Relational Database Systems 2
1. System Architecture

Christoph Lofi
Benjamin Köhncke
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de
Organizational Information

• **Lecture**
  – 07. April 2011 – 14. July 2011, 15:00h-17:15h
  – We will interleave lectures and exercises

• **5 Credits**

• **Oral Exams**

• [http://www.ifis.cs.tu-bs.de/teaching/ss-11/rdb2](http://www.ifis.cs.tu-bs.de/teaching/ss-11/rdb2)
• Scope of the lecture

- Architecture of RDBs
- Storage Hardware
- Indexing
  - Basic Index Structures
  - Tree Index Structures
- Query Processing
  - Basics of QP (Algebraic)
  - Query Optimization (Heuristical)
  - Query Optimization (Statistical & Join-Order)
- Transaction Management
  - Locking Protocols
  - Alternative Protocols
- Data Protection
  - Recovery & Durability
  - Security
- Trends & Alternatives
  - Alternative Implementations
  - Alternative Applications
Recommended Literature

• **Fundamentals of Database Systems (EN)**
  – Elmasri & Navathe
  – Addison Wesley, ISBN 032141506X

• **Database Systems Concepts (SKS)**
  – Silberschatz, Korth & Sudarshan

• **Database Systems (GUW)**
  – Garcia-Molina, Ullman & Widom
  – Prentice Hall, ISBN 0130319953

• **Datenbanksysteme (KE)**
  – Kemper & Eickler
  – Oldenbourg, ISBN 3486576909
Recommended Literature

- **Transactional Information Systems (WV)**
  - Weikum & Vossen
  - Morgan Kaufmann, ISBN 1558605088

- **Transaction Processing (GR)**
  - Gray & Reuter
  - Morgan Kaufmann, ISBN 1558601902

- **Database Security (CFMS)**
  - Castano, Fugini, Martella & Samarati
  - Addison Wesley, ISBN 0201593750
1. System Architecture

1.1 Characteristics of Databases

1.2 Data-Models and Schemas
   - Data Independence
   - Three Schema Architecture
   - System Catalogs

1.3 System Structure

1.4 Quality Benchmarks
1.1 What is a Database?

- A database (DB) is a collection of related data
  - Represents some aspects of the real world
    - Universe of Discourse (UoD)
  - Data is logically coherent
  - Is provided for an intended group of users and applications
- A database management system (DBMS) is a collection of programs to maintain a database, i.e. for
  - Definition of Data and Structure
  - Physical Construction
  - Manipulation
  - Sharing/Protecting
  - Persistence/Recovery
1.1 Example

- Classic Example: Banking Systems
  - DBMS used in banking since ca. 1960
  - Huge amounts of data on customers, accounts, loans, balances,…
• File management systems are **physical** interfaces
1.1 File Systems

• Advantages
  – Fast and easy access

• Disadvantages
  – Uncontrolled redundancy
  – Inconsistent data
  – Limited data sharing and access rights
  – Poor enforcement of standards
  – Excessive data and access paths maintenance
1.1 Databases

• Databases are **logical** interfaces
  – Controlled redundancy
  – Data consistency & integrity constraints
  – Integration of data
  – Effective and secure data sharing
  – Backup and recovery

• However…
  – More complex
  – More expensive data access
I.1 Example

• **DBMS** replaced previously dominant file-based systems in **banking** due to special requirements
  
  – Simultaneous and quick access is necessary
  
  – Failures and loss of data **cannot** be tolerated
  
  – Data always has to remain in a **consistent** state
  
  – Frequent queries and modifications
1.1 Characteristics of Databases

• Databases control redundancy
  – Same data used by different applications/tasks is only stored once
  – Access via a single interface provided by DBMS
  – Redundancy only purposefully used to speed up data access (e.g. materialized views)

• Problems of uncontrolled redundancy
  – Difficulties in consistently updating data
  – Waste of storage space
1.1 Characteristics of Databases

- Databases are **well-structured** (e.g. ER-Model)
  - **Catalog** (data dictionary) contains all **meta-data**
  - Defines the **structure** of the data in the database

- Example: ER-Model
  - Simple banking system

```
ID

firstname

lastname

address

customer

has

account

AccNo

type

balance

AccNo

has

```
1.1 Characteristics of Databases

- Databases aim at **efficient** manipulation of data
  - Physical tuning allows for good data allocation
  - Indexes speed up search and access
  - Query plans are optimized for improved performance

- Example: Simple Index

<table>
<thead>
<tr>
<th>AccNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1278945</td>
<td>saving</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>saving</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
<td>€ -43.03</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
<td>€ -345.17</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>
• **Isolation** between applications and data

