Retrospect on DB Models & Languages

CPS 296.1

Database and Programming Languages: Crossing the Chasm

Jun Yang
Duke University
January 19, 2010

† Thanks to contents/ideas borrowed from Hellerstein (http://...) and Lefevre (http://...)

Image from http://www.bbm.edu/...

Announcements

- Sign up (by email) to lead discussions by Wednesday
  - Student-led discussions start next week!
- Read the two papers on making DBMS object-relational for Thursday
  - Review for "Inclusion of New Types..." due by 9am on Thursday
  - Submit your review on the Blackboard class forum by replying to my post
  - Again, see course website for instructions on reviewing and submission, as well as tips on reading research papers

On leading discussions

- Three types of discussion: research papers, survey papers, or tutorials of systems/platforms
- As leaders, you must finish reading/researching in advance
- I will meet you to talk about the lecture
  - By default, during office hours on the day of the lecture before the one you are leading
    - Thursday lecture → meet on Tuesday; Tuesday lecture → meet on Thursday of the preceding week
- Besides providing summary, critique, and answering questions, strive to generate discussion from class
  - A good way is to ask "facilitating" questions

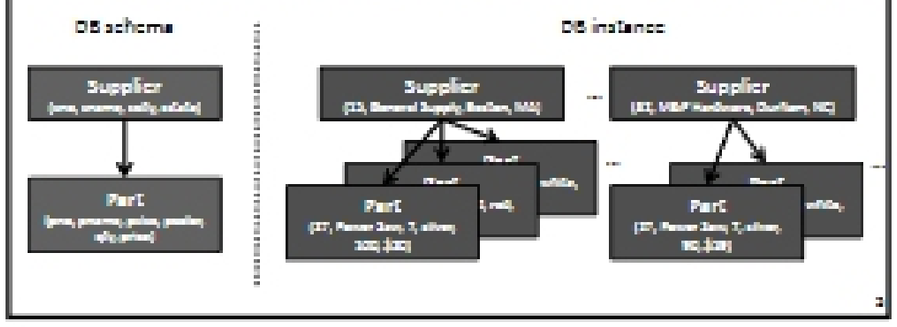
Overview

- Stonebraker & Hellerstein. "What Goes Around Comes Around." In *Readings in Database Systems* (aka "the red book"), 4th ed., 2005
  - A retrospective survey of DBMS data models and query languages
  - Lessons to learn from past experience
  - And why XML is doomed

➲What do *you* think?

Hierarchical: IMS (1968)

- Organize record types in hierarchies
  - Each non-root type has a single parent type
  - Each record of a non-root type has one parent of the parent type
    - Corollary: each record has a unique HSK (Hierarchical Sequence Key)
- Model simplicity facilitates simple language & implementation



Issues with model

- Information is repeated
  - Schema #1: parts info repeated across suppliers
  - Schema #2: supplier info repeated across parts
- Existence depends on parent data
  - Schema #1: what if nobody supplies a part?
  - Schema #2: what if a supplier doesn't supply anything?

Issues with language (DL/1)

- Conceptually, records are laid out in HSK order: depth-first, left-to-right → get ingrained in language constructs
- Record-at-a-time language
- Programmer writes an algorithm for solving each query, e.g.:

```
get unique Supplier with sno = 15
until failure do
    get next within parent with pcolor = red
```

Why is this bad?

- Different underlying storage formats (sequential/B-tree/hash) → different restrictions on commands
  - Heavy coupling between storage and client apps
- Different data characteristics → different optimal algorithms
  - Optimization is performed by programmer and DB designer
- Not declarative, poor physical data independence

Hacking the model

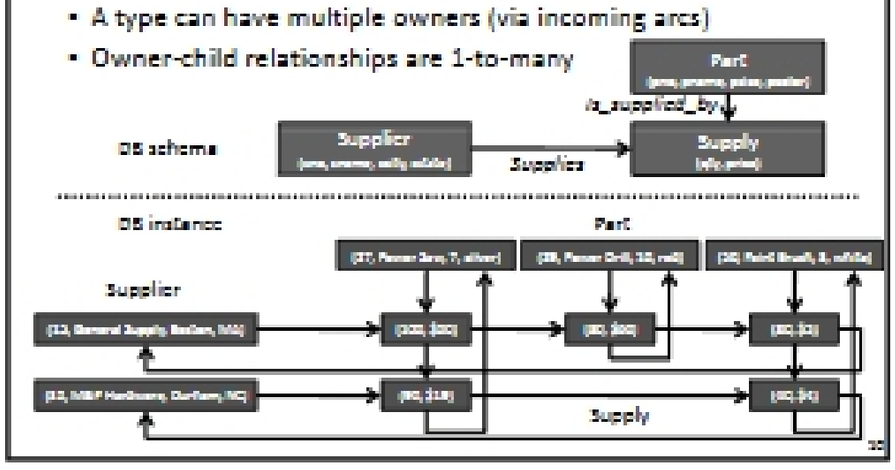
- Store data in two physical databases
  - No redundancy
- IMS grafts together the two to present a logical view to programmers
  - But lots of restrictions and complexity
  - No complete transparency

