Designing Databases for e-Business

- What’s different about e-Business?
- Design methodology
- Design issues
- Conclusions
What’s Different About e-Business?

- New types of applications
  - Net markets, supply chain automation
  - Customer care, personalization
- Different processing characteristics
  - Less defined, more dynamic workloads
  - Increased data accesses for display, validation
  - Tendency toward online, less batch
- More complex architectures
  - Multi-tier architectures and platforms
  - Disparate data sources, content data
- Greater degree of integration
  - Increased usage of software packages, products
  - Integration with legacy systems, data
What’s Different About e-Business Requirements?

- **Scalability**
  - Skewed peak volumes
  - Unlimited user community
  - Unpredictable, sudden growth

- **Availability and reliability**
  - Near 24 by 7 availability
  - Immediate recovery or fail-over
  - Maintenance operations non-intrusive

- **Response time**
  - All processing and data online to user
  - Blurring of batch and online processing
  - Volatile access rates

---

What’s Different About e-Business Requirements? (Cont.)

- **Data accessibility**
  - Data accessible online, more up to date
  - Unstructured and semi-structured data
  - Data access across multiple systems, sources
  - Data exchange, internal and external
  - Security

- **Flexibility**
  - Rapidly changing business requirements
  - Short window of opportunity
  - Components frequently replaced or added
  - Need to integrate with everything
  - Internationalization
e-Business Database Design Methodology

Designing Databases for Success

- Database design methodology is fundamentally the same
  - Apply same rigor as for ‘traditional’ systems
  - Adopt for compressed time schedule
- Database architecture is essential
  - Scalability, availability, performance must be in architecture
  - Database architecture part of integrated system architecture
- Performance engineering approach
  - Design for performance and scalability
  - Design for peak volumes and work down
- Collaboration with application developers
Use Performance Engineering Approach

- Apply throughout development lifecycle
- Pay particular attention to applications with:
  - Heavy workloads
  - Scalability/performance questions
  - Unproven technology
- Use appropriate means to prove:
  - Scalability
  - Performance
  - Availability

Techniques to Estimate Capacity and Performance

Accuracy vs. Difficulty

- Rules of Thumb
- Predictive Modeling
- Simulation
- Benchmark
- Real System

Relative scale only
Predictive modeling includes analytical modeling
Collaborate With Application Developers

- Work with application designers and developers
  - Architecture
  - Data access protocols
  - Component and process design
  - Predictive performance modeling

- Design issues posed with component based development
  - Object-relational mapping
  - Granularity of components
  - Generated SQL masks data access workload

- Recognize opportunities to use modern DBMS features
  - Object-relational
  - Advanced SQL
  - Other new features

Object-relational Mapping

Component Technologies
- EJB
- CORBA
- DCOM

Business object

Data object

Relational Tables
Design Issues

Architectural Tiers

Database Guidelines

- Choose type of tiered architecture
  - 2 tier vs. 3 tier vs. multi-tier
  - Applets, servlets, components
- Separate business processing from database
  - Vertical scalability
  - Off loads business processing from database server
- Move data accesses close to database
  - Avoid SQL connections from client browser
  - Beware of locking and network latencies
- Use stored procedures to batch data access calls
  - Encapsulates data processing with data
  - Additional processing on database server can affect scalability
Architectural Tiers

Database Guidelines (Cont.)

- Consider replication for performance, integration
  - Read only data, data warehouses
  - Necessary because of system integration, packages
- Use caching
  - Web servers, application servers
  - Database
- Design database and component based architecture
  - Object-relational mapping
  - Enterprise JavaBeans
  - Stored procedures and components

Scalability and Capacity

- Options for scalability
  - Horizontal architecture
  - Vertical architecture
- Web servers and application servers
  - Easy to grow horizontally
  - Many methods to save state data
  - Many transaction routing options
  - Database is usually a shared resource
- Database design options
  - Single database
  - Database partitioning
  - Application partitioning
  - Data replication
Sample Database Architecture

Application and Database Partitioning

Application Partitioning
DB2 EE

DB2
SMP 1

DB2
SMP 2

Application Servers
Transaction routing required

Database Partitioning
DB2 EEE

DB2
Node 1

DB2
Node n

Part 1
Switch
Part n
Sample Database Architecture

**Application and Database Partitioning**

Application Partitioning
Oracle

Oracle A
SMP 1

Oracle B
SMP 2

Database Partitioning
Oracle OPS

Oracle
SMP 1

Oracle
SMP 2

Transaction routing required

Application routing

Data accessible from each node

Data affinity load balancing

Database Architecture

**Replication**

Near Real Time Replication

DBMS A
SMP 1

DBMS B
SMP 2

Database Replication

Utilities/Application

Snapshot
Scalability and Capacity

Database Design Guidelines

- Determine degree of scalability
  - Determine need for partitioning and/or replication
  - Plan for worst case
  - Select granularity of adding scale
- Design as part of architecture
  - System architecture
  - Database architecture
- Choose appropriate DBMS options
  - Match database architecture with hardware architecture
- Combine approaches as needed
  - Balance scalability and availability

Scalability and Capacity

Database Design Guidelines (Cont.)

