Introduction to database management systems

Lecture 1

What is database

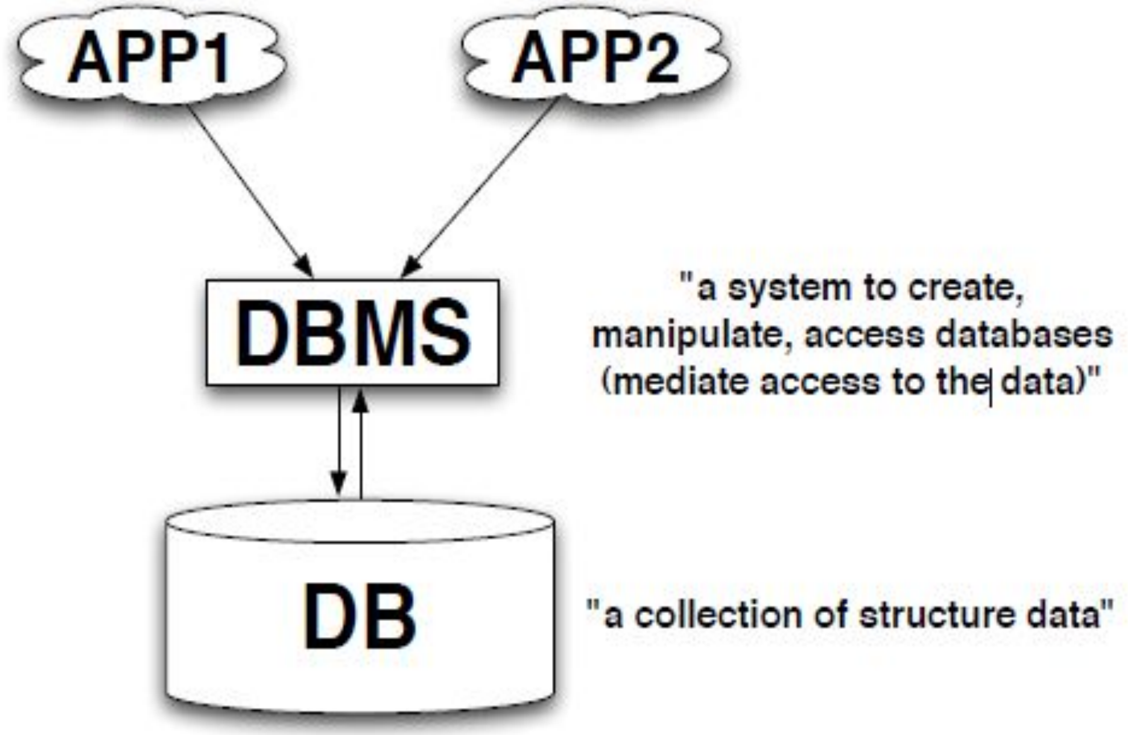
A database is a collection of structured data. A database captures an abstract representation of the domain of an application.

- Typically organized as “records” (traditionally, large numbers, on disk)
- and relationships between records

What is DBMS

- A DBMS is a (usually complex) piece of software that sits in front of a collection of data, and mediates applications accesses to the data, guaranteeing many properties about the data and the accesses.

SCHEMA


APP1
APP2
DBMS
"a system to create, manipulate, access databases (mediate access to the data)"
DB
"a collection of structure data"

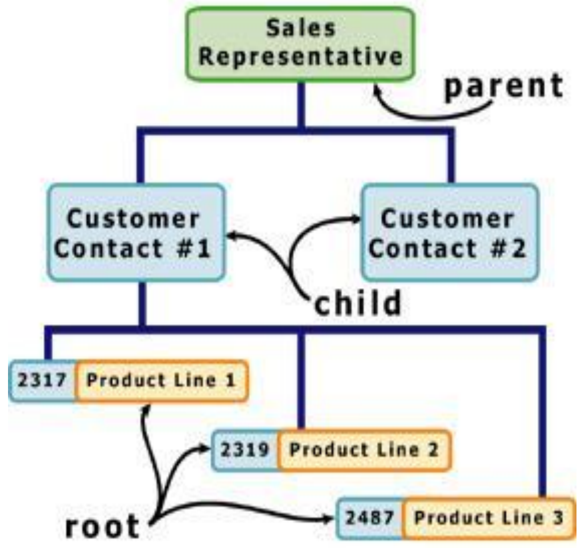
Brief History

- The first general-purpose DBMS designed by Charles Bachman at General Electric in the early 1960s, and formed the basis for network database model
- In the late 1960s, IBM developed the Information Management System (IMS), and formed the basis for hierarchical database model
- In 1970, Edgar Codd, at IBM's San Jose Research Laboratory, proposed a new data representation framework called the *relational data model*
- SQL was standardized in the late 1980s, and the current standard, SQL:1999, was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).

