

Data Engineering: Data Modeling

LTAT.02.007

Ass Prof. Riccardo Tommasini

Assistants: **Fabiano Spiga, Mohamed Ragab, Hassan Eldeeb**

- <https://courses.cs.ut.ee/2020/dataeng>
- Forum
- Moodle

What is Data Modeling?

It is the process of defining the structure of the data for the purpose of communicating¹ or to develop an information systems².

¹ between functional and technical people to show data needed for business processes

² between components of the information system, how data is stored and accessed.

What is a data model?

A data model represents the structure and the integrity of the data elements of a (single) applications [2](#)

Data models provide a framework for data to be used within information systems by providing specific definition and format.

The literature of data management is rich of data models that aim at providing increased expressiveness to the modeler and capturing a richer set of semantics.

Data models are perhaps the most important part of developing software. They have such a profound effect not only on how the software is written, but also on how we think about the problem that we are solving¹.

– *Martin Kleppmann*

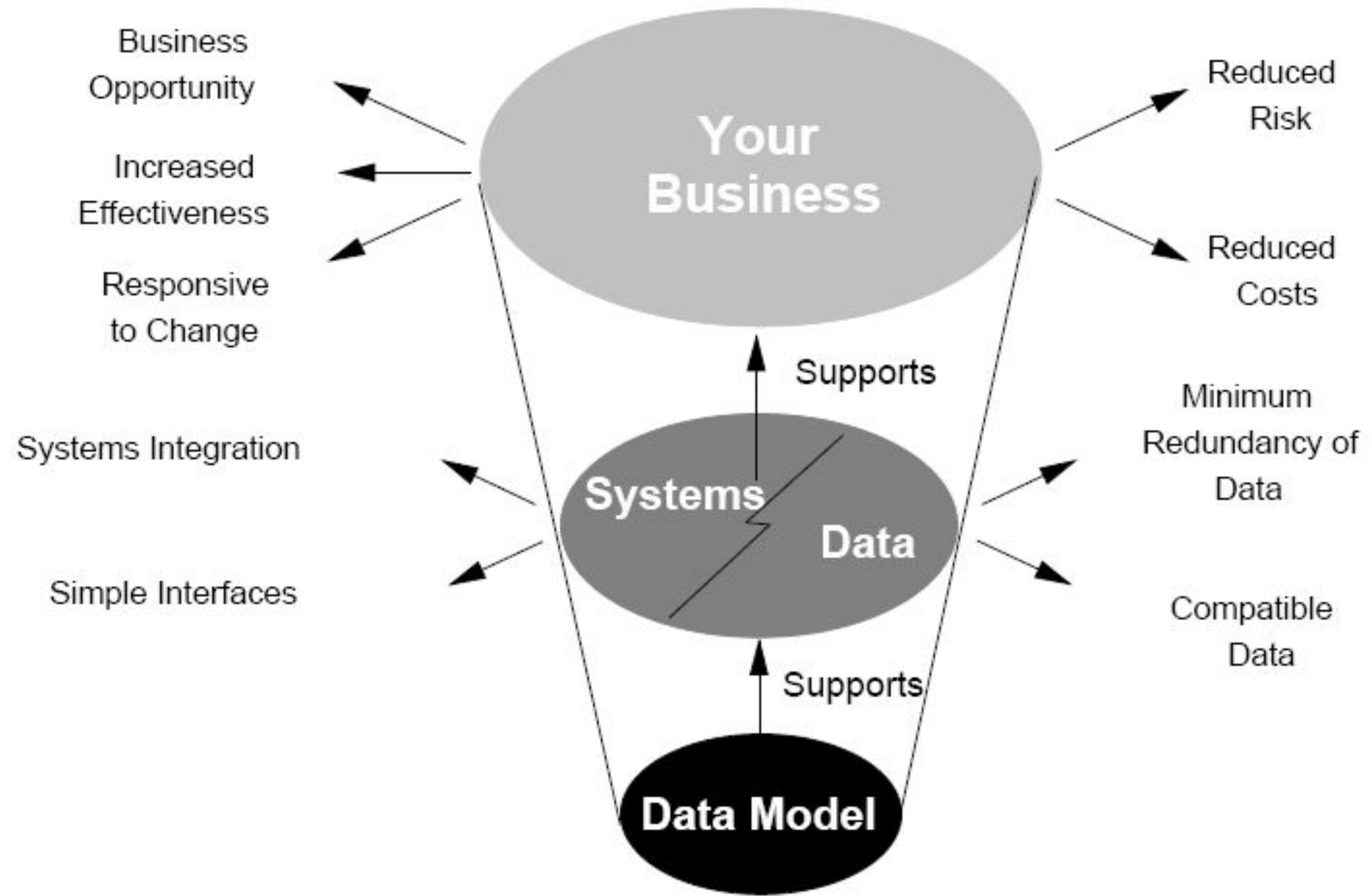
¹ between functional and technical people to show data needed for business processes

Any Example?

Example: The STUDENT table

ID	Last Name	First Name	Year	QPI
081111	Cruz	Juan	2	3.5
072222	Smith	John	3	2.2
089999	Cruz	Maria	3	4.0
077676	Santos	Linda	4	3.0

There are four rows/records in this example



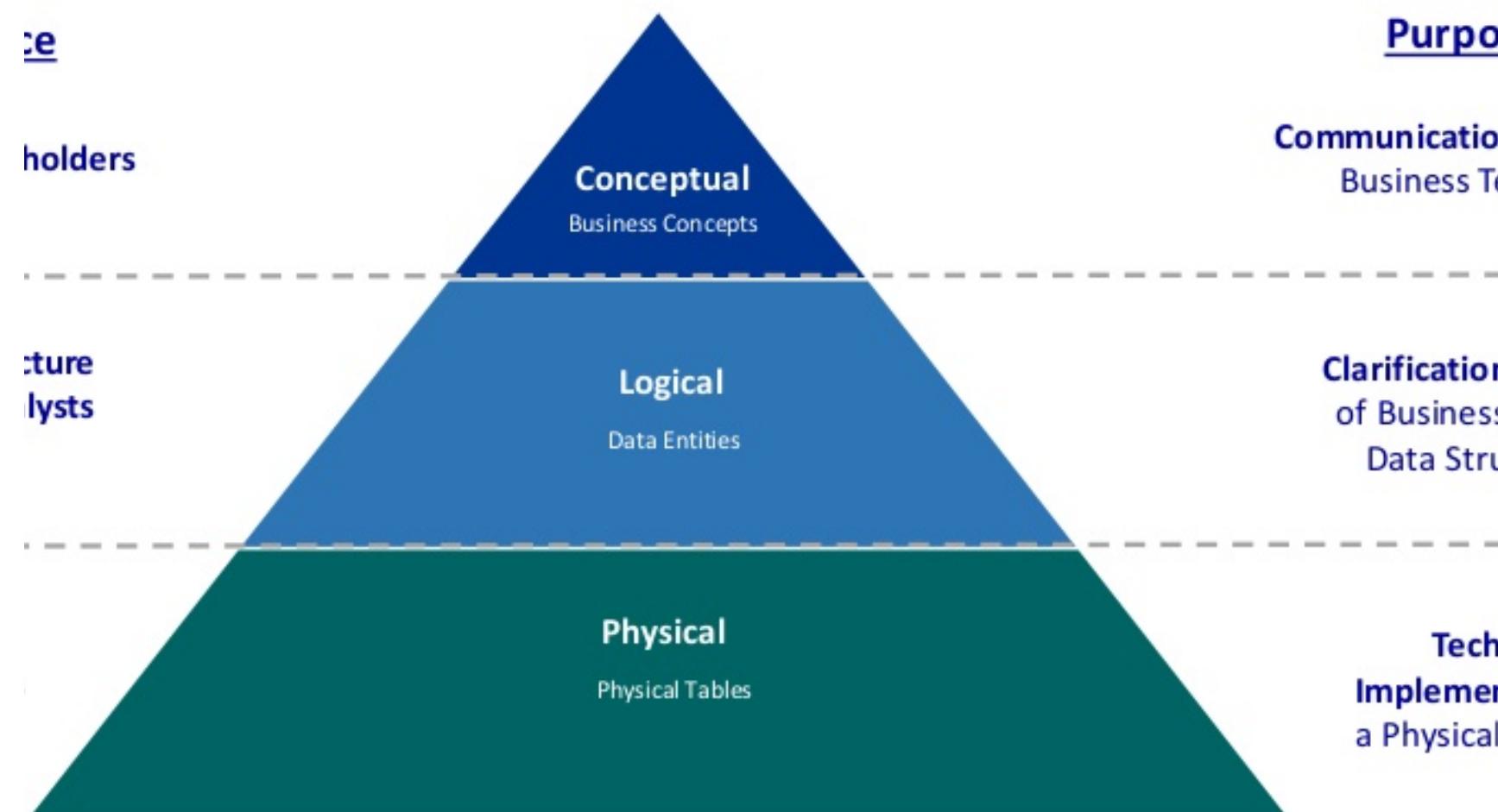
Level of Data Modeling

Conceptual: The data model defines *WHAT* the system contains.

Logical: Defines *HOW* the system should be implemented regardless of the DBMS.

Physical: This Data Model describes *HOW* the system will be implemented using a specific DBMS system³.

Levels of Data Modeling

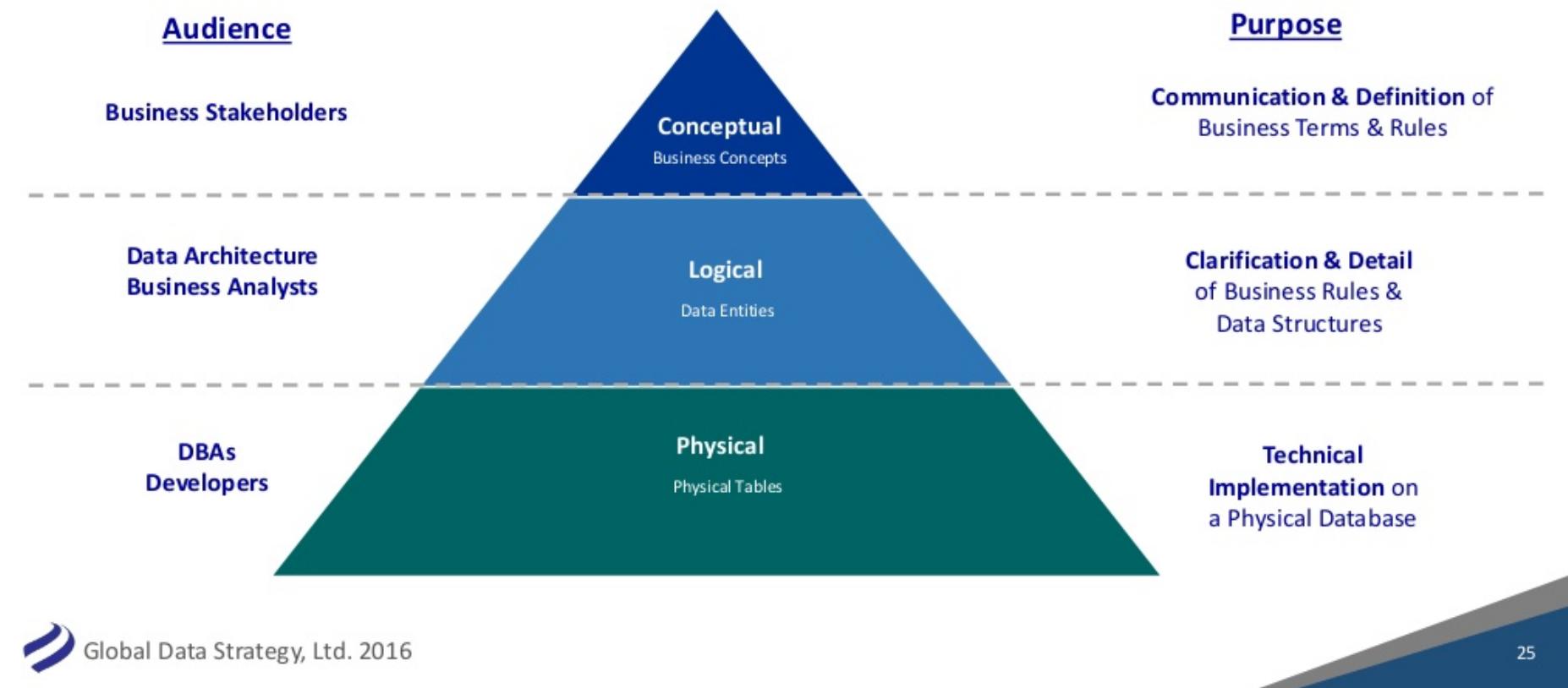


egy, Ltd. 2016

³ Designing Data-Intensive Applications

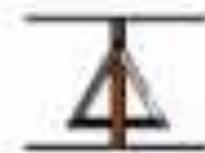
A Closer Look⁴

Levels of Data Modeling



⁴ physical

We need help from Rudyard Kipling



Copyright © 2012, Essential Strategies, Inc.

2/34

Conceptual

- Semantic Model (divergent)
 - Describes an enterprise in terms of the language it uses (the jargon).
 - It also tracks inconsistencies, i.e., semantic conflicts
- Architectural Model (convergent)
 - More fundamental, abstract categories across enterprise

Logical

Already bound to a technology, it typically refers already to implementation details

- Relational
- Hierarchical
- Key-Value
- Object-Oriented
- Graph

Physical

The physical level describes how data are **Stored** on a device.

- Data formats
- Distribution
- Indexes
- Data Partitions
- Data Replications

...and you are in the Big Data World

A Question

Why should you, ~~an application developer~~
a data engineer, care how the database
handles storage and retrieval internally?

– Martin Kleppmann

I mean, you're probably not going to
implement your own storage engine from
scratch...



The Students



But...

- You do need to select a storage engine that is appropriate for your application, from the many that are available
- You need to tune a storage engine to perform well on your kind of workload
- You are going to experiment with different access patterns and data formats

Therefore, you must have a rough idea of what the storage engine is doing under the hood

Also the Students



Data Formats

- In memory, data are kept in objects, structs, lists, arrays, hash tables, trees, and so on. These data structures are optimized for efficient access and manipulation by the CPU (typically using pointers).
- On Disk (or over the network), data are encoded into a self-contained sequence of bytes (for example, a JSON document).

Encoding and decoding

Encoding is the translation from the in-memory representation to a byte sequence (also known as serialization or marshalling)

Decoding is the reverse translation from the byte sequence to a memory layout (also known as parsing, deserialization, unmarshalling)

The encoding is often tied to a particular programming language, and reading the data in another language is very difficult

Memory vs Disk

Data layout is much less important in memory than on disk.

An efficient disk-resident data structure must allow quick access to it, i.e., find a way to serialize and deserialize data rapidly and in a compacted way.

In general, pointers do not make sense outside memory, thus the sequence-of-bytes representation looks quite different from the data structures that are normally used in memory.

Popular (textual) File Formats

JSON

- has a schema
- cannot distinguish between integers and floating-point numbers
- have good support for Unicode character string
- do not support sequences of bytes without a character encoding

XML

- has a schema
- cannot distinguish between a number and a string
- have good support for Unicode character string
- do not support sequences of bytes without a character encoding

CSV

- cannot distinguish between a number and a string
- does not have any schema

Avro

Avro is a binary encoding format that uses a schema to specify the structure of the data being encoded.

Avro's encoding consists only of values concatenated together, and there is nothing to identify fields or their datatypes in the byte sequence.

