CSC 546: Client/Server Fundamentals

Fall 2000

Networked SQL

- Format & Protocol (FAP), FAP gateways
  - DAL, EDA/SQL, RDA, DRDA
- distributed aspects
  - stored procedures, distributed updates, replicated data
- performance aspects
  - query optimization, locking issues, bottlenecks, indexing
- online analytical processing aspects

Format and Protocol (FAP)

SQL provides a standard language for database operations

APIs provide standard(?) interfaces for invoking SQL commands from applications

REMAINING QUESTIONS:

- how does the database client interoperate with the database server?
- what application-layer protocols is used?
- what happens when try to interact with many distributed databases?
- what are the major performance issues involved with networked databases?
- . . .

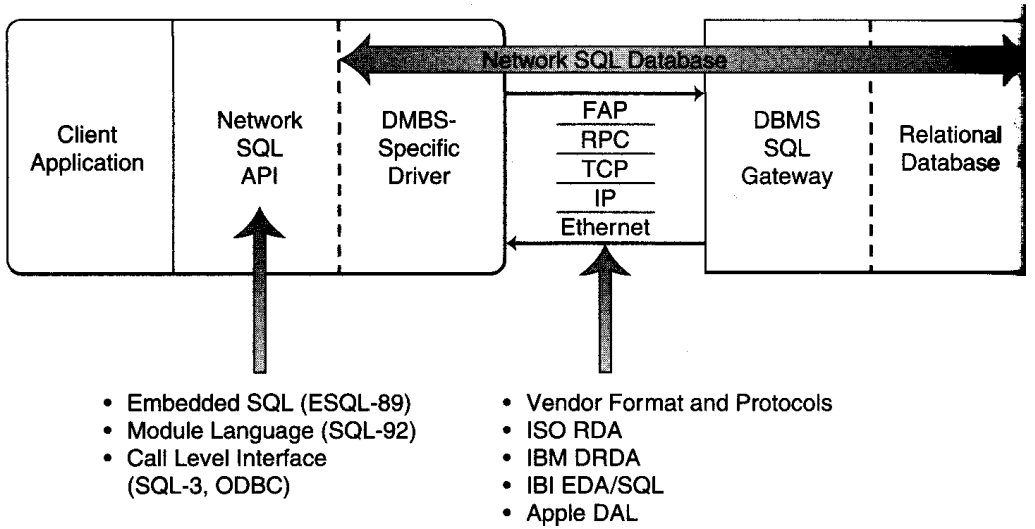
FAP: set of formats and protocols used by a particular networked database

- format of encoded SQL commands
- format of response packets, return codes, and diagnostic information
- type of RPC and network protocols

FAP software

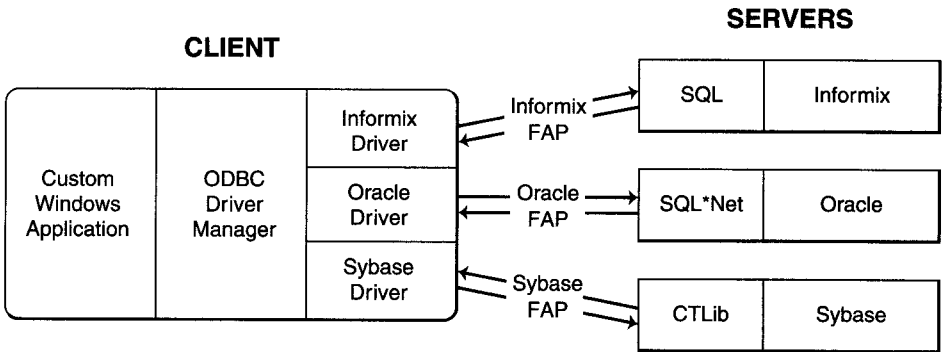
typically, each database vendor has their own proprietary FAP