Types of Databases

- Hierarchical database
- Network database
- Relational database
- Object-oriented database
- NoSQL
  - Graph Oriented Database - OrientDB
  - Column Oriented Database - HBase
  - Document Oriented Database - MongoDB

HIERARCHICAL DATABASE



- *A DBMS is said to be hierarchical if the relationships among data in the database are established in such a way that one data item is present as the subordinate of another one.*
- Here subordinate means that items have 'parent-child' relationships among them. Direct relationships exist between any two records that are stored consecutively. The data structure "tree" is followed by the DBMS to structure the database. No backward movement is possible/allowed in the hierarchical database.

NETWORK DATABASE

- *A DBMS is said to be a Network DBMS if the relationships among data in the database are of type many-to-many.*
- The relationships among many-to-many appears in the form of a network. Thus the structure of a network database is extremely complicated because of these many-to-many relationships in which one record can be used as a key of the entire database. A network database is structured in the form of a graph that is also a data structure.

RELATIONAL DATABASE

- *A DBMS is said to be a Relational DBMS or RDBMS if the database relationships are treated in the form of a table.* there are three keys on relational DBMS 1)relation 2)domain 3)attributes.
- *A network means it contains fundamental constructs sets or records.sets contains one to many relationship, records contains fields statical table that is composed of rows and columns is used to organize the database and its structure and is actually a two dimension array in the computer memory. A number of RDBMSs are available, some popular examples are Oracle, Sybase, Ingress, Informix, Microsoft SQL Server, and Microsoft Access.*

OBJECT-ORIENTED DATABASE

- *Object-oriented databases use small, reusable chunks of software called objects. The objects themselves are stored in the object-oriented database. Each object consists of two elements: 1) a piece of data (e.g., sound, video, text, or graphics), and 2) the instructions, or software programs called methods, for what to do with the data.*
- *Object-oriented databases have two disadvantages. First, they are more costly to develop. Second, most organizations are reluctant to abandon or convert from those databases that they have already invested money in developing and implementing. However, the benefits to object-oriented databases are compelling. The ability to mix and match reusable objects provides incredible multimedia capability.*

FILE SYSTEMS VERSUS A DBMS

- Data independence – physical storage system is hidden from the final user
- Efficient Data access – the procedures to store and extract data handled by the DBMS core
- Data Integrity and Security – Intrinsic Authentications and Authorizations. Relations of the entities monitored by DBMS
- Data Administration
- Concurrent Access and Crash Recovery
- Application Development Time

LEVELS OF ABSTRACTION

- Conceptual
  - Entities and Relations between Them
- Physical
  - File organization, storage selection for different kind of DBMS elemtns like indexes, relations,
- External
  - Usually interpreted like business cases level where conceptual schema transformed to the business needs

Queries in a DBMS

A very attractive feature of the relational model is that it supports powerful query languages. Relational calculus is a formal query language based on mathematical logic, and queries in this language have an intuitive, precise meaning. Relational algebra is another formal query language, based on a collection of operators for manipulating relations, which is equivalent in power to the calculus.

- **Data Description Language**
- **Data Manipulation Language**

TRANSACTION MANAGEMENT

- Airline reservations

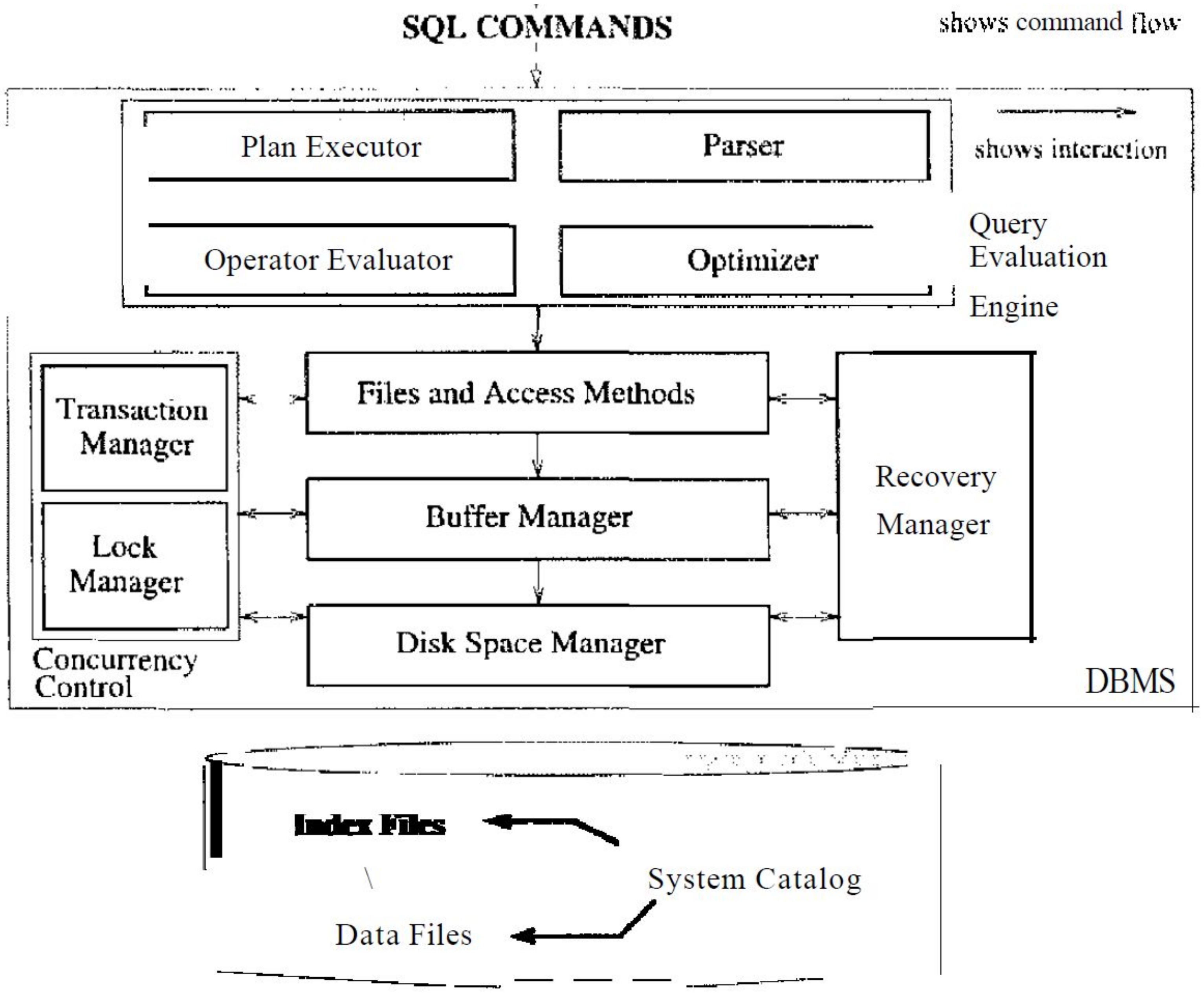
when one travel agent looks up Flight 100 on some given day and finds an empty seat, another travel agent may simultaneously be making a reservation for that seat, thereby making the information seen by the first agent obsolete.

- Bank’s database

While one user's application program is computing the total deposits, another application may transfer money from an account that the first application has just 'seen' to an account that has not yet been seen, thereby causing the total to appear larger than it should be.

Concurrency, Control and Recovery

- Every object that is read or written by a transaction is first locked in shared or exclusive mode, respectively. Placing a lock on an object restricts its availability to other transactions and thereby affects performance.
- For efficient log maintenance, the DBMS must be able to selectively force a collection of pages in main memory to disk. Operating system support for this operation is not always satisfactory.
- Periodic checkpointing can reduce the time needed to recover from a crash. Of course, this must be balanced against the fact that checkpointing too often slows down normal execution.


SQL COMMANDS
shows command flow
shows interaction
Plan Executor
Parser
Operator Evaluator
Optimizer
Query Evaluation Engine
Files and Access Methods
Buffer Manager
Disk Space Manager
Transaction Manager
Lock Manager
Concurrency Control
Recovery Manager
DBMS
Index Files
System Catalog
Data Files

TWO CONCEPTUAL USERS OF DBMS

- Application programmers
- Database Administrators where administrators responsibilities are often next:
    - Design of the Conceptual and Physical Schemas
    - Security and Authorization
    - Data Availability and Recovery from Failures
    - Database Tuning

Questions

- What are the main benefits of using a DBMS to manage data in applications involving extensive data access?
- When would you store data in a DBMS instead of in operating system files and vice-versa?
- What is a data model? What is the relational data model? What is data independence and how does a DBMS support it?
- Explain the advantages of using a query language instead of custom programs to process data.
- What is a transaction? What guarantees does a DBMS offer with respect to transactions?
- What are locks in a DBMS, and why are they used? What is write-ahead logging, and why is it used? What is checkpointing and why is it used?
- Identify the main components in a DBMS and briefly explain what they do.
- Explain the different roles of database administrators, application programmers, and end users of a database. Who needs to know the most about database systems?

EXERCISES

Exercise 1.1 Why would you choose a database system instead of simply storing data in operating system files? When would it make sense *not* to use a database system?

Exercise 1.2 What is logical data independence and why is it important?

Exercise 1.3 Explain the difference between logical and physical data independence.

Exercise 1.4 Explain the difference between external, internal, and conceptual schemas. How are these different schema layers related to the concepts of logical and physical data independence?

Exercise 1.5 What are the responsibilities of a DBA? If we assume that the DBA is never interested in running his or her own queries, does the DBA still need to understand query optimization? Why?

Exercise 1.6 Scrooge McNugget wants to store information (names, addresses, descriptions of embarrassing moments, etc.) about the many ducks on his payroll. Not surprisingly, the volume of data compels him to buy a database system. To save money, he wants to buy one with the fewest possible features, and he plans to run it as a stand-alone application on his PC clone. Of course, Scrooge does not plan to share his list with anyone. Indicate which of the following DBMS features Scrooge should pay for; in each case, also indicate why Scrooge should (or should not) pay for that feature in the system he buys.

1. A security facility.
2. Concurrency control.
3. Crash recovery.
4. A view mechanism.
5. A query language.

Exercise 1.7 Which of the following plays an important role in *representing* information about the real world in a database'? Explain briefly.

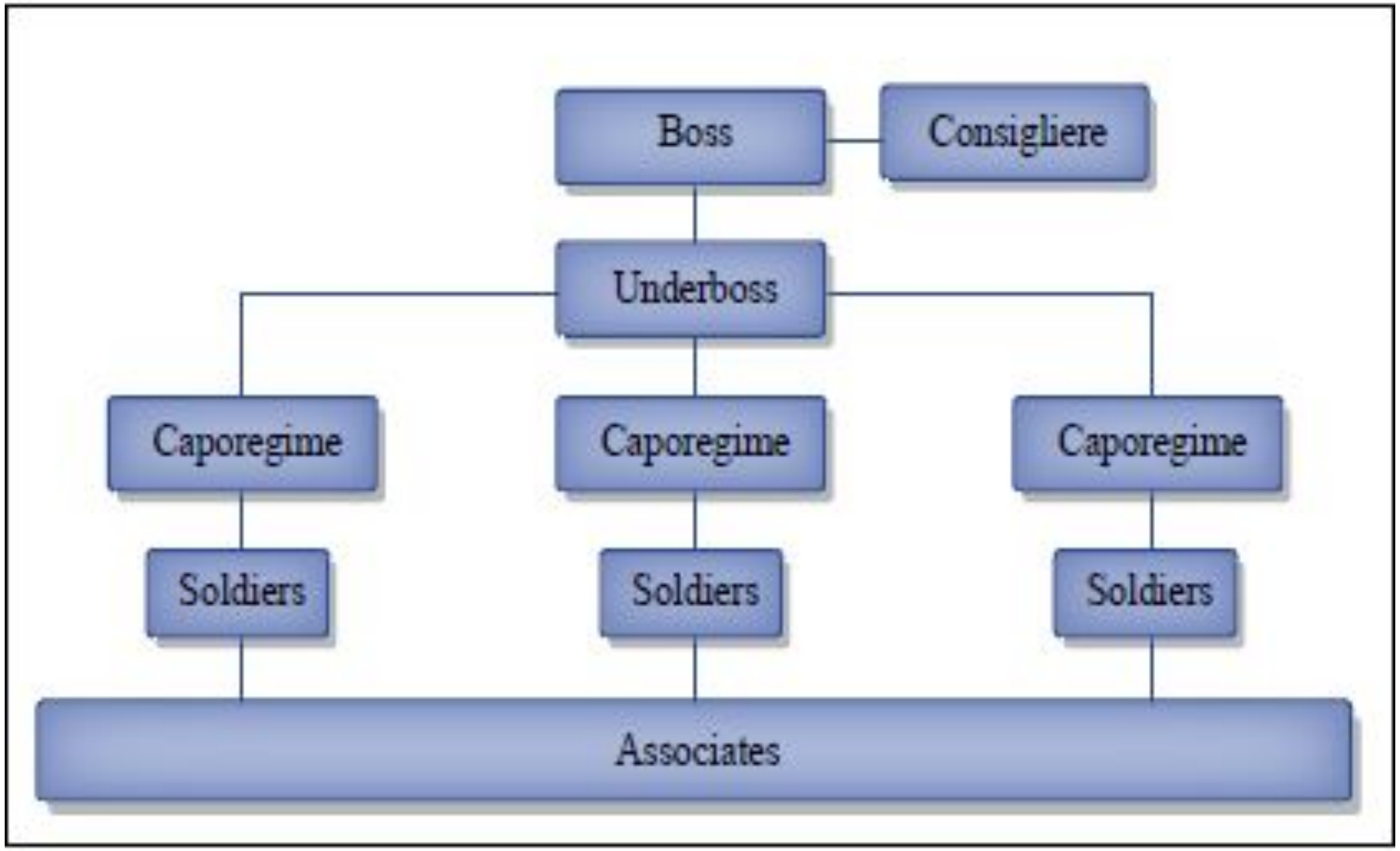
1. The data definition language.
2. The data manipulation language.
3. The buffer manager.
4. The data model.

Exercise **1.8** Describe the structure of a DBMS. If your operating system is upgraded to support some new functions on as files (e.g., the ability to force some sequence of bytes to disk), which layer(s) of the DBMS would you have to rewrite to take advantage of these new functions?

Exercise 1.9 Answer the following questions:

1. What is a transaction?
2. Why does a DBMS interleave the actions of different transactions instead of executing transactions one after the other?
3. What must a user guarantee with respect to a transaction and database consistency? What should a DBMS guarantee with respect to concurrent execution of several transactions and database consistency'?
4. Explain the strict two-phase locking protocol.
5. What is the WAL property, and why is it important?

Practical case - MAFIA



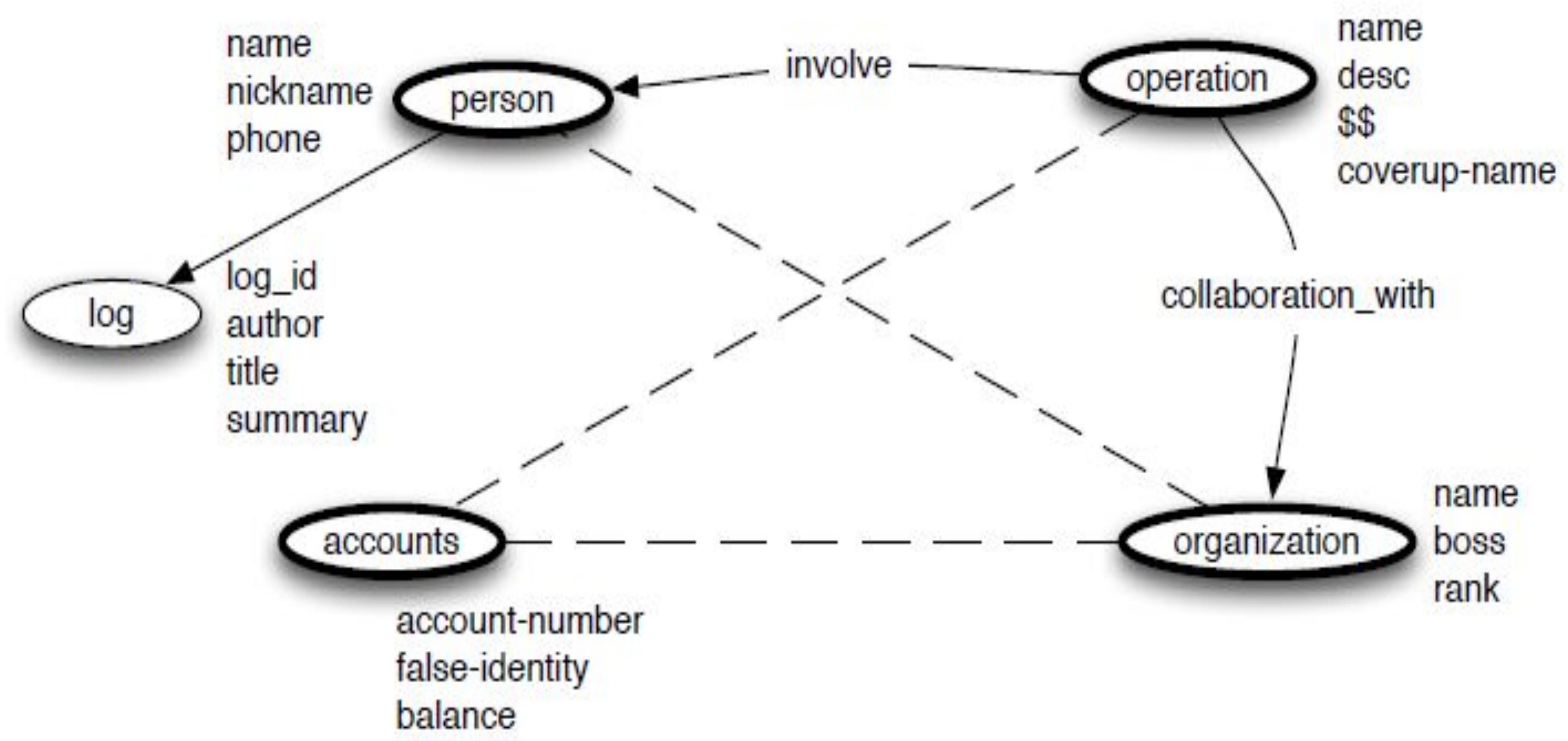
I need to store information about

- people that work for me (soldiers, caporegime, etc..)
- organizations I do business with (police, ’Ndrangheta, politicians)
- completed and open operations:
    - protection rackets
    - arms trafficking
    - drug trafficking
    - loan sharking
    - control of contracting/politics
    - I need to avoid that any of my man is involved in burglary, mugging, kidnapping (too much police attention)
    - cover-up operations/businesses
    - money laundry and funds tracking
- assignment of soldiers to operations
- etc...

I will need to share some of this information with external organizations I work with, protecting some of the information.

- Therefore I need:
  - the boss, underboss and consigliere should be able to access all the data and do any kind of operations (assign soldiers to operations, create or shutdown operations, pay cops, check the total state of money movements, etc...)
  - the accountants (20 of them) access to perform money book-keeping (track money laundering operations, move money from bank to bank, report bribing expenses)
  - the soldiers (5000) need to report daily misdeeds in a daily-log, and report money expenses and collections
  - the semi-public interface accessible by other bosses I collaborate with (search for cops on our books, check areas we already cover, etc..)

What data to store



What to Consider

- What to represent:, what are the key entities in the real world I need to represent? how many details?
- How to store data: maybe we can use just files: people.txt, organizations.txt, operations.txt, money.txt, daily-log.txt. Each files contains a textual representation of the information with one item per line.
- Control access credentials at low granularity: accountants should know about money movement, but not the names and addresses of our soldiers. Soldiers should know about operations, but not access money information
- How to access data: we could write a separate procedural program opening one or more files, scanning through them and reading/writing information in them.
- Access patterns and performance: how to find shop we didn’t collected money from for the longest time (and at least 1 month)? scan the huge operation file, sort by time, pick the oldest, measure time? (need to be timely or they will stop paying, and this get the boss mad... you surely don’t want that, and make sure no one is accessing it right now). “Tony Schifezza” is a mole, we need to find all the operations and people he was involved or knew about and shut them down... quick... like REAL quick!!!

- Atomicity: when an accountant moves money from one place to another you need to guarantee that either money are removed from account A and added to account B, or nothing at all happens... (You do not want to have money vanishing, unless you plan to vanish too!).
- Consistency: guarantee that the data are always in a valid state (e.g., there are no two operations with the same name)
- Isolation: multiple soldiers need to add to daily-log.txt at the same time (risk is that they override each other work, and someone get "fired" because not productive!!)
- Durability: in case of a computer crash we need to make sure we don't lose any data, nor that data get scrambled (e.g., If the system says the payment of a cop went through, we must guarantee that after reboot the operation will be present in the system and completed. The risk is police taking down our operation!)