Avro Schema Definition

```
record Person {  
    string userName;  
    union { null, long } favoriteNumber = null;  
    array<string> interests;  
}
```

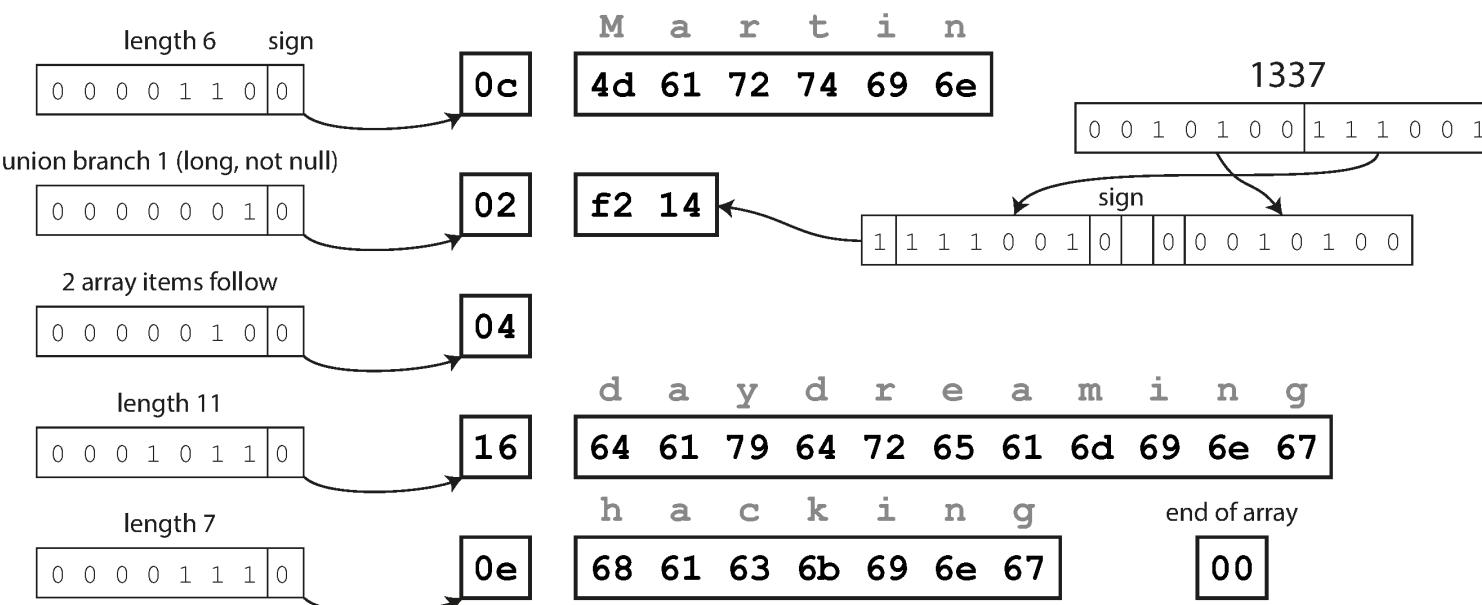
Example

Avro

Byte sequence (32 bytes):

0c	4d	61	72	74	69	6e	02	f2	14	04	16	64	61	79	64	72	65	61	6d
69	6e	67	0e	68	61	63	6b	69	6e	67	00								

Breakdown:



Source

Encoding and Decoding

- Encoding requires the writer's schema
- Decoding requires the reader's schema.
- Avro does not require that the writer's schema and the reader's schema are the same, they only need to be **compatible**

Schema Evolution Rules

- If the code reading the data encounters a field that appears in the writer's schema but not in the reader's schema, it is ignored.
- If the code reading the data expects some field, but the writer's schema does not contain a field of that name, it is filled in with a default value declared in the reader's schema.

Compatibility

- forward compatibility: there is a new version of the writer's schema and an old version of the reader's schema
- backwards compatibility: there is a new version of the reader's schema and an old version of the writer's schema

Worth Mentioning⁸

- Apache Thrift and Protocol Buffers are binary encoding libraries
 - require a schema for any data that is encoded.
 - come with a code generation tool that takes a schema definitions to reproduce the schema in various programming languages

```
struct Person {  
    1: required string userName,  
    2: optional i64 favoriteNumber,  
    3: optional list<string> interests  
}
```

```
message Person {  
    required string user_name = 1;  
    optional int64 favorite_number = 2;  
    repeated string interests = 3;  
}
```

⁸ Chapter 4. Encoding and Evolution

A dense, colorful pile of various LEGO components, including bricks of different sizes and colors (red, blue, yellow, green, black, white), as well as gears, axles, and other structural elements.

Distribution

CAP Theorem (Brewer's Theorem)

It is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

- **Consistency**: all nodes see the same data at the same time
- **Availability**: Node failures do not prevent other survivors from continuing to operate (a guarantee that every request receives a response whether it succeeded or failed)
- **Partition tolerance**: the system continues to operate despite arbitrary partitioning due to network failures (e.g., message loss)

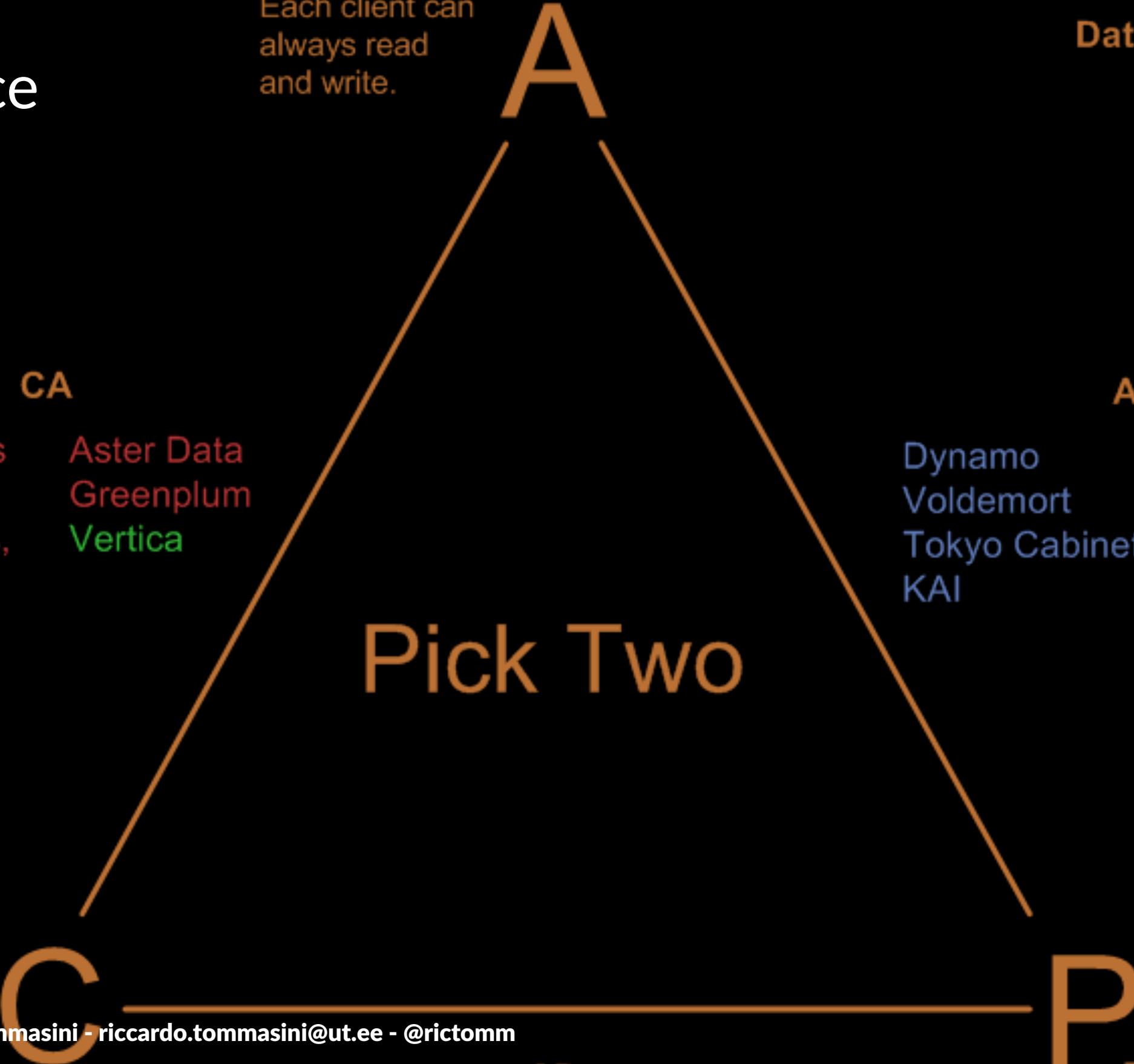
A distributed system can satisfy any two of these guarantees at the same time but not all three.

source

Availability:
Each client can
always read
and write.

Data Models

Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented



The network is not reliable

In a distributed system, **a network (of networks)* * failures can, and will, occur.

We cannot neglect Partition Tolerance

The remaining option is choosing between **Consistency** and **Availability**.

We cannot neglect Partition Tolerance

Not necessarily in a mutually exclusive manner:

- CP: A partitioned node returns
 - the correct value
 - a timeout error or an error, otherwise
- AP: A partitioned node returns the most recent version of the data, which could be stale.

Indexing

- Indices are critical for efficient processing of queries in (any kind of) databases.
- basic idea is trading some computational cost for space, i.e., materialize a convenient data structure to answer a set of queries.
- The caveat is that we must maintain indexes up-to-date upon changes

Basics Terms

- Ordered indices. Based on a sorted ordering of the values.
- Hash indices. Using an hash-function that assigns values across a range of buckets.
- Primary Index: denotes an index on a primary key
- Secondary Index: denotes an index on non primary values

Data Replication

Replication means keeping a copy of the same data on multiple machines that are connected via a network



Reasons for Replication

- Increase data locality
- Fault tolerance
- Concurrent processing (read queries)

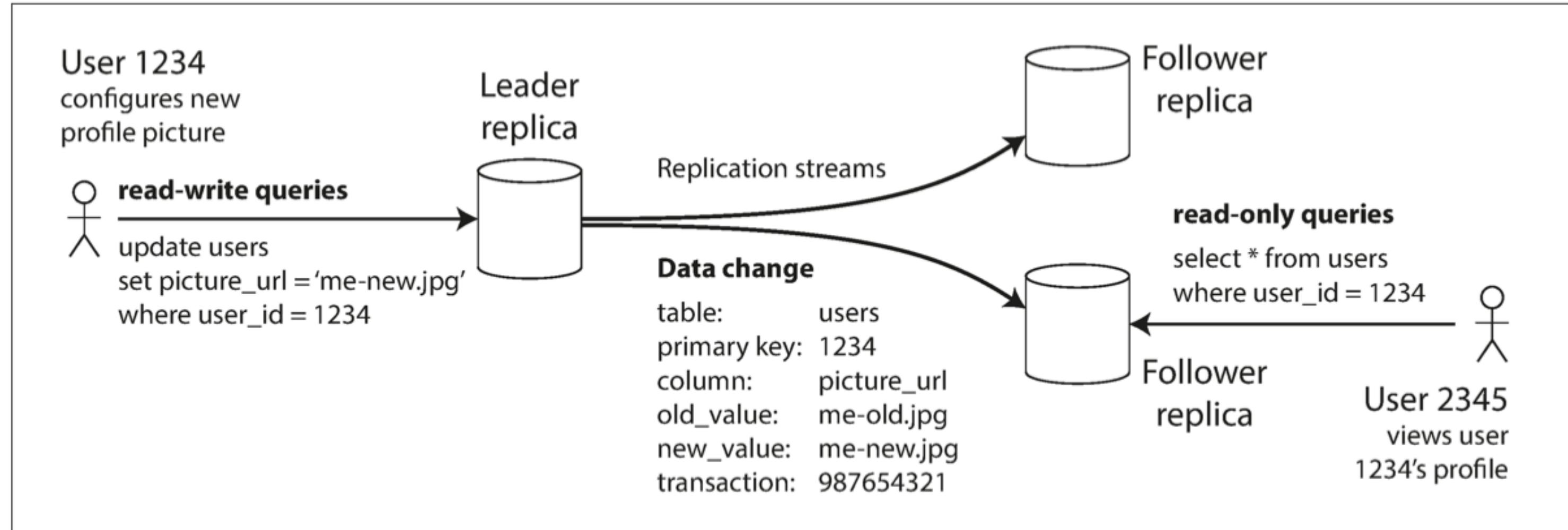
Approaches

- Synchronous vs Asynchronous Replication
 - The advantage of synchronous replication is that the follower is guaranteed to have an up-to-date copy
 - The advantage of asynchronous replication is that follower's availability is not a requirement (cf CAP Theorem)
- Leader - Follower (Most common cf Kafka)

Leaders and Followers

- One of the replicas is designated as the leader
- Write requests go to the leader
- leader sends data to followers for replication
- Read request may be directed to leaders or followers





Source is³

³ Designing Data-Intensive Applications

Caveats

Only one: handling changes to replicated data is extremely hard.

Data Partitioning (Sharding)

breaking a large database down into smaller ones

Reasons for Partitioning

- The main reason for wanting to partition data is scalability³

³ Designing Data-Intensive Applications

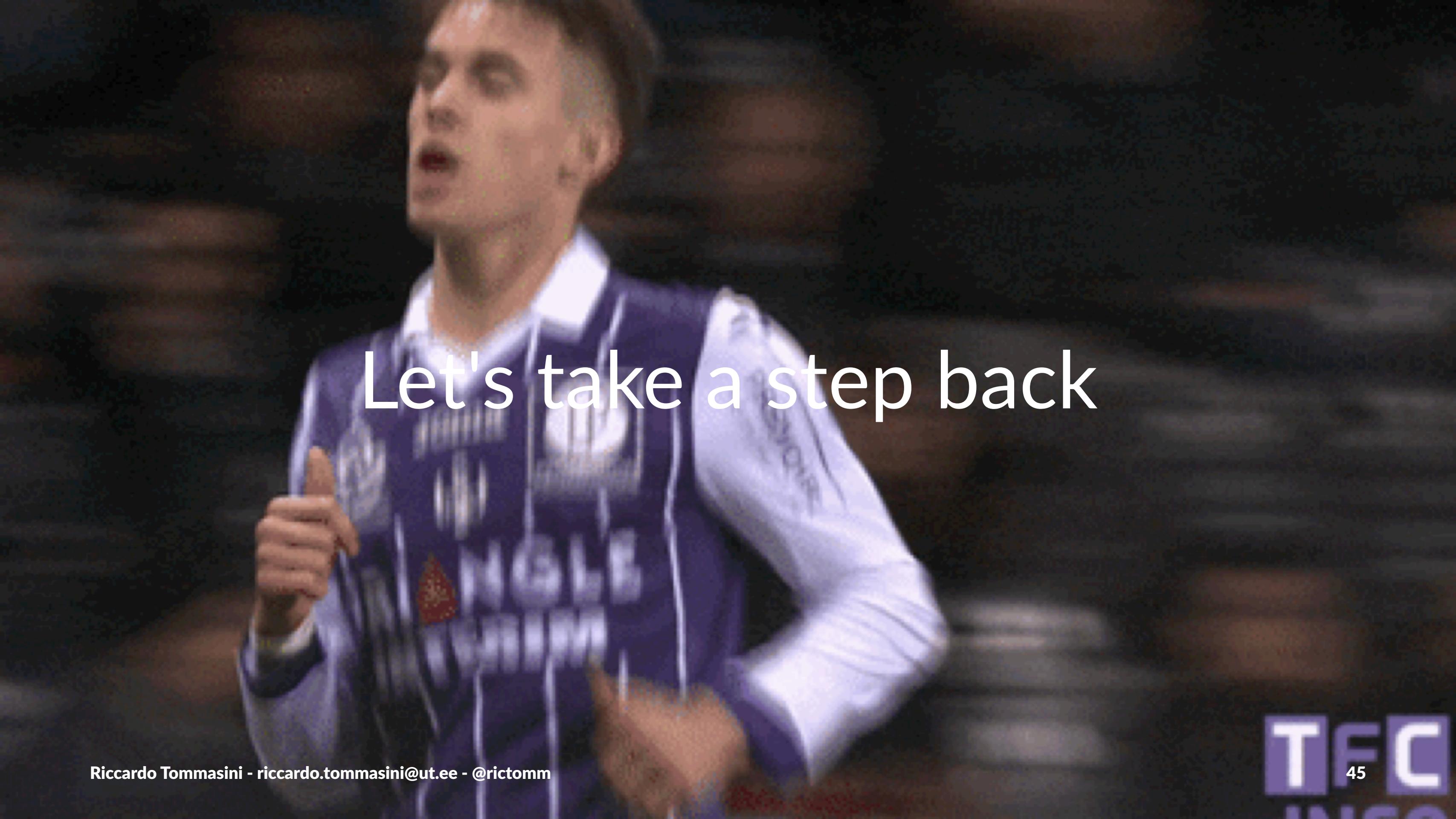
What to know

- If some partitions have more data or queries than others the partitioning is **skewed**
- A partition with disproportionately high load is called a **hot spot**
- For reaching maximum scalability (linear) partitions should be balanced

Let's consider some partitioning strategies, for simplicity we consider Key,Value data.

