Data modelling overview (+ paper introduction)

COSC430—Advanced Databases

David Eyers

Overview of today's lecture

- Course introduction
  - Goals
  - Teaching team
  - Course outline
  - Assessment information

- Overview of how database data models evolved, e.g.
  - hierarchical; network; relational; object; non-relational; …

Goals of COSC430

- Understand advanced **database theory and research**
  - investigate advanced data models
  - explore distributed, time-series, embedded and graph databases
  - examine data mining approaches
  - discuss new and emerging database technologies
- Perform **practical exercises**
  - relational database administration
  - Cassandra, Neo4j and embedded database practical labs
- Develop **research skills**
  - Paper reading, critical thinking, problem solving, report writing

Teaching team

- **David Eyers** (course coordinator)
  - Room: 125   email: dme@cs.otago.ac.nz
- **Cathy Chandra**
  - Room: 121   email: cathy@cs.otago.ac.nz

Textbook and resources

- No specific recommended textbooks
  - Reading materials and online references will be provided throughout the semester as needed
  - Some material derived from Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe. Addison-Wesley. **(E&N)**

- Blackboard pages will typically link to material hosted on the Computer Science website
  - I want my teaching material to be publicly available
  - https://www.cs.otago.ac.nz/cosc430/

Assessment

- Assignments (40%)
  - Details will be available on the COSC430 website
  - **Assignment 1**: data modelling (8%)
    - Due at 4pm on 20th March (Friday)
  - **Assignment 2**: Oracle database administration (15%)
    - Part 1: due 4pm on 27th March (Friday)
    - Part 2: due 4pm on 8th May (Friday)
  - **Assignment 3**: project (17%)
    - Due 4pm on 25th May (Monday)
- **Examination** (60%)

Course outline

Week	Date	Lecture topic		Notes
		First hour	Second hour	
1	24 February	Data modelling		
2	2 March	Relational model		
3	9 March	Oracle DBA lab		
4	16 March	NoSQL models		Prepare Spanner paper
5	23 March	Distributed models	Spanner discussion	
6	30 March	Cassandra lab		Prepare Apriori paper
7	6 April	Data mining	Apriori discussion	Prepare Gorilla paper
	13 April	(Easter break)		
8	20 April	Time-series databases	Gorilla discussion	Prepare Trinity paper
9	27 April	(ANZAC Day holiday)		
10	4 May	Graph databases	Trinity discussion	
11	11 May	Neo4j lab		
12	18 May	Embedded / ORMs lab		
13	25 May	Temporal and geographic databases		

Academic integrity and misconduct

- Academic integrity policy
  - Being honest in your study and assessments
  - https://www.otago.ac.nz/administration/policies/otago116838.html

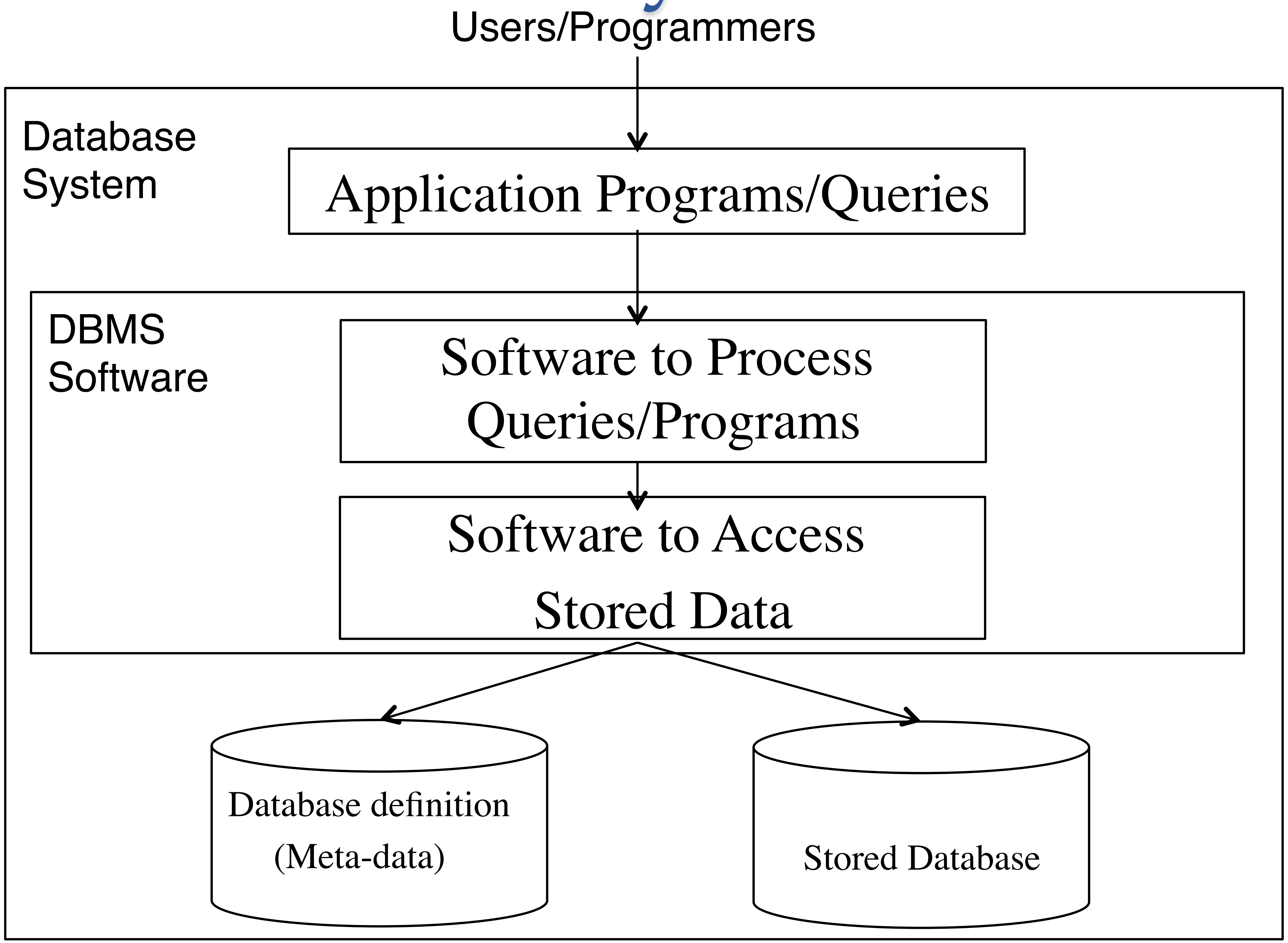
- Student academic misconduct procedures
  - Types of misconduct include plagiarism, copying, unauthorised collaboration, using unauthorised material, assisting someone else's misconduct, etc.
  - https://www.otago.ac.nz/administration/policies/otago116850.html

Learning objectives

You should understand:

- What a **data model** is within database design and why is it needed
- What **data independence** is and why is it important
- Evolution of data models:
    - **Hierarchical** model
    - **Network** model
    - **Relational** model
    - **Object-relational** model
    - **NoSQL** models
- What the object-relational "**impedance mismatch**" is
- What the pros and cons are of **schema** and **schema-less** designs

Simplified database system environment



An example of a relational database

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-1965	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-1955	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-1968	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-Jun-1941	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-1962	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-1972	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-Mar-1969	980 Dallas, Houston, YX	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-1937	450 Stone, Houston, TX	M	55000	NULL	1

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerisation	10	Stafford	4
Reorganisation	20	Houston	1
NewBenefits	30	Stafford	4

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-May-1988
Administration	4	987654321	1-Jan-1995
Headquarters	1	888665555	19-Jun-1981
Dummies	0	111100000	31-Dec-2004

DEPT_LOCATION

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPENDENT

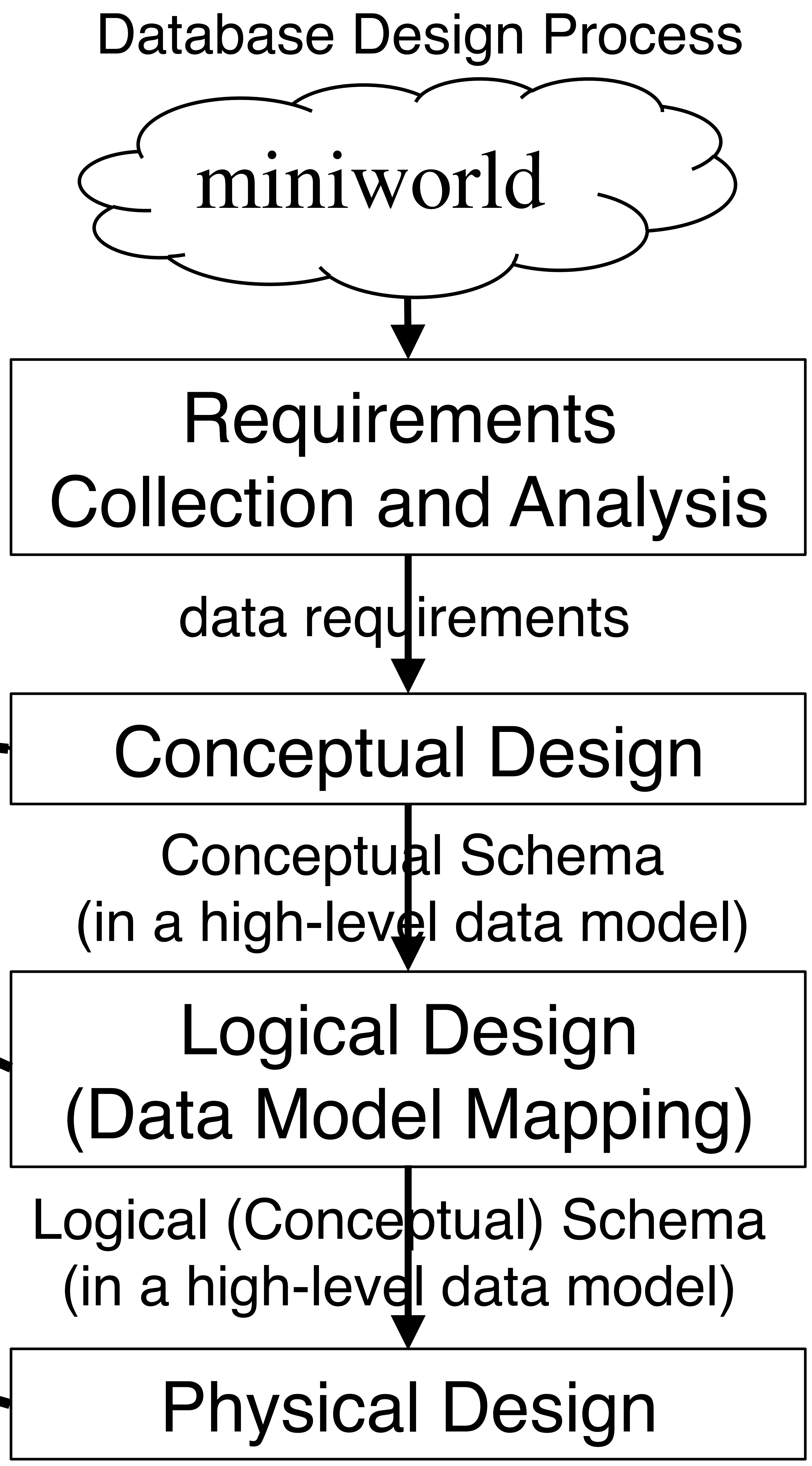
ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	5-Apr-1986	Daughter
333445555	Theodore	M	25-Oct-1983	Son
333445555	Joy	F	3-May-1958	Spouse
987654321	Abner	M	28-Feb-1942	Spouse
123456789	Michael	M	4-Jan-1988	Son
123456789	Alice	F	30-Dec-1988	Daughter
123456789	Elizabeth	F	5-May-1967	Spouse

Defining the term “data model”

- A **data model** is an abstract model that describes how data is organised/represented in an information system or a database management system
  - is a system of concepts and their interrelations
  - is the “language” used to describe data
  - defines syntax and semantics
  - defines operations for data manipulation
- Why do we need data models?
- **database schema**: description of database structure

