977, which developed the original version of **Oracle**.
  - The name **Oracle** comes from the code-name of a CIA-funded project Ellison had worked before.
SQL

- SQL: Structured Query Language Invented in 1974 by Donald Chamberlin and Raymond Boyce for IBM
  - Initially called SEQUEL, changed to SQL due to trademark issue
  - In late 1970s, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL in Oracle V2
  - Multiple standard revisions and multiple flavors (implementations) exist
SQL Standard Revisions

- SEQUEL/Original SQL - 1974
- SQL86: ratification and acceptance of a formal SQL standard by ANSI and ISO
- SQL2 (a.k.a. SQL92): still strictly relational, with new primitive data types, operations and join types
- SQL3: working documents discussing new specs for OR systems, but also for recursion, active rules, OLAP
- SQL:1999: added user defined types, etc
- SQL:2003: added XML-related features, etc
- SQL:2006: increased support for XML support for XQuery, an XML-SQL interface standard
- SQL:2011: added temporal support

And evolution continues…
# NoSQL Systems

<table>
<thead>
<tr>
<th>Category</th>
<th>Data Model</th>
<th>Example Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-Value</td>
<td>(Global) collection of K-V pairs</td>
<td>BerkeleyDB, LevelDB, Memcached, Project Voldemort, Redis, Riak</td>
</tr>
<tr>
<td>Column Families</td>
<td>Big table, column families</td>
<td>Amazon SimpleDB, Cassandra, HBase, Hypertable</td>
</tr>
<tr>
<td>Document</td>
<td>Collections of K-V Collections</td>
<td>CouchDB, MongoDB, OrientDB, RavenDB, Terrastore</td>
</tr>
<tr>
<td>Graph</td>
<td>Nodes, relations, K-V on both</td>
<td>Apache Tinkerpop, FlockDB, HerperGraphDB, Infinite Graph, AllegroGraph, Neo4j, OrientDB</td>
</tr>
<tr>
<td>Search engines</td>
<td>Inverted indexes, tries, Information retrieval</td>
<td>Apache Lucene, Apache Solr, Elasticsearch</td>
</tr>
</tbody>
</table>
Distributed DBMS: CAP

Partition Tolerance

Consistency

Availability

RDBMS

NoSQL (most)
More on evolution of DBMS

• Hierarchical model (~1968):
  • record types arranged as a hierarchy
  • each type has a single parent

```
"Type Hierarchy" (Schema)

Supplier
(sno, sname, scity, sstate)

Part
(pno, pname, psize, pcolor, qty, price)

Sample Instances

16, General Supply, Boston, MA

27, Power Saw, 7, silver, 100, $20
(Each record has a key)
```
More on evolution of DBMS

- Hierarchical model (~1968):
  - some problems:
    - Information repeated:
      - Schema #1: part info repeated for each supplier that supplies the part
      - Schema #2: supplier info repeated for each part

![Diagram of Hierarchical Model](image)
More on evolution of DBMS

- Hierarchical model (~1968):
  - some problems:
    - Existence depends on parent data
    - Schema#1: what if there is a part not currently supplied by anyone?
More on evolution of DBMS

- Hierarchical model (~1968):
  - DL/1 programming language for IMS: "record-at-a-time" language: the programmer constructs an algorithm for solving a query and IMS executes it

```plaintext
Supplier
  (sno, sname, scity, sstate)

Part
  (pno, pname, psize, pcolor, qty, price)

Find red parts supplied by Supplier 16

Get unique Supplier (sno = 16)
Until no-more {
  Get next within parent (color = red)
}

Until no-more {
  Get next Part (color = red)
}
```
More on evolution of DBMS

- Hierarchical model (~1968):
  - Different underlying storage = different restrictions on commands: heavy coupling between storage format used (sequential/B-tree/hashed) and client application
  - Different sets of data = different optimization opportunities
    - even if the optimization is programmed by the programmer
More on evolution of DBMS

- **Data Independence:**
  - A Simple Idea: Applications should be insulated from how data is structured and stored
  - **Logical data independence:** protection from changes in the logical structure of data.
  - **Physical data independence:** protection from changes in the physical structure of data.
More on evolution of DBMS