Lessons

- Lesson 1: physical/logical data independence is good
  - Δ data $\gg \Delta$ app
  - Changes to physical/logical representations of data should not require expensive changes to apps
- Lesson 2: tree-structured data models are restrictive
  - Force navigation one way
  - Need extensions/hacks to be general
- Lesson 3: logical reorganization of tree-structured data is hard
- Lesson 4: record-at-a-time interface forces programmer to do manual query optimization

Graph/network: CODASYL (1969)

- A directed graph where nodes are records types and arcs are "sets" (relationships)
  - A type can have multiple owners (via incoming arcs)
  - Owner-child relationships are 1-to-many



CODASYL programming

- Bachmann (Turing Award 1973): program by navigation

```
get unique Supplier with sno = 15
until failure do
    get next Supply in Supplies
    get owner Part through Is_supplied_by
    check pcolor = red
```

- Alternatively, start navigating from Parts

Improvements & limitations

- Model is powerful itself to avoid redundancy and dependency on owners' existence
- Arcs are just binary, though n-ary relationships can be simulated
- Language is still record-at-a-time
- Programming over graphs is harder than over trees
- Less logical data independence than IMS
- Still no physical data independence

Lessons

- Lesson 5: graphs are more flexible than trees but more complex
- Lesson 6: loading and recovering graphs is harder than trees

Relational (1970)

- Started with the 1970 proposal by Codd (Turing Award 1981)
  - Motivated by heavy maintenance required with IMS applications
- Data stored in flat tables—no nesting
- High-level, set-oriented language
- Underlying physical storage is completely up to vendors
- Example schema and query

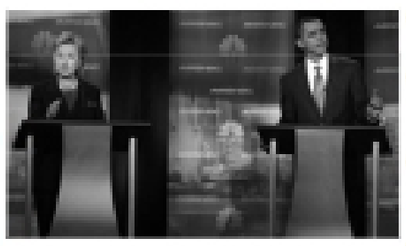
  Supplier(sno, sname, scity, sstate)
  Part(pno, pname, psize, pcolor)
  Supplies(sno, pno, qty, price)

```
SELECT * FROM Supplier, Supplies, Part
WHERE Supplier.sno = 16 AND Part.pcolor = 'red'
AND Supplier.sno = Supplies.sno AND Supplies.pno = Part.pno;
```

The "Great Debate"

- Ideological battle throughout the 1970's
  - Codd et al. advocating relational
  - Bachman et al. advocating CODASYL (graph/network)

- *Relational languages too hard*
- *Implementing relational model efficiently too difficult*
- *CODASYL able to simulate relational*



- *CODASYL too complex*
- *Too much dependence on data layout*
- *Record-at-time too hard to optimize*
- *Relational better for complex relationships*

Image by NY Times, http://graphics8.nytimes.com/images/2007/10/31/us/1104debate-sarge8.jpg

How was it resolved?

- Both parties adopted many of each other's policies while pretending to remain at oppose sites of the ideological spectrum
- IBM advocated the relational model, and won in the marketplace due to its dominant position in the microcomputers industry

Lessons

- Lesson 7: set-at-time languages offer better physical data independence
  - Up to the DBMS to optimize physical structure based on data/workload characteristics
- Lesson 8: simpler data models lend themselves to better logical data independence
- Lesson 9: technological debates are often settled by dollars rather than ideas
- Lesson 10: query optimizers almost always better than a programmer optimizing manually
  - ➲ Are there exceptions?

E/R model

- Schema expressed in diagrams with "entity" sets connected by "relationship" sets

- Never caught on as a physical/implementation model, but very successful for modeling and DB design
  - Automatic mapping to relational schema possible
- Lesson 11: "relationships" are easier to understand than "functional dependencies"