CSCI 403
Database Management

2

High-Level Concepts

History

CS@Mines

What is data? What is a database?

Discussion time!

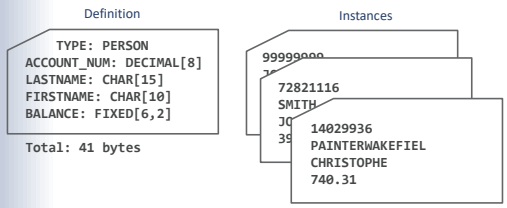
EARLY DAYS

Early Data Storage

Typical approach:

- Define application-specific fixed-length "record"
  - Within the record, fixed-length "fields"
  - Byte-for-byte equivalent of structure in memory
  - Store records on some storage medium:
    - Punch card
    - Paper or magnetic tape
    - Disk

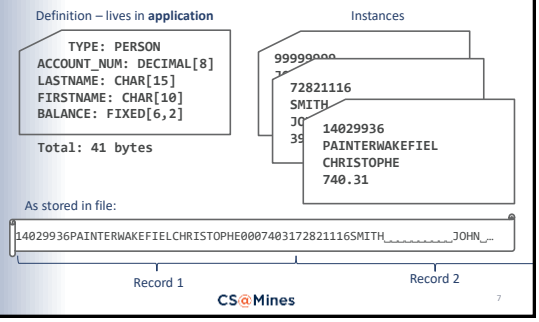
Records



Data Access

- Punch cards/tape:
  - Little idea – that was before my time ☺
  - Usually involved a specialist computer *operator*
  - Often just loaded everything into memory
- Files on disk (random access):
  - If you want record *n*, multiply by record size to find byte offset from start of file
  - Indices to speed search on specified fields

Before Databases



Application Code Example

```
typedef struct {
    char account_number[8];
    char lastname[15];
    char firstname[10];
    char balance[8];
} person_record;

…

person get_person_by_index(int n) {
    FILE* person_file;
    person_record p;
    person_file = fopen("person.dat", "r");
    fseek(person_file, n * sizeof(person_record), SEEK_SET);
    fread(&p, sizeof(person_record), 1, person_file);
    fclose(person_file);
    return p;
}
```

Note, this is without error checking code…

Code Example, continued

```
person find_person_by_account_number(char[] account) {
    FILE* person_file;
    person_record p;
    person_file = fopen("person.dat", "r");
    while (!feof(person_file)) {
        fread(&p, sizeof(person_record), 1, person_file);
        if (!strncmp(p.account_number, account, sizeof(p.account_number))) {
            fclose(person_file);
            return p;
        }
        fseek(person_file, sizeof(person_record) - sizeof(p.account_number),
              SEEK_CUR);
    }
    fclose(person_file);
    strcpy(p.account_number, "NOMATCH");
    return p;
}
```


Some Issues

- Hard-coded record definition in code: what if definition needed to change?
  - Re-write, rebuild, test all software
  - Write special code to migrate data from old to new format
- Performance
  - Sequential search expensive, unless you can hold all data in memory
  - Could make an index – but then have to maintain index as well
- Flexible access – how do we add/insert/delete?
- Application specific
  - This code only works for person records
  - Must write the same code over and over for different applications

MODERN ERA

Modern Databases

Typical properties:

- Self-describing
- Program-data separation
- Storage abstraction
- Network multi-user access
- Client-server architecture

Self-Describing

Suppose your "database" contains:

Without knowing the record structure, can you figure out what data is being stored?

You need the record definition, too…

Metadata

- Store data description (metadata) with data
  - The metadata is stored in the database *catalog*
  - The catalog is in same format as any other data
    - Thus, the catalog metadata is stored in the catalog!
- Just need to know how to read metadata from the catalog
  - Lets us describe/store many record types
  - Any application that can query the catalog can query any record type

Program-Data Separation

- Programs can evolve independently of data
- Without separation, a change to definition →
  - Re-code and rebuild all software
  - Migrate all data (by loading all records and re-writing in new format)
- Follows from self-describing
  - Many changes to data definition are non-breaking to application (some still are, though…)

Data Abstraction

Owner to DBMS (Database Management System):

Q. Here's my data. How will you store it?

A. Why should you care?
- When you give me data, I will store it.
- When you ask for the data, I will give it to you.
- I may change how I store your data from time to time.

A DBMS is much like a bank - you don't get (or need) the keys to the safe.