Partitioning Strategies

- **Round-robin** randomly assigns new keys to the partitions.
 - Ensures an even distribution of tuples across nodes;
- **Range partitioning** assigns a contiguous key range to each node.
 - Not necessarily balanced, because data may not be evenly distributed
- **Hash partitioning** uses a hash function to determine the target partition. - If the hash function returns i , then the tuple is placed

A man in a dark suit and white shirt is looking upwards and slightly to his left. He is holding a thick, dark book or folder against his chest with both hands. The background is dark and out of focus.

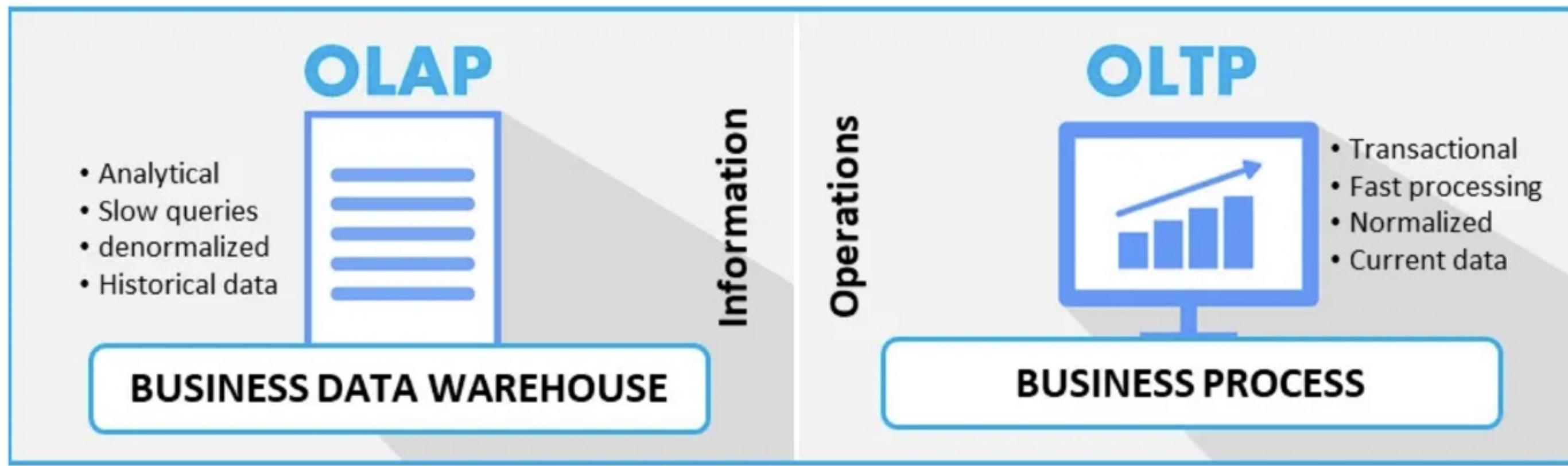
Let's take a step back



To the future

Let's Talk about Workloads

OLAP Vs OLTP



Online Transactional Processing

Because these applications are interactive, the access pattern became known as **online**

Transactional means allowing clients to make low-latency reads and writes—as opposed to batch processing jobs, which only run periodically (for example, once per day).

Refresh on ACID Properties

- ACID, which stands for Atomicity, Consistency, Isolation, and Durability¹⁰
- **Atomicity** refers to something that cannot be broken down into smaller parts.
 - It is not about concurrency (which comes with the I)
- **Consistency** (overused term), that here relates to the data *invariants* (integrity would be a better term IMHO)
- **Isolation** means that concurrently executing transactions are isolated from each other.
 - Typically associated with serializability, but there weaker options.
- **Durability** means (fault-tolerant) persistency of the data, once the transaction is completed.
- ^ The terms was coined in 1983 by Theo Härdter and Andreas Reuter¹⁰

¹⁰ Theo Härdter and Andreas Reuter: "Principles of Transaction-Oriented Database Recovery," ACM Computing Surveys, volume 15, number 4, pages 287–317, December 1983. doi:10.1145/289.291

Online Analytical Processing

An OLAP system allows a data analyst to look at different cross-tabs on the same data by interactively selecting the attributes in the cross-tab

Statistical analysis often requires grouping on multiple attributes.

Example⁹

Consider this is a simplified version of the sales fact table joined with the dimension tables, and many attributes removed (and some renamed)

sales (*itemname*, *color*, *clothesize*, quantity)

⁹ Database System Concepts Seventh Edition Avi Silberschatz Henry F. Korth, S. Sudarshan McGraw-Hill ISBN 9780078022159 [link](#)

item_name	color	clothes_size	quantity
dress	dark	small	2
dress	dark	medium	6
...
pants	pastel	medium	0
pants	pastel	large	1
pants	white	small	3
pants	white	medium	0
shirt	white	medium	1
...
shirt	white	large	10
skirt	dark	small	2
skirt	dark	medium	5
...

Cross-tabulation of sales by item name and color

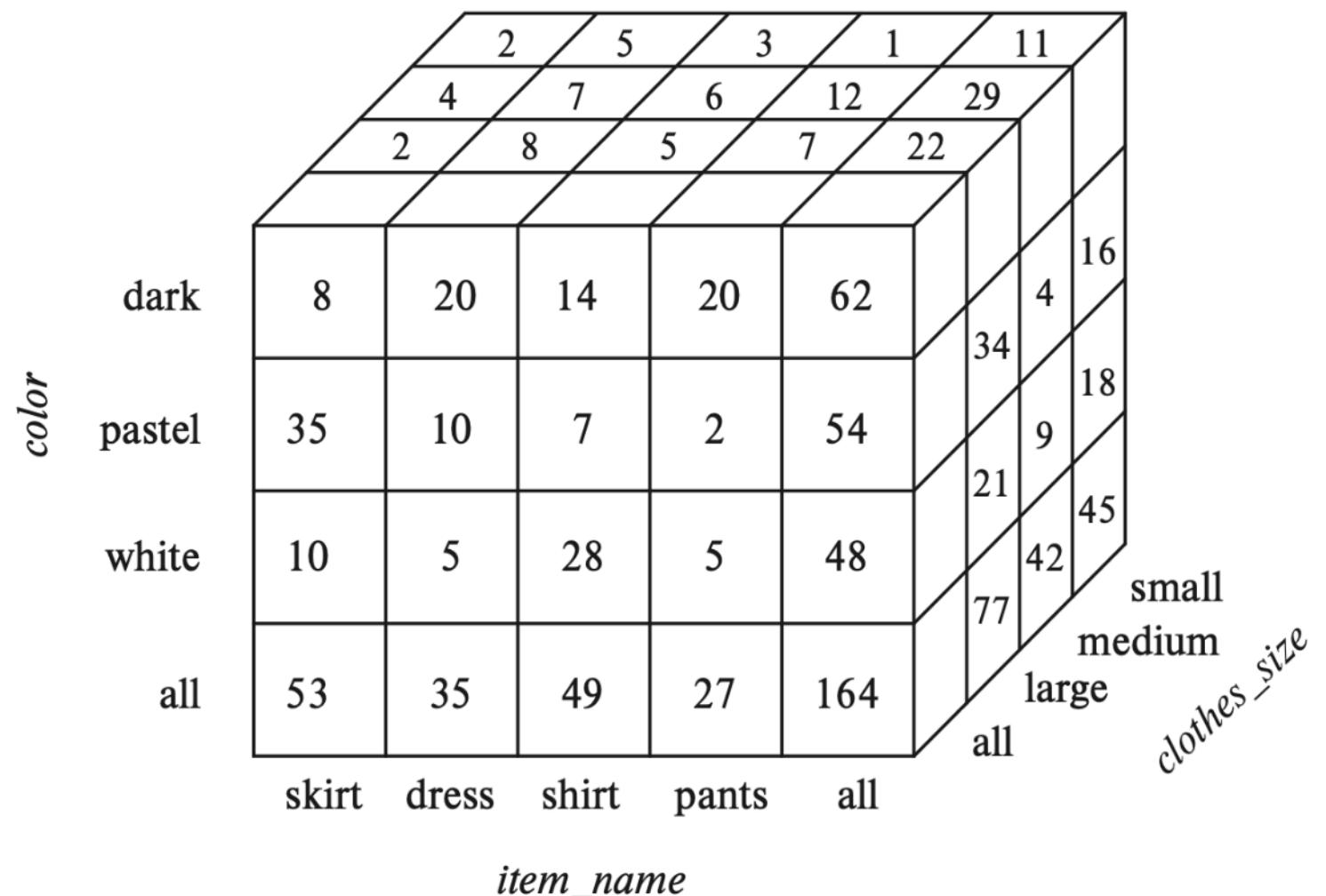
	dark	pastel	white	total
skirt	8	35	10	53
dress	20	11	5	36
shirt	22	4	46	72
pants	23	42	25	90
total	73	92	102	267

columns header: color

rows header: item name

Data Cube⁹

- It is the generalization of a Cross-tabulation



⁹ Database System Concepts Seventh Edition Avi Silberschatz Henry F. Korth, S. Sudarshan McGraw-Hill ISBN 9780078022159 [link](#)

Cheat Sheet of OLAP Operations⁹

- **Pivoting:** changing the dimensions used in a cross-tab
 - E.g. moving colors to column names
- **Slicing:** creating a cross-tab for fixed values only
 - E.g fixing color to white and size to small
 - Sometimes called dicing, particularly when values for multiple dimensions are fixed.
- **Rollup:** moving from finer-granularity data to a coarser granularity
 - E.g. aggregating away an attribute
 - E.g. moving from aggregates by day to aggregates by month or year
- **Drill down:** The opposite operation - that of moving from coarser granularity data to finer-granularity data

⁹ Database System Concepts Seventh Edition Avi Silberschatz Henry F. Korth, S. Sudarshan McGraw-Hill ISBN 9780078022159 [link](#)

Summary OLTP vs OLAP³

Property	OLTP	OLAP
Main read pattern	Small number of records per query, fetched by key	Aggregate over large number of records
Main write pattern	Random-access, low-latency writes from user input	Bulk import (ETL) or event stream
Primarily used by	End user/customer, via web application	Internal analyst, for decision support
What data represents	Latest state of data (current point in time)	History of events that happened over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

³ Designing Data-Intensive Applications

Modeling for (relational) Database⁶

- Works in phases related to the aforementioned levels of abstractions
- Uses different data models depending on the need:
 - Relational, Graph, Document...
- Tries to avoid two major pitfalls:
 - **Redundancy:** A design should not repeat information
 - **Incompleteness:** A design should not make certain aspects of the enterprise difficult or impossible to model
- Optimized for OLTP

⁶ Also known as Database Design

Before, let's refresh

Relational Database

A relational database consists of...

- a set of relations (tables)
- a set of integrity constraints

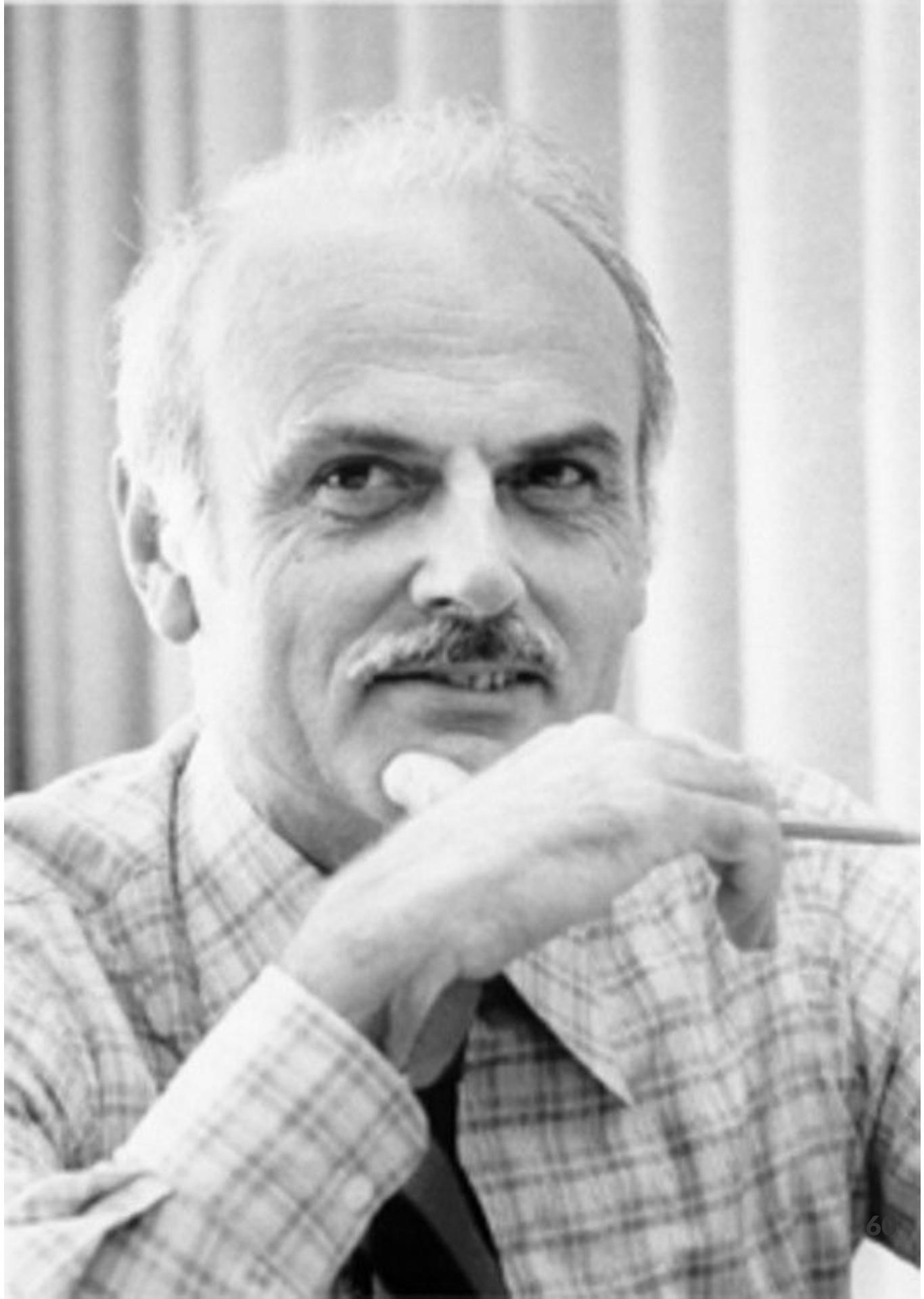
If the database satisfies all the constraints we said it is in a valid state.

An important distinction regards the **database schema**, which is the logical design of the database, and the **database instance**, which is a snapshot of the data in the database at a given instant in time.

