

# Chapter 25: Advanced Application Development

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan See www.db-book.com for conditions on re-use



## Hardware Tuning: Choice of RAID Level

- To use RAID 1 or RAID 5?
  - Depends on ratio of reads and writes
    - RAID 5 requires 2 block reads and 2 block writes to write out one data block
- If an application requires r reads and w writes per second
  - RAID 1 requires r + 2w I/O operations per second
  - RAID 5 requires: r + 4w I/O operations per second
- For reasonably large r and w, this requires lots of disks to handle workload
  - RAID 5 may require more disks than RAID 1 to handle load!
  - Apparent saving of number of disks by RAID 5 (by using parity, as opposed to the mirroring done by RAID 1) may be illusory!
- Thumb rule: RAID 5 is fine when writes are rare and data is very large, but RAID 1 is preferable otherwise
  - If you need more disks to handle I/O load, just mirror them since disk capacities these days are enormous!



## **Tuning the Database Design (Cont.)**

#### **Materialized Views**

- Materialized views can help speed up certain queries
  - Particularly aggregate queries
- Overheads
  - Space
  - Time for view maintenance
    - Immediate view maintenance: done as part of update txn
      - time overhead paid by update transaction
    - Deferred view maintenance: done only when required
      - update transaction is not affected, but system time is spent on view maintenance
        - until updated, the view may be out-of-date
- Preferable to denormalized schema since view maintenance is systems responsibility, not programmers
  - Avoids inconsistencies caused by errors in update programs



## **Tuning the Database Design (Cont.)**

- How to choose set of materialized views
  - Helping one transaction type by introducing a materialized view may hurt others
  - Choice of materialized views depends on costs
    - Users often have no idea of actual cost of operations
  - Overall, manual selection of materialized views is tedious
- Some database systems provide tools to help DBA choose views to materialize
  - "Materialized view selection wizards"



## **Tuning of Transactions (Cont.)**

- Reducing lock contention
- Long transactions (typically read-only) that examine large parts of a relation result in lock contention with update transactions
  - E.g., large query to compute bank statistics and regular bank transactions
- To reduce contention
  - Use multi-version concurrency control
    - E.g., Oracle "snapshots" which support multi-version 2PL
  - Use degree-two consistency (cursor-stability) for long transactions
    - Drawback: result may be approximate



## **Tuning of Transactions (Cont.)**

- Long update transactions cause several problems
  - Exhaust lock space
  - Exhaust log space
    - and also greatly increase recovery time after a crash, and may even exhaust log space during recovery if recovery algorithm is badly designed!
- Use mini-batch transactions to limit number of updates that a single transaction can carry out. E.g., if a single large transaction updates every record of a very large relation, log may grow too big.
  - Split large transaction into batch of "mini-transactions," each performing part of the updates
  - Hold locks across transactions in a mini-batch to ensure serializability
    - If lock table size is a problem can release locks, but at the cost of serializability
  - In case of failure during a mini-batch, must complete its remaining portion on recovery, to ensure atomicity.



## **Performance Simulation**

- Performance simulation using queuing model useful to predict bottlenecks as well as the effects of tuning changes, even without access to real system
- Queuing model as we saw earlier
  - Models activities that go on in parallel
- Simulation model is quite detailed, but usually omits some low level details
  - Model service time, but disregard details of service
  - E.g., approximate disk read time by using an average disk read time
- Experiments can be run on model, and provide an estimate of measures such as average throughput/response time
- Parameters can be tuned in model and then replicated in real system
  - E.g., number of disks, memory, algorithms, etc.



## **Database Application Classes**

- Online transaction processing (OLTP)
  - requires high concurrency and clever techniques to speed up commit processing, to support a high rate of update transactions.
- Decision support applications
  - including online analytical processing, or OLAP applications
  - require good query evaluation algorithms and query optimization.
- Architecture of some database systems tuned to one of the two classes
  - E.g., Teradata is tuned to decision support
- Others try to balance the two requirements
  - E.g., Oracle, with snapshot support for long read-only transaction