Network Multi-User Access

- Name kind of says it all
  - Multiple users with simultaneous access
  - Accessed remotely via network
- Eliminates bottlenecks
- Requires sophisticated *transaction control*
  - Updates from one user should not destroy updates from another user
  - Airline ticketing example

Client-Server Architecture

- DBMS software lives on server
- Applications talk to server to via standard protocol
- Similar to e.g. web browser/web server:
  - One web server (http://mines.edu)
  - Many users (and browsers – Chrome, Safari, Firefox…)
- Supports data abstraction, program-data separation

DATABASE HISTORY

Quick Timeline

- 1956 – Hard drive technology introduced (IBM)
- 1962 – "data-base" term invented (per OED)
- 1964 – Integrated Data Store (IDS) released (GE)
  - First "network model" DBMS
- 1966-1968 – Information Management System (IMS) released (IBM)
  - First "hierarchical model" DBMS
- 1969 – CODASYL network database standard
- 1970 – Paper by E.F. Codd (IBM) on relational model
- 1973 – Start of INGRES (Berkeley research project)
- 1977 – System R released (IBM)
  - First commercial relational model DBMS
  - Introduction of SQL
- 1979 – Oracle released (Relational Software, now Oracle)
- 1985 – Start of POSTGRES (Berkeley, successor to INGRES)

Network Model

- 1964 - Integrated Data Store (IDS)
  - Created by Charles Bachman (1924-2017) of GE
    - 1973 Turing Award winner
    - Also invented (1965) early transaction control system
  - Graph-based storage of records
    - Records organized into named types
    - "Sets" defining relationships between record types
  - All records keyed with unique ID
    - Disk location computable from ID
    - Allowed fast navigation between records linked by ID
- Later standardized by CODASYL
  - Closely tied to COBOL language
  - Many vendors – at least one still sold today!

Hierarchical Model

- 1966-1968 – Information Management System (IMS)
  - Invented at IBM
  - Created for Apollo space program (tracking parts for the Saturn V rocket)
  - Still sold today!
- Records form a tree structure
  - Think file structure on modern OS
  - Fast navigation by pointers

(not actual size)

Navigational DBMS

- Network and hierarchical model are "navigational"
  - Access to record is predicated on knowing key value
  - Data retrieval follows linkages (like pointers)
- The relational model is completely different…

Relational Model

- 1970 – "A Relational Model of Data for Large Shared Data Banks"
  - E.F. Codd (1923-2003) of IBM
    - Turing Award, 1981
  - Initially met resistance inside IBM
    - No desire to cannibalize success of IMS
  - Eventually productized as System R (1977)
    - System R also introduced SEQUEL (later SQL)
- More on the relational model soon…

PostgreSQL

- INGRES project (Berkeley) started in 1973
  - Michael Stonebraker (1943-)
    - Turing Award, 2014
  - Eugene Wong
  - Based on technical papers from System R project
  - QUEL query language
  - Briefly commercialized
  - Students from this project later founded Sybase
    - Technology from Sybase now MS SQL Server
- POSTGRES project (Berkeley) started in 1985
  - Successor to INGRES
  - Goal to address problems with relational databases of the time
  - Open sourced in 1994
  - Postgres95 with SQL in 1995 (renamed PostgreSQL in 1996)

Relational Model Preview

- Moves away from pointer-based (navigational)
- Based on set theory
- Flexible – dynamic views of data created as needed
- Initially slow compared to navigational, but now the dominant technology
  - Dramatically improves data abstraction and program-data separation
  - Oracle, SQL Server, PostgreSQL, MySQL, etc.

New (Old) Ideas

- 1990s – OODBMSes
  - Persistent store for objects
  - Came with rise of object-oriented programming (OOP)
  - Essentially reverts to navigational model
  - Subsumed by RDBMSes like Oracle, PostgreSQL
- New data types – XML, BLOB, GIS
  - Also subsumed by RDBMSes
- Recent: NoSQL ("Not Only SQL")
  - Response to demands of Big Data
  - Lots of flavors
  - We'll talk more about these near the end of the course
  - Some reversion to navigational in these, too
- Current: NewSQL - relational guarantees + Big Data robustness

Up Next

- Next lecture:

  Informal introduction to queries in SQL.

- Reading: Chapter 6: "Basic SQL"

- Friday, January 11
  - Project 0 due
  - Project 1 – Connect assigned