Relational Model ¹⁷

A formal mathematical basis for databases based on set theory and first-order predicate logic

Underpins of SQL



¹⁷ Extra Read Codd, Edgar F. "A relational model of data for large shared data banks." *Communications of the ACM* 13.6 (1970): 377-38z

Relation

Relation R is a set of tuples (d_1, d_2, \dots, d_n) , where each element d_j is a member of D_j , a data domain.

A Data Domain refers to all the values which a data element may contain, e.g., N.

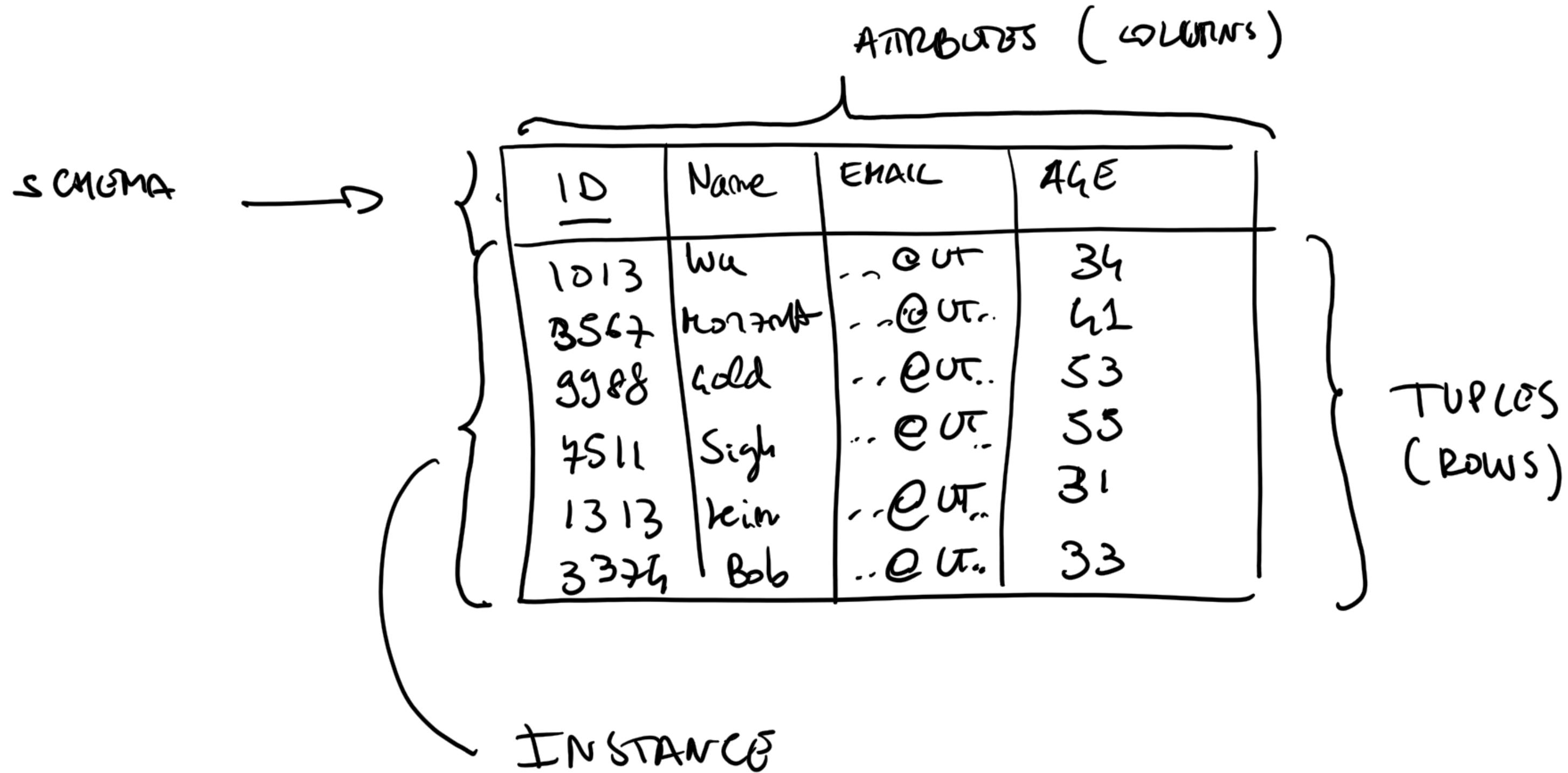
Note that in the relational model the **term relation is used to refer to a table**, while the term **tuple is used to refer to a row**

ATTRIBUTES (columns)

ID	Name	EMAIL	AGE
1013	Wu	@ut.	34
3567	Moritz	@ut.	41
9988	Gold	@ut.	53
7511	Sigh	@ut.	55
1313	Kein	@ut.	31
3324	Bob	@ut.	33

Relation Schema

- corresponds to the notion of **type** in programming languages
- consists of a list of **attributes** and their corresponding domains
- a **relation instance** corresponds to the programming-language notion of a value of a variable



Keys

- A **superkey** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation
- **candidate keys** are superkeys for which no proper subset is a superkey
- primary key is the chosen candidate key
- foreign key is a set of attributes from a referenced relation.

ATTRIBUTES (columns)

SCHEMA

ID **Name** **EMAIL** **AGE**

ID	Name	EMAIL	AGE
1013	Wu	...@ut.	34
3567	Moritz	...@ut.	41
9988	Godd	...@ut.	53
7511	Sigh	...@ut.	55
1313	lein	...@ut.	31
3324	Bob	...@ut.	33

TUPLES (rows)

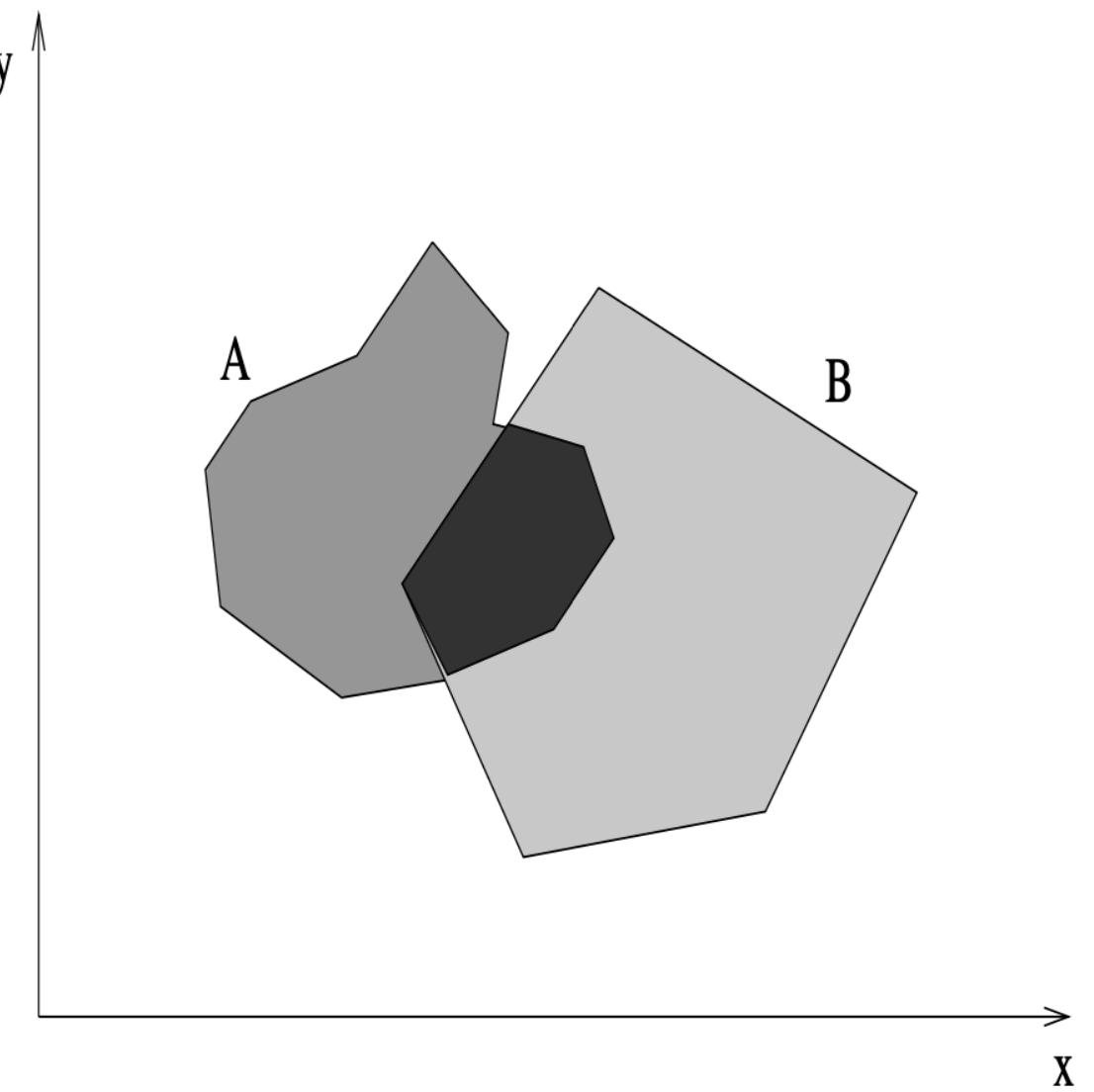
key

Relational Algebra (On Practice)

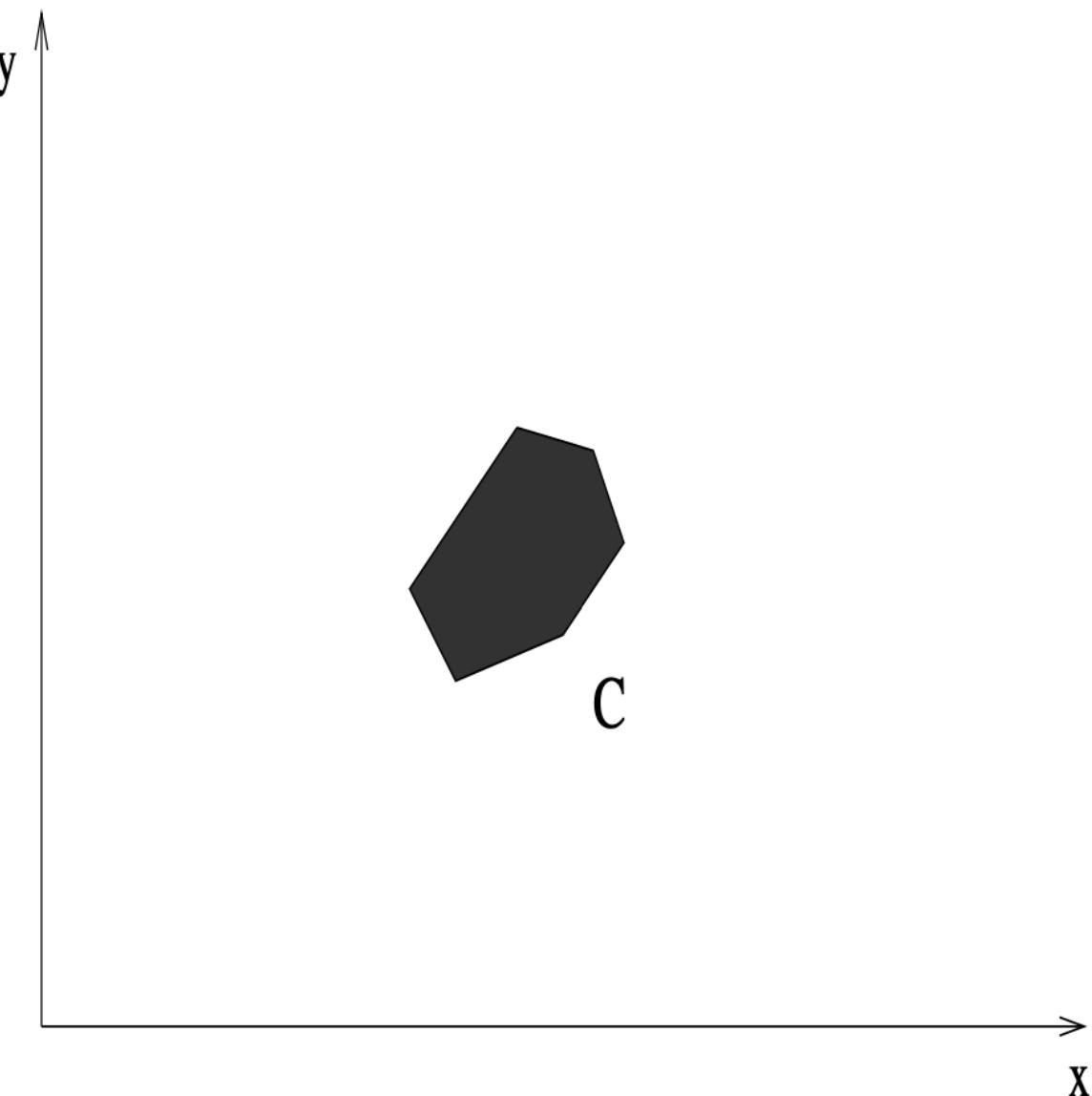
is a procedural language consisting of a six basic operations that take one or two relations as input and produce a new relation as their result:

- select: σ
- project: Π
- union: \cup
- set difference: $-$
- Cartesian product: \times
- rename: ρ