<table>
<thead>
<tr>
<th>Scaling type</th>
<th>Single Database</th>
<th>Database Partitioning</th>
<th>Application Partitioning</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling limits</td>
<td>Vertical</td>
<td>Horizontal</td>
<td>Horizontal</td>
<td>Horizontal</td>
</tr>
<tr>
<td>Application impact</td>
<td>Largest server</td>
<td>Practically unlimited</td>
<td>Limited to management issues</td>
<td>Limited to management issues</td>
</tr>
<tr>
<td>Design issues</td>
<td>Architecture to match database</td>
<td>Architecture and process routing</td>
<td>Mostly database</td>
<td></td>
</tr>
<tr>
<td>Application characteristics</td>
<td>Need for Database partitioning</td>
<td>Design for DBMS architecture</td>
<td>Global view of database</td>
<td>Latency and refresh</td>
</tr>
</tbody>
</table>

Growth limited | Growth unlimited | Process affinity practical | Static data, high volume access |
Availability and Reliability

**Database Design Options**

- **Hardware options**
  - Mutual takeover
  - Hot standby
- **DBMS options**
- **Partitioning strategies**
  - Application partitioning
  - Database partitioning
- **Replication strategies**
  - Near real time replication
  - Snapshots

---

Availability and Reliability

Application Servers

Transaction routing required

Route to any node

DBMS A

Buffer Pool

SMP 1

Mutual Takeover Configuration

DBMS B

Buffer Pool

SMP 2

Application Servers

Hot Standby Configuration

DBMS A

Buffer Pool

SMP 1

DBMS A

Buffer Pool

SMP 2
Availability and Reliability

**Replication**

- Near Real Time
- Transaction routing required

- Snapshot
- Utilities/application

Availability and Reliability

**Database Design Guidelines**

- High availability different than reliability
- Keep it simple
  - More components, more points of failure
- Avoid single point of failures
  - Redundant components
  - Database fail over
- Plan for:
  - Database fail over
  - Database fail back
  - Database maintenance
  - Disaster recovery
Availability and Reliability

Database Design Guidelines (Cont.)

<table>
<thead>
<tr>
<th></th>
<th>Mutual Standby</th>
<th>Hot Standby</th>
<th>Partitioning</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to resume service</td>
<td>Time to restart DBMS</td>
<td>Time to restart DBMS</td>
<td>No disruption of other partitions</td>
<td>Time to switch to replicate</td>
</tr>
<tr>
<td>Extent of outage</td>
<td>Database</td>
<td>Database</td>
<td>One partition</td>
<td>Database</td>
</tr>
<tr>
<td>Degraded services</td>
<td>Reduced CPU capacity</td>
<td>Depends on standby system</td>
<td>One partition inaccessible</td>
<td>Reduced functionality and data</td>
</tr>
<tr>
<td>Fall back</td>
<td>Stop/Start DBMS</td>
<td>Stop/Start DBMS</td>
<td>No impact</td>
<td>Application dependencies</td>
</tr>
<tr>
<td>Ease of implementation</td>
<td>Simple, proven</td>
<td>Simple, proven</td>
<td>Application impact</td>
<td>Application impact</td>
</tr>
</tbody>
</table>

Response Time and Performance SQL Interfaces

- ODBC, JDBC
  - Keep data accesses close to database
  - Type of drivers
  - Performance features

- SQLJ
  - Embedded SQL for Java, less coding than JDBC
  - Standard parser
  - Strong type checking

- Stored procedure languages
  - C, Java
  - Persistent Stored Module (PSM)

- Java supports advanced data types
Unstructured and Semi-structured Data

- **Unstructured Data**
  - No structure
  - Examples: video, audio

- **Semi-structured data**
  - “schema-less” data
  - Self describing data
  - Examples: text, documents, CAD/CAM

Unstructured and Semi-structured Data

**Design Options**

- **Store as OS files**
  - Current location
  - Integration with content management products

- **Store in specialized database**
  - Proprietary database
  - OODBMS

- **Store or manage with DBMS**
  - Large objects (BLOB, CLOB, DBCLOB)
  - DB2 Extenders or Oracle Cartridges
  - DB2 DataLinks, Oracle interMedia
Unstructured and Semi-structured Data DB2

Datalinks Architecture

Unstructured and Semi-structured Data DB2

Database Design Guidelines - DB2

<table>
<thead>
<tr>
<th></th>
<th>LOB</th>
<th>Text/Video Extender</th>
<th>XML Extender</th>
<th>DataLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data storage</td>
<td>Database</td>
<td>Database</td>
<td>Database</td>
<td>OS files</td>
</tr>
<tr>
<td>Unstructured Data</td>
<td>Yes</td>
<td>Advanced features</td>
<td>Probably not</td>
<td>Most efficient</td>
</tr>
<tr>
<td>Semi-structured Data</td>
<td>No structure supported</td>
<td>Advanced features</td>
<td>Yes</td>
<td>No structure supported</td>
</tr>
<tr>
<td>Data exchange</td>
<td>No direct support</td>
<td>No direct support</td>
<td>Advanced features</td>
<td>No direct support</td>
</tr>
<tr>
<td>Data streaming</td>
<td>Yes</td>
<td>Yes</td>
<td>Probably not</td>
<td>Most efficient</td>
</tr>
</tbody>
</table>
Other Architectural Issues

- Security
- Internationalization
  - Double byte character set
  - Time and language
  - UDF
- Software packages
  - Integration, EAI, Queuing
  - Performance tuning
- Data integration with DBMS
  - DataJoiner
  - Data wrappers
  - Portals

Conclusions
Conclusions

■ e-Business has new and different requirements
  – More challenging
■ Designers need to apply same rigor as traditional systems
  – Architecture is essential
  – Design for scalability, availability, and performance
  – Adopt for speed of development
■ Database and application designers must work together
  – Database and internet system architects
  – Database and application designers and developers
■ Database designers must be knowledgeable of technology
  – DBMS offers extensive set of ‘interesting’ features
  – New opportunities
  – Rapidly changing