  – Database employs **data abstraction** by providing **data models**
  
  – Applications work only on the **conceptual representation** of data
    
    • Data is strictly **typed** (Integer, Timestamp, VarChar, …)
    
    • Details on where data is actually stored and how it is accessed is **hidden** by the DBMS
    
    • Applications can access and manipulate data by invoking **abstract operations** (e.g. SQL Select statements)

  – DBMS-controlled parts of the file system are **strongly protected** against outside manipulation (tablespaces)
• **Example:** Schema is changed and table-space moved without an application noticing

```sql
SELECT AccNo FROM account WHERE balance > 0
```
• **Example:** Schema is changed and table-space moved without an application noticing

\[
\text{SELECT AccNo FROM account WHERE balance}>0
\]
1.1 Characteristics of Databases

• Supports multiple **views** of the data
  – Views provide a different perspective of the DB
    • A user’s conceptual understanding or task-based excerpt of all data (e.g. aggregations)
    • Security considerations and access control (e.g. projections)
  – For the application, a view does not differ from a table
  – Views may contain **subsets** of a DB and/or contain **virtual data**
    • Virtual data is **derived** from the DB (mostly by simple SQL statements, e.g. joins over several tables)
    • Can either be computed at query time or **materialized** upfront
### 1.1 Characteristics of Databases

**Example Views: Projection**

- Saving account clerk vs. checking account clerk

<table>
<thead>
<tr>
<th>AccNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1278945</td>
<td>saving</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>saving</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>4543032</td>
<td>checking</td>
<td>-€ 43.03</td>
</tr>
<tr>
<td>5539783</td>
<td>saving</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>7809849</td>
<td>checking</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>checking</td>
<td>-€ 345.17</td>
</tr>
<tr>
<td>9134354</td>
<td>saving</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>saving</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>

**Saving View**

<table>
<thead>
<tr>
<th>AccNo</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1278945</td>
<td>€ 312.10</td>
</tr>
<tr>
<td>2437954</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td>5539783</td>
<td>€ 12.54</td>
</tr>
<tr>
<td>9134354</td>
<td>€ 2.22</td>
</tr>
<tr>
<td>9543252</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>

**Checking View**

<table>
<thead>
<tr>
<th>AccNo</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>4543032</td>
<td>-€ 43.03</td>
</tr>
<tr>
<td>7809849</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td>8942214</td>
<td>-€ 345.17</td>
</tr>
</tbody>
</table>
1.1 Characteristics of Databases

- **Sharing** of data and support for **atomic multi-user transactions**
  - Multiple user and applications may access the DB at the same time
  - **Concurrency control** is necessary for maintaining consistency
  - Transactions need to be **atomic** and **isolated** from each other
1.1 Characteristics of Databases

• Example: Atomic Transactions

  – Program:
    Transfer X Euro from Account 1 to Account 2
    1. Deduce amount X from Account 1
    2. Add amount X to Account 2
• Example: Atomic Transactions
  – Program:
    Transfer X Euro from Account 1 to Account 2
    1. Deduce amount X from Account 1
    2. Add amount X to Account 2
  – But what happens if system fails between step 1 and 2?
• **Example: Multi-User Transactions**
  
  – **Program: Deduce amount X from Account 1**
    1. Read old balance from DB
    3. Write new balance back to the DB
  
  – **Problem: Dirty Read**
    • Account 1 has €500
    • User 1 wants to deduce €20
    • User 2 wants to deduce €80 **at the same time**
  
  – Without multi-user transaction, account will have either €480 or €420, but not the correct €400
1.1 Characteristics of Databases

• **Persistence** of data and disaster **recovery**
  – Data needs to be persistent and accessible at all times
  – Quick recovery from system crashes **without data loss**
  – Recovery from natural disasters (fire, earthquakes,…)
• What concepts does a DBMS need and how do you actually implement the concepts to build a DBMS?

– Basic concepts
– Query processing and optimization
– Transaction concept and implementing concurrent usage
– Logging and recovery concepts
– Implementing access control
1.2 Data Models

• **A Data Model** describes data objects, operations and their effects

• **Data Definition Language (DDL)**
  – Create Table, Create View, Constraint/Check, etc.

• **Data Manipulation Language (DML)**
  – Select, Insert, Delete, Update, etc.