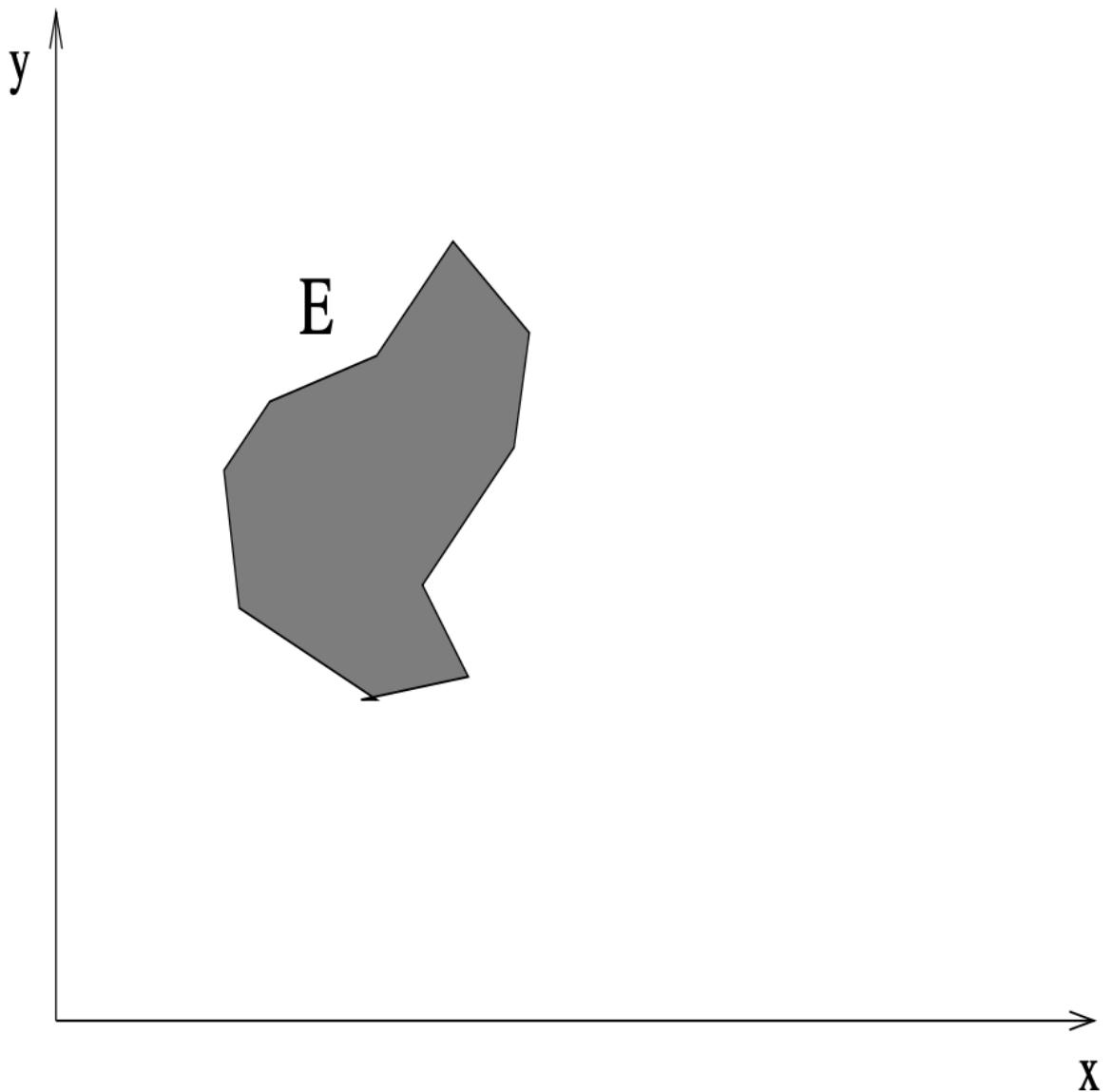
Two Sets



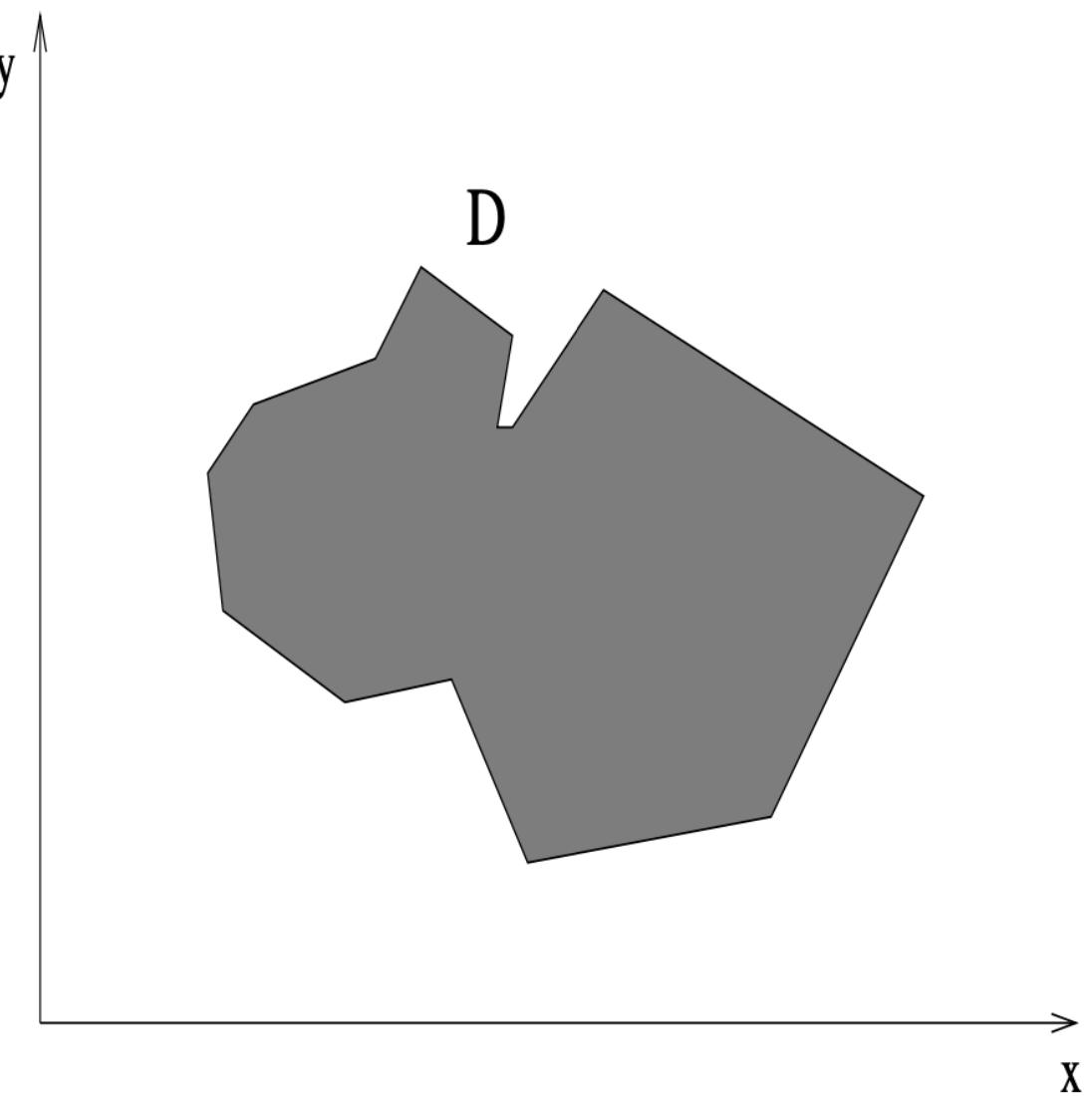
Intersection

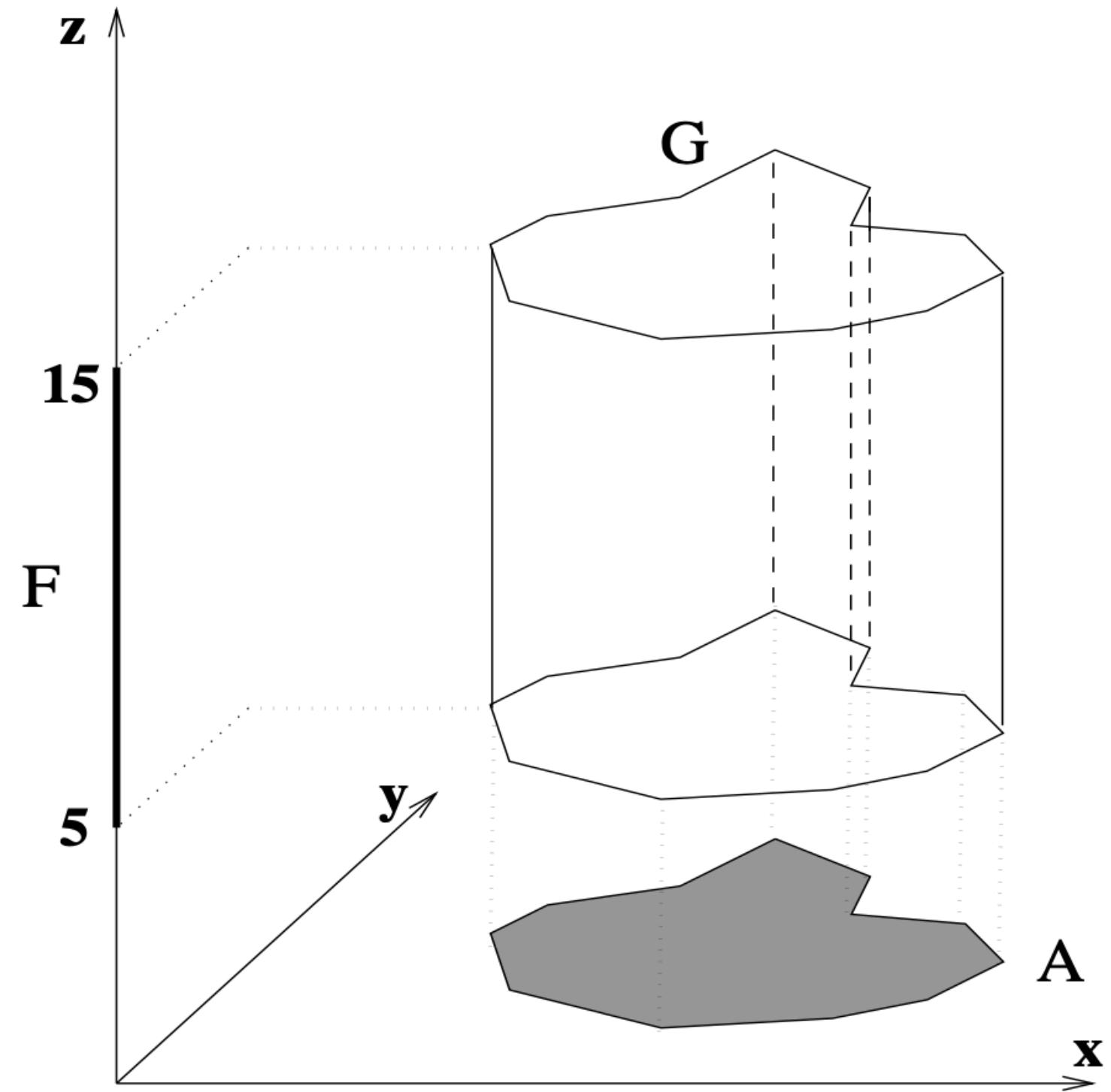


Difference

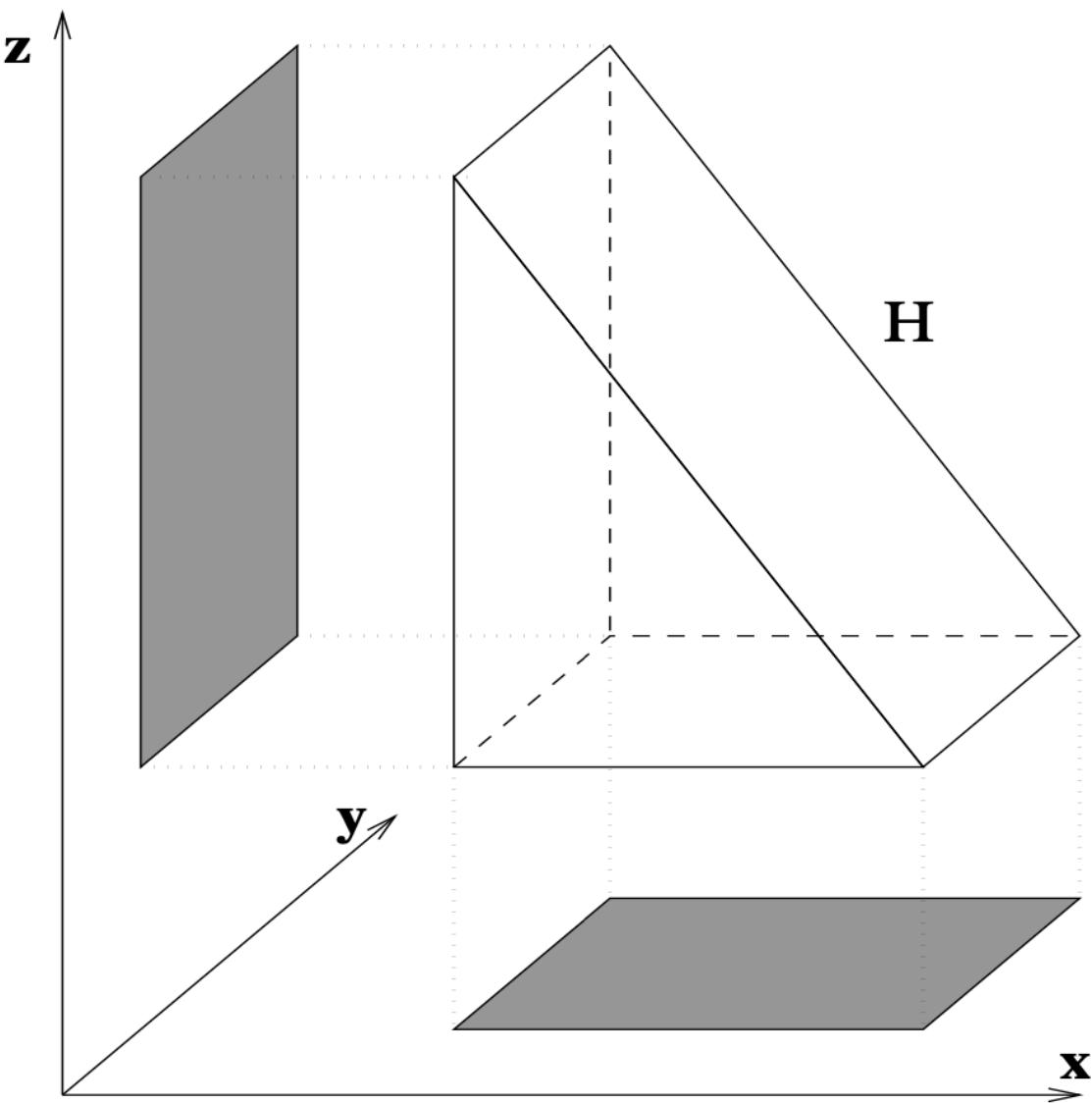


Union

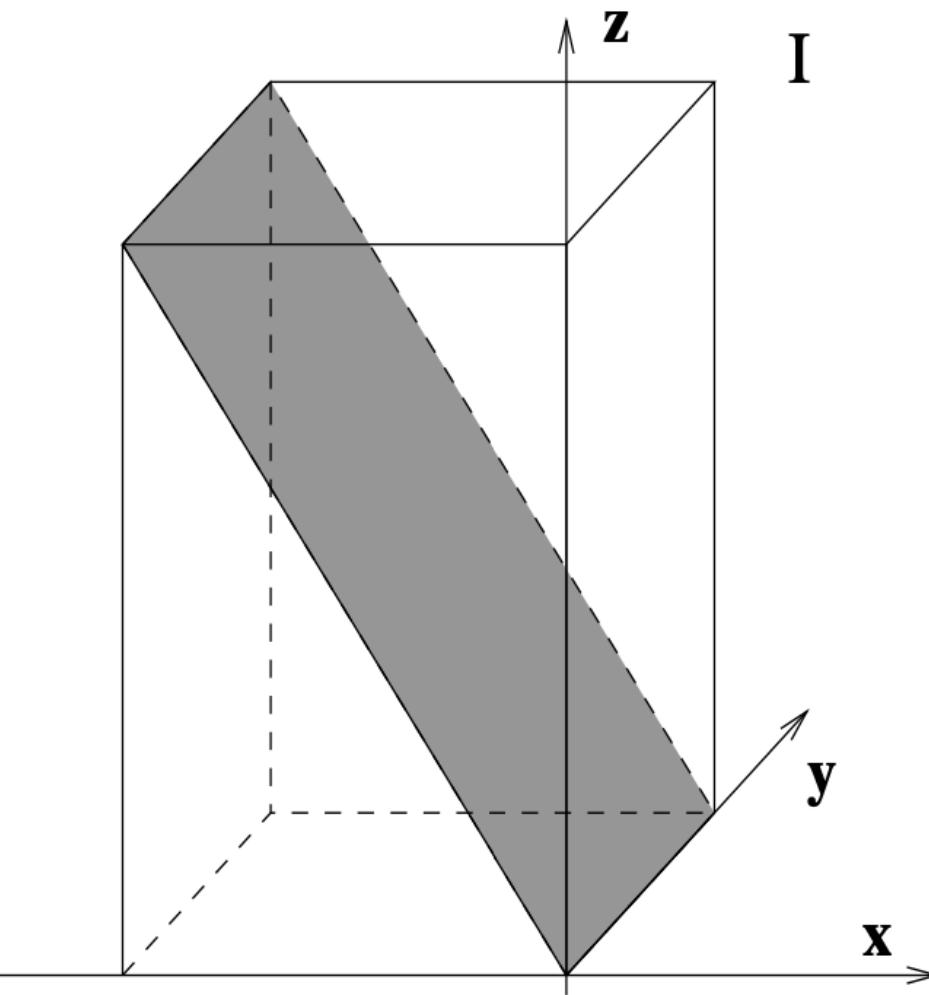




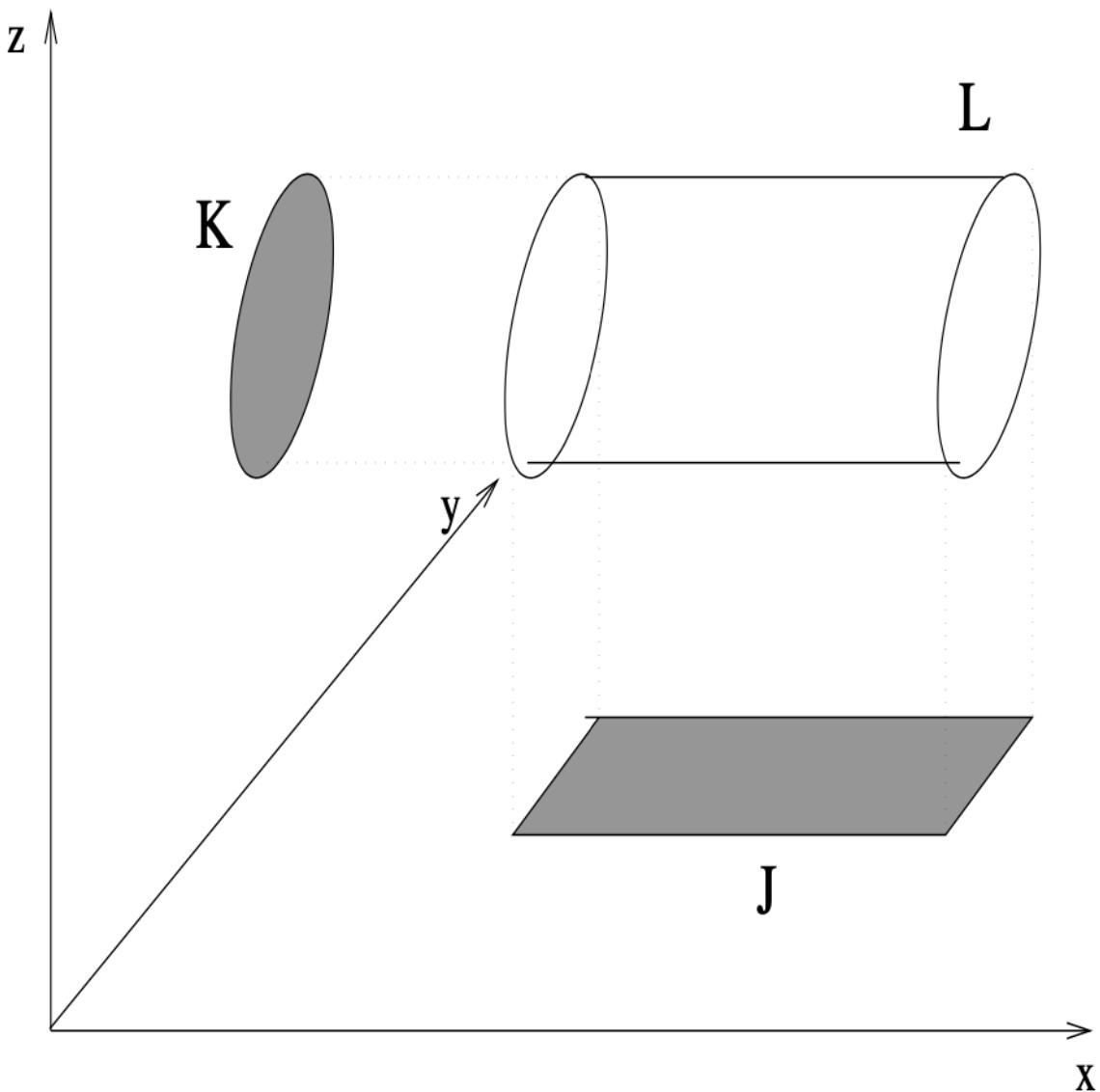
Projection



Selection



Natural JOIN



Entity-Relationship (ER) Model

- Outputs a conceptual schema.
- The ER data model employs three basic concepts:
 - entity sets
 - relationship sets and
 - attributes.
- It is also associated with diagrammatic representation [try out](#)

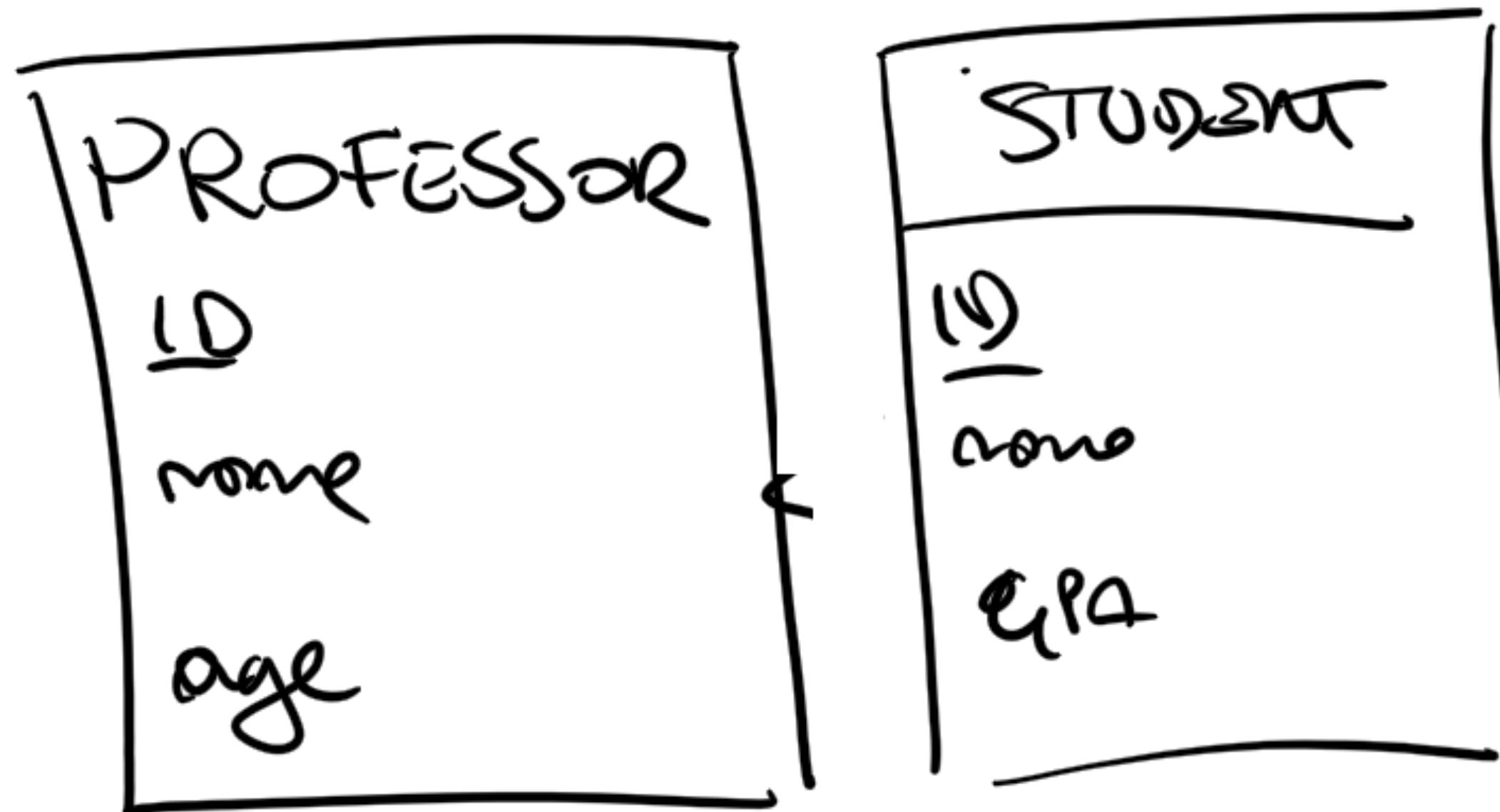
Entities And Entity Sets