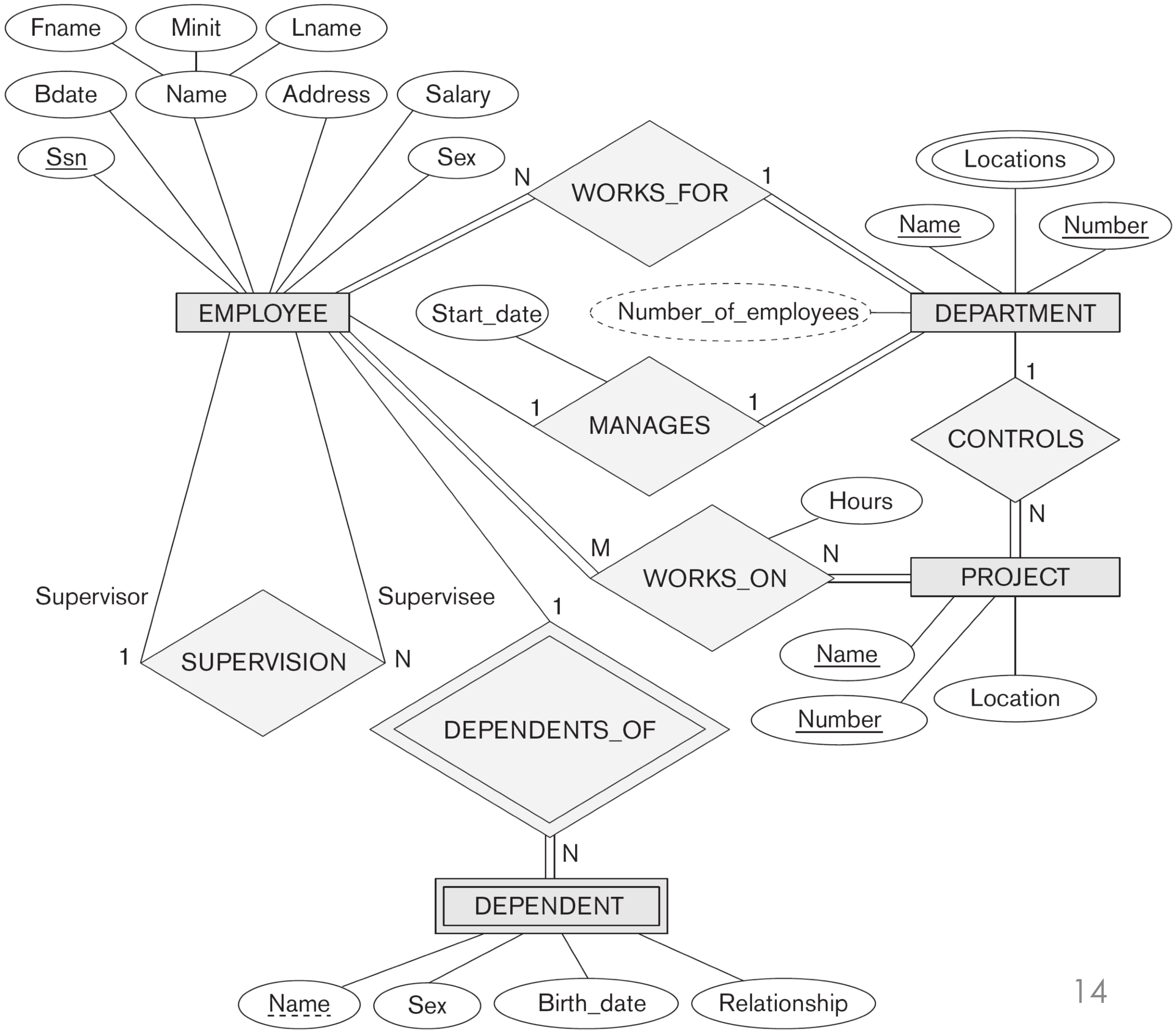
Database data models

- Tightly coupled with DBMS
  - implements one or more data models
- **conceptual** data model
  - concepts and relationships, ER model
- **implementation** data model
  - logical model without storage details
- **physical** data model
  - storage details



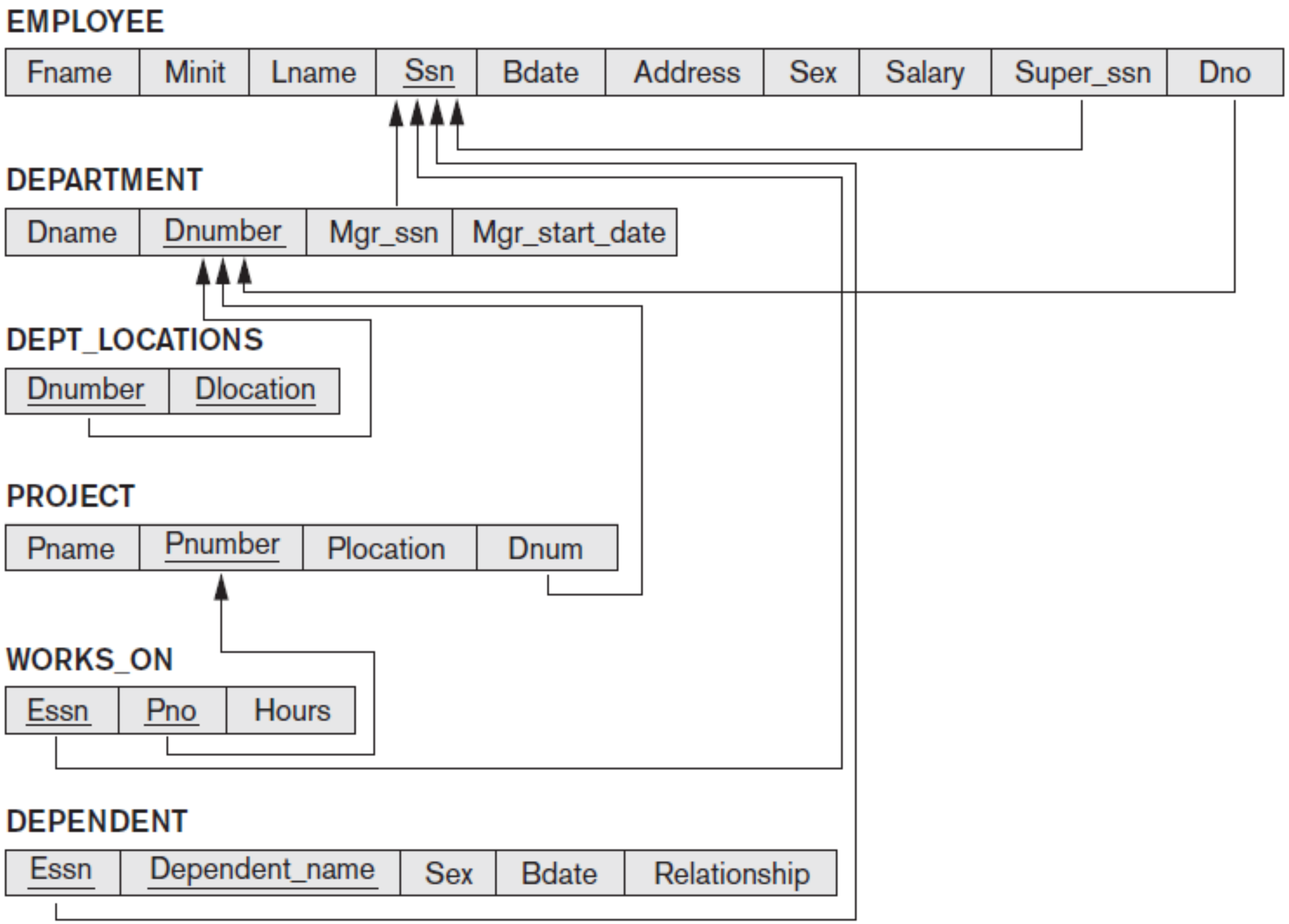
Example conceptual schema

- An ER diagram for a company database
- Figure 7.2 in E&N (6th edition)



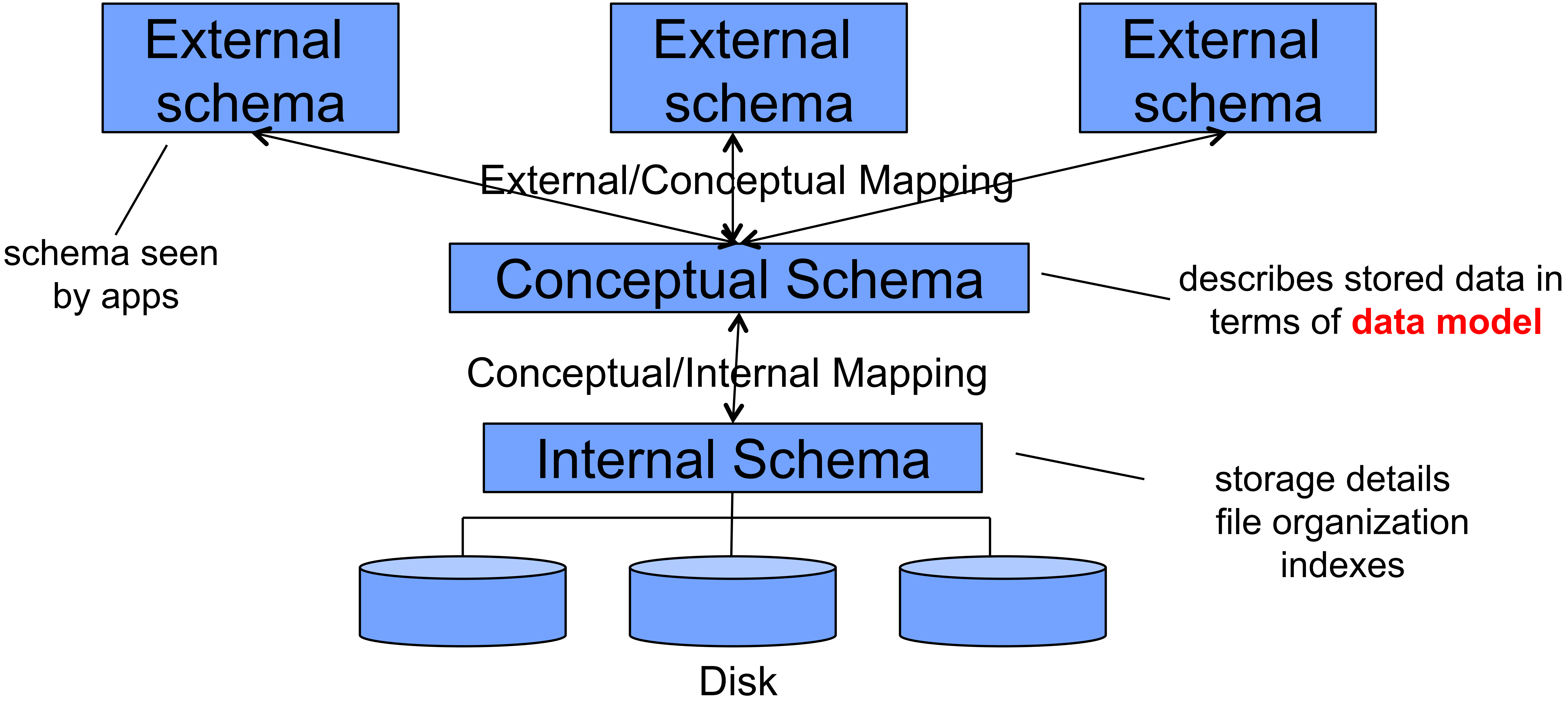
Example logical schema

- A relational model for a company database
- Figure 3.7 in E&N (6th edition)



Level of data abstraction

- ANSI/SPARC architecture

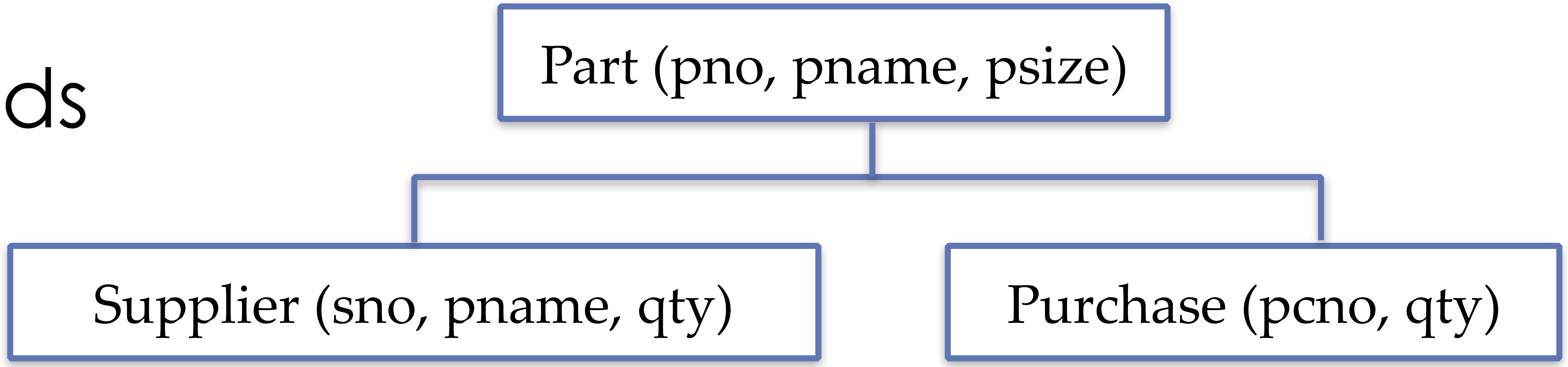


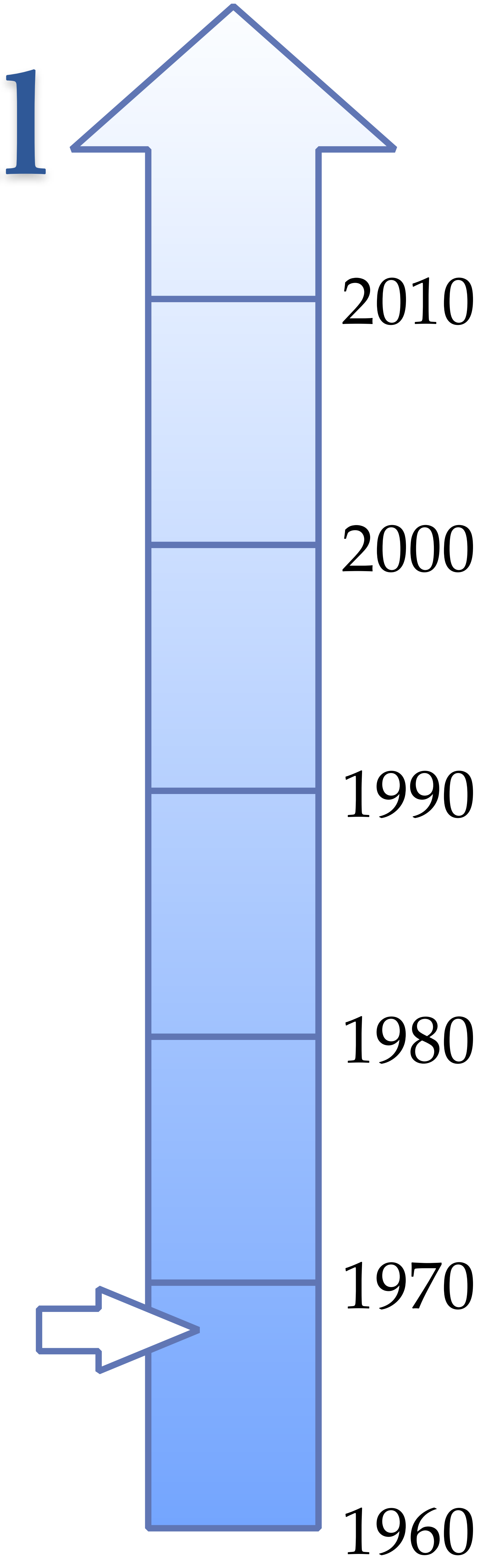
Data independence has benefits

- **Logical** data independence:
  - gives capacity to change the conceptual schema without having to change external schemas
- **Physical** data independence:
  - gives capacity to change the internal schema without having to change the conceptual schemas
- Why are these properties important?
  - **Reduce maintenance cost** since:
    - logical design can change over time
    - physical design may need to be tuned to improve performance