- competing standards exist, must have compatible software at each end



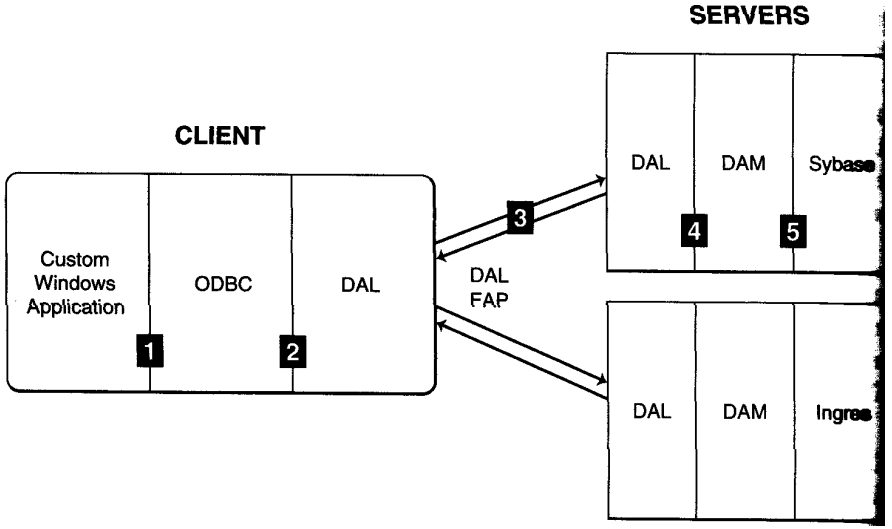
if multiple databases are accessed, may need many drivers on client

- can cause conflicts over system resources, leads to *versionitis*



Standard FAPs

- Apple's Data Access Language (DAL)
  DAL client software encodes & transmits generic SQL to DAL server software
  vendor specific Data Adaptor Modules (DAMs) convert to native SQL format



- IBI's Enterprise Data Access/SQL (EDA/SQL)
  provides read-only support for many databases, protocols, and operating systems

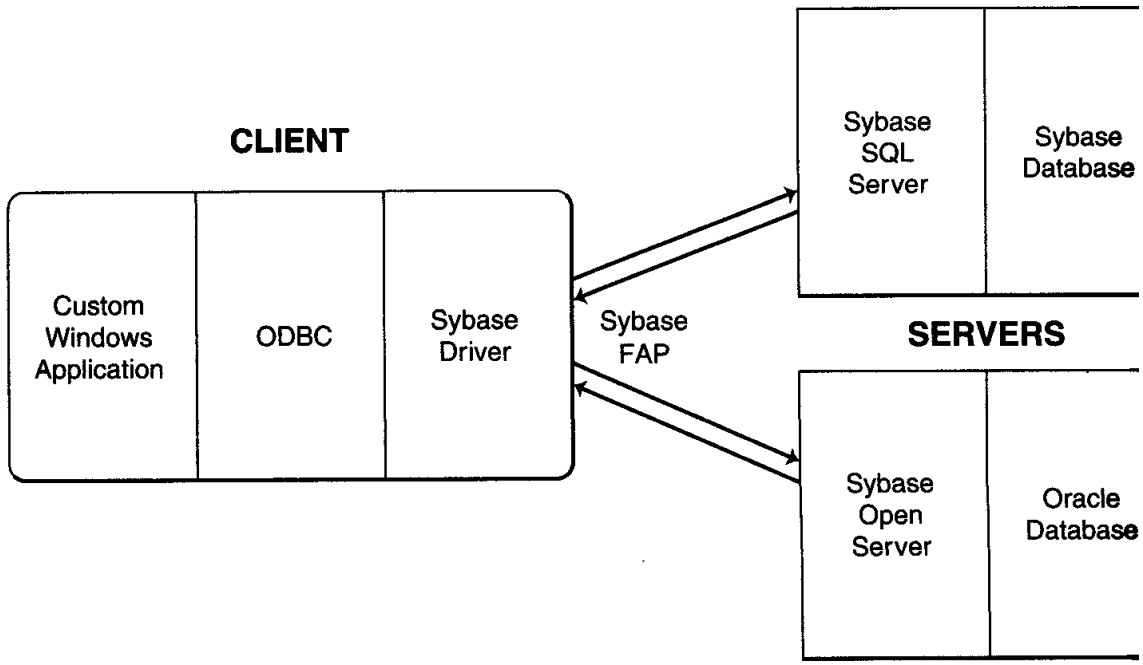
- ISO's Remote Database Access (RDA)
  designated by ISO as a layer-7 OSI protocol, limited to SQL-92 standard