An entity can be any object in the real world that is distinguishable from all other objects.

An **entity set** contains entities of the same type that share the same properties, or attributes.

NB We work at *set* level

Syntax

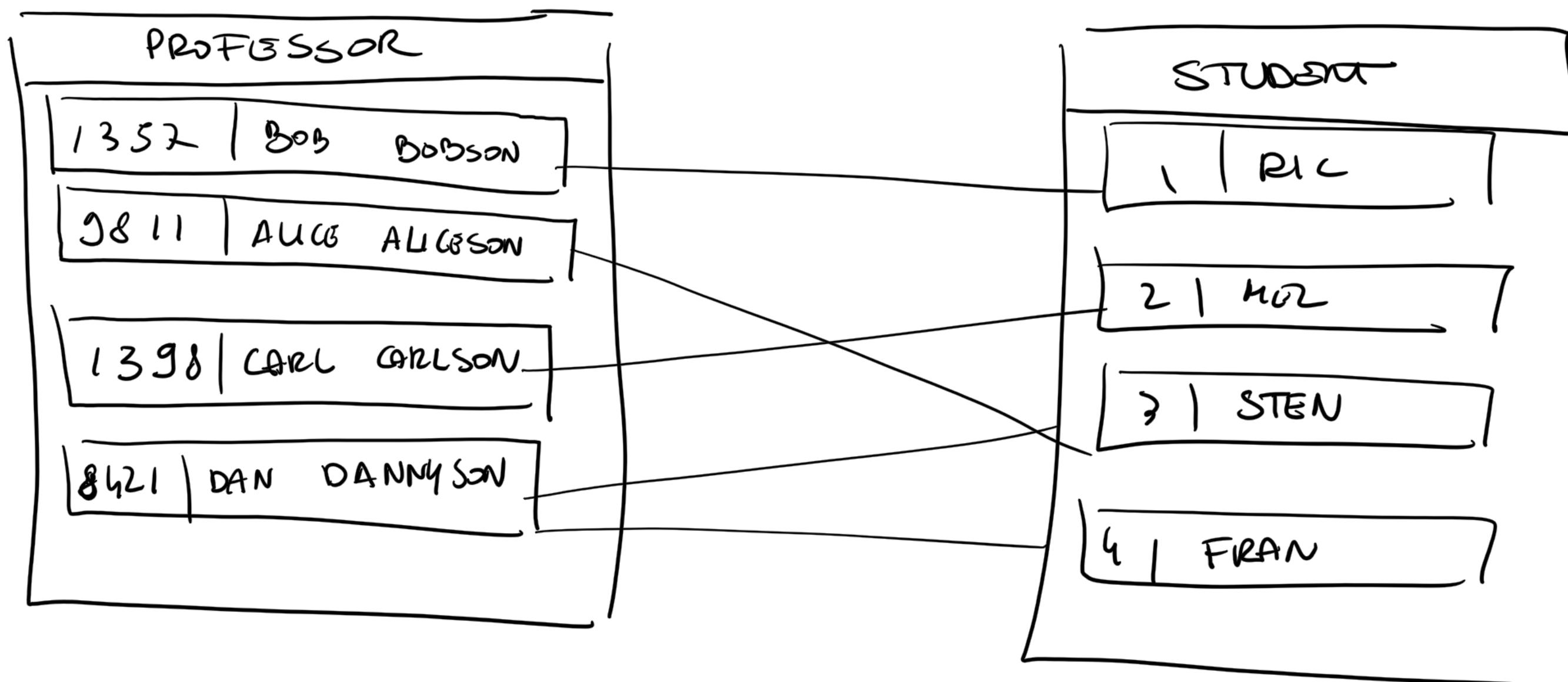


Relationships and Relationship Sets

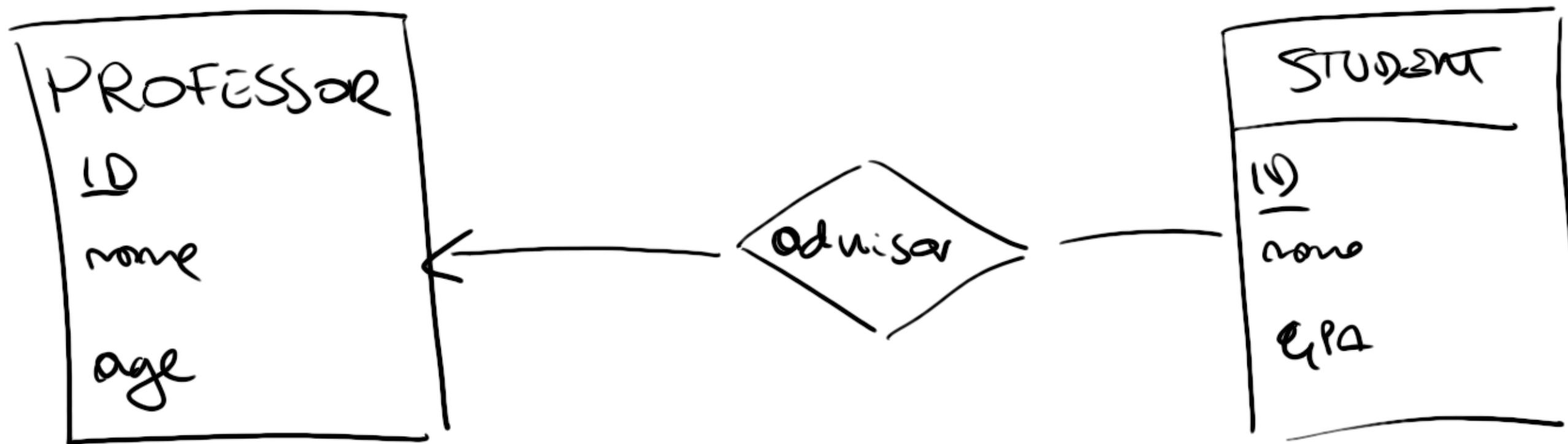
A **relationship** is an association among several entities.

A **relationship set** is a set of relationships of the same type.

Intuition



Syntax



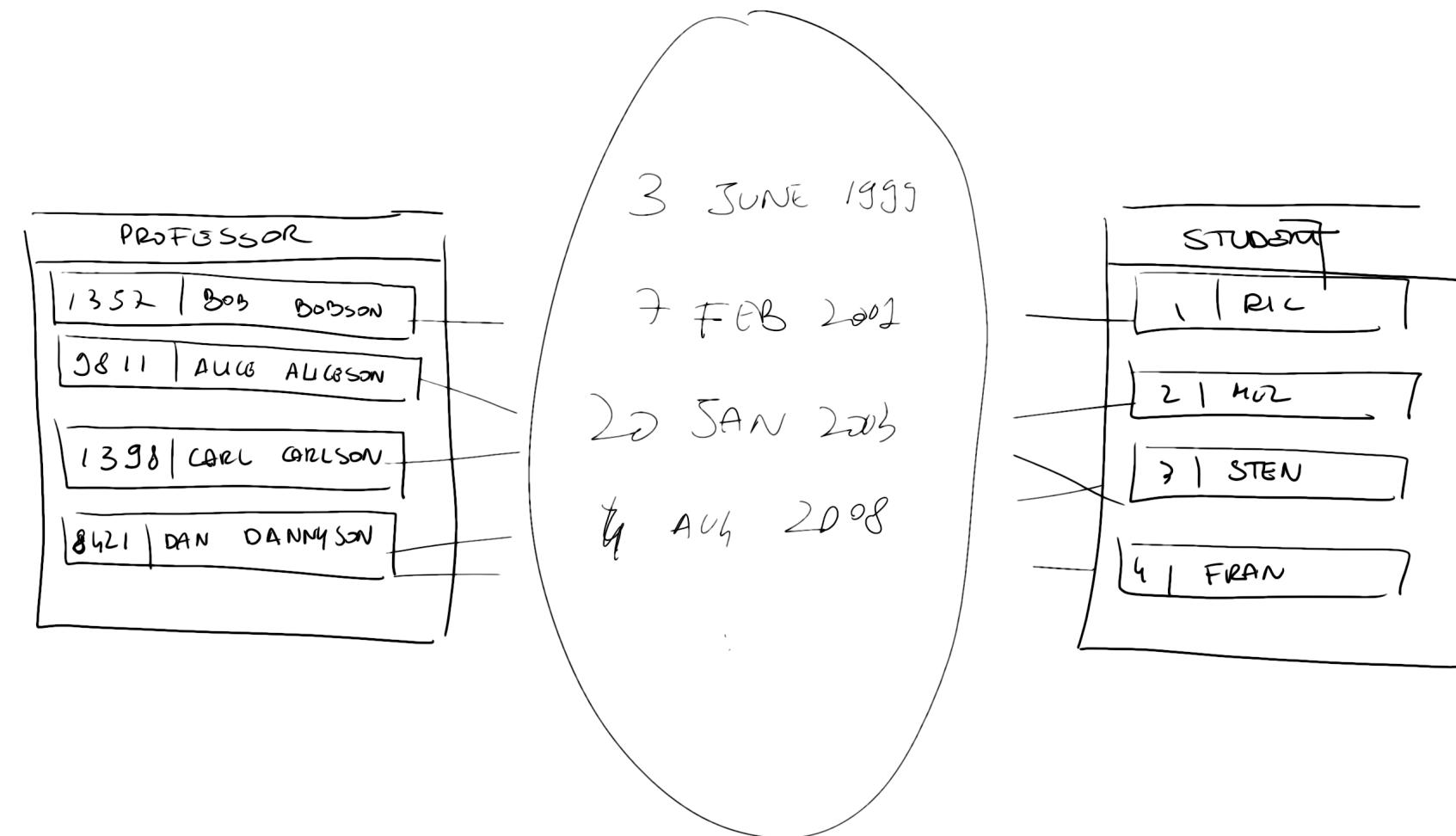
Attributes and Values

attributes. Attributes are descriptive properties possessed by each member of an entity set.