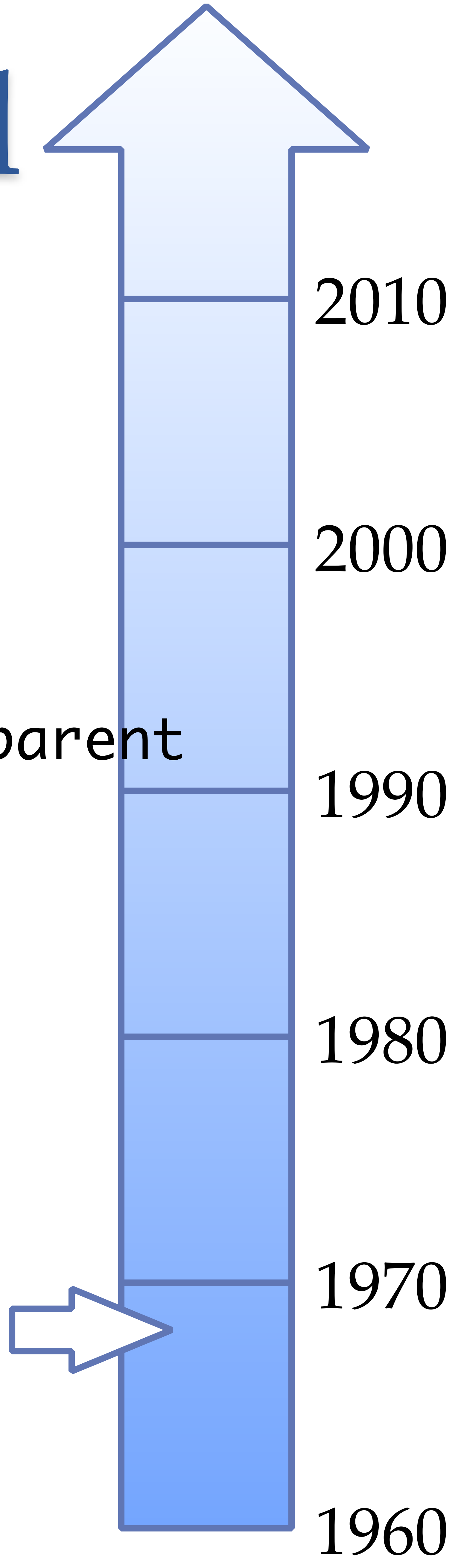
Evolution of data models: **Hierarchical**

- IBM's IMS—Information Management System
  - Record type: a collection of named fields
  - Each record type must have a key
  - Record types must be arranged in a tree structure
- Tree-structure has two drawbacks
  - Redundant data (e.g., pname, ….)
  - Existence of a record depends on its parents



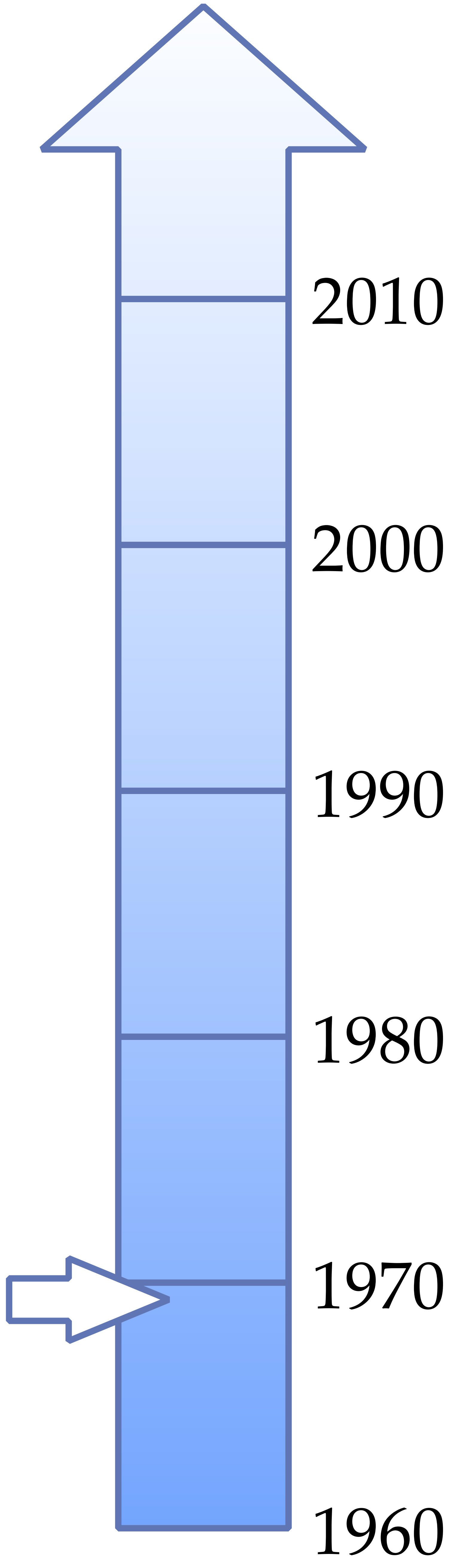


Evolution of data models: **Hierarchical**

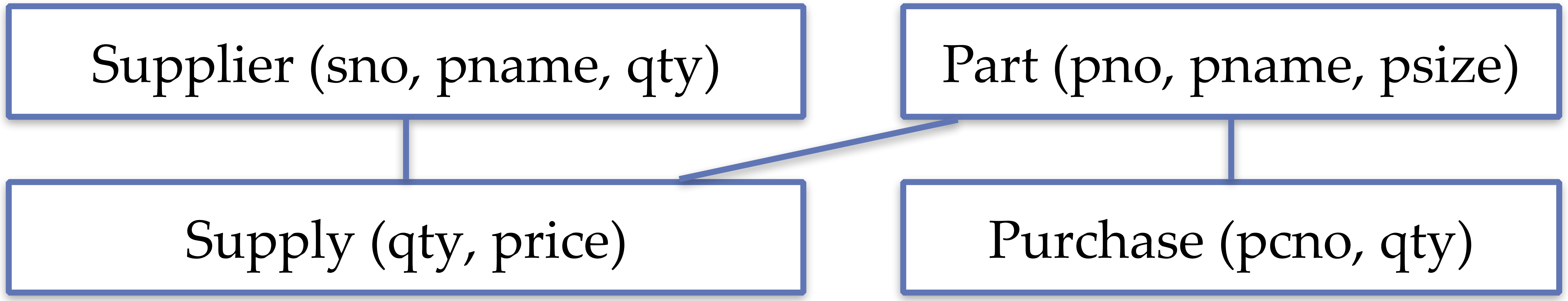


- Data manipulation language: DL/1
  - Each record has a hierarchical sequence key (HSK)
  - Records are ordered: depth-first and left-to-right
  - HSK defines semantics of commands: `get_next`; `get_next_within_parent`
  - A **record-at-a-time** language
    - Programmer constructs provided for solving and optimising query
- Data independence
  - Very **limited physical independence**
    - Records are stored sequentially based on HSK
  - Includes a **small amount of logical independence**
    - DL/1 programs run on logical database

Evolution of data models: **Network**

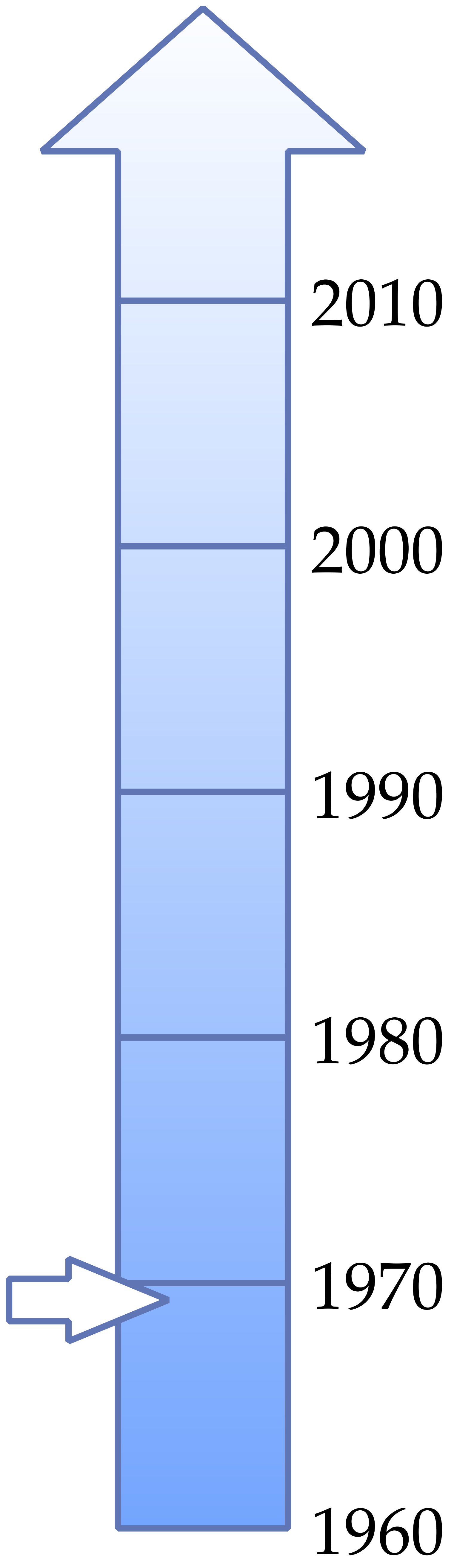


- Also known as the CODASYL data model
- Similar to hierarchical model:
  - Collection of record types, each with keys
  - Record-at-a-time data manipulation language
- Different from hierarchical model:
  - Record types are organised into a network
    - A given record instance can have **multiple parents**
    - At least one entry point to the **network graph**



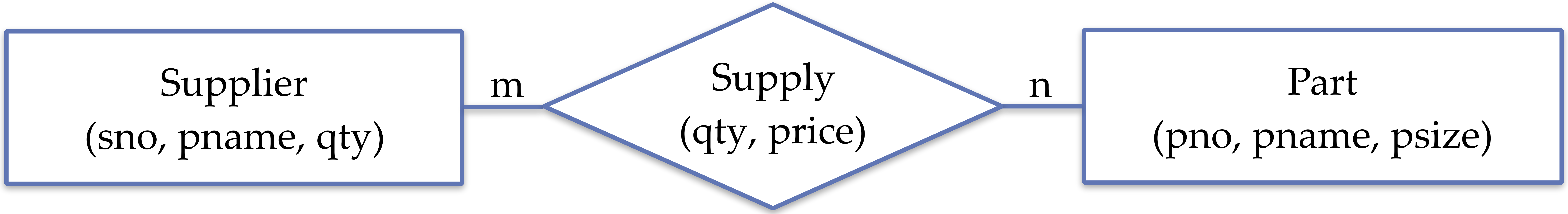
Evolution of data models: **Network**

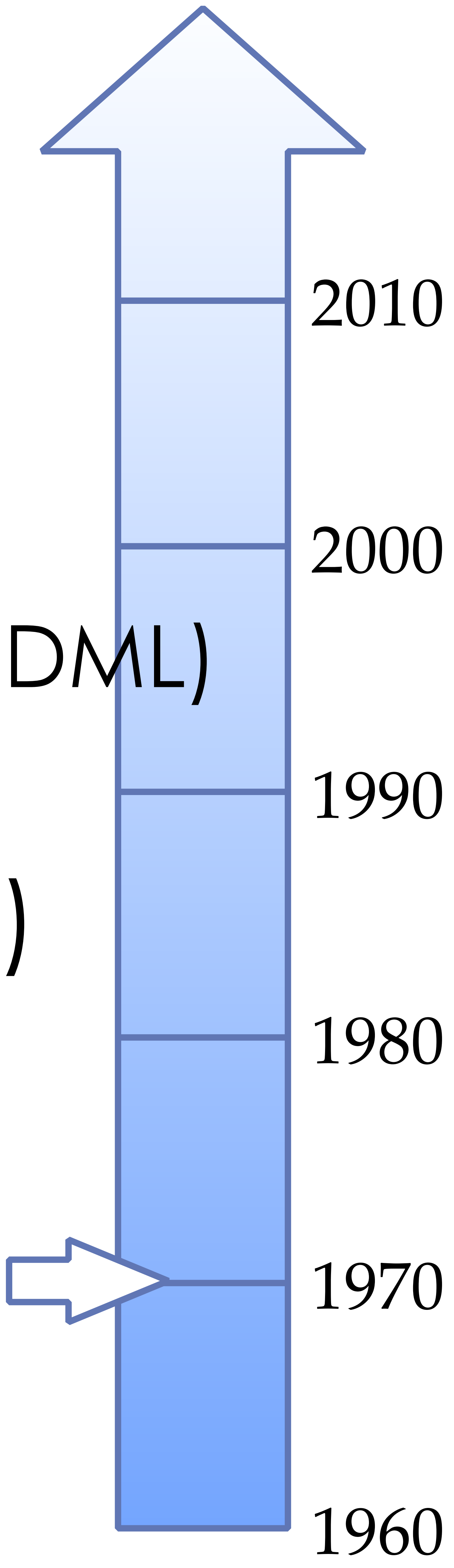
- It provides more flexibility but is more complex than the hierarchical model
  - Only need to navigate a hierarchical space in the hierarchical model
  - Need to navigate a multi-dimensional space in the network model
- Data independence
  - **No physical independence**
  - **No logical independence**