- IBM's Distributed Relational Database Architecture (DRDA)
  provides federated database interoperability (independent but collaborative)

FAP gateways

to aid interoperability, many database vendors provide gateways for accessing other vendors' databases

e.g., Sybase provides a gateway to Oracle databases as well as its own
→ only need one driver on client, appropriate gateways on servers



Distributed aspects

QUESTION 1: how should processing be distributed between client & server?

- usually: want data access and manipulation performed on the client
  limit server to retrieving, updating, and organizing data

- but not always:
  suppose wanted to average household incomes from a large table
  using SELECT and a cursor requires downloading all records, averaging by client
  better to access and average on the server, avoid network traffic

many databases allow for *stored procedures*

i.e., user-defined routines for accessing/manipulating data directly on the server

stored procedures can be

- explicitly invoked by special network SQL commands on the client, OR
- *triggered* automatically whenever a specific event occurs in the database

Distributed aspects (cont.)

QUESTION 2: how should data be distributed between multiple servers?

- clearly, can improve performance by balancing the load
- however, have to worry about consistency & coordination

"To the user, a distributed system should look exactly like a nondistributed system"

1. Each server should be autonomous to the maximum extent possible.
2. There should be no reliance on a central server.
3. Continuous operation of the database should always be possible.

4. Users should not have to know where data is located.
5. Users should not have to know how data is fragmented for distribution.
6. Users should not have to know if any data has been replicated among servers.

7. Network-based optimization should be performed to minimize traffic. *
8. Distributed transaction management should be performed to coordinate consistency and recovery. *

9. Heterogeneous platforms, operating systems, protocols, and interfaces should be supported.

Distributed queries

7. Network-based optimization should be performed to minimize traffic.

e.g., suppose want to join a huge table and a tiny table

need to combine two tables into one, but at which location?

better to join smaller to huge, copy rows from tiny table & add to huge

e.g., suppose have a large table of inventory and a small table of suppliers, want to select all *computer* parts in stock from *foreign* suppliers

need to perform select on each table, but in what order?

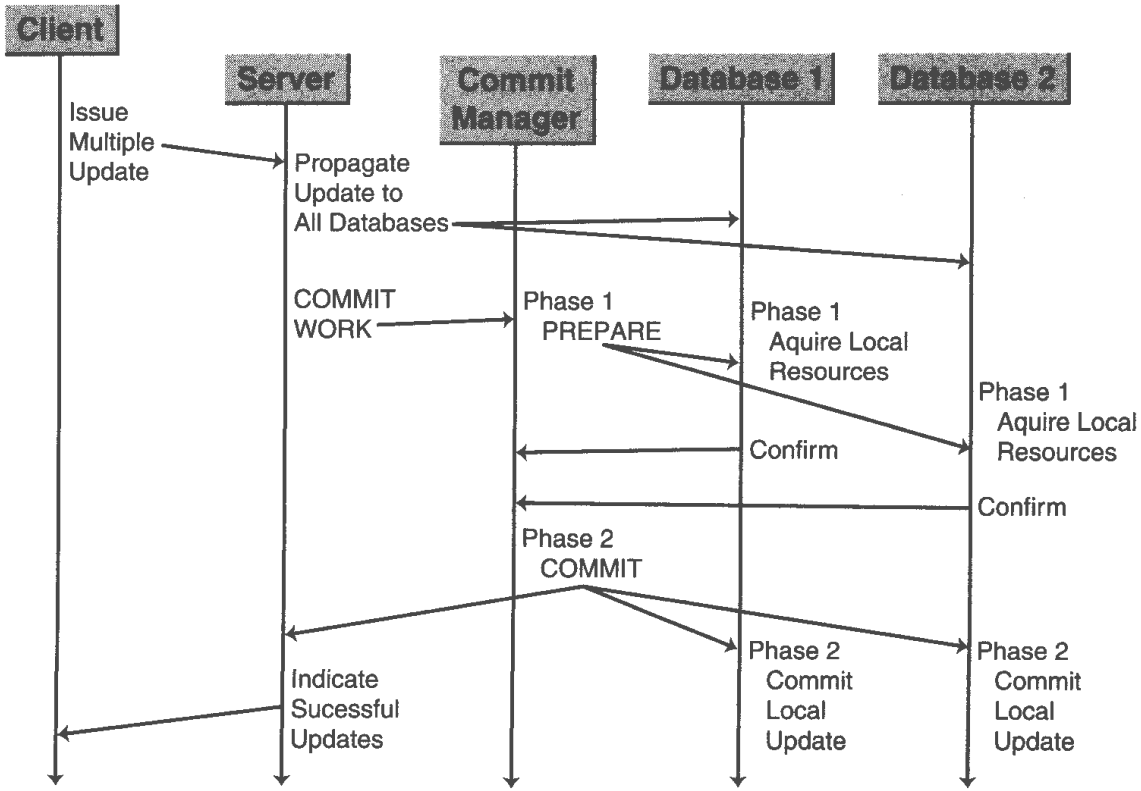
assuming there are many computer parts in stock & few foreign suppliers, better to identify foreign suppliers first, use results to constrain parts search

