CMSC724: What goes around comes around

Amol Deshpande

University of Maryland, College Park

January 31, 2012

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

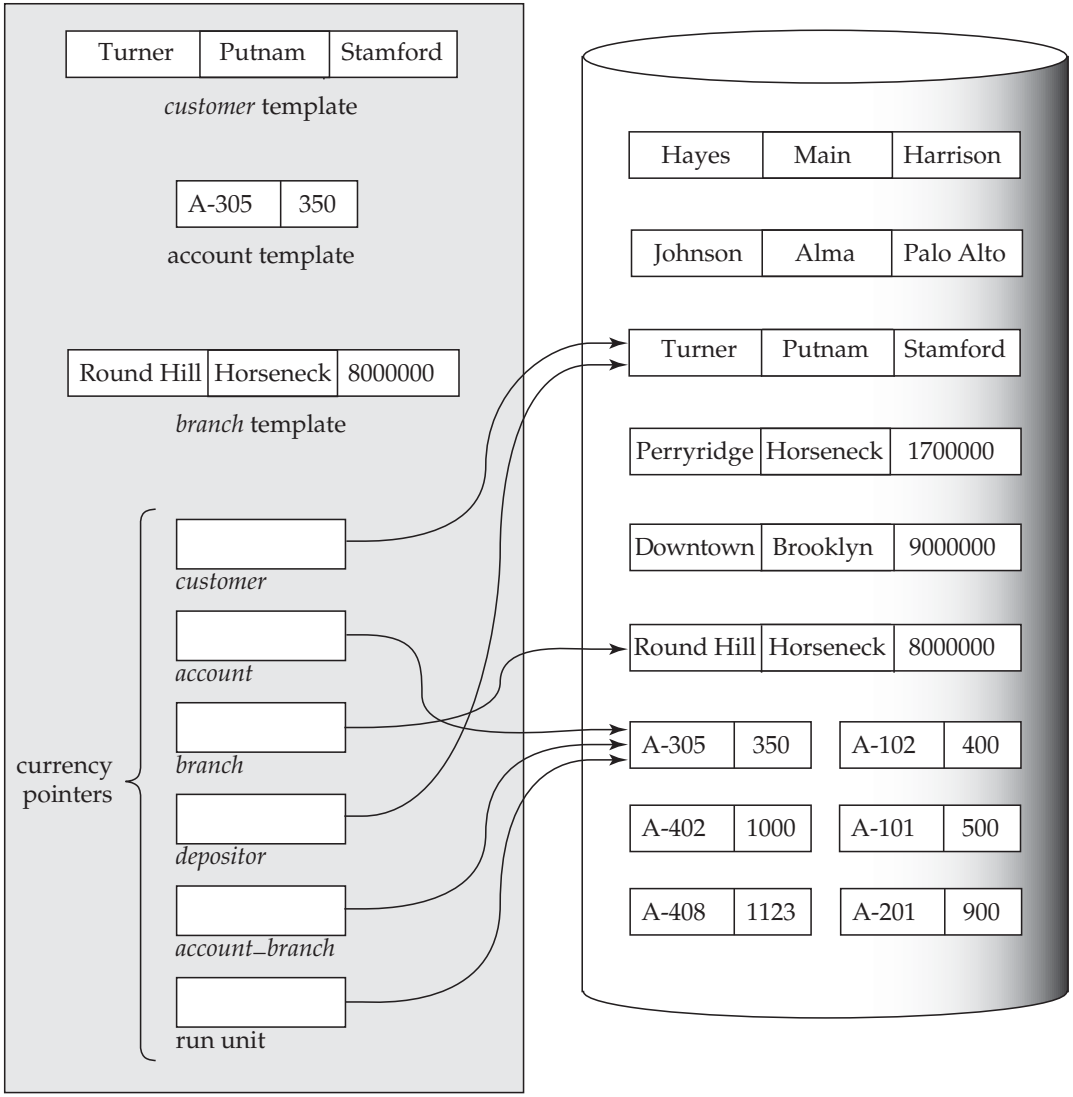
Data Models

- 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.”

Network Data Model: Example Program

- From article in: ACM Computing Surveys, 1976

```
IDENTIFICATION DIVISION.
PROGRAM-NAME. SAMPLE-QUERY.
ENVIRONMENT DIVISION.
   identification of machine environment and
   declaration of non-data-base files
   (i.e., standard COBOL files)

DATA DIVISION.
FILE SECTION.
DB   PRES-ADMIN-STATE-INFO WITHIN PRESIDENTIAL.
FD   REPORT-FILE.
       remainder of data item entries which make
       up a standard COBOL file.
WORKING-STORAGE SECTION.
77   PRESIDENT-COUNT USAGE COMPUTATIONAL   PIC 999.
77   DONE   PIC 9(5) VALUE "04021".
77   NO-MORE-SONS   PIC A(5).
77   NO-MORE-STATES   PIC A(5).


   PROCEDURE DIVISION.
   DECLARATIVES.
   EXPECTED-ERROR SECTION.
     USE FOR DATABASE-EXCEPTION ON "04021".
   EXPECTED-ERROR-HANDLING.
     EXIT.
   UNEXPECTED-ERROR SECTION.
     USE FOR DATABASE-EXCEPTION ON OTHER.
   UNEXPECTED-ERROR-HANDLING.
     here we would process unexpected error
     conditions.
   END DECLARATIVES.
```

```
INITIALIZATION.
  READY PRESIDENTIAL-AREA.
  OPEN non-database COBOL files.
  MOVE "FALSE" TO NO-MORE-STATES.
  FIND FIRST STATE IN ALL-STATES-SS.
  PERFORM PROCESS-STATE THRU FINISH-STATE
    UNTIL NO-MORE-STATES = "TRUE".
  GO TO FINISH-UP.
PROCESS-STATE.
  MOVE 0 TO PRESIDENT-COUNT.
  IF NATIVE-SON IS EMPTY
    MOVE "TRUE" TO NO-MORE-SONS,
    ELSE MOVE "FALSE" TO NO-MORE-SONS.
  PERFORM COUNT-NATIVE-SONS
    UNTIL NO-MORE-SONS = "TRUE".
  GO TO FINISH-STATE.
COUNT-NATIVE-SONS.
  FIND NEXT PRESIDENT IN NATIVE-SON.
  IF DATABASE-STATUS = DONE
    MOVE "TRUE" TO NO-MORE-SONS
    ELSE ADD 1 TO PRESIDENT-COUNT.
FINISH-STATE.
  IF PRESIDENT-COUNT IS GREATER THAN 1
    FIND STATE CURRENT,
    GET STATE,
    write out state name and president count.
  FIND NEXT STATE IN ALL-STATES-SS
  IF DATABASE-STATUS = DONE
    MOVE "TRUE" TO NO-MORE-STATES.
FINISH-UP.
  FINISH PRESIDENTIAL-AREA.
  CLOSE non-database COBOL files.
  STOP RUN.
```

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 heterogeniety
  - 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