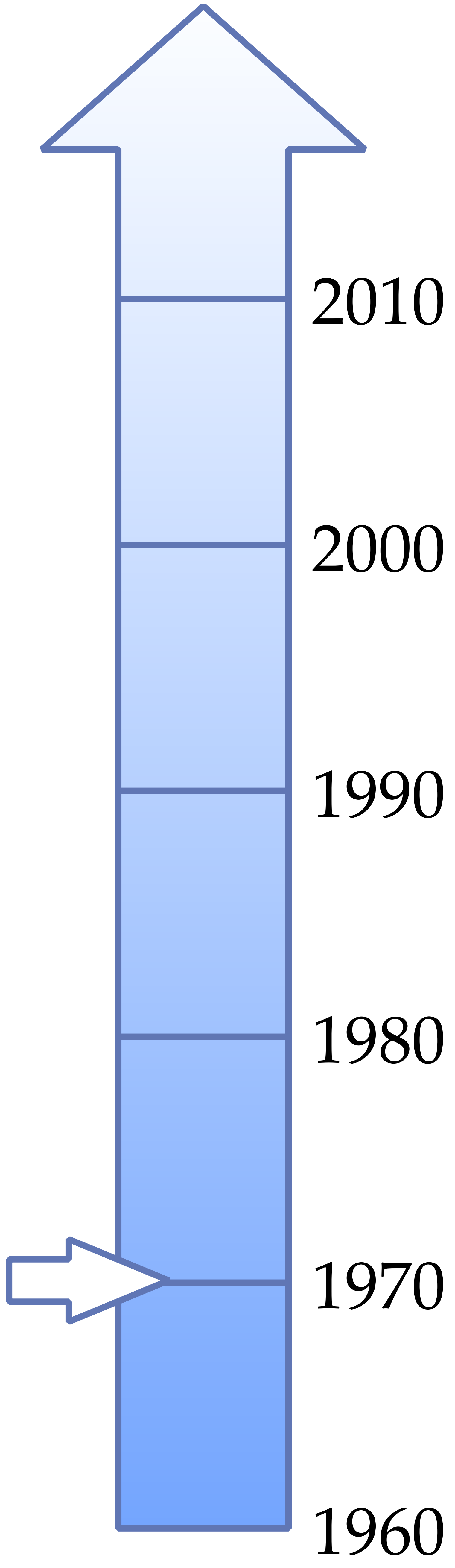
Evolution of data models: **Relational**

- Invented by **E.F. Codd**, IBM Research, in 1970
  - Simple data structure (relation, table)
  - High-level **set-at-a-time** data manipulation language (DML)
  - Define logical schema only, no physical schema
- Entity-Relationship (E-R) model (**Peter Chen**, 1970)
  - Entities; Attributes; Relationships





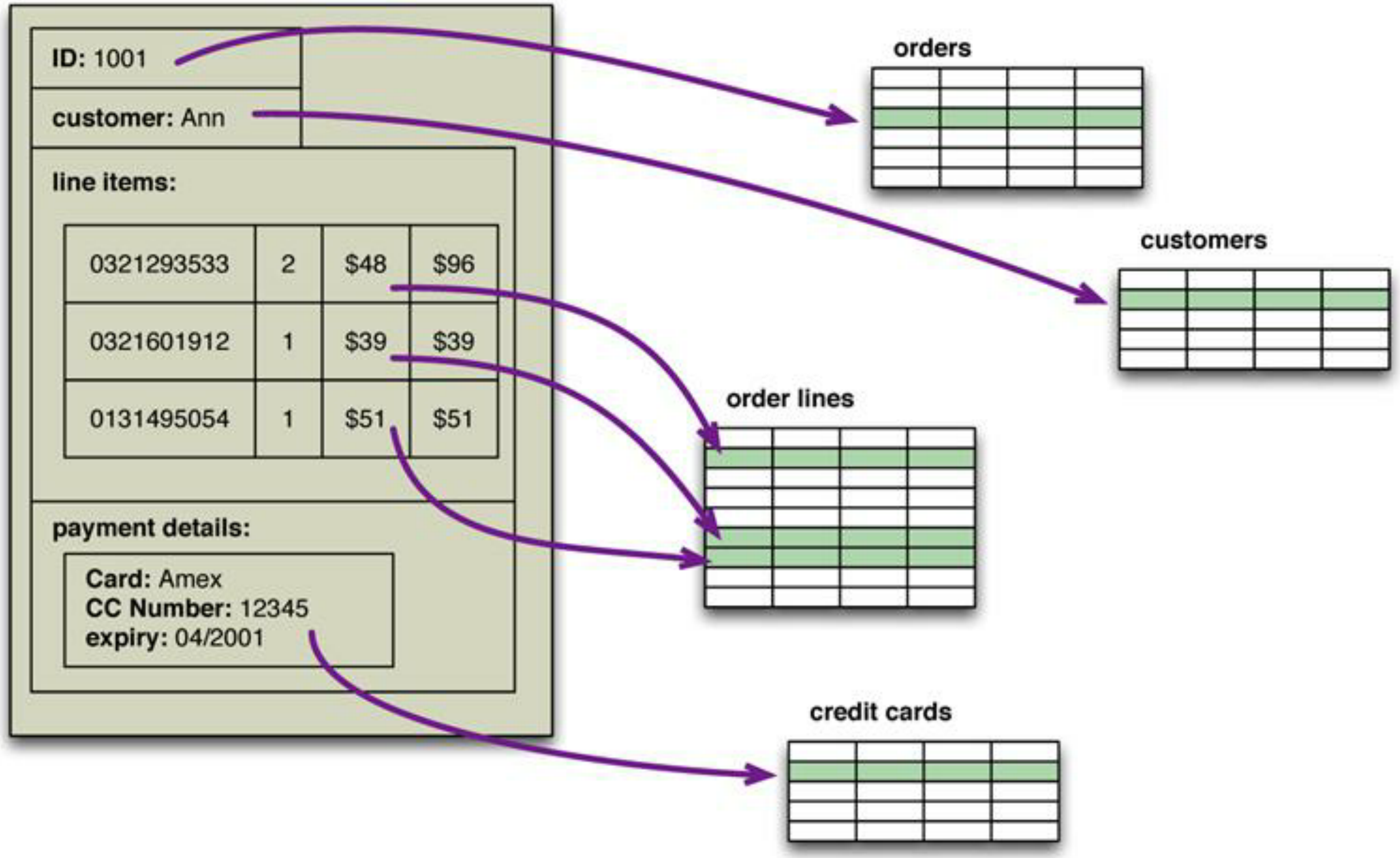
Evolution of data models: **Relational**



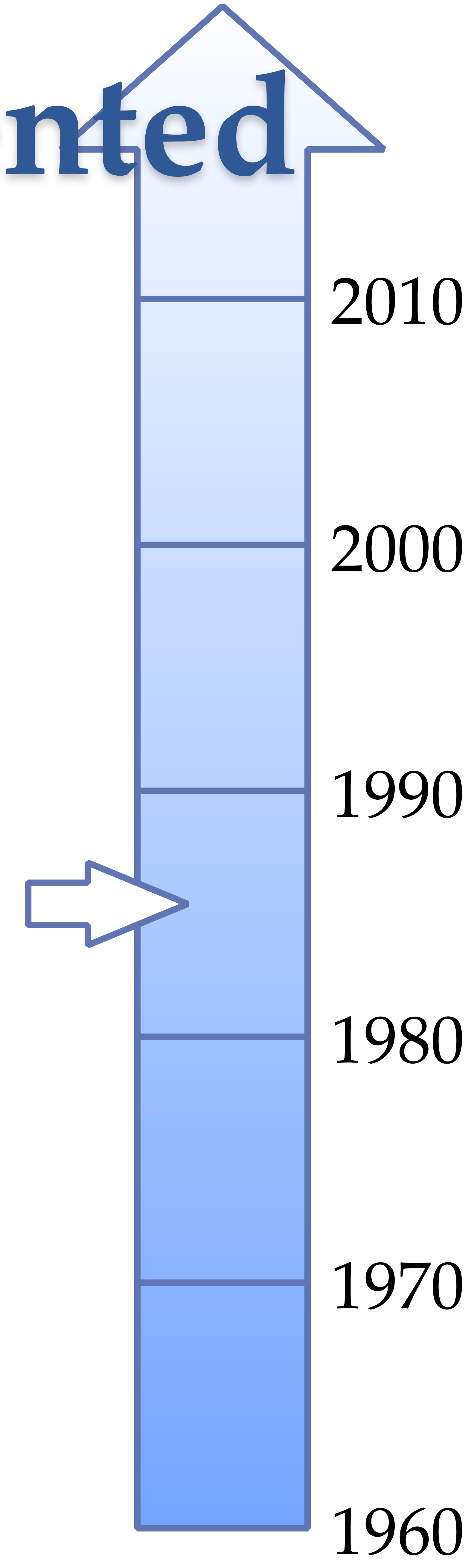
- Has the flexibility to represent almost anything
- **Set-at-a-time** DML offers substantial advantages over record-at-a-time DML
  - e.g., can optimise general queries within the RDBMS
- Data independence
  - **Provides physical independence**: no specification of what storage looks like
  - **Provides logical independence** through use of views

Object↔relational impedance mismatch

- Problems encountered when relational model and object oriented (OO) model work together
  - Difference between the relational model and OO's in-memory data structures
  - Object-relational mapping bridges differences (e.g. Hibernate and iBATIS)



Evolution of data models: **Object Oriented**

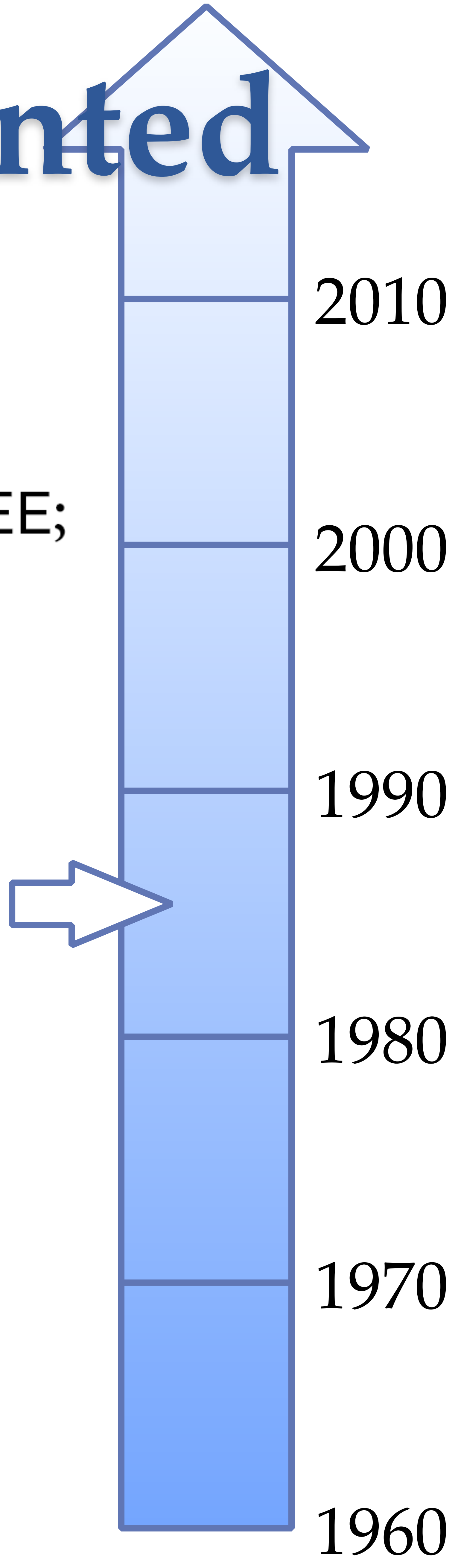


- Motivation for the OO data model within databases:
  - Need for DBs to support more complex applications
  - Need for DBs to support additional data modelling features
  - Popularity of OO programming languages
- Object
  - Components: state (value) and behaviour (operations)
  - Can have a **complex structure** as well as specific **operations** defined by the programmer
- Some commercial products (e.g. O2, Objectstore), but **not much impact** on mainstream data management

Evolution of data models: **Object Oriented**

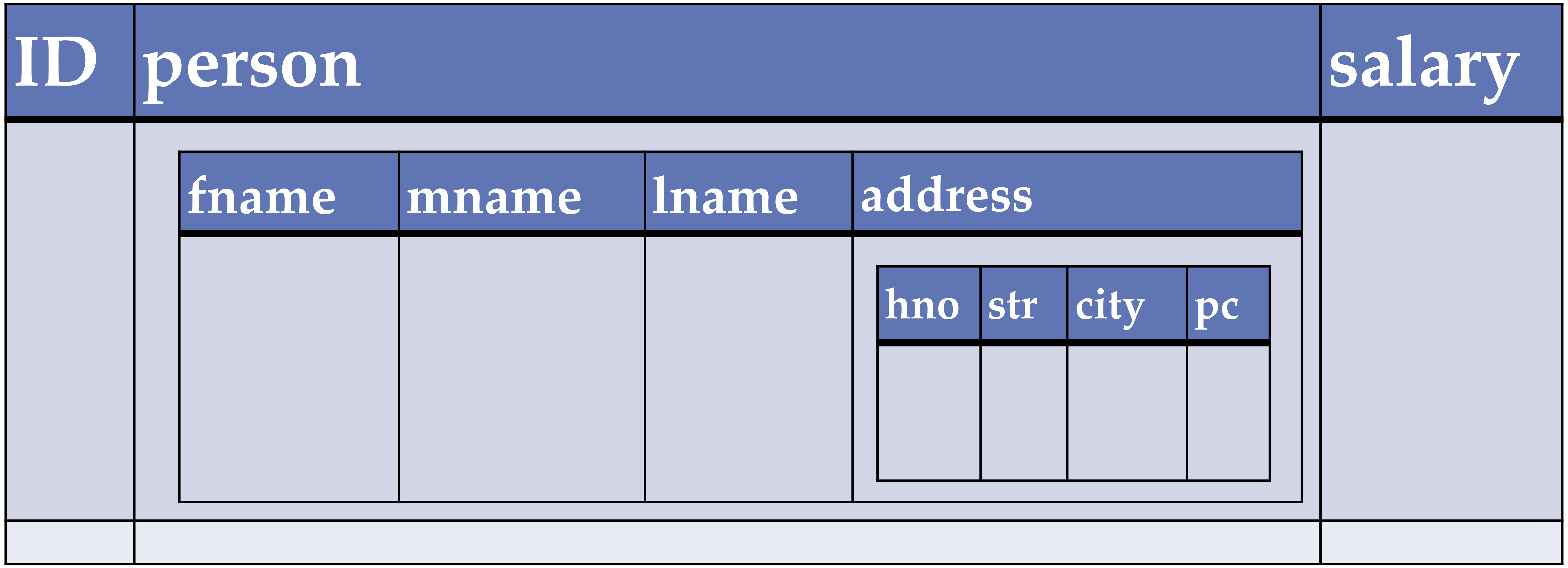
- An example OO database schema
- Figure 11.2 from E&M (6th edition)