Distributed updates

8. Distributed transaction management should be performed to coordinate consistency and recovery.

typically, use a *two-phase commit strategy*

1. transaction commit manager sends requests for resource
2. each database server responds with a lock confirmation



DANGER: deadlock is possible and must be actively avoided

Replicated data

8. Distributed transaction management should be performed to coordinate consistency and recovery.

once data is distributed, only a 2-phase commit strategy can ensure consistency

- requires that all databases be online and accessible at all times

there are times when this is not feasible, or real-time consistency is not nec.

could use a *replicated distribution strategy*

- a copy of data managed at one location is made available to others
- can control access to copied data (e.g., cache-like)
- can control updates to copied data (e.g., read-only)

The Principle of Data Uncertainty: the probability of knowing the true value for a specific data element declines proportionally by the extent to which it is shared.

Performance aspects

when many clients can share database server(s), performance is affected by:

- query optimizations
  most database servers can optimize inefficient queries (e.g., huge+tiny join)

- locking granularity
  the smaller the page size, the more concurrency there can be (but more overhead)

- access bottlenecks
  busy tables should be spread across multiple disks to reduce disk head contentions
  if numerous collisions, can possibly spread data across multiple servers

- indexing
  all databases allow you to specify the way data is to be organized and indexed
  different organizations have different tradeoffs
  frequent inserts → B-tree *searches only → clustered ISAM*
  multiple-match queries → hash table *small table → array (no index)*

Online Analytical Processing (OLAP)

business trend:

- don’t just access & report on data (Online Transaction Processing – OLTP)
- analyze and navigate through data to discover trends, spot exceptions, and understand the details driving the business