  – **DML** and **DDL** are usually clearly separated, since they handle **data** and **meta-data**, respectively
1.2 Data Models

- Conceptual Data Models
  - **ER Model**
  - Semantic Data Models
  - **UML class diagrams**

- Logical Data Models
  - Model Types
    - **Relational Data Model** (in this lecture)
    - Network Models
    - Object Models
  - **Schema** describing
    - Structure
    - High Level Operations

- Physical Data Models
  - Describes how data is stored, i.e. formats, ordering and **access paths** like tablespaces or indexes
1.2 DBMS Meta-Data Environments

• Schemas
  – Describe a part of the structure of the stored data as tables, attributes, views, constraints, relationships, etc. (Meta-Data)

• System Catalogs
  – A collection of schemas
  – Contain special schemas describing the schema collection

• Clusters (optionally)
  – A collection of catalogs
  – May be individually defined for each user (access control)
  – Represent the maximal query scope
1.2 DBMS Meta-Data Environments

DBMS Environment

Cluster = Max. Query Scope

Catalog
Schema
Schema
Catalog
Schema
Schema
Catalog
Schema
Schema
Catalog
Schema
Schema
1.2 Meta-Data - Example

- **DBMS:** IBM DB2 V9
- **Catalog:** HORIZON
- **Example Meta-Data View:** SYSIBM.TABLES

  -- Describes all tables of the catalog

<table>
<thead>
<tr>
<th>TABLE_CATALOG</th>
<th>TABLE_SCHEMA</th>
<th>TABLE_NAME</th>
<th>TABLE_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSUSEROPTIONS</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSURAPROPTIONS</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSCOLGROUPDIST</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSSXSOBJECTHIERARCHIES</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSTUNINGINFO</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSCONTEXTATTRIBUTES</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSVARIABLES</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>SYSSERVICECLASSES</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSTOOLS</td>
<td>POLICY</td>
<td>BASE TABLE</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSIBM</td>
<td>ROUTINES_S</td>
<td>VIEW</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSCAT</td>
<td>COlGROUPDIST</td>
<td>VIEW</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSCAT</td>
<td>EVENTS</td>
<td>VIEW</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSCAT</td>
<td>FUNCMAPPINGS</td>
<td>VIEW</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSCAT</td>
<td>PACKAGEDEP</td>
<td>VIEW</td>
</tr>
<tr>
<td>HORIZON</td>
<td>SYSCAT</td>
<td>PROCPARAMS</td>
<td>VIEW</td>
</tr>
</tbody>
</table>
1.2 Schemas and Instances

- Schemas describe the **structure** of part of the DB data (*intensional database*)
  - Entities (a real world concept) as **tables** and their **attributes** (a property of an entity)
  - **Types** of attributes and **integrity constraints**
  - **Relationships** between entities as tables
  - Schemas are intended to be **stable** and not change often

- **Basic operations**
  - Operations for selections, insertions and updates
  - Optionally **user defined operations** (User Defined Functions (UDFs), stored procedures) and **types** (UDTs)
    - May be used for more complex computations on data
• The actually stored data is called an instance of a schema (extensional database)

  – Warning: some DBMS (e.g. IBM DB2) call a set of schemas and physical parameters (tablespaces, etc.) “instances” of a database

<table>
<thead>
<tr>
<th></th>
<th>AccNo</th>
<th>type</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensional DB</td>
<td>1278945</td>
<td>saving</td>
<td>€ 312.10</td>
</tr>
<tr>
<td></td>
<td>2437954</td>
<td>saving</td>
<td>€ 1324.82</td>
</tr>
<tr>
<td></td>
<td>4543032</td>
<td>checking</td>
<td>€ 43.03</td>
</tr>
<tr>
<td></td>
<td>5539783</td>
<td>saving</td>
<td>€ 12.54</td>
</tr>
<tr>
<td></td>
<td>7809849</td>
<td>checking</td>
<td>€ 7643.89</td>
</tr>
<tr>
<td></td>
<td>8942214</td>
<td>checking</td>
<td>€ 345.17</td>
</tr>
<tr>
<td></td>
<td>9134354</td>
<td>saving</td>
<td>€ 2.22</td>
</tr>
<tr>
<td></td>
<td>9543252</td>
<td>saving</td>
<td>€ 524.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Extensional DB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Check balance &gt; 0</td>
</tr>
</tbody>
</table>

Table account

Primary key AccNo
1.2 Three Schema Architecture

• **Remember:**
  – DBs should be **well structured and efficient**
  – Programs and data should be **isolated**
  – Different **views** for different user groups are necessary

