CSE 544
Principles of Database Management Systems

Magdalena Balazinska

Winter 2009

Lecture 2 – Early data models

References

- M. Stonebraker and J. Hellerstein. What Goes Around Comes Around. In "Readings in Database Systems" (aka the Red Book). 4th ed.
- R&G Book. Chapter 3: “The relational model”

Outline

- Different types of data
- Early data models
  - IMS
  - CODASYL
- Relational model (introduction)
- Other data models

Different Types of Data

- **Structured data**
  - All data conforms to a schema. Ex: business data
- **Semistructured data**
  - Some structure in the data but implicit and irregular
  - Ex: resume, ads
- **Unstructured data**
  - No structure in data. Ex: text, sound, video, images

- Our focus: structured data & relational DBMSs

Outline

- Different types of data
- Early data models
  - IMS – late 1960’s and 1970’s
  - CODASYL – 1970’s
- Relational model (introduction)
- Other data models

Early Proposal 1: IMS

- **Hierarchical data model**

- **Record**
  - **Type**: collection of named fields with data types (+)
  - **Instance**: must match type definition (+)
  - Each instance must have a **key** (+)
  - Record types must be arranged in a **tree** (-)

- **IMS database** is collection of instances of record types organized in a tree

IMS Example

- See Figure 2 in paper “What goes around comes around”

Data Manipulation Language: DL/1

- Each record has a hierarchical sequence key (HSK)
  - Records are totally ordered: depth-first and left-to-right
- HSK defines semantics of commands:
  - get_next
  - get_next_within_parent
- **DL/1 is a record-at-a-time language**
  - Programmer constructs an algorithm for solving the query
  - Programmer must worry about query optimization

Data storage

- Root records
  - Stored sequentially (sorted on key)
  - Indexed in a B-tree using the key of the record
  - Hashed using the key of the record

- Dependent records
  - Physically sequential
  - Various forms of pointers

- Selected organizations restrict DL/1 commands
  - No updates allowed with sequential organization
  - No “get-next” for hashed organization

Data Independence

- **Physical data independence**: Applications are insulated from changes in **physical storage details**
- **Logical data independence**: Applications are insulated from changes to **logical structure of the data**
- **Why are these properties important?**
  - Reduce program maintenance as
  - Logical database design changes over time
  - Physical database design tuned for performance

IMS Limitations

- **Tree-structured data model**
  - **Redundant** data, existence **depends on parent, artificial structure**
- **Record-at-a-time** user interface
  - User must specifies **algorithm** to access data
- **Very limited physical independence**
  - Phys. organization limits possible operations
  - Application programs break if organization changes
- Provides **some logical independence**
  - DL/1 program runs on logical database
  - Difficult to achieve good logical data independence with a tree model

Early Proposal 2: CODASYL

- **Networked data model**

- Primitives are also **record types** with **keys** (+)
- Record types are organized into **network** (-)
  - A record can have multiple parents
  - Arcs between records are named
  - At least one entry point to the network
- Network model is **more flexible than hierarchy**
  - Ex: no existence dependence
- **Record-at-a-time** data manipulation language (-)

CODASYL Example

- See Figure 5 in paper “What goes around comes around”

CODASYL Limitations

- **No physical data independence**
  - Application programs break if organization changes
- **No logical data independence**
  - Application programs break if organization changes
- Very **complex**
- Programs must “**navigate** the hyperspace”
- Load and recover as **one gigantic object**

Outline

- Different types of data
- Early data models
  - IMS
  - CODASYL
- Relational model (introduction) – 1970's and early 1980's
- Other data models

Relational Model Overview

- Proposed by Ted Codd in 1970
- Motivation: better logical and physical data independence
- Overview
  - Store data in a **simple data structure** (table)
    - Facilitates logical data independence
    - Flexible enough to represent almost anything
  - Access data through **set-at-a-time** language
    - Facilitates physical data independence
  - **No need for physical storage proposal**

Great Debate

- Pro relational
  - CODASYL is too complex
  - CODASYL does not provide sufficient data independence
  - Record-at-a-time languages are too hard to optimize
  - Trees/networks not flexible enough to represent common cases
- Against relational
  - COBOL programmers cannot understand relational languages
  - Impossible to represent the relational model efficiently
  - CODASYL can represent tables
- Ultimately settled by the market place

Outline

- Different types of data
- Early data models
  - IMS
  - CODASYL
- Relational model (introduction)
- Other data models

Other Data Models

- Entity-Relationship: 1970's
  - Successful in logical database design (lecture 4)
- Extended Relational: 1980's
- Semantic: late 1970's and 1980's
- Object-oriented: late 1980's and early 1990's
  - Address impedance mismatch: relational dbs ←→ OO languages
  - Interesting but ultimately failed (several reasons, see paper)
- Object-relational: late 1980's and early 1990's
  - User-defined types, ops, functions, and access methods
- Semi-structured: late 1990's to the present

Summary

- Data independence is desirable
  - Both physical and logical
  - Early data models provided very limited data independence
  - Relational model facilitates data independence
    - Set-at-a-time languages facilitate phys. indep. [more next lecture]
    - Simple data models facilitate logical indep. [more next lecture]
- Flat models are also simpler, more flexible
- User should specify what they want not how to get it
  - Query optimizer does better job than human
- New data model proposals must
  - Solve a “major pain” or provide significant performance gains