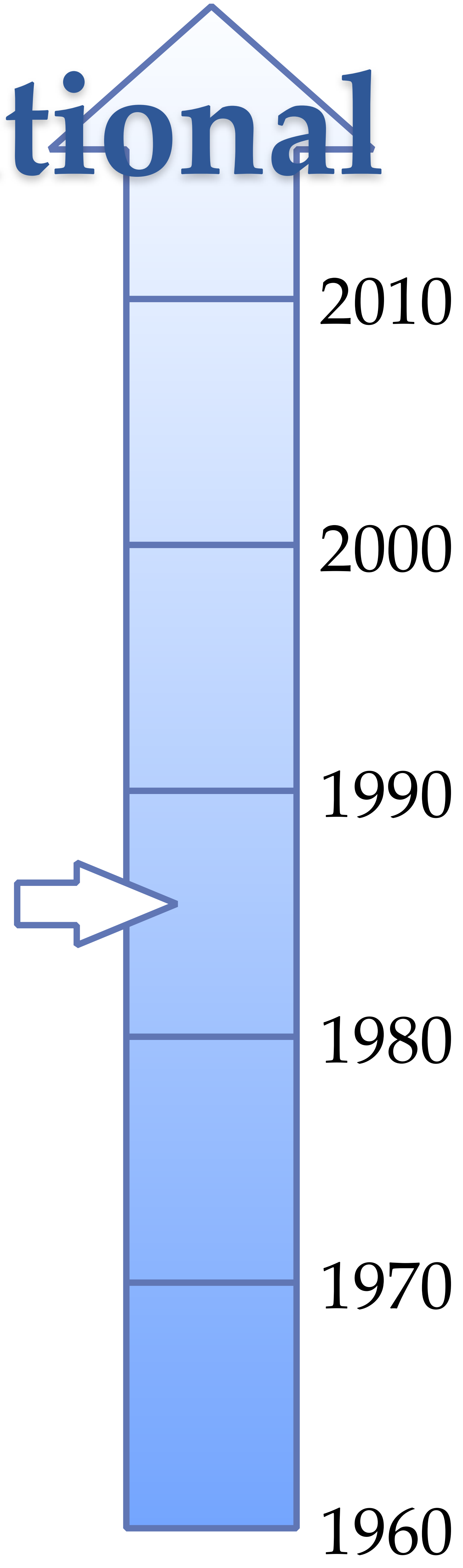
```
define class DEPARTMENT
    type tuple (  Dname:           string;
                  Dnumber:         integer;
                  Mgr:             tuple ( Manager:      EMPLOYEE;
                                           Start_date:   DATE;   );
                  Locations:       set (string);
                  Employees:       set (EMPLOYEE);
                  Projects         set(PROJECT);     );
    operations    no_of_emps:      integer;
                  create_dept:     DEPARTMENT;
                  destroy_dept:    boolean;
                  assign_emp(e:    EMPLOYEE): boolean;
                  (* adds an employee to the department *)
                  remove_emp(e: EMPLOYEE): boolean;
                  (* removes an employee from the department *)
end DEPARTMENT;
```



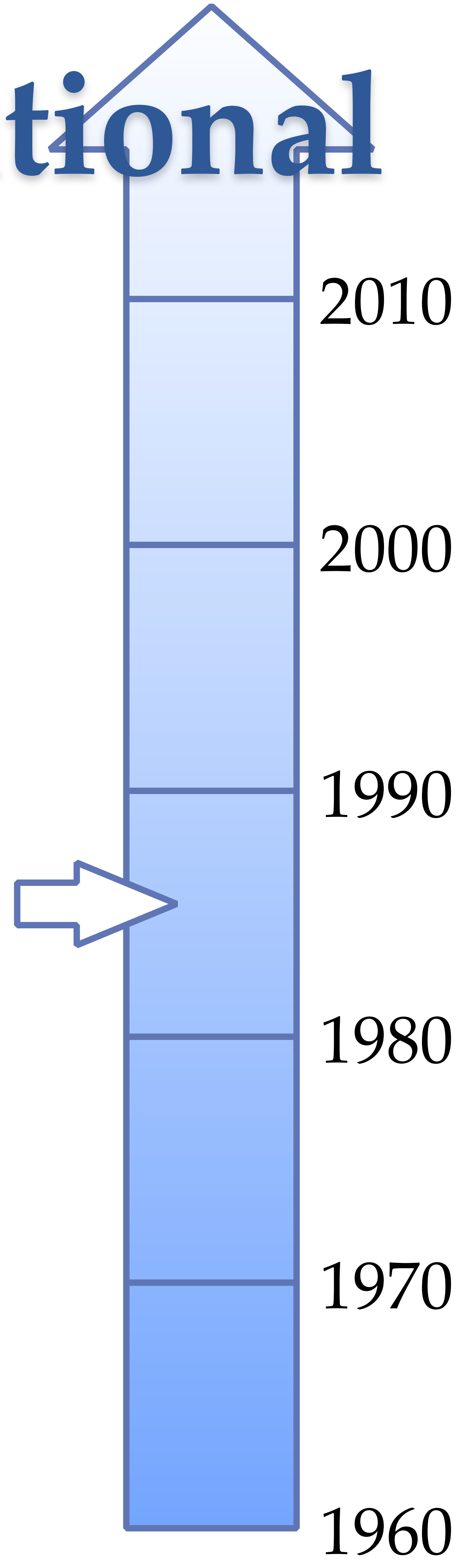
Evolution of data models: **Object-Relational**

- Relation is the fundamental structure
  - … but allows constructs such as **nested relations**
  - often implementations support user-supplied **methods**





Evolution of data models: **Object-Relational**



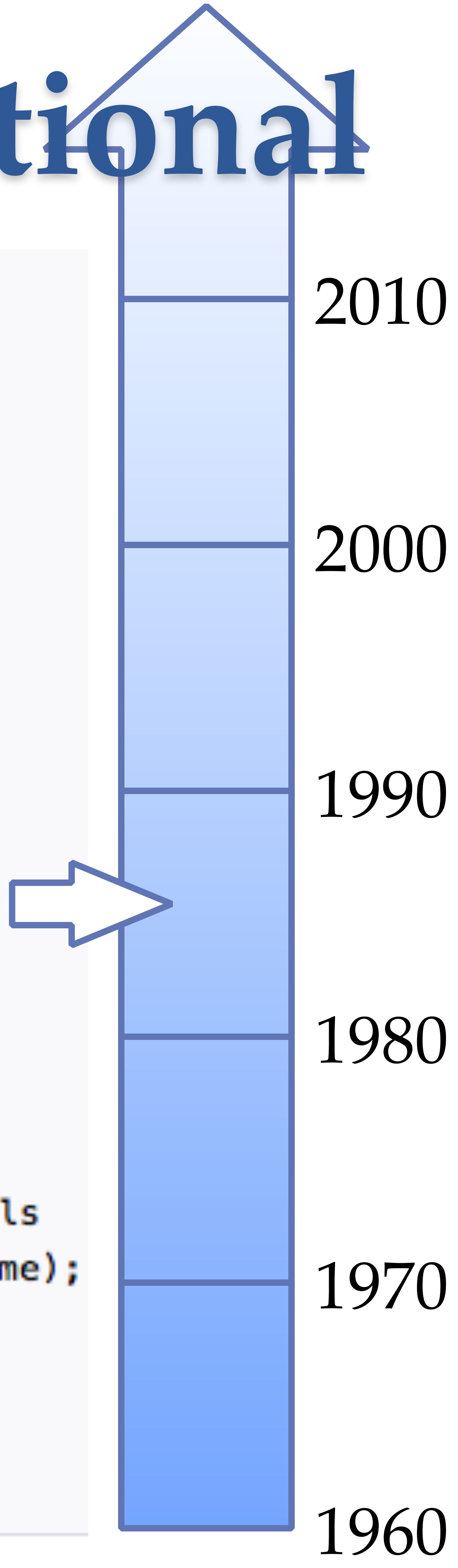
- Extend the relational model to the OO domain
  - **Type system** with user-defined types (UDT)
    - Including set, bag, array, list collection types
    - Including structures like records
    - Can use type inheritance
  - **Methods**
    - Special operations can be defined over the UDTs
    - Special operators for complex types, e.g. images
  - **References**
    - Several ways to reference and de-reference objects, e.g. using pointers to avoid redundant use of storage
- Implemented in most popular relational RDBMSs

Evolution of data models: **Object Relational**

- Example: Creating an object type in Oracle

```
CREATE TYPE person_typ AS OBJECT (
  idno           NUMBER,
  first_name     VARCHAR2(20),
  last_name      VARCHAR2(25),
  email          VARCHAR2(25),
  phone          VARCHAR2(20),
  MAP MEMBER FUNCTION get_idno RETURN NUMBER,
  MEMBER PROCEDURE display_details ( SELF IN OUT NOCOPY person_typ ));
/

CREATE TYPE BODY person_typ AS
  MAP MEMBER FUNCTION get_idno RETURN NUMBER IS
  BEGIN
    RETURN idno;
  END;
  MEMBER PROCEDURE display_details ( SELF IN OUT NOCOPY person_typ ) IS
  BEGIN
    -- use the PUT_LINE procedure of the DBMS_OUTPUT package to display details
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(idno) || ' ' || first_name || ' ' || last_name);
    DBMS_OUTPUT.PUT_LINE(email || ' '  || phone);
  END;
END;
/
```



Different types of data

- **Structured** data:
  - Has a strict format, highly organised, conforms to a schema
  - Example: data stored in a relational database
  - Probably represents only 5–10% of data
- **Semi-structured** data
  - Has certain structure but not all information collected has identical structure
  - Example: XML and JSON documents
  - Probably represents another 5–10% of data
- **Unstructured** data
  - Has no structure
  - Examples: free text, videos, images
  - Probably represents around 80% of data

The Big Data era—but what is big data?

- *"Big Data is high-volume, high-velocity, and/or high-variety information asset that requires new forms of processing to enable enhanced decision making, insight discovery and process optimization."*

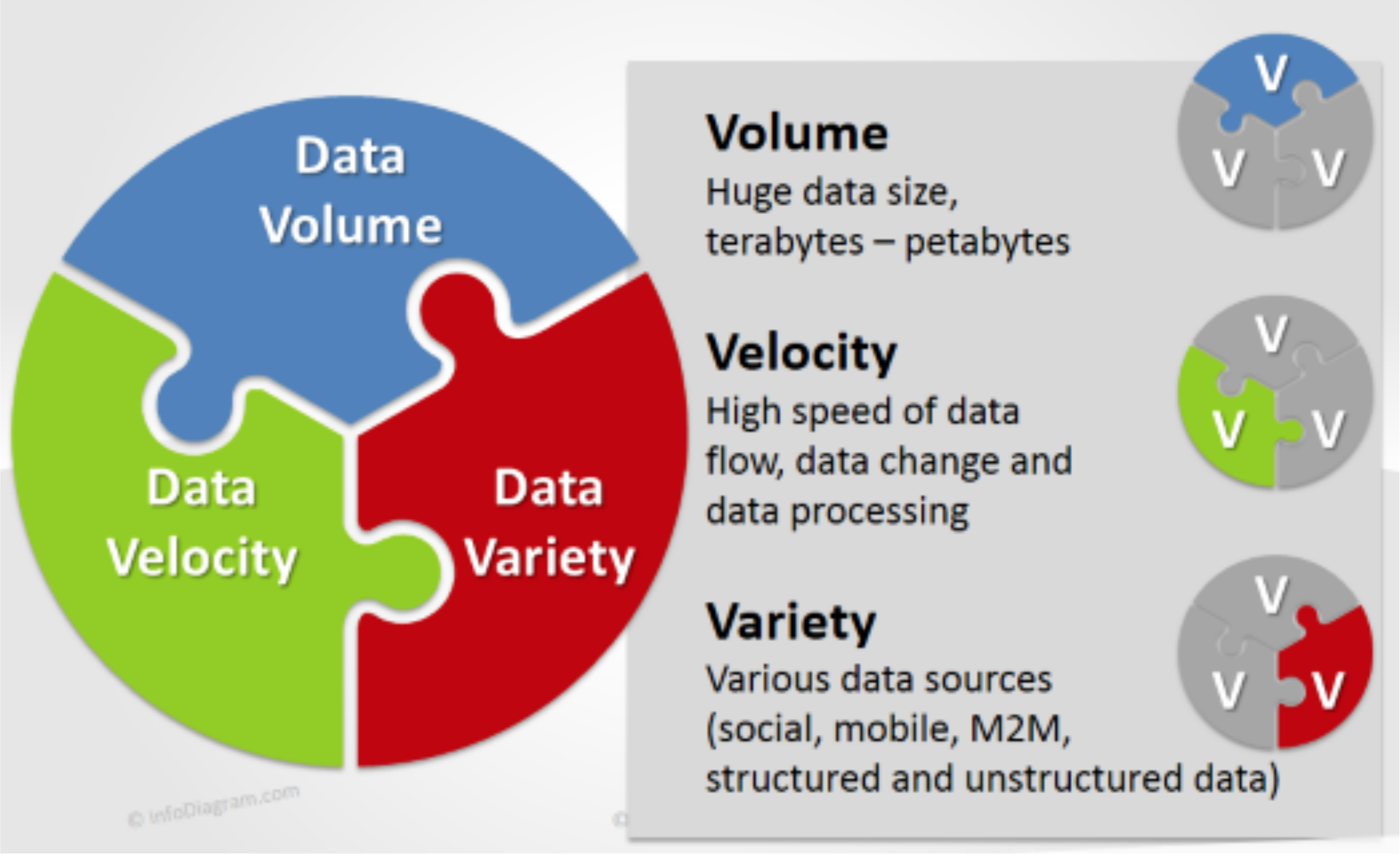
  —Gartner (2012)