data mining: scan for anomalies or patterns in the data

data modeling: correlate driving factors, determine causes and effects

OLAP has different query & response time characteristics from OLTP

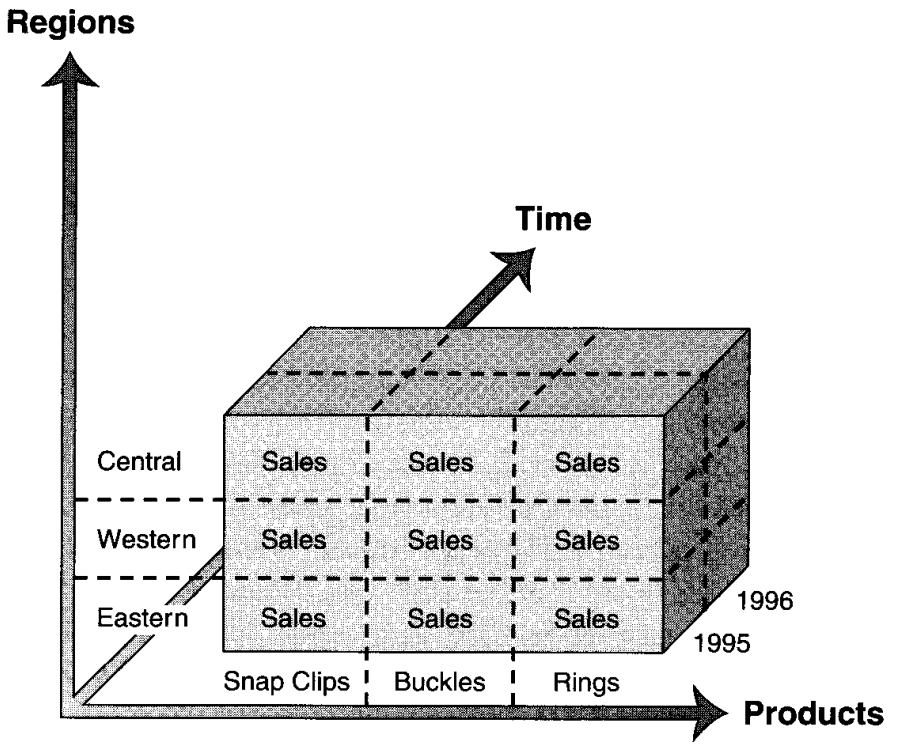
OLTP: read & update small numbers of closely related records

OLAP: read, integrate, and consolidate very large volumes of diverse data

OLAP example

to allow for efficient analysis, OLAP data is optimized into a multidimensional hypercube that is pivoted to support different analyses

Date	Market	Region	Product	Channel	Units	Sales
Jan 1995	Fasteners	Central	Snap Clips	Direct	5,000	$10,000
Jan 1995	Spacers	Central	Rings	Direct	20,000	$20,000
Feb 1995	Fasteners	Western	Snap Clips	Mail Order	6,000	$15,000
Feb 1995	Fasteners	Central	Buckles	Direct	20,000	$85,000
Mar 1995	Fasteners	Central	Snap Clips	Mail Order	50,000	$90,000



each column of data defines a dimension

2 dimensions → table

3 dimensions → cube

. . .

n dimensions → hypercube

Navigating OLAP data

can "drill down" deeper into the data

e.g., break year down into months → subdivide hypercube entry

can "roll up" out of the data

e.g., coalesce months back into years, years into decades, etc.

can "slice and dice" to look at the data differently

dicing: changing the data being viewed (i.e., focus on different features)
slicing: rearrange the data differently for analysis purposes

example: Central regional sales manager wants to understand sales

- organizes data to show sales by quarter for all products sold in the region
- noting poor sales of fasteners in first quarter, can drill into that specific info
- after finding snap clips to be the cause, can slice & dice data to see if other regions saw similar trends
- . . .

OLAP characteristics

OLAP requires higher performance access/manipulation than OLTP

- dynamic access to large quantities of data
- on-the-fly aggregation due to drilling down or rolling up
- rapid change of perspective due to slicing or dicing
- calculations on the raw data, relative data, or derived data

in theory, could implement OLAP using 2-D relational databases

- need to keep 2-D tables for each pair of relevant dimensions
- drilling down or rolling up requires generating new tables (or precomputing)

- lots of redundancy, difficult to perform calculations on different aggregates

in practice, OLAP databases uses specialized methods

- relational databases with enhanced OLAP features such as bit-mapped indexes or star schema tables (e.g., Oracle Express, DSS/Server)
- pure multidimensional databases that use complex data structures tailored to hypercube aggregates (e.g., Essbase, TM/1, LightShip Server)

Next week…

client/server implementation aspects

- principles of design & development
- structural aspects
- distributed systems management

course overview

read Chapters 15 and 16

As always, be prepared for a short quiz