• Thus, DBs are organized using 3 levels of schemas
  – **Internal** Schema (physical schema)
    • Describes the **physical storage and access paths**
    • Uses physical models
  – **Conceptual** Schema (logical schema)
    • Describes **structure** of the whole DB, hiding physical details
    • Uses conceptual models
  – **External** Schema (views)
    • Describes **parts** of the DB structure for a certain **user group** as views
    • Hides the conceptual details
1.2 Three Schema Architecture

- **End Users or Applications**
  - View 1
  - View 2
  - View n

- **Conceptual Schema**

- **Internal Schema**

- **Stored Data**

**ANSI/SPARC (1975)**
American National Standards Institute / Standards Planning and Requirements Committee
1.2 Data Independence

- Ability to change schema of one level without changing the others
- **Logical** Data Independence
  - Change of *conceptual* schema without change of *external* schemas (and thus applications)
  - Examples: adding attributes, changing constraints,…
  - But: for example dropping an attribute used in some user’s/application’s view will violate independence
1.2 Data Independence

**Physical Data Independence**

- Changes of the *internal* schema do not affect the *conceptual* schema
  - Important for reorganizing data on the disk (moving or splitting tablespaces)
  - Adding or changing access paths (new indices, etc.)
- Physical tuning is one of the most important *maintenance tasks* of DB administrators
- Physical independence is also supported by having a *declarative query language* in relational databases
  - What to access vs. how to access
1.3 System Structure - Overview

• Database characteristics lead to layered architecture

• Query Processor
  – Query Optimization
  – Query Planning

• Storage Manager
  – Access Paths
  – Physical sets, pages, buffers
  – Accesses disks through OS
    • May be avoided using ‘raw devices’ for direct data access
1.3 System Structure
1.3 Storage Manager

- The **storage manager** provides the interface between the data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for:
  - Interaction with the file manager
  - Efficient storing, retrieving and updating of data
- Tasks:
  - Storage access
  - File organization
  - Indexing and hashing
The query processor parses queries, optimizes query plans and evaluates the query

- Alternative ways of evaluating a given query due to equivalent expressions
- Different algorithms for each operation
- Cost difference between good and bad ways of evaluating a query can be enormous

Needs to estimate the cost of operations

- Depends critically on statistical information about relations which the DBMS maintains
- Need to estimate statistics for intermediate results to compute cost of complex expressions (join order, etc.)
• A **transaction** is a collection of operations that performs a single logical function in a database application

• The **transaction manager**
  – Ensures that the database remains in a correct state despite system failures (like power failures, operating system crashes, or transaction failures)
  – Controls the interaction among concurrent transactions to ensure the database consistency
• How do you know whether you built or bought a **good DBMS**?

• **Always**: depends on the application
  
  – Analyze data volume, typical DB queries and transactions (what do you really need?)
  
  – Analyze expected frequency of invocation of queries and transactions (what has to be supported?)
  
  – Analyze time constraints of queries and transactions (how fast does it have to be?)
  
  – Analyze expected frequency of update operations (does it deal with rather static or with volatile data?)
1.4 Performance Measures

• Basically analytical & experimental approaches on typical characteristics like
  
  – **Response time:** how long can a query/update be expected to take?
    • On average or at peak times (worst case)
  
  – **Transaction throughput:** how many transactions can be processed per second/millisecond?
    • On average or at peak times (worst case)
1.4 Industry Standard Benchmarks

• How to compare database performance across vendors?

• The **Transaction Processing Performance Council**
  
  – Aims are “significant disk input/output, moderate system and application execution time, and transaction integrity”
  
  – Defines certain scenarios with **standard data sets, schemas and queries**

http://www.tpc.org
1.4 Industry Standard Benchmarks

- **Performance Metrics:**
  - Throughput measured in transactions per second (tps)
  - Response time of transaction (transaction elapse time)
  - Cost metric (in $/tps)

- **TPC-D**
  - Ad hoc business questions, e.g. sales trends
  - Decision Support Applications
    - Long, complex read-only queries
    - Infrequent updates
    - Access large portions of the database
  - Used until 1999
1.4 Current TPC Benchmarks

• TPC-C
  – Standard for comparing On-Line Transaction Processing (OLTP) performance on various hardware and software configurations since 1992
  – Regular business operations, e.g. order-entry processing
  – OLTP applications
    • Update intensive
    • Shorter transactions that access a small portion of a database
• Numbers in entity blocks represent the cardinality of the tables (number of rows). These numbers are factored by $W$.
• The numbers next to the relationship arrows represent the cardinality of the relationships.
• The plus (+) symbol is used after the cardinality of a relationship or table to illustrate that this number is subject to small variations in the initial database population over the measurement interval as rows are added or deleted.
• New-Order Transaction
  – Entering a complete order through a single database transaction.