- *"Big data describes data that are so voluminous and complex that traditional data processing application software are inadequate to deal with them."*

  —Wikipedia

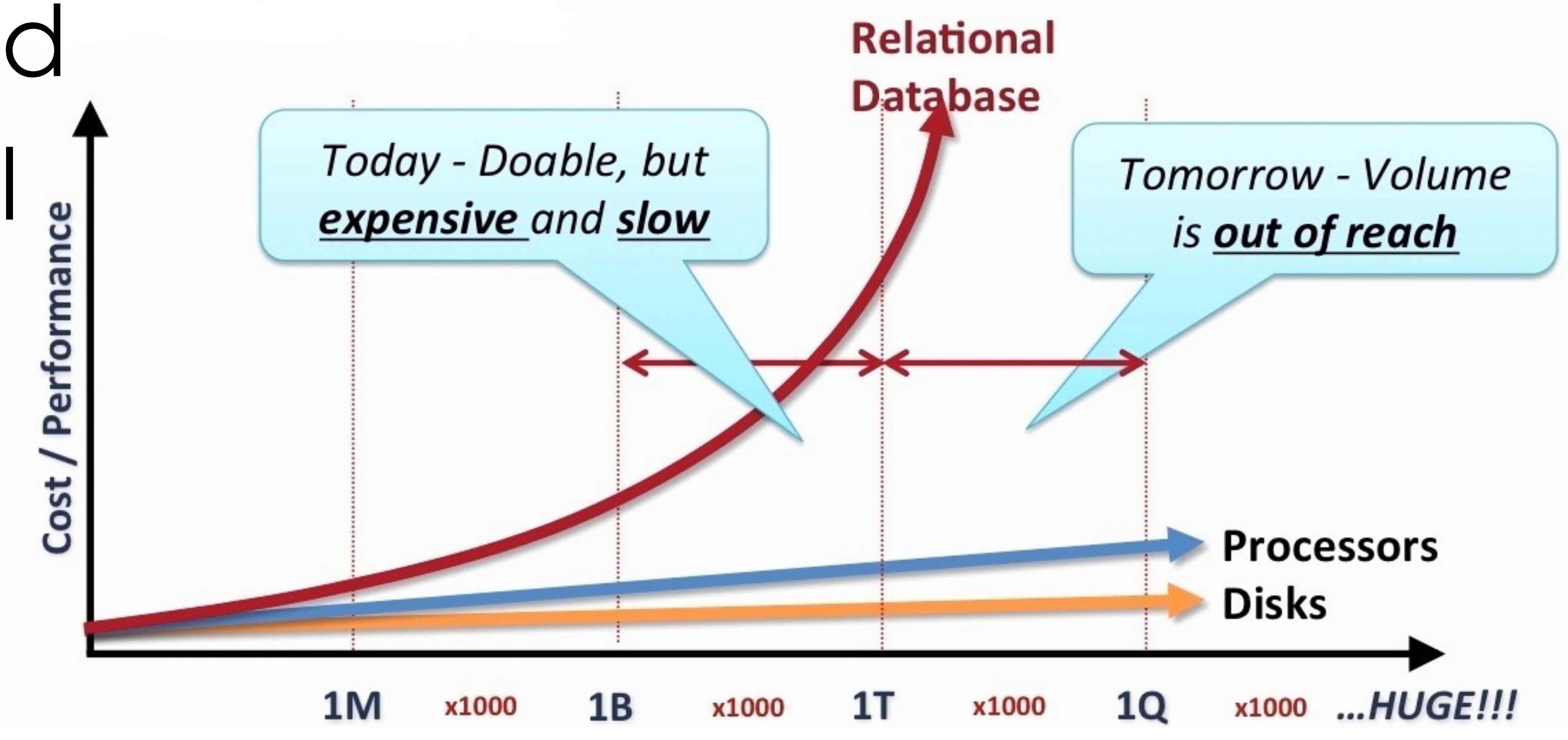
Big Data characteristics

- Facebook's activity every day at least:
  - Generates 500+ terabytes of data
  - Involves recording 2.7 billion "Like" actions
  - Includes 300 million photos being uploaded
  - Scanning roughly 105 terabytes of data each half hour…
  - (Figures from ~2017)



Difficulties using RDBMSs for Big Data

- Size of data may challenge RDBMS storage engines
- Data processing may require cluster processing
  - but many RDBMSs only operate on a single server
- Data is often unstructured
- SQL may not be an ideal query language
  - Not sure what to analyse!

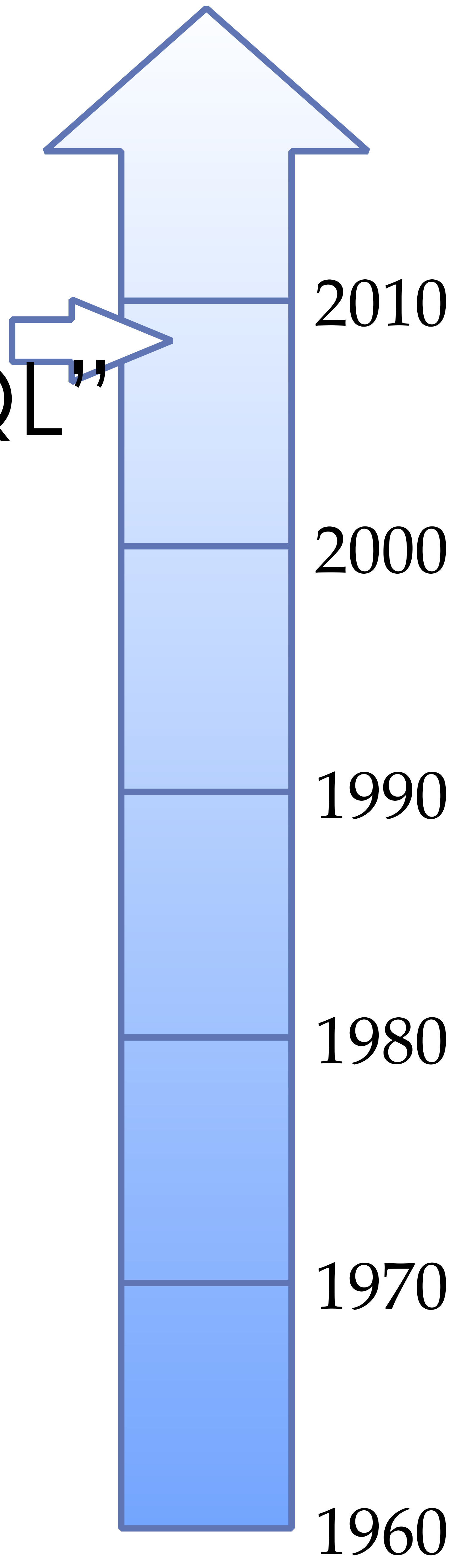


Clusters become commonplace DB tools

- A shift from **scale-up** to **scale-out** designs:
  - Architectures using clusters of commodity hardware emerged as the solution to dealing with the explosion of data volumes
  - But existing relational databases didn't support clusters well
- Many organisations developed new solutions to data storage, including:
  - BigTable (Google)
  - Dynamo (Amazon)
  - (… and many newer examples)

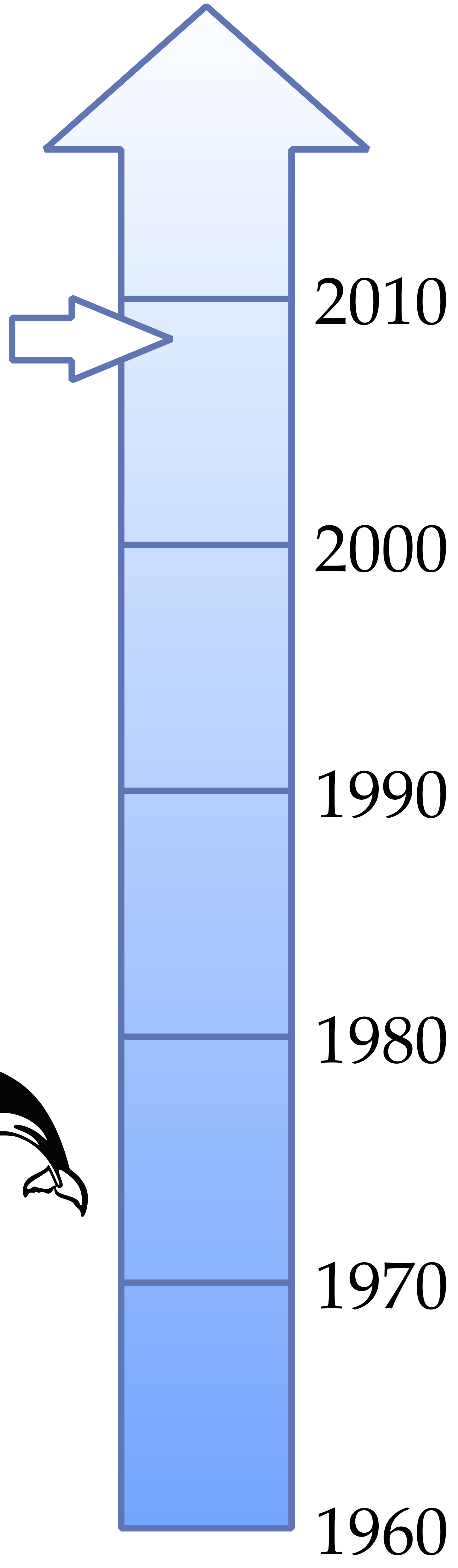
Evolution of data models: **NoSQL**

- **NoSQL** now typically interpreted as “Not Only SQL”
  - No precise definition—term emerged around 2009
    - (… and don’t confuse it with C. Strozzi’s 1998 NoSQL DB)
- Typical features of NoSQL databases:
  - Not limited to using SQL and the relational model
  - Implementations often run on clusters
  - Implementations may be schema-less
  - Implementations are often open-source software



Evolution of data models: **NoSQL**

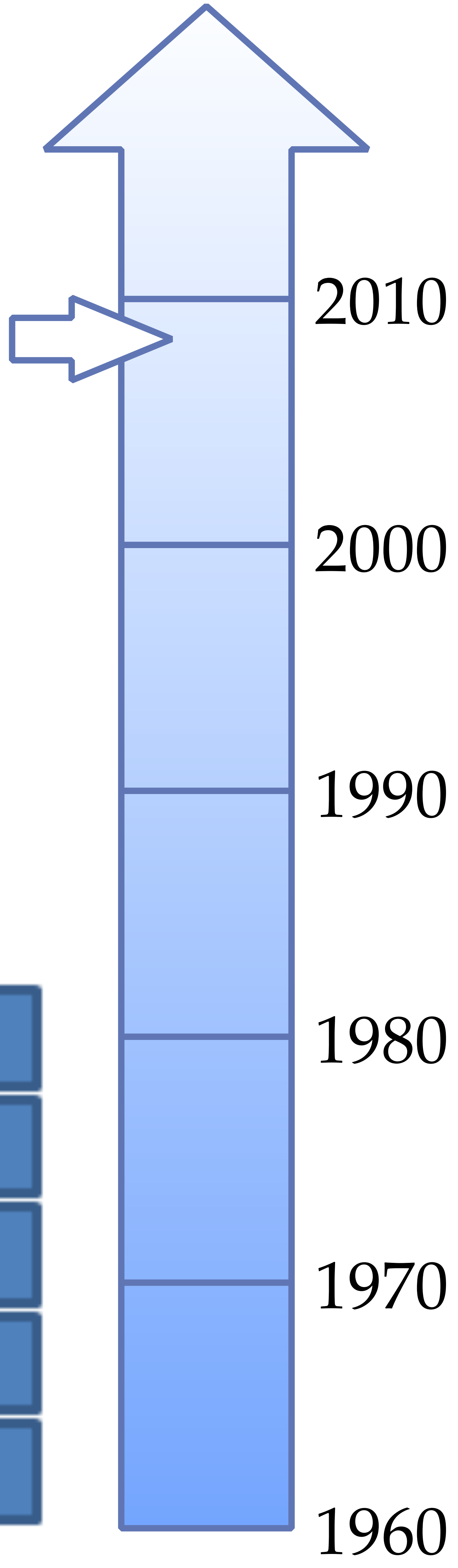
- Four important NoSQL data models are:
  - **Key-value**
    - e.g., AWS DynamoDB
  - **Document**
    - e.g., mongoDB
  - **Column-oriented**
    - e.g., Apache Cassandra,
    - Apache HBase
  - **Graph**
    - e.g., neo4j



Evolution of data models: **NoSQL**

- **Key-value** data model
  - A collection of <key,value> pairs
  - Works like a hash table or hash map
  - Values can be complex compound structures
    - DB doesn't care what they are
  - Look up based on the key
    - simple with no joins and foreign key constraints
  - Very easy to distribute across a cluster, partitioning on keys