## **Benchmarks Suites (Cont.)**

- TPC benchmarks (cont.)
  - TPC-D: complex decision support application
    - Superceded by TPC-H and TPC-R
  - **TPC-H**: (H for ad hoc) based on TPC-D with some extra queries
    - Models ad hoc queries which are not known beforehand
      - Total of 22 queries with emphasis on aggregation
    - prohibits materialized views
    - permits indices only on primary and foreign keys
  - TPC-R: (R for reporting) same as TPC-H, but without any restrictions on materialized views and indices
  - TPC-W: (W for Web) End-to-end Web service benchmark modeling a Web bookstore, with combination of static and dynamically generated pages



#### **TPC Performance Measures**

- Two types of tests for TPC-H and TPC-R
  - Power test: runs queries and updates sequentially, then takes mean to find queries per hour
  - Throughput test: runs queries and updates concurrently
    - multiple streams running in parallel each generates queries, with one parallel update stream
  - Composite query per hour metric: square root of product of power and throughput metrics
  - Composite price/performance metric



### **Other Benchmarks**

- OODB transactions require a different set of benchmarks.
  - OO7 benchmark has several different operations, and provides a separate benchmark number for each kind of operation
  - Reason: hard to define what is a typical OODB application
- Benchmarks for XML being discussed



## **SQL Standards History (Cont.)**

- SQL:1999
  - Adds variety of new features --- extended data types, object orientation, procedures, triggers, etc.
  - Broken into several parts
    - SQL/Framework (Part 1): overview
    - SQL/Foundation (Part 2): types, schemas, tables, query/update statements, security, etc.
    - SQL/CLI (Call Level Interface) (Part 3): API interface
    - SQL/PSM (Persistent Stored Modules) (Part 4): procedural extensions
    - SQL/Bindings (Part 5): embedded SQL for different embedding languages



## **SQL Standards History (Cont.)**

- More parts undergoing standardization process
  - Part 7: SQL/Temporal: temporal data
  - Part 9: SQL/MED (Management of External Data)
    - Interfacing of database to external data sources
      - Allows other databases, even files, can be viewed as part of the database
  - Part 10 SQL/OLB (Object Language Bindings): embedding SQL in Java
  - Missing part numbers 6 and 8 cover features that are not near standardization yet



## **XML-Based Standards**

- Several XML based Standards for E-commerce
  - E.g., RosettaNet (supply chain), BizTalk
  - Define catalogs, service descriptions, invoices, purchase orders, etc.
  - XML wrappers are used to export information from relational databases to XML
- Simple Object Access Protocol (SOAP): XML based remote procedure call standard
  - Uses XML to encode data, HTTP as transport protocol
  - Standards based on SOAP for specific applications
    - E.g., OLAP and Data Mining standards from Microsoft



## **E-Commerce**

- E-commerce is the process of carrying out various activities related to commerce through electronic means
- Activities include:
  - Presale activities: catalogs, advertisements, etc.
  - Sale process: negotiations on price/quality of service
  - Marketplace: e.g., stock exchange, auctions, reverse auctions
  - Payment for sale
  - Delivery related activities: electronic shipping, or electronic tracking of order processing/shipping
  - Customer support and post-sale service



## **E-Catalogs**

- Product catalogs must provide searching and browsing facilities
  - Organize products into intuitive hierarchy
  - Keyword search
  - Help customer with comparison of products
- Customization of catalog
  - Negotiated pricing for specific organizations
  - Special discounts for customers based on past history
    - E.g., loyalty discount
  - Legal restrictions on sales
    - Certain items not exposed to under-age customers
- Customization requires extensive customer-specific information



## **Marketplaces**

- Marketplaces help in negotiating the price of a product when there are multiple sellers and buyers
- Several types of marketplaces
  - Reverse auction
  - Auction
  - Exchange
- Real world marketplaces can be quite complicated due to product differentiation
- Database issues:
  - Authenticate bidders
  - Record buy/sell bids securely
  - Communicate bids quickly to participants
    - Delays can lead to financial loss to some participants
  - Need to handle very large volumes of trade at times
    - E.g., at the end of an auction



## **Types of Marketplace**

- Reverse auction system: single buyer, multiple sellers.
  - Buyer states requirements, sellers bid for supplying items. Lowest bidder wins. (also known as tender system)
  - Open bidding vs. closed bidding
- Auction: Multiple buyers, single seller
  - Simplest case: only one instance of each item is being sold
  - Highest bidder for an item wins
  - More complicated with multiple copies, and buyers bid for specific number of copies
- Exchange: multiple buyers, multiple sellers
  - E.g., stock exchange
  - Buyers specify maximum price, sellers specify minimum price
  - exchange matches buy and sell bids, deciding on price for the trade
    - e.g., average of buy/sell bids