• Payment Transaction
  – Updates the customer's balance and reflects the payment on the district and warehouse sales statistics.

• Order-Status Transaction
  – Queries the status of a customer's last order.
• Delivery Transaction
  – Consists of processing a batch of 10 new (not yet delivered) orders.
  – Each order is processed (delivered) in full within the scope of a read-write database transaction.

• Stock-Level Transaction
  – Determines the number of recently sold items that have a stock level below a specified threshold.
### TPC-C Results

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company</th>
<th>System</th>
<th>Performance (tpmC)</th>
<th>Price/tpmC</th>
<th>Watts/Ktmpc</th>
<th>System Availability</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ORACLE</td>
<td>SPARC SuperCluster with T3-4 Servers</td>
<td>30,249,603</td>
<td>1.01 USD</td>
<td>NR</td>
<td>06/01/11</td>
<td>Oracle Database 11g Release 2 Ent. Ed. w/Real Application Clusters w/Partitioning</td>
</tr>
<tr>
<td>2</td>
<td>IBM</td>
<td>IBM Power 780 Server Model 9179-MHB</td>
<td>10,356,254</td>
<td>1.38 USD</td>
<td>NR</td>
<td>10/13/10</td>
<td>DB2 9.7</td>
</tr>
<tr>
<td>3</td>
<td>ORACLE</td>
<td>Sun SPARC Enterprise TS440 Server Cluster</td>
<td>7,646,485</td>
<td>2.36 USD</td>
<td>NR</td>
<td>03/19/10</td>
<td>Oracle Database 11g Ent. Ed. w/Real Application Clusters w/Partitioning</td>
</tr>
<tr>
<td>4</td>
<td>IBM</td>
<td>IBM Power 595 Server Model 9119-FHA</td>
<td>6,085,166</td>
<td>2.81 USD</td>
<td>NR</td>
<td>12/10/08</td>
<td>DB2 9.5</td>
</tr>
<tr>
<td>5</td>
<td>Bull</td>
<td>Bull Scalix P6460R</td>
<td>6,085,166</td>
<td>2.81 USD</td>
<td>NR</td>
<td>12/15/08</td>
<td>DB2 9.5</td>
</tr>
<tr>
<td>6</td>
<td>HP</td>
<td>HP Integrity Superdome-itanium2/1.6GHz/24MB iL3</td>
<td>4,092,799</td>
<td>2.93 USD</td>
<td>NR</td>
<td>08/06/07</td>
<td>Oracle Database 10g R2 Enterprise Edt w/Partitioning</td>
</tr>
<tr>
<td>7</td>
<td>IBM</td>
<td>IBM System p5 595</td>
<td>4,033,378</td>
<td>2.97 USD</td>
<td>NR</td>
<td>01/22/07</td>
<td>DB2 9</td>
</tr>
<tr>
<td>8</td>
<td>IBM</td>
<td>IBM eServer p5 595</td>
<td>3,210,540</td>
<td>5.07 USD</td>
<td>NR</td>
<td>05/14/05</td>
<td>DB2 UDB 8.2</td>
</tr>
<tr>
<td>9</td>
<td>FUJITSU</td>
<td>PRIMEQUEST 580A 32p/64c</td>
<td>2,322,032</td>
<td>3.76 USD</td>
<td>NR</td>
<td>12/04/08</td>
<td>Oracle Database 10g R2 Enterprise Edt w/Partitioning</td>
</tr>
<tr>
<td>10</td>
<td>IBM</td>
<td>IBM System x3850 X5</td>
<td>2,308,099</td>
<td>.64 USD</td>
<td>NR</td>
<td>05/20/11</td>
<td>DB2 ESE 9.7</td>
</tr>
<tr>
<td></td>
<td>FUJITSU</td>
<td>PRIMEQUEST 580 32p/64c</td>
<td>2,196,255</td>
<td>4.70 USD</td>
<td>NR</td>
<td>04/30/08</td>
<td>Oracle 10g Enterprise Ed R2 w/ Partitioning</td>
</tr>
</tbody>
</table>
1.4 Current TPC Benchmarks

- **TPC-E**
  - New OLTP workload benchmark
  - Simulates the OLTP workload of a brokerage firm focusing on a central database that executes transactions related to the firm’s customer accounts

- **TPC-H**
  - Ad-hoc, decision support benchmark
  - Consists of a suite of business oriented ad-hoc queries and concurrent data modifications
• How is data physically stored?
  – Memory types
  – Hard disks
  – RAID
  – SAN
  – NAS
  – etc.