• Logical data independence:
  • changes to physical/logical structure should not require changes at the application level (ideally)
  • in general, should not require expensive changes to apps
• Impossible to achieve in the hierarchical model, where:
  • trees are difficult to reorganize
  • the record-at-a-time language delegates the optimization to the programmer
More on evolution of DBMS

- Graph / Network model (CODASYL 1969):
  - Schema arranged in a graph model

```
Supplier (sno, sname, scity, sstate)
```

```
Part (pno, pname, psize, pcolor)
```

```
Supply(qty, price)
```

```
Supplies
```

```
Supplied_by
```
More on evolution of DBMS

- Graph / Network model (CODASYL 1969):
  - Instances

![Diagram showing instances and relationships in a graph/network model.]

(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)
More on evolution of DBMS

- **Graph / Network model (CODASYL 1969):**
  - **Improvement:**
    - entities can exist without their parents
  - **Limitations:**
    - still using the record-at-a-time DML language
    - still no physical independence
    - more difficult to program against a graph than a tree
    - graphs are more complex: the whole graph must be loaded at once (IMS trees could be loaded individually)
More on evolution of DBMS

• Relational model (1970)
  • Ted Codd was motivated by the heavy maintenance required by the IMS applications
  • data stored in tables (see next class)
• High level, set oriented DML
• underlying physical storage is up to vendors
More on evolution of DBMS

- Entity Relationship (mid 1970s)
  - Proposed by Peter Chen
  - Relationships with attributes and multiplicities

As a physical model: never caught on (little benefit)
As a conceptual model: widely used for database schema design because it offers a methodology for creating initial tables and some normalization on E-R models can be done automatically
More on evolution of DBMS

- Semantic data model (early 1980s)
  - View relations as classes
    - multiple inheritance
    - class-wide attributes
  - Vendors were more concerned with performance
  - Can be simulated with relational
More on evolution of DBMS

- OO DBs (mid 1980s)
  - Integrate data persistency into OO programming languages
    - extend a OO programming language (e.g., C++) with database functionality to support data persistence
    - initial work targeted towards engineering niche market (e.g., CAD)

```cpp
Persistent part p;
Persistent int i;
```

- Did not go because vendors did not want change!
More on evolution of DBMS

- Object-Relational DBs (mid 1980s)
  - motivated by spatial queries: INGRES team had a "haunting" interest in GIS (geographical information system)
    - B-trees are inefficient to solve such queries
    - User defined data types (box) and operators (box intersects box, R-tree indexing)
  - Major prototype: Postgres showed how to build a DBMS engine so new types and functions can be plugged in
  - Also Sybase contributed with stored procedures: user defined functions for application logic, not just operators
  - Postgres was commercialized by Illustra (acquired by Informix)
More on evolution of DBMS

- Semi-structured era (~2000+)
  - Schema Evolution / Schema "later": data is self describing
  
  ```
  Person:
  Name: Joe Jones
  Wages: 14.75
  Employer: My_accounting
  Hobbies: skiing, bicycling
  Works for: ref (Fred Smith)
  Favorite joke: Why did the chicken cross the road? To get to the other side
  Office number: 247
  Major skill: accountant
  End Person
  
  Person:
  Name: Smith, Vanessa
  Wages: 2000
  Favorite coffee: Arabian
  Pastimes: sewing, swimming
  Works_for: Between jobs
  Favorite restaurant: Panera
  Number of children: 3
  End Person:
  ```
  
  Semantic Heterogeneity:
  * Different sets of attributes
  * Same attributes have different formats
  * Different attributes have same meaning

- Complex graph oriented data models

- Also, a Response to the growth of Web services and XML as a language (same for JSON as Javascript)
More on evolution of DBMS

• Semi-structured era (~2000+)
  • Schema Evolution / Schema "later": data is self describing
    • Relational DBMS have heavy-weight mechanisms to change schema (ALTER)
    • XML and JSON as a data model:
      • records can be hierarchical,
      • records can reference to other records
      • schema can be defined "later" in DTDs and XMLSchema
      • XQuery is essentially an Object-Relational SQL
      • OR DBMSs adapted to support XML
Finally

Stop following me!

http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques