Introduction to Databases

Thomas Schwarz, SJ

Beginnings: Data Base Management Systems

- 1960: “Computerization”
  - Needs “totally integrated management information”
    - Beyond the capabilities of computation then
      - But:
        - disk storage + tapes
        - better processors
        - interactive terminals
  - Actually: batch processing of large files
    - Weekly payroll processing, accounts payable, customer statement generation

Beginnings: IDS

- 1960: General Electric: Bachman (Turing Award):
  - MIACS Materials/Manufacturing/Management Information and Control system
  - Integrated Data Store (IDS)
  - Problem Controller operating system
    - early transactional support
  - Data Structure Diagram

Beginnings: IDS

- Two big problems:
    - Data is stored on tape in programmer defined formats
        - Adding a field or changing the format means rewriting all data
        - But the data is connected
        - So, one change breaks many programs
    - Today: We have things like file systems and standardized file formats that makes this easier

Beginnings: IDS

- Second problem:
  - Using Random access storage (disks) efficiently
    - Easy to instruct program to pull data from a disk at a certain spot
    - But need to find the spot
- Today: File systems make this so much easier

Beginnings: IDS

- IDS integrates data files
- Allows programming in high level programs
- IDS introduces:
    - Storing and manipulating metadata about records
    - Enforces relationships between different record types
        - Ergo: protects database integrity
- Introduces concept of transaction
    - A set of changes to records that need to be executed all or none at all
- First system that interacts with teletypers

Beginnings: IDS

- 1964: Backup and recovery system was added
- Report Generation was not integrated:
    - All access to database is programmatic

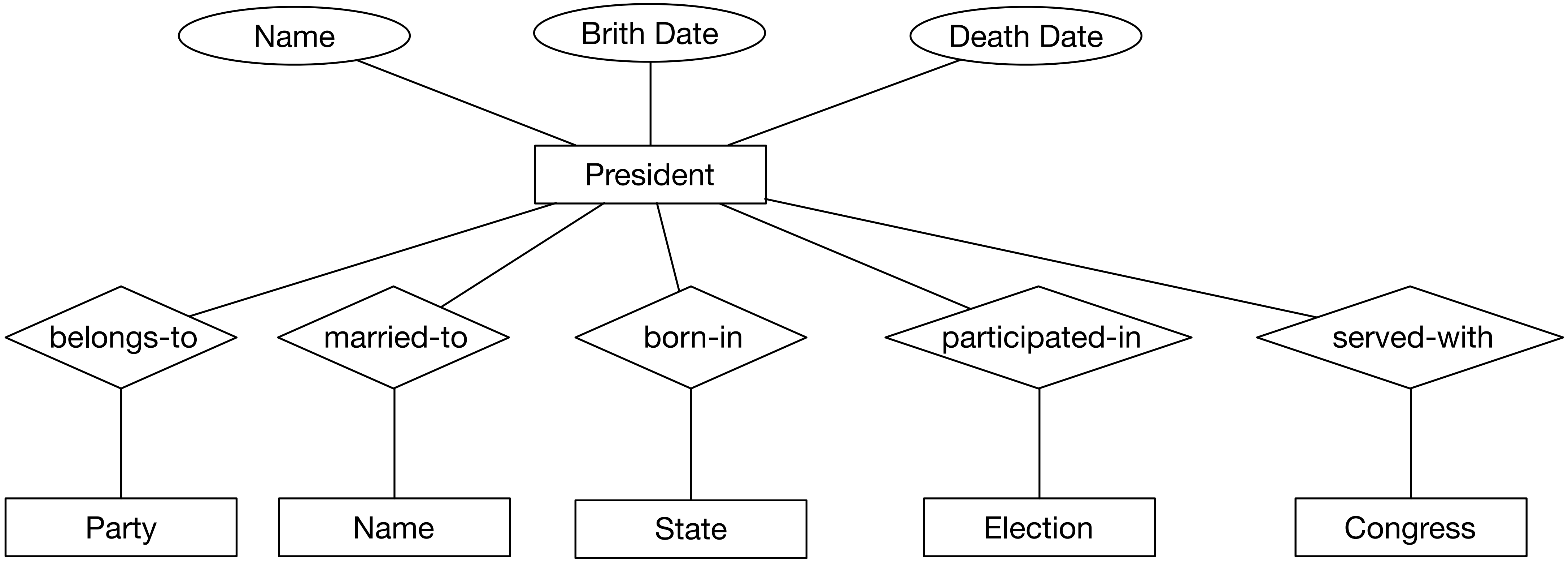
Beginnings: Codasyl

- CODASYL - standards body for data processing industry
  - Created and promoted COBOL
  - 1969: Report documenting foundational concepts and vocabulary:
    - Data Definition Language
    - Data Manipulation Language
    - Schemas
    - Data independence and program independence
  - **Adds:**
    - “Privacy locks”
    - Sub-schemas (like modern views)

Underlying technology

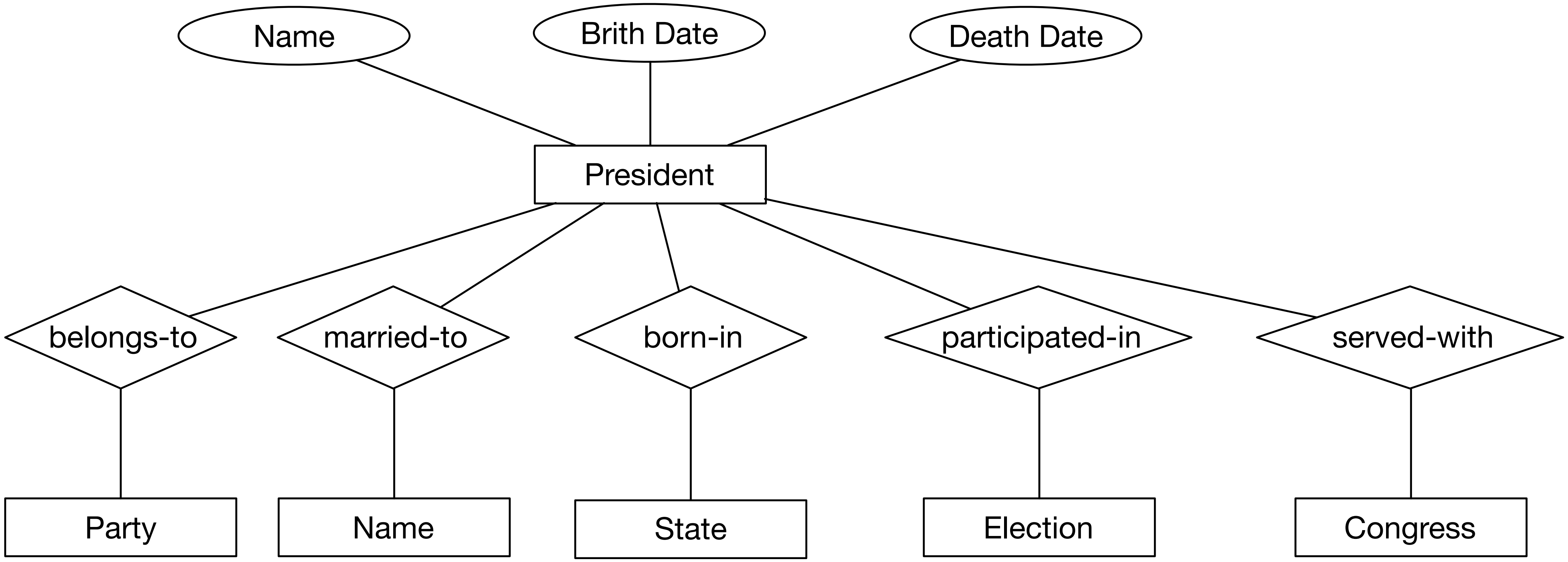
- “Network Data Model”
  - Data items are collections of *records*
  - Relationships among data is represented by *links*
    - basically, pointers

Presidents



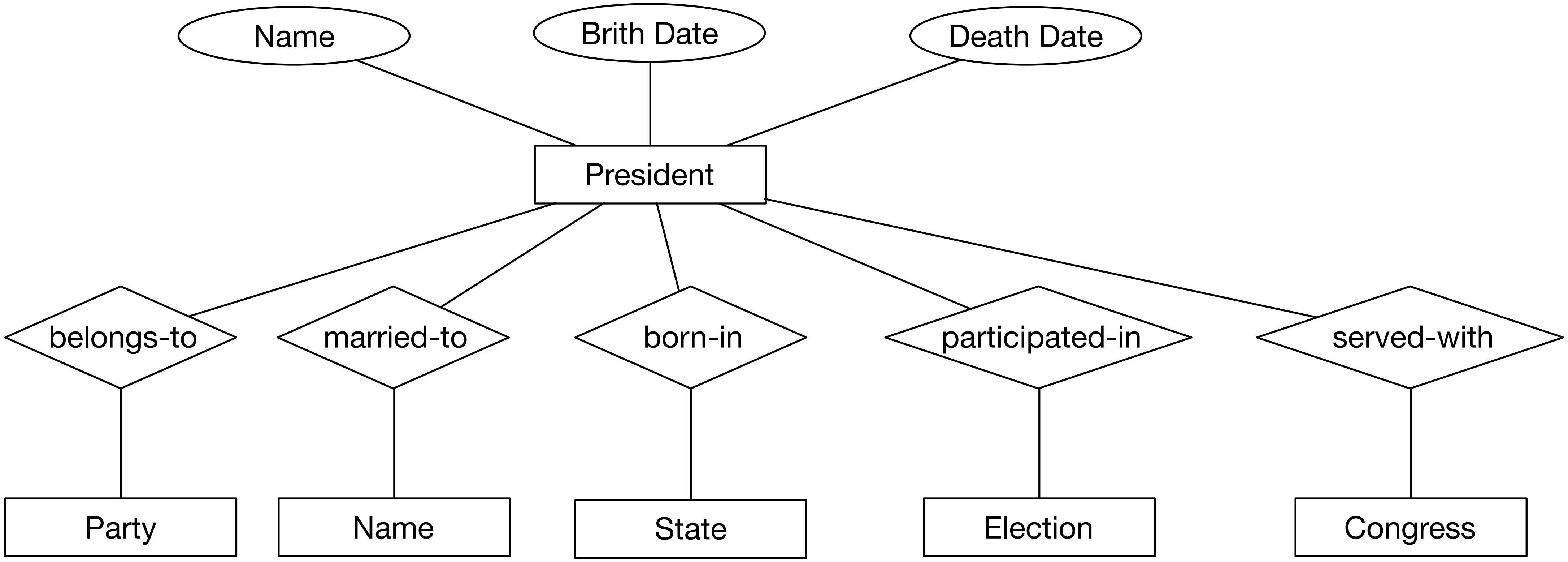
- Start out with the attributes of a president
    - Presidents have a name, a birth date, and a death date, though the latter might still be in the future
    - They belong to parties, but parties cannot be attributes, since some presidents belonged to more than one party.
        - Abraham Lincoln was a Whig who joined the Republican party when it was founded
        - Monroe did not want to belong to a party, but he was not a Federalist

Presidents



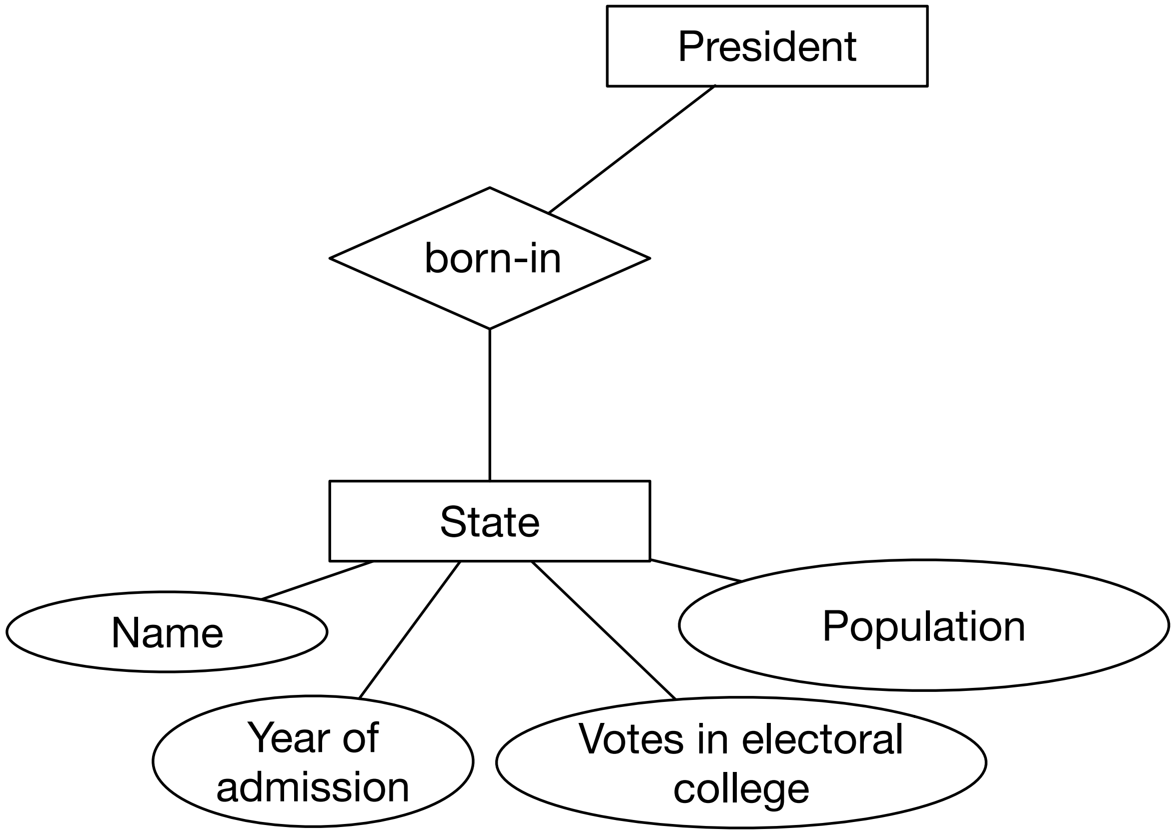
- Start out with the attributes of a president
    - Some presidents where married more than once, so the spouse cannot be an attribute

Presidents



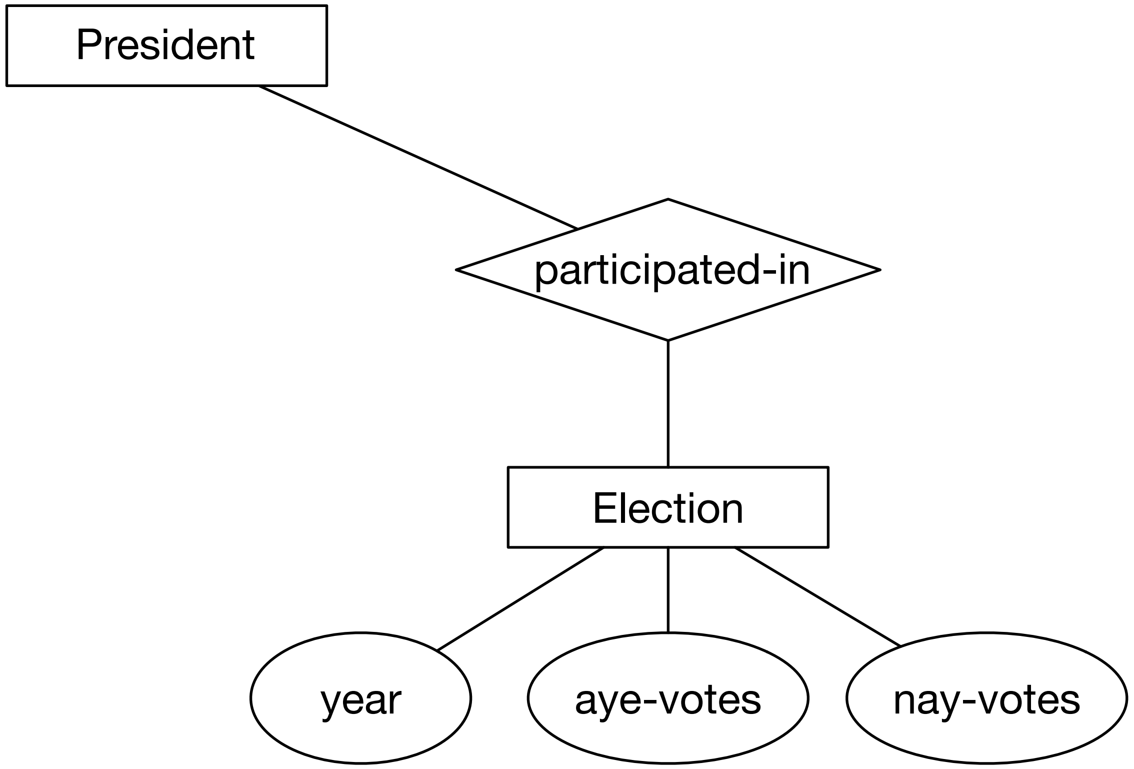
- Relationships are with other entities.
    - We have an anemic entity name that stands for a person
    - By design, we assume that the name of the first lady (first spouse?) is the only thing that we care about.
    - Parties similarly are entities with only one attribute, namely their name
    - States, Elections, and Congresses however are more involved.
    - Even that is complicated: Andrew Jackson's marriage might have been illegal and therefore void

Presidents



- The State-of-the-Union part of the E/R model
  - A state has the “born-in” or “native-son” relationship with presidents
  - A state has attributes: her name, the year of admission into the union (whatever that might be for Delaware or Vermont), an official population number, and the votes in the electoral college
  - Strictly speaking, this is not good design since the number of votes in the electoral college and the official population varies with each new census. A reelected president might span two different census and in the case of F. D. Roosevelt, three.

Presidents



- Elections (in the electoral college) provide even more challenges.
  - Try to extend the data model to give information about the votes of the candidates!
  - In the current model, we only have votes for and votes for other than the winner
  - It would be nice to have the names of people with votes in the college together with their party
  - This is made more complicated because we can have several people. For example, Wallace got 46 electoral votes as a third-party candidate and the 1860 election had four, even though Lincoln garnered a comfortable majority in the college

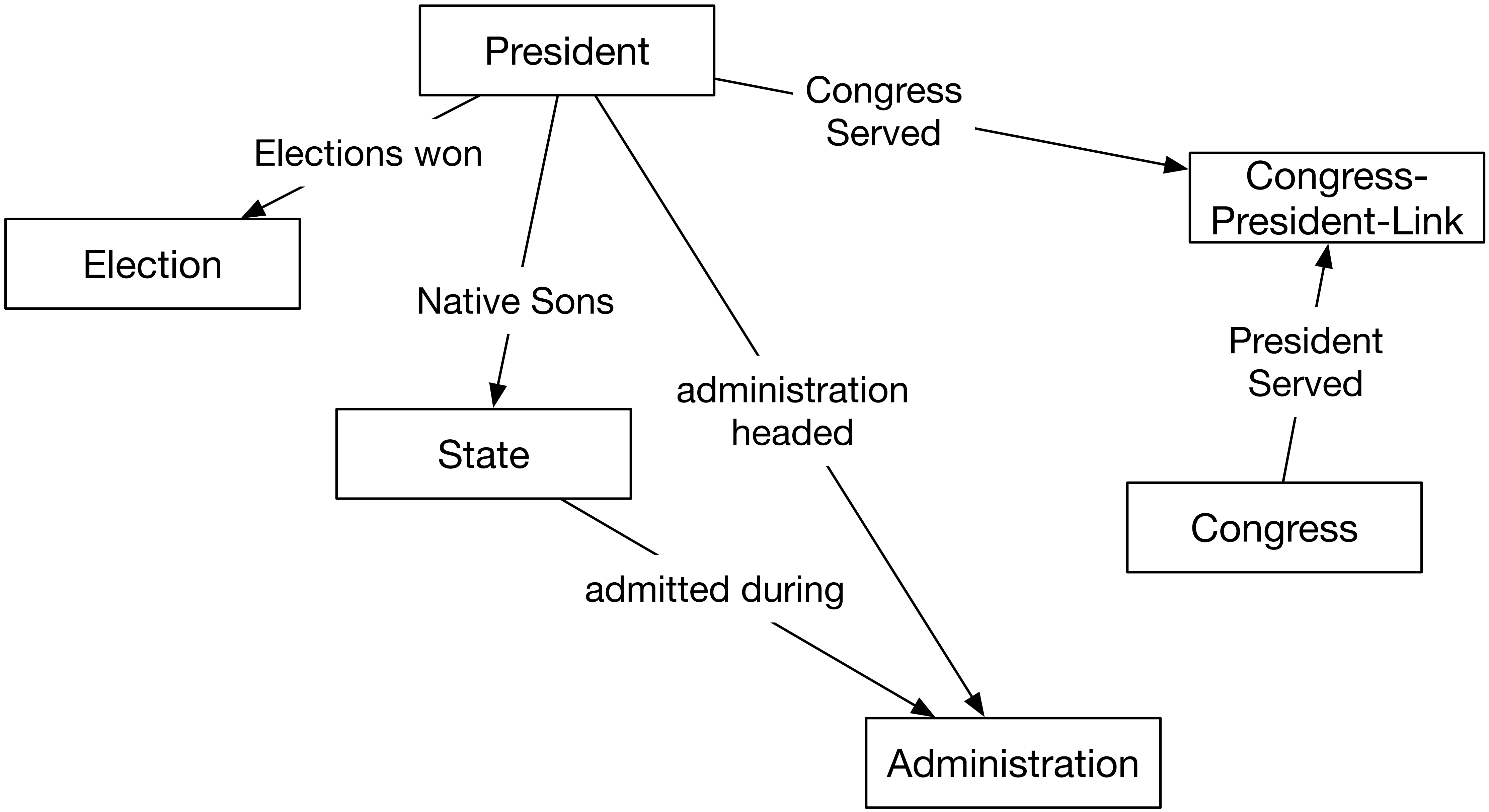
Network Model

- Attribute relationship: Between attributes of the same entity
    - Presidential candidate has a name
    - Presidential candidate is affiliated with a party
  - Represented by a record type: collection of data items
- Entity relationship: Between two different entities
  - E.g. relationship between president and election
    - Can be represented by a graph

Network Model

- Relationships can be:
    - one-to-one
    - one-to-many
        - E.g. “administrations headed”, “administrated during”
    - many-to-many:
        - E.g. Presidents - Congress
        - Introduce a “link record”

Network Model



President

Congress Served

Congress-President-Link

Elections won

Election

Native Sons

State

administration headed

President Served

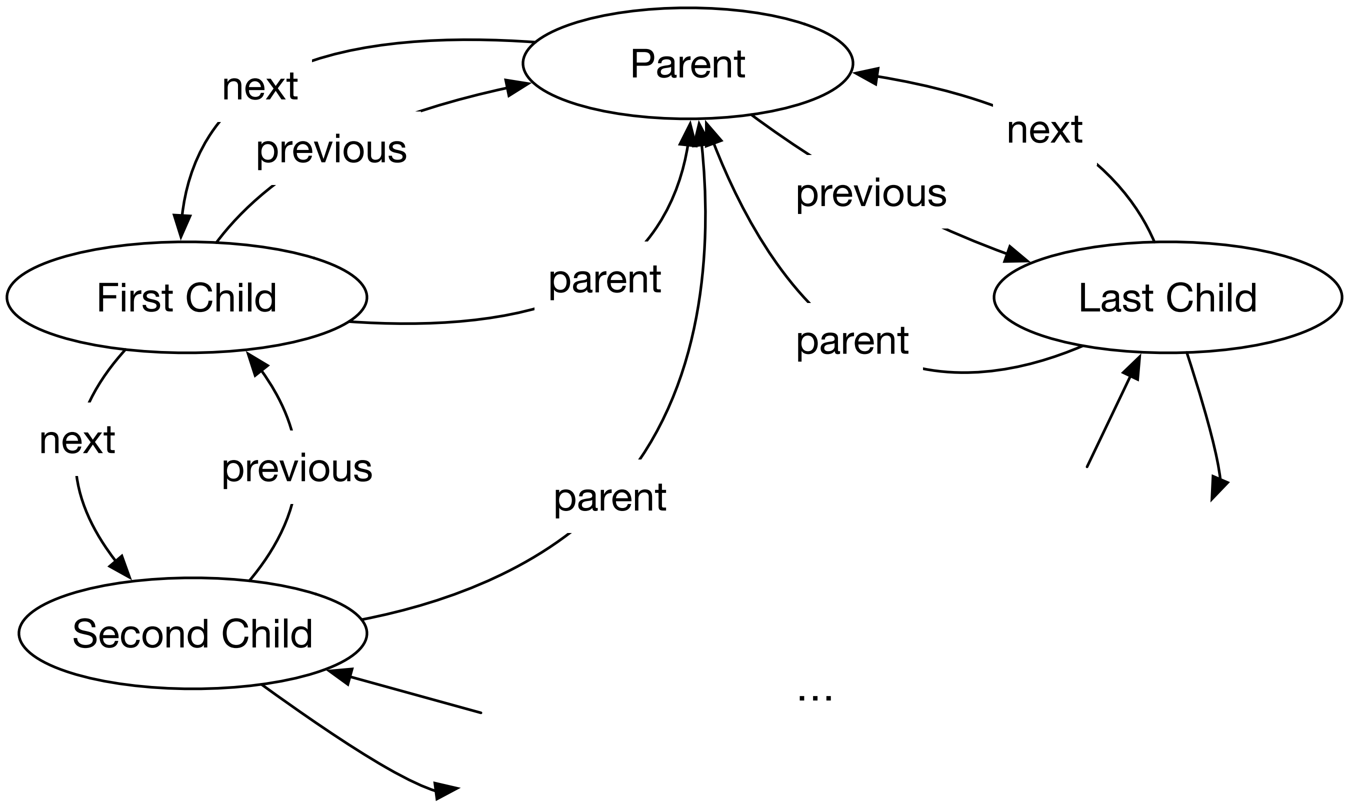
Congress

admitted during

Administration

Network Model

- Implementing network model:
  - Double linked list of record (pointers) for implementing a one-to-many relationship

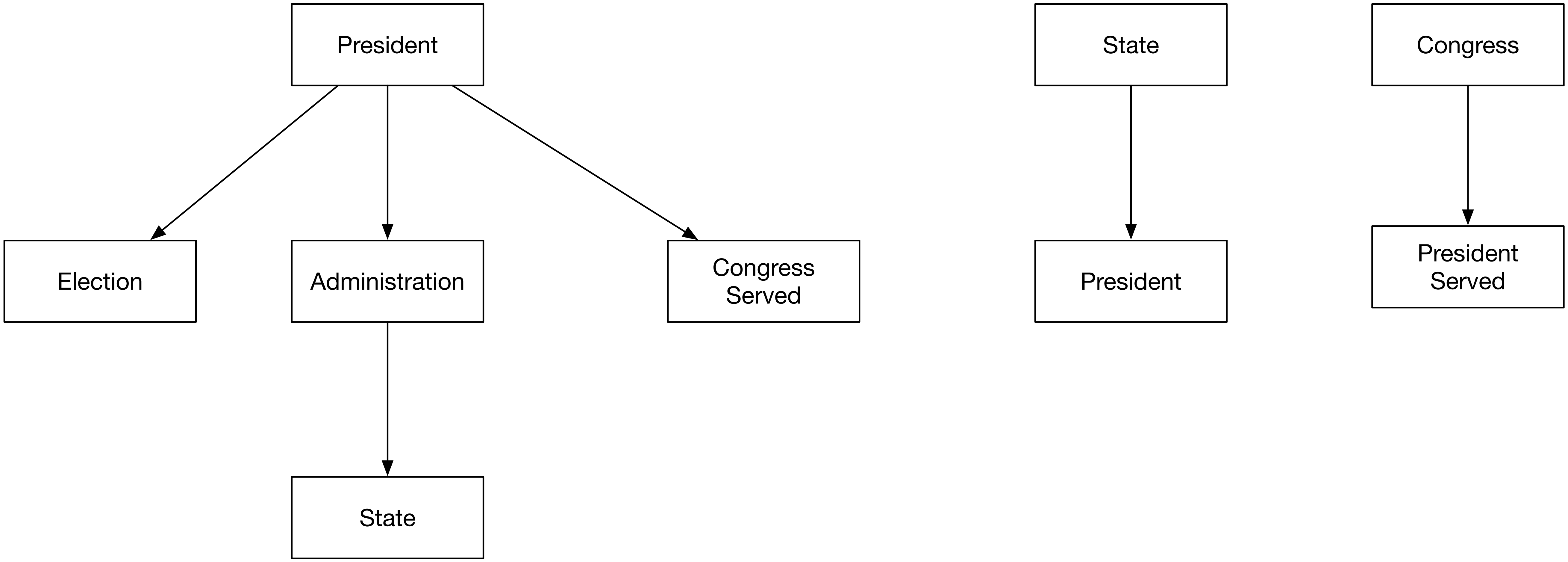


Network Model

- Programmer is a *navigator*

Hierarchical Model

- Use only hierarchical definition trees
  - No need to label relationships



Hierarchical Model

- Implement via pointers
- Smart systems limit the number of pointers
  - E.g. Only one pointer to child, then pointers to siblings
- Implement via *traces*
  - records have logical addresses based on their position in the tree (
    - e.g. path from root)
  - trace is a translation between logical and physical addresses

Hierarchical Model

- Programmer is a *navigator*
  - Use tree traversal systems

Database Organization

- We can observe that the hierarchical and less the network model of databases is tied to the logical organization of data access
- Network model based databases were commercially successful
- In order to allow untrained or untrainable users to interact with them, manipulation mechanisms became more sophisticated
- Network model based databases still have problems with parallelism and record protection

Relational Databases

- E. F. Codd (IBM, San José, CA) proposed relational databases in a 1970 paper
- Pressured IBM into developing System R, with a non-relational access language called Sequel
- Based on preprints of papers, Ellison founded Oracle, with a similar language called SQL

Relational Databases

"Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). ... Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.
"Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model."

Codd: ”A Relational Model of Data for Large Shared Data Banks”, CACM 1970

https://dl.acm.org/citation.cfm?id=362685

Relational Databases

- Data is stored as tuples.
  - A tuple is an array of values.
  - Each coordinate is an *attribute*
- Customary to present tuples as rows in a matrix

title	year	length	genre
Gone with the wind	1939	231	drama
Star wars	1977	124	SciFi
Wayne’s World	1992	95	comedy

Relational Databases

- Columns are called attributes
- Name and the set of attributes are called the *scheme*
  - `Movies(title, year, length, genre)`
- Entries are called tuples
- Strict relational model requires that all attributes are atomic: an elementary type
  - `Movies(title:str, year:int, length:int, genre:str)`

Relational Databases

- Relational databases change over time through inserts and deletions
    - The state of a database at one time is called the *current instance*

Relational Databases

- Keys:
    - Relations are not ordered
        - To allow fast access, need indices
    - Information represented by the data also needs to be coherent
        - A change in the information should result in a single update to a tuple
        - Otherwise, programming errors are likely to render the information incoherent

Relational Databases

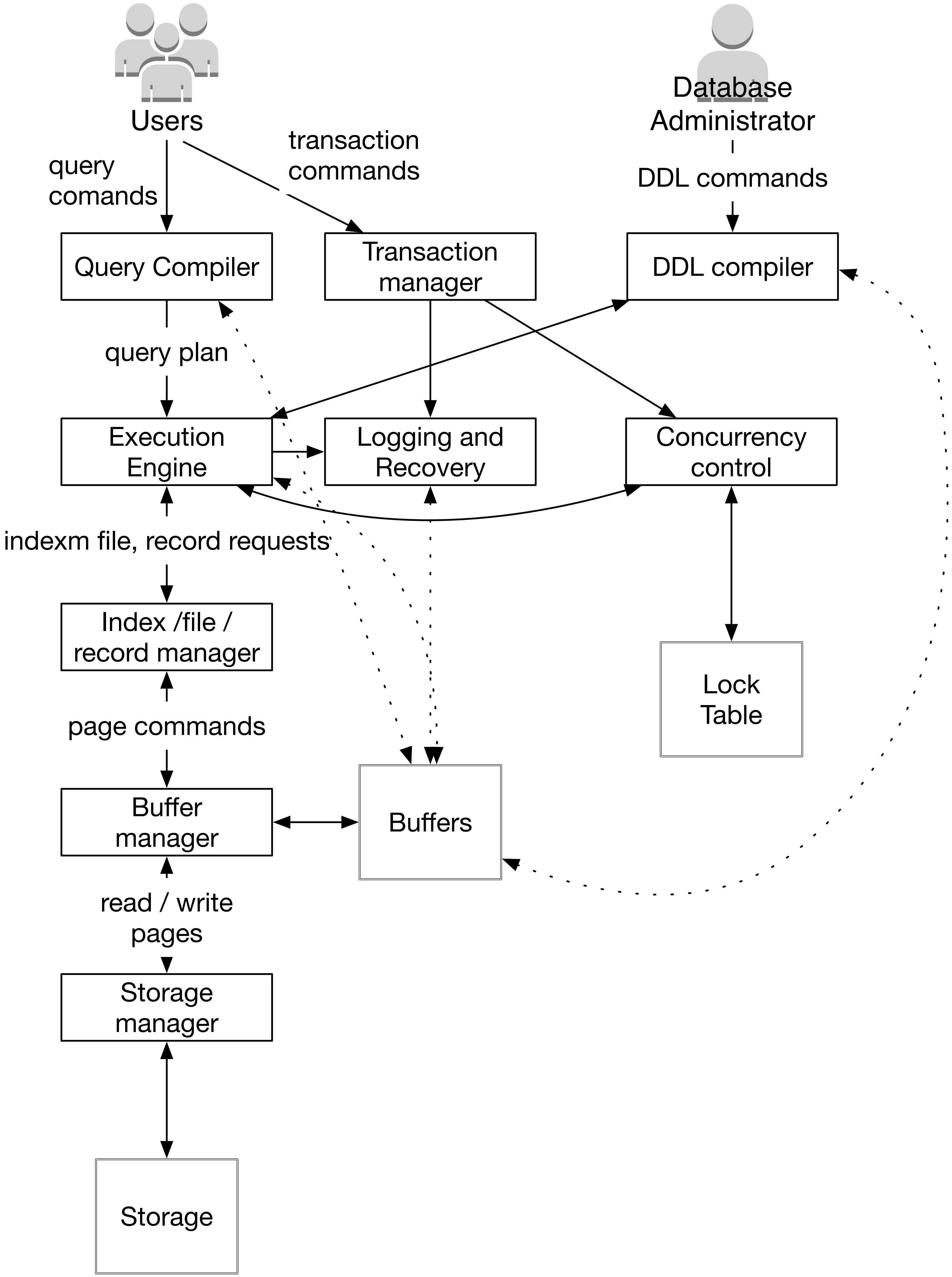
- Notation of keys supports both
    - Artificial keys: an auto-generated ID that characterizes each tuple uniquely
    - A or a combination of attributes that are unique to the tuple (for all eternity)
        - Movie database example:
            - Title is not sufficient, there were two King Kong movies
            - Underline keys in a scheme
            - `Movies(title, year, length, genre)`

Relational Databases

- SQL is **descriptive**
  - Does not provide a way to obtain the result
    - This is left to the DBMS itself

Database Management System

- DBMS:
  - Conventional users and application programs ask for data or modify data
  - Database administrator: define and change *structure* (***schema***) of a data base
- Data Definition Language
  - Commands alter schema



Users

Database Administrator

query comands

transaction commands

DDL commands

Query Compiler

Transaction manager

DDL compiler

query plan

Execution Engine

Logging and Recovery

Concurrency control

indexm file, record requests

Index /file / record manager

Lock Table

page commands

Buffer manager

Buffers

read / write pages

Storage manager

Storage

Database Management System

- Query Processing
  - Parsed and optimized by a query compiler
  - Results in query plan
  - Execution engine requests small pieces of data
    - Records or tuples of a relation
  - Made to Index / File / Record Manager
    - Knows about data files, format and size of records
    - Knows about index files (to find records faster)

Database Management System

- Query Processing (cont)
  - Requests for data are passed to the buffer manager
  - Buffer manager brings pages / disk blocks from storage
  - Mediated by storage manager

Database Management System

- Transaction Processing
  - Goal:
    - Atomicity
      - Updates either happen or do not happen at all
    - Isolation
      - Updates appear as if they were the only ones processed
    - Durability
      - Updates do not get lost
    - Consistency
      - Properties of / among data is maintained

Database Management System

- Transaction Processing
  - Concurrency control manager aka scheduler
    - Responsible for atomicity and isolation
  - Logging and recovery manager
    - Responsible for durability of transactions

Database Management System

- Storage and Buffer Management
    - Data usually resides in secondary storage
        - Hard disk
        - SSD
    - Needs to be brought into RAM for reading / processing
    - Storage manager controls placement of data on disk / SSD

Database Management System

- Buffer manager partitions main memory into buffers
    - (Page-sized regions of memory)
- All other components interact with buffer manager for information on
    - Data
    - Metadata
    - Log Records
    - Statistics
    - Indexes

Database Management System

- Transaction processing
  - Transaction manager:
    - *Logging*
    - *Concurrency control*
      - Typically uses locks
        - Stored in the lock table
    - *Deadlock Resolution*
      - Usually involves abort or rollback

Database Management System

- Query processor
  - Query compiler generates query plan
  - Query compiler consists of
    - Query parser
    - Query preprocessor
      - Semantic checks / changes parse tree
    - Query optimizer
      - Transforms initial query plan

Database Management System

- Query processor
    - Execution Engine