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

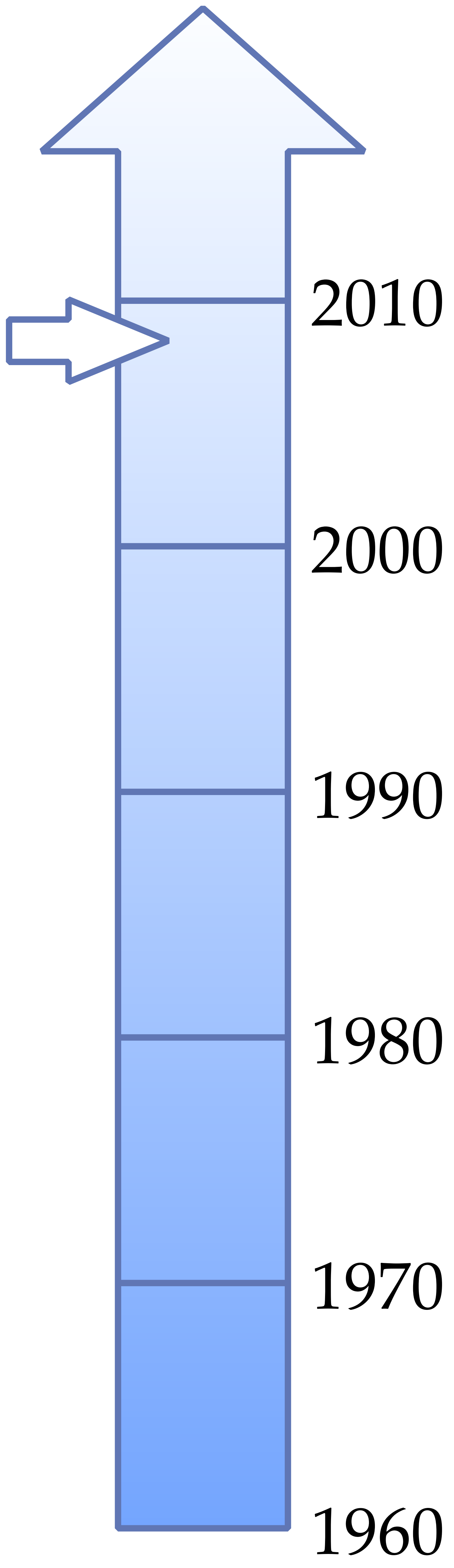


Evolution of data models: **NoSQL**

- **Document** data model
  - A collection of <key, document > pairs
  - A document is a block of data with some format or encoding such as JSON or XML
  - Allow to query based on content or metadata
  - No fixed schemas

```
<shipto>
  <name>Ola Nordmann</name>
  <address>Langgt 23</address>
  <city>4000 Stavanger</city>
  <country>Norway</country>
</shipto>
```

```
{
  "name":"John",
  "age":30,
  "cars": {
    "car1":"Ford",
    "car2":"BMW",
    "car3":"Fiat"
  }
}
```



Evolution of data models: **NoSQL**

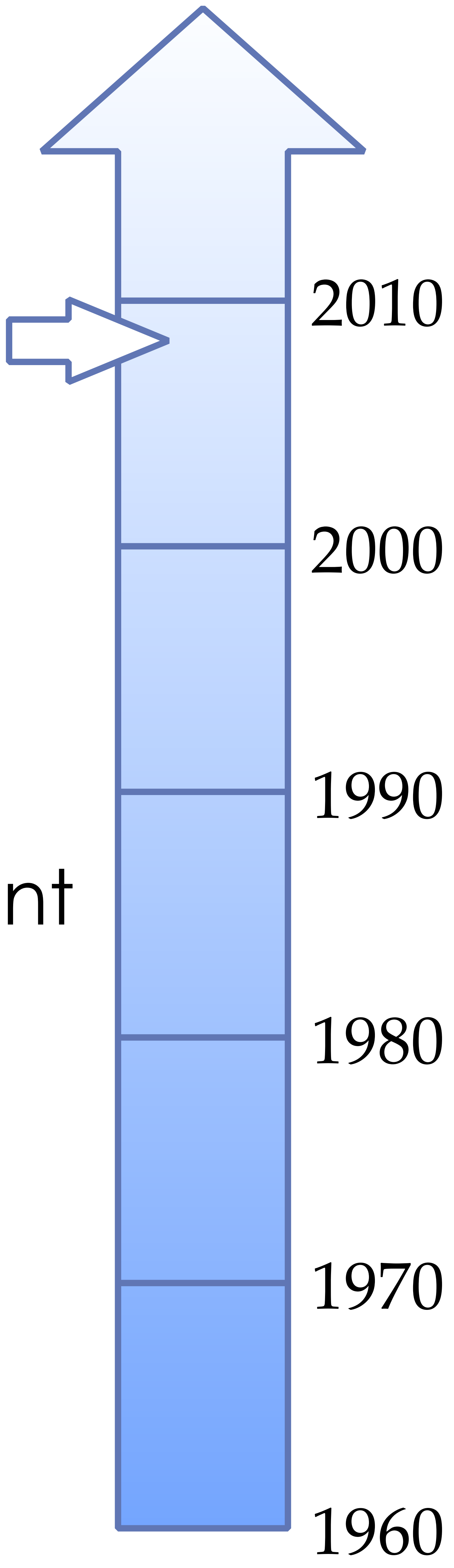
- **Column-oriented** data model (column store)
  - Organises table data by column rather than by row
  - Table of data on left stored in parts shown on right
  - **Column families** are analogous to tables in an RDBMS
    - but any data row may have different columns' data present

ID	*FName*	*LName*	*DOB*
1	Celia	Smith	1994-02-20
2	Alicia	Taylor	1991-04-02
3	John	Wong	1995-10-14
4	John	Taylor	1992-08-06

```
Celia:1; Alicia:2; John:3,4
```

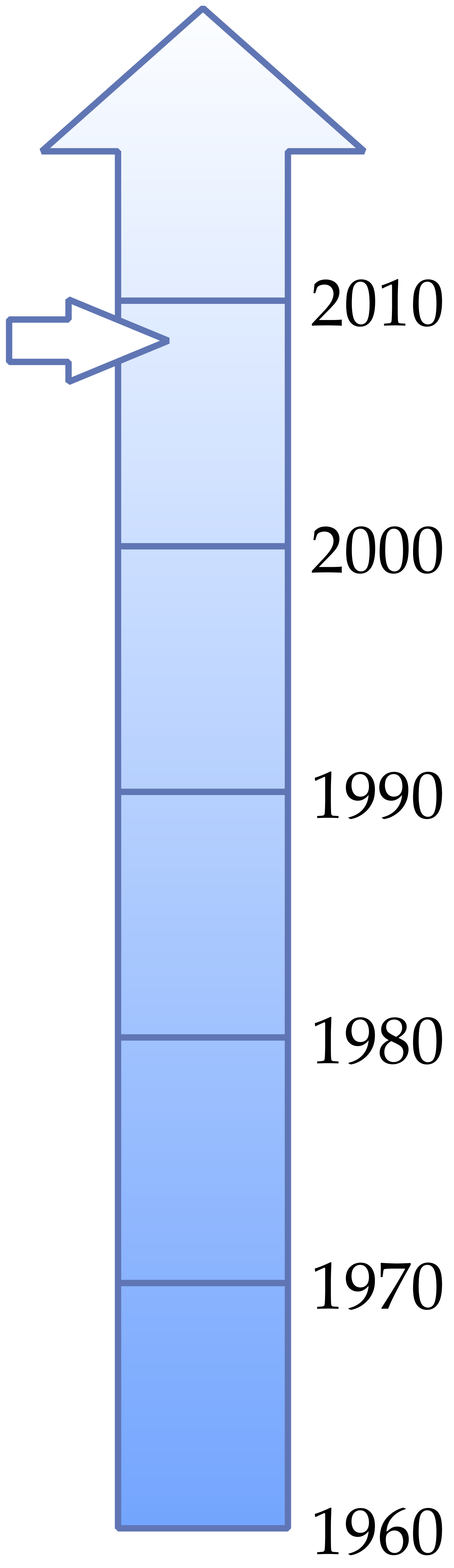
```
Smith:1; Taylor:2,4; Wong:3
```

```
1994-02-20:1; 1991-04-02:2;
1995-10-14:3; 1992-08-06:4
```



Evolution of data models: **NoSQL**

- Column-oriented database benefits:
  - Good data compression (collects similar data)
  - Performs well with aggregation queries such as summation, average, etc.
  - Scales up over clusters of machines easily due to data partitioning
  - Will speed up loading and querying of data for certain workloads

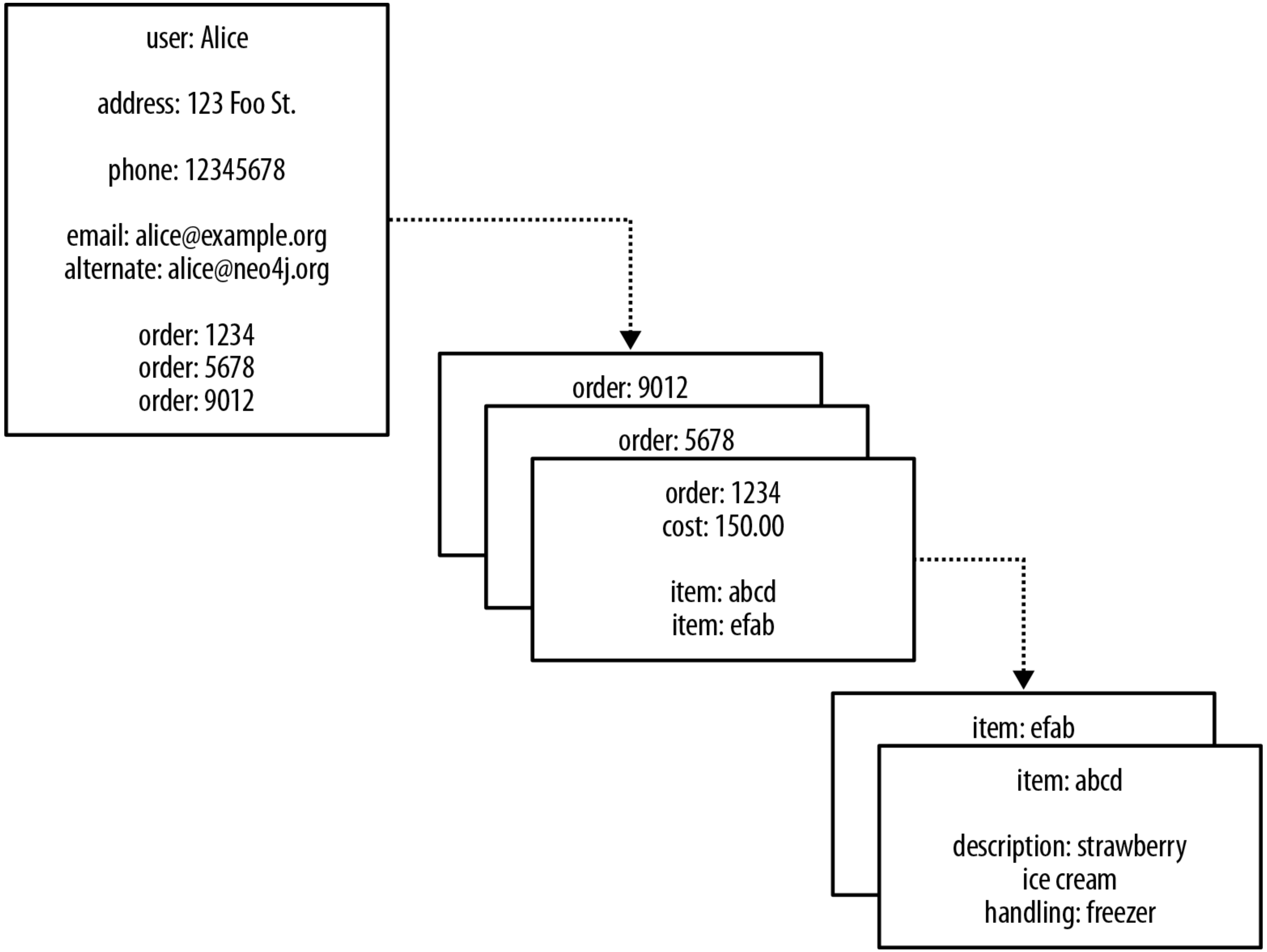


Relationships in different data models

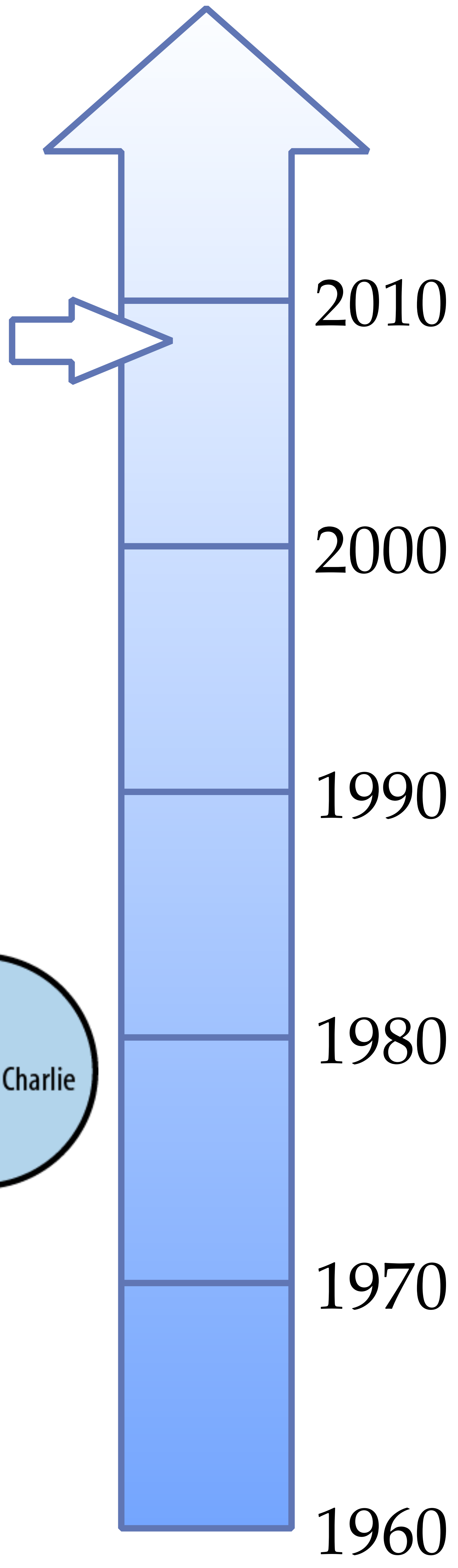
- Relational data model
  - Relationships exist only between tables (**foreign keys**)—not good for highly connected domains
  - Special checking required for nullable columns
  - Relationship traversal can be very expensive (**joins**)
- NoSQL data model (key-value, document, column)
  - Store set of disconnected values/documents
  - Embed an aggregate's identifier inside the field belonging to another aggregate (join at application level: expensive)