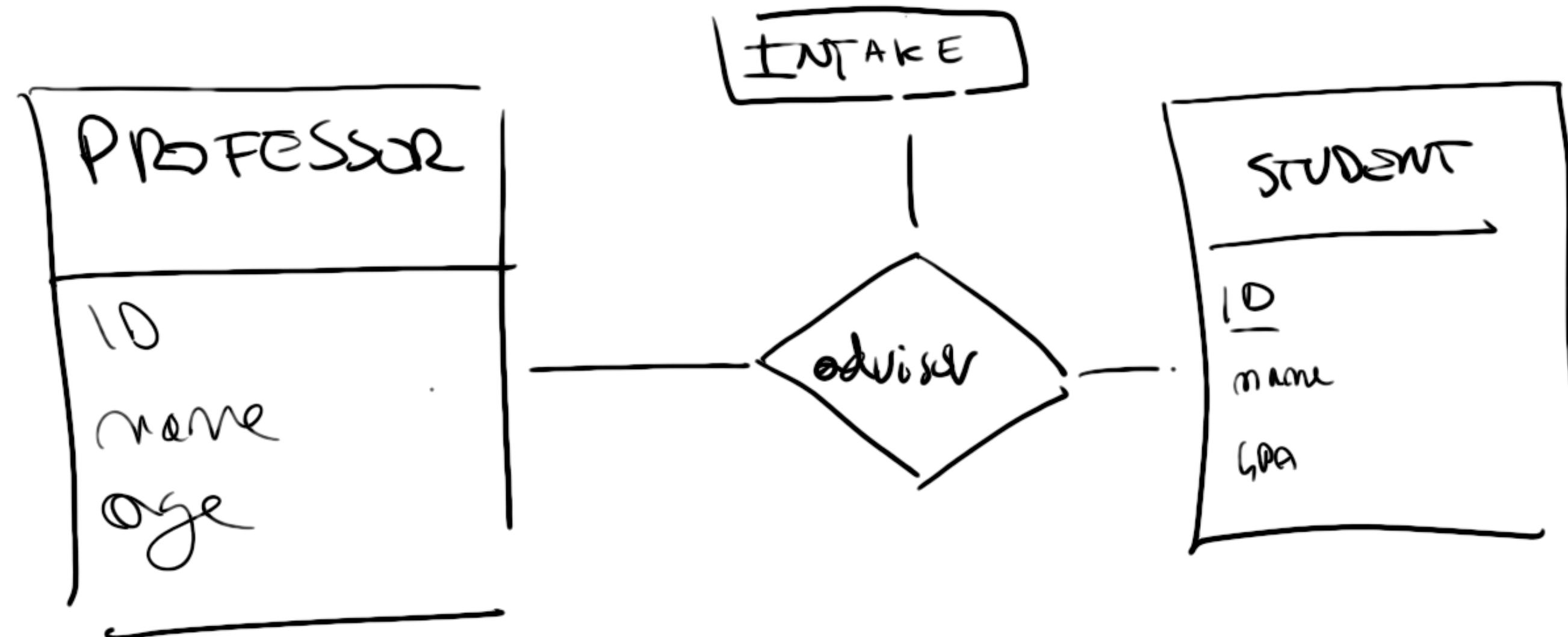
Each entity has a **value** for each of its attributes.

Also relationships may have attributes called **descriptive attributes**.

Intuition



Syntax



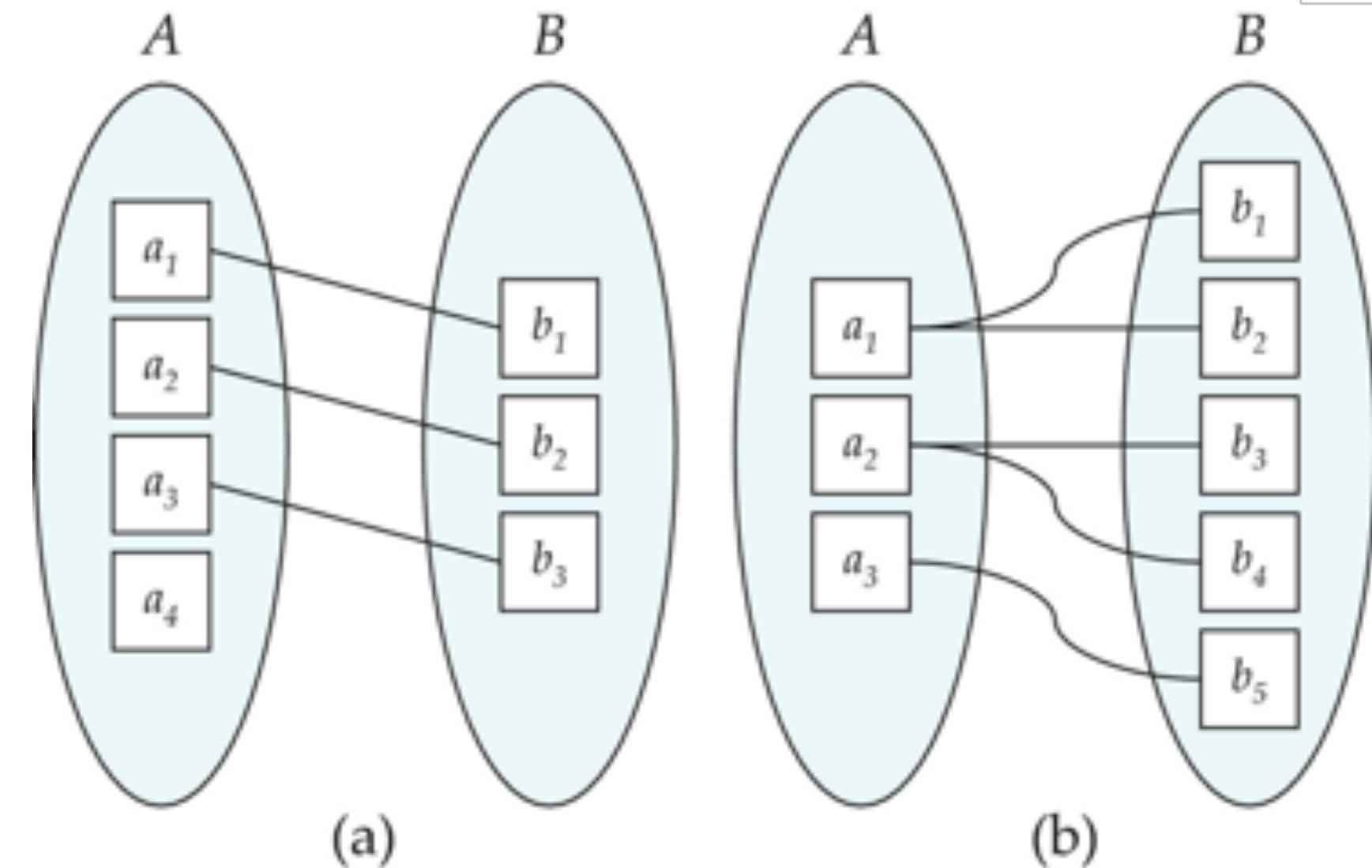
Cardinality

For a binary relationship set the mapping cardinality must be one of the following types:

