

CMSC724: What goes around comes around

Amol Deshpande

University of Maryland, College Park

February 3, 2009

Data Modeling

- A data model theory typically involves
 - “structural part” – collection of concepts/constructs to *represent* data
 - “integrity part” – constraints to *ensure data integrity*
 - “manipulation part” – constructs for *manipulating* the data
- We would like it to be:
 - Sufficiently expressive – can capture real-world data well
 - Easy to use
 - Lends itself to good performance

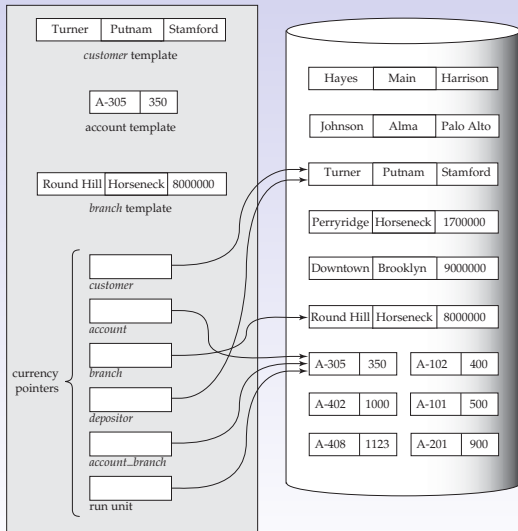
Data Models

- Hierarchical/IMS
 - Constructs: Hierarchy, keys, record types, DL/1 lang.
 - Issues:
 - Navigational interaction
 - No physical data independence
 - Repetition of information (m-to-n relationships)
 - Inability to represent information
 - Last two solved by a latter extension
 - Some logical data independence

Data Models

- Network/CODASYL
 - Constructs: “set” type, network structure
 - “programmer as a navigator”
 - Cons:
 - No physical or logical data independence
 - Bulk loading
 - 3-way relationships
 - Very complex programming constructs

Data Models: Network/CODASYL



```
customer.customer_city := "Harrison";  
find any customer using customer_city;  
while DB-status = 0 do  
  begin  
    get customer;  
    print (customer.customer_name);  
    find duplicate customer using customer_city;  
  end;
```

Figure A.20 Program work area.

Data Models

- Relational

- Constructs: Relations, relational algebra/calculus, functional dependencies
- Physical and logical data independence
- Cons:
 - Transitive closure
 - (initially) performance
 - (initially) too complex and mathematical languages

Data Models

- Don Chamberlin of IBM was an early CODASYL advocate (later co-invented SQL)

“He (Codd) gave a seminar and a lot of us went to listen to him. This was as I say a revelation for me because Codd had a bunch of queries that were fairly complicated queries and since I'd been studying CODASYL, I could imagine how those queries would have been represented in CODASYL by programs that were five pages long that would navigate through this labyrinth of pointers and stuff. Codd would sort of write them down as one-liners. These would be queries like, "Find the employees who earn more than their managers." [laughter] He just whacked them out and you could sort of read them, and they weren't complicated at all, and I said, "Wow." This was kind of a conversion experience for me, that I understood what the relational thing was about after that.”

Data Models

- Entity-relational
 - Constructs: entity, relationship
 - Limitations: Lack of query language, ease of mapping into relational
 - The conceptual model of choice to design the schema
- Relational++
 - Constructs: set-valued attributes, aggregation, generalization etc
 - Limitations: Didn't prove terribly useful either functionally or performance-wise

Data Models

- Semantic data models
 - Constructs: class, class variable, multiple inheritance etc
- OO
 - Essentially persistent PLs
 - Less impedance mismatch
 - Weak support for Xions, queries etc.
 - Niche market (e.g. CAD)

Data Models

- OR
 - Constructs: User-defined functions, types, access methods
- Semi-structured
 - “Schema last”
 - Constructs: DTD, XMLSchema, union types etc
 - Issues: Limited applicability ??, doesn't really solve semantic heterogeneity
 - Standard for wire format

Some lessons

- Physical/logical data independence desirable
- Record-at-a-time, navigational interfaces force manual query optimization and won't scale
- Technical debates settled by the marketplace in many cases
- KISS
- OO: Packages will not sell to users without “major pain”
- User-defined functions and access method effective
- Schema-last is probably a niche market
- Semantic heterogeneity not solved by XML