Relationships in different data models

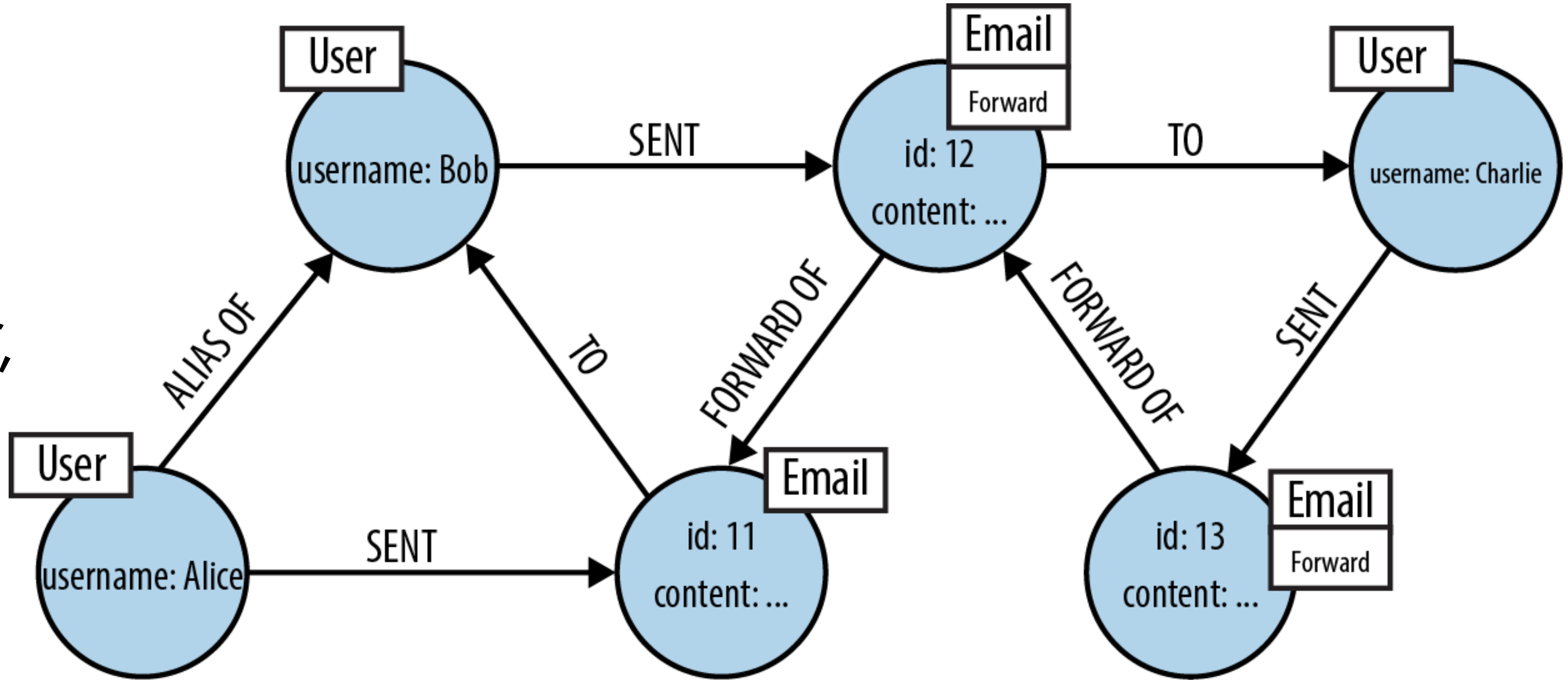
- In a highly-connected domain, consider storing graph directly
  - (Example from O'Reily *Graph Databases*)



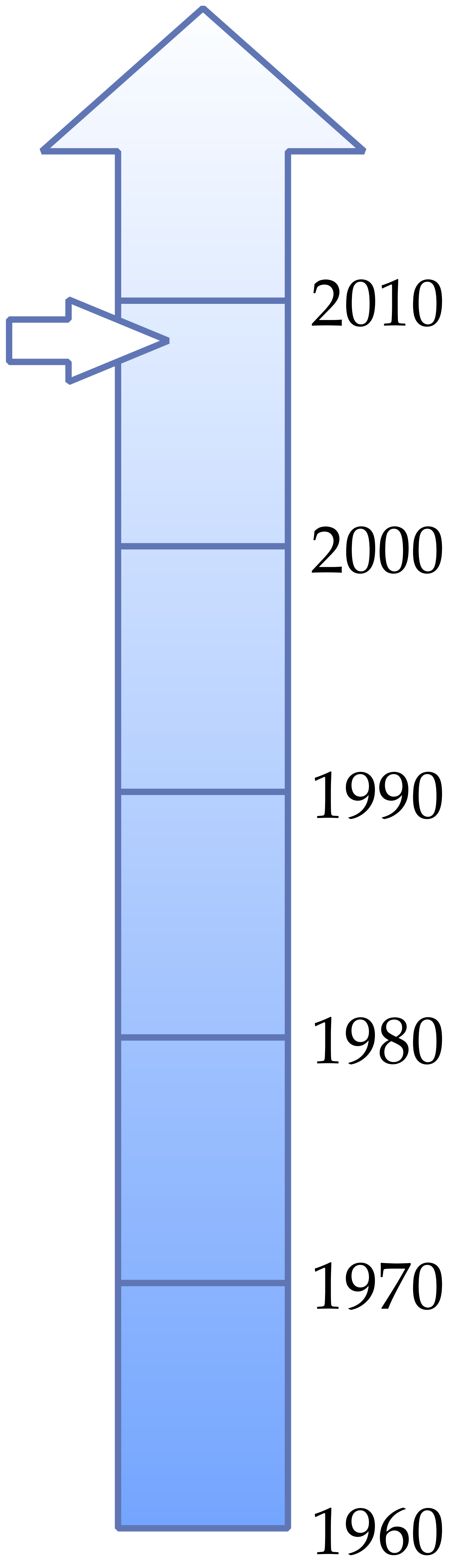
Evolution of data models: **NoSQL**



- **Graph data** model—directly store graph
- Labeled Property Graph Model:
  - **Nodes**: contain properties, with one or more labels
  - **Relationships**: connect nodes; can have properties
  - **Properties**: key-value pairs
  - **Labels**: group nodes together, e.g., based on "data type"



Evolution of data models: **NoSQL**



- Advantages of graph databases
  - Explicit graph structure is stored directly:
    - semantic dependencies are made explicit
  - New nodes and new relationships can be easily added without data migration and restructuring
  - Relationships correspond to paths; querying the database involves traversing the graph
  - Convenience of schema-free operation
  - Suitable to model complex, highly-connected data
    - e.g., social networks, web data, product preferences

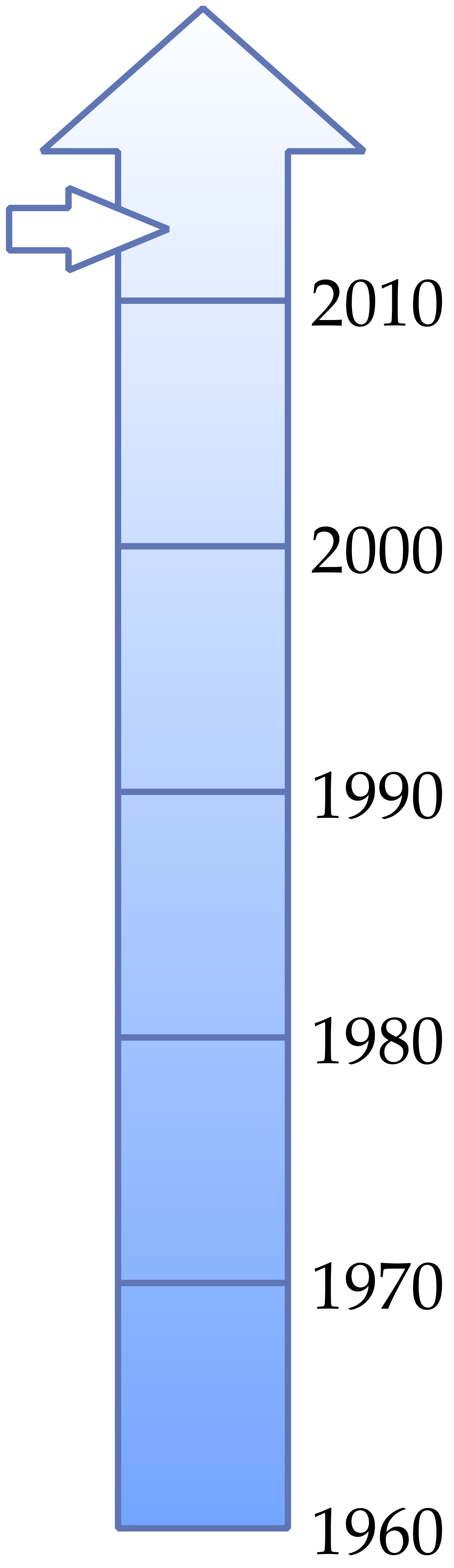
Schema-less databases

- Schema-less operation: a common theme across many forms of NoSQL databases:
  - Key-value stores allow any data under a given key
  - Document databases make no restriction on the structure of the documents
  - Column-oriented databases can produce tuples that contain any set of columns (values in each column point to rowid)
  - Graph databases allow new nodes to be added freely into graphs, also for properties and relationships to be updated

Schema versus schema-less databases

- Schema:
  - Global data definition—tuple data types, constraints
  - Can help optimal data storage, management, and access
  - Less flexibility—must maintain schema, be aware of data types
- Schema-less:
  - Flexibility—any kind of data, easily change data organisation
  - Ease of use and maintenance
  - Poor integrity
  - Performance suffers—implicit schema shifted to app.'s code

Evolution of data models: **NewSQL**



- Not actually a new data model:
  - Uses the relational model
  - … but mixes in many NoSQL advantages
  - Term "NewSQL" first used around 2011

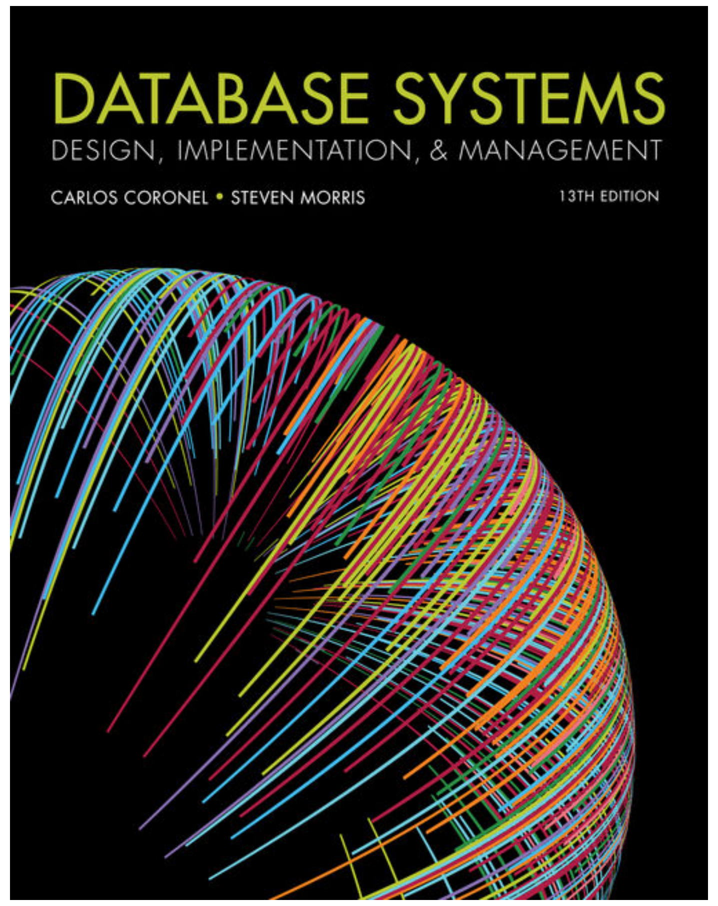
- NewSQL implementations usually:
  - Scale-out easily across clusters
  - …yet provide strong RDBMS-style consistency (ACID)
  - Provide a standard SQL interface

Summary

- Concept of data model
- Data abstraction and independence
- Evolution of data models:
  - Hierarchical data model
  - Network data model
  - Relational data model
  - Key-value store
  - Document
  - Column-family
  - Graph
  - (NewSQL)
- Characteristics of “big data”
- Schema-less databases

Useful reading material

- Database Systems Design, Implementation, and Management (13th edition)
  - Carlos Coronel, Steven Morris and Peter Rob
  - Chapter 2



- NoSQL Distilled: A Brief Guide to the Emerging World of Ployglot Persistence
  - Pramod J. Sadalage and Martin Fowler
  - Chapters 1–3