## **Order Settlement**

- Order settlement: payment for goods and delivery
- Insecure means for electronic payment: send credit card number
  - Buyers may present some one else's credit card numbers
  - Seller has to be trusted to bill only for agreed-on item
  - Seller has to be trusted not to pass on the credit card number to unauthorized people
- Need secure payment systems
  - Avoid above-mentioned problems
  - Provide greater degree of privacy
    - E.g., not reveal buyers identity to seller
  - Ensure that anyone monitoring the electronic transmissions cannot access critical information



## **Secure Payment Systems**

- All information must be encrypted to prevent eavesdropping
  - Public/private key encryption widely used
- Must prevent person-in-the-middle attacks
  - E.g., someone impersonates seller or bank/credit card company and fools buyer into revealing information
    - Encrypting messages alone doesn't solve this problem
    - More on this in next slide
- Three-way communication between seller, buyer and credit-card company to make payment
  - Credit card company credits amount to seller
  - Credit card company consolidates all payments from a buyer and collects them together
    - E.g., via buyer's bank through physical/electronic check payment



## **Secure Payment Systems (Cont.)**

- Digital certificates are used to prevent impersonation/man-in-the middle attack
  - Certification agency creates digital certificate by encrypting, e.g., seller's public key using its own private key
    - Verifies sellers identity by external means first!
  - Seller sends certificate to buyer
  - Customer uses public key of certification agency to decrypt certificate and find sellers public key
    - Man-in-the-middle cannot send fake public key
  - Sellers public key used for setting up secure communication
- Several secure payment protocols
  - E.g., Secure Electronic Transaction (SET)



## **Digital Cash**

- Credit-card payment does not provide anonymity
  - The SET protocol hides buyers identity from seller
  - But even with SET, buyer can be traced with help of credit card company
- Digital cash systems provide anonymity similar to that provided by physical cash
  - E.g., Dig Cash
  - Based on encryption techniques that make it impossible to find out who purchased digital cash from the bank
  - Digital cash can be spent by purchaser in parts
    - much like writing a check on an account whose owner is anonymous



## **Legacy Systems**

- Legacy systems are older-generation systems that are incompatible with current generation standards and systems but still in production use
  - E.g., applications written in Cobol that run on mainframes
    - Today's hot new system is tomorrows legacy system!
- Porting legacy system applications to a more modern environment is problematic
  - Very expensive, since legacy system may involve millions of lines of code, written over decades
    - Original programmers usually no longer available
  - Switching over from old system to new system is a problem
    - more on this later
- One approach: build a wrapper layer on top of legacy application to allow interoperation between newer systems and legacy application
  - E.g., use ODBC or OLE-DB as wrapper



## **Legacy Systems (Cont.)**

- Rewriting legacy application requires a first phase of understanding what it does
  - Often legacy code has no documentation or outdated documentation
  - reverse engineering: process of going over legacy code to
    - Come up with schema designs in ER or OO model
    - Find out what procedures and processes are implemented, to get a high level view of system
- Re-engineering: reverse engineering followed by design of new system
  - Improvements are made on existing system design in this process



## **Legacy Systems (Cont.)**

- Switching over from old to new system is a major problem
  - Production systems are in every day, generating new data
  - Stopping the system may bring all of a company's activities to a halt, causing enormous losses

#### Big-bang approach:

- Implement complete new system
- 2. Populate it with data from old system
  - 1. No transactions while this step is executed
  - scripts are created to do this quickly
- 3. Shut down old system and start using new system
- Danger with this approach: what if new code has bugs or performance problems, or missing features
  - Company may be brought to a halt



## **Legacy Systems (Cont.)**

- Chicken-little approach:
  - Replace legacy system one piece at a time
  - Use wrappers to interoperate between legacy and new code
    - E.g., replace front end first, with wrappers on legacy backend
      - Old front end can continue working in this phase in case of problems with new front end
    - Replace back end, one functional unit at a time
      - All parts that share a database may have to be replaced together, or wrapper is needed on database also
  - Drawback: significant extra development effort to build wrappers and ensure smooth interoperation
    - Still worth it if company's life depends on system



# **End of Chapter 25**