- One to one
- One to many
- Many to one
- Many to many

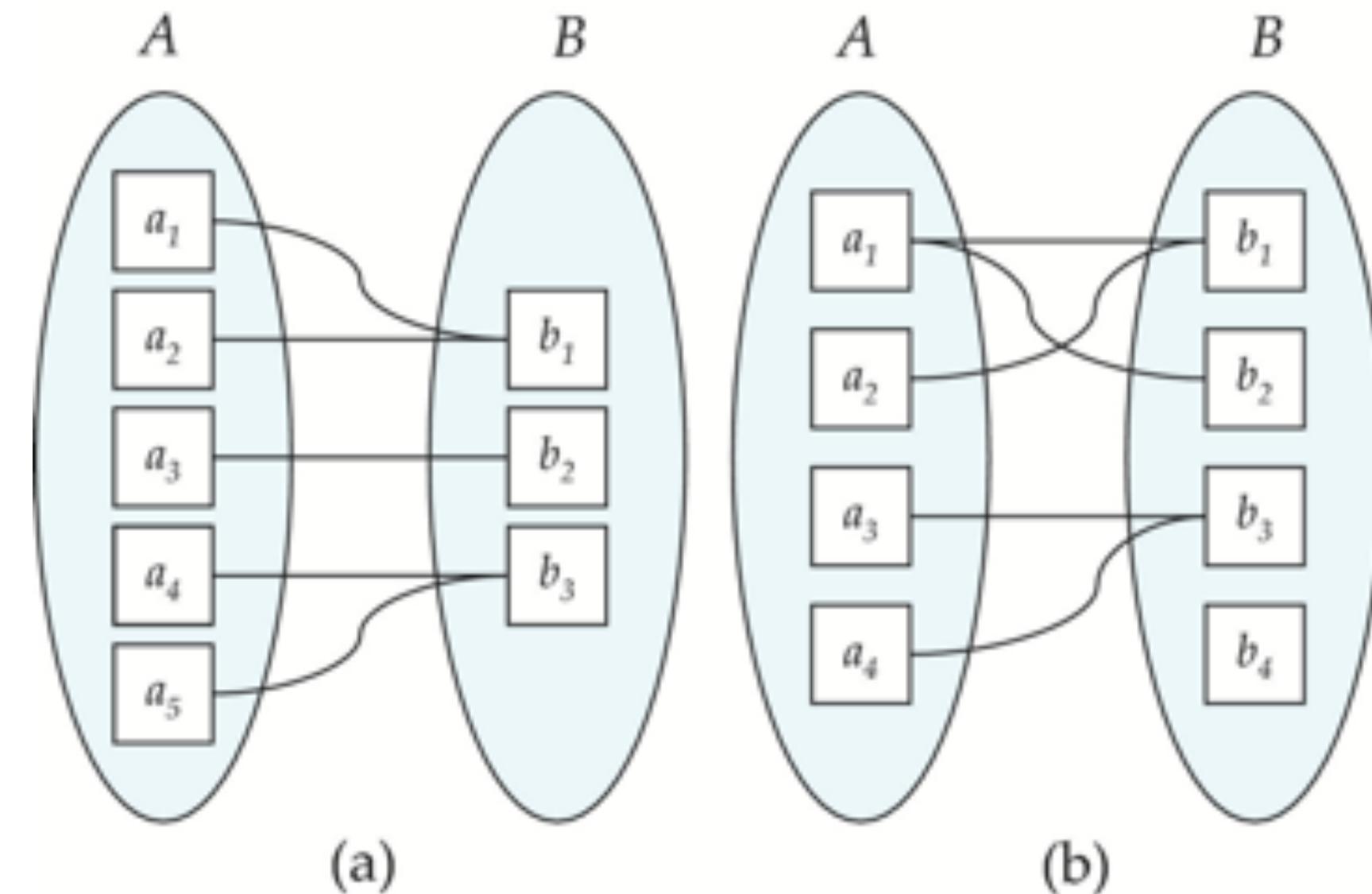
Cardinality Visualized

- (a) One to One
- (b) One to Many

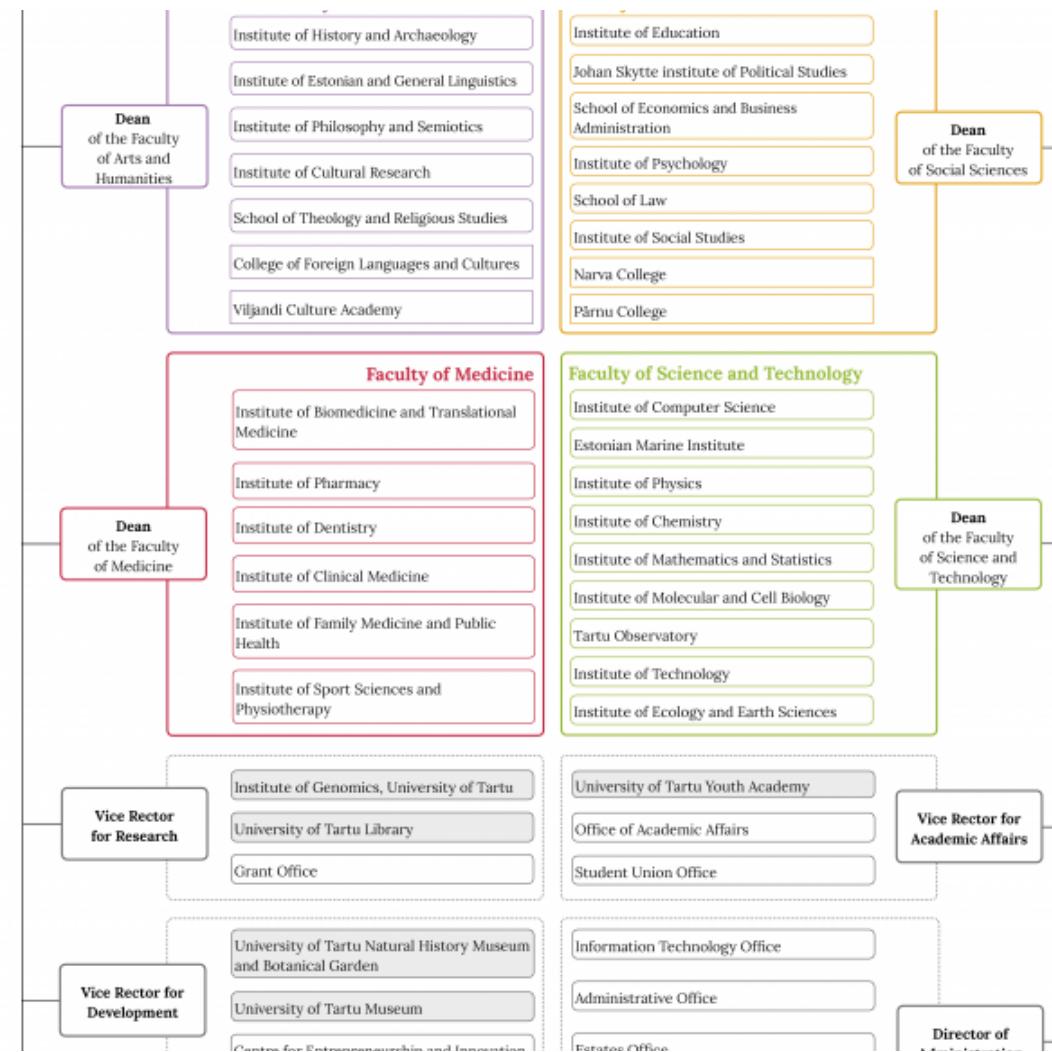


Cardinality Visualized

- (a) Many to One
- (b) Many to Many

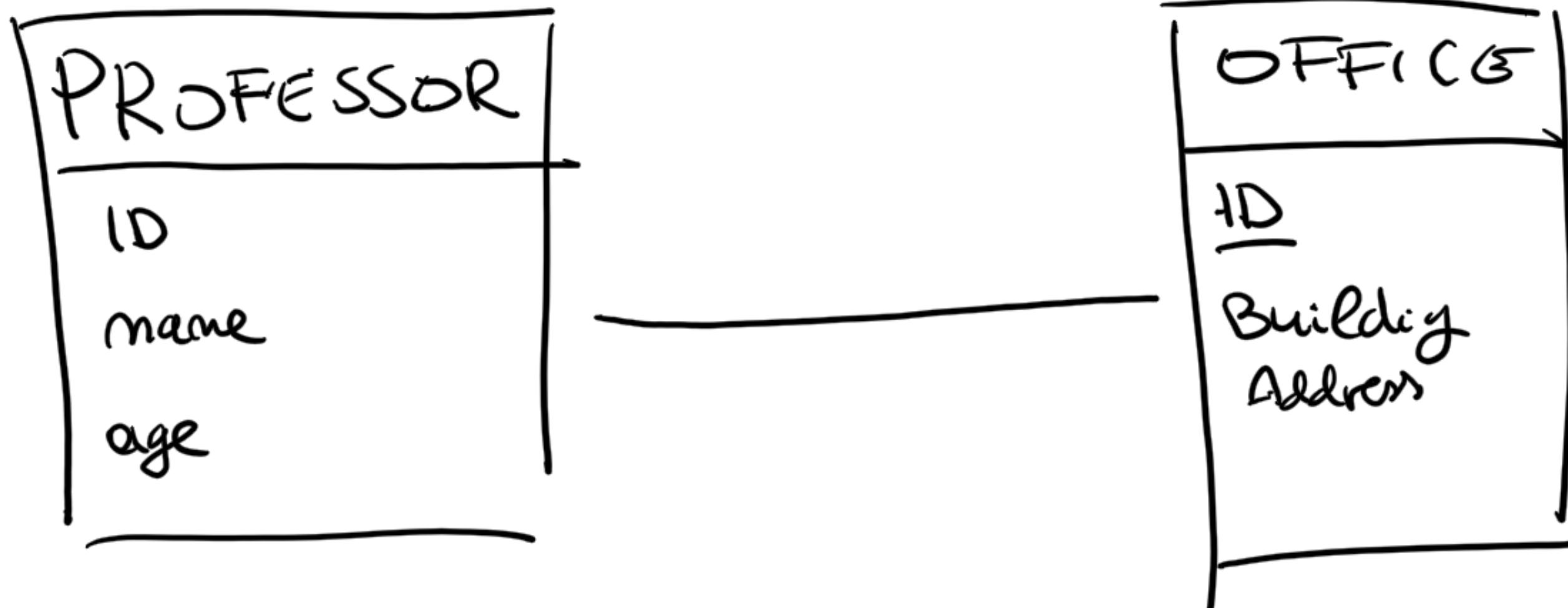


University of Tartu Example



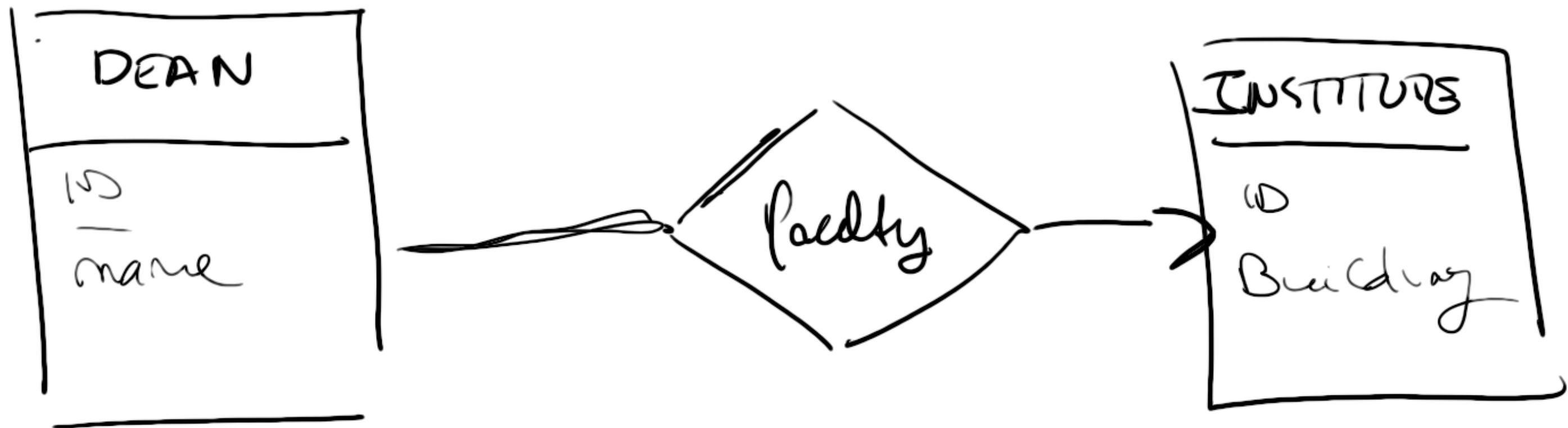
source

One to Many



A (full) professor has one office
an office hosts one full professor

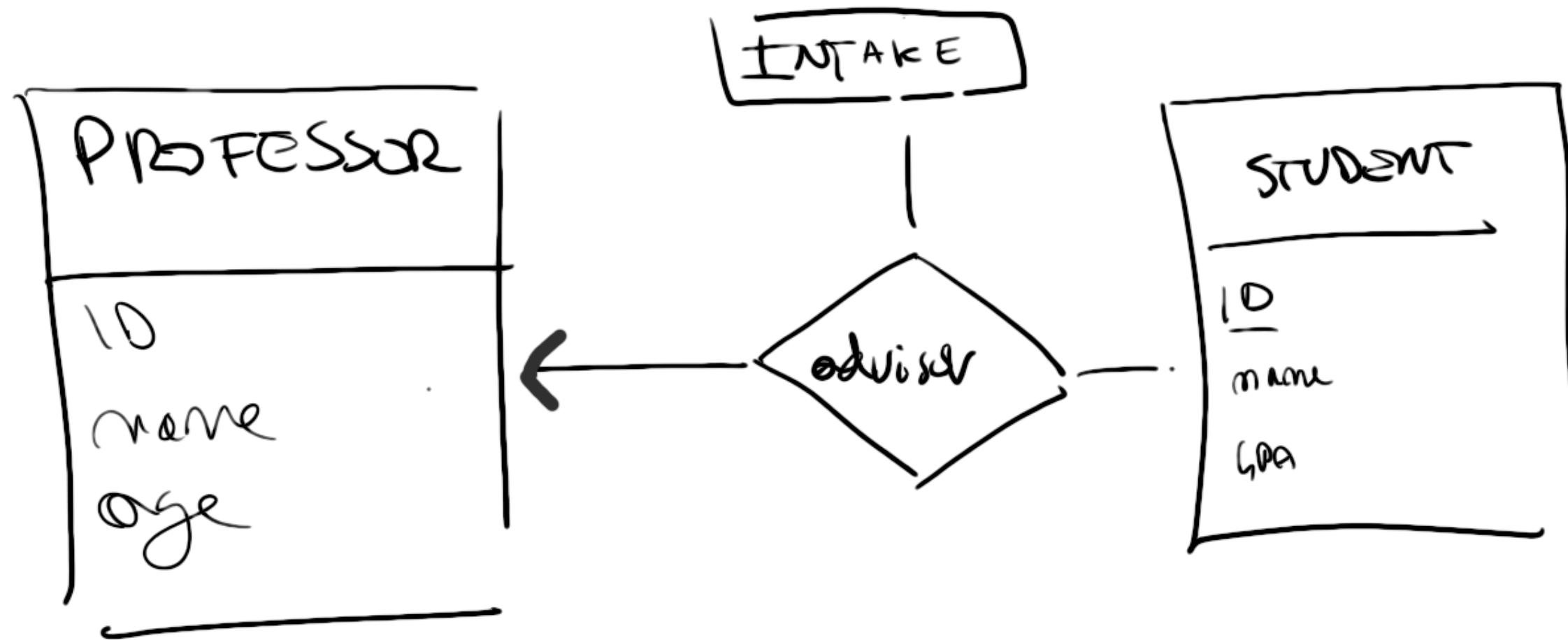
One to Many



A Dean is associated with many institutes

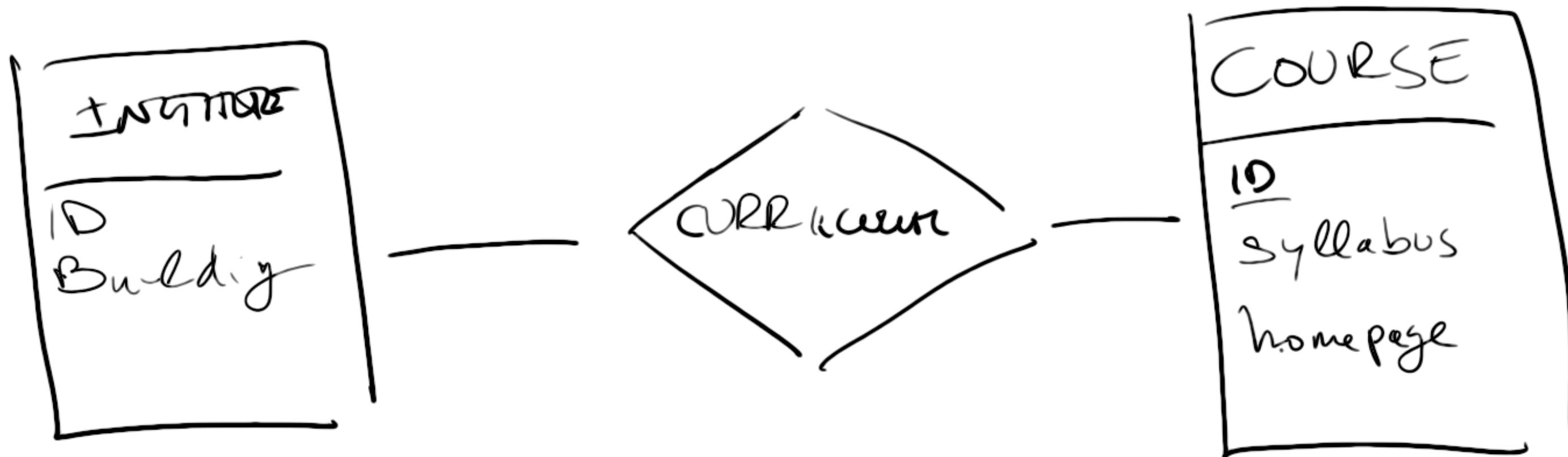
An Institute has only one dean

Many to One



A professor advises many students but a student has only one advisor.

Many to Many



A course is associated to many insitute in the context of a curriculum
An institute offers many courses within a curriculum

Keys

- Provide a way to specify how entities and relations are distinguished.
- *Primary key* for Entity Sets
 - By definition, individual entities are distinct (set)
 - From database perspective, the differences among them must be expressed in terms of their attributes
- *Primary Key* for Relationship Sets
 - We use the individual primary keys of the entities in the relationship set.
 - The choice depends on the mapping cardinality of the relationship set.

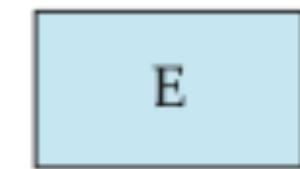
Choice of Primary key for Binary Relationship

- One-to-one relationships. The primary key of either one of the participating entity sets forms a minimal superkey, and either one can be chosen as the primary key.
- One-to-Many relationships and Many-to-one relationships
 - The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- Many-to-Many relationships:
 - The preceding union of the primary keys is a minimal superkey and is chosen as the primary key.

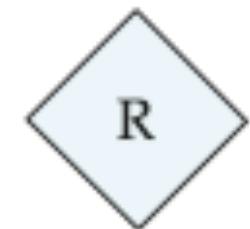
Weak Entity Sets

- A weak entity set is one whose existence is dependent on another entity,
called its **identifying entity**
- A weak entity set is one whose existence is dependent on another entity,
called its identifying entity

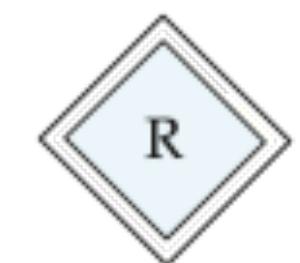
Summary of Symbols



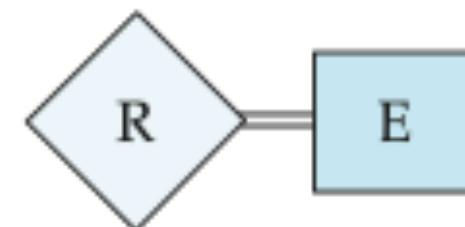
entity set



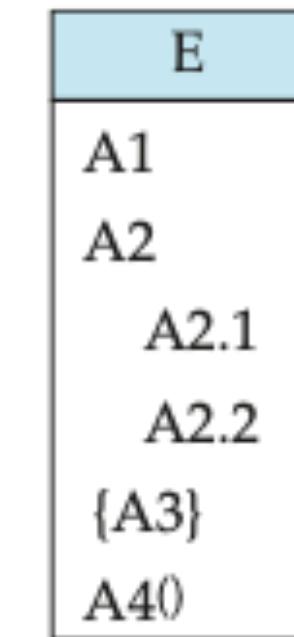
relationship set



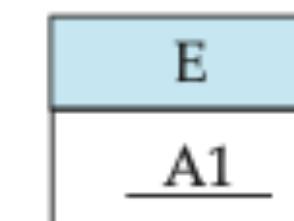
identifying
relationship set
for weak entity set



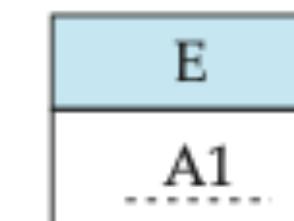
total participation
of entity set in
relationship



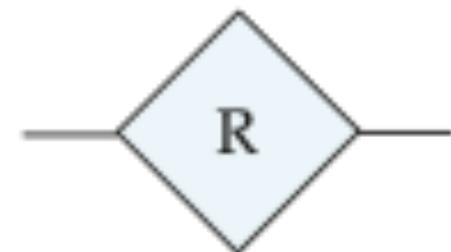
attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)



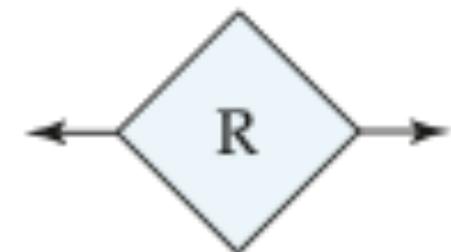
primary key



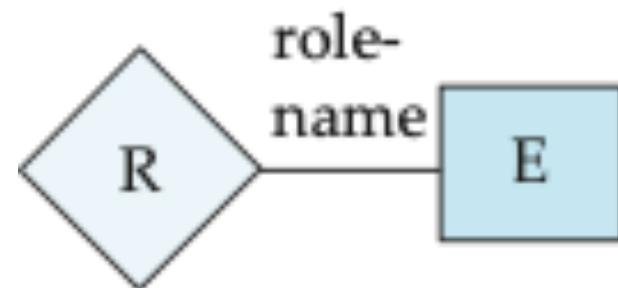
discriminating
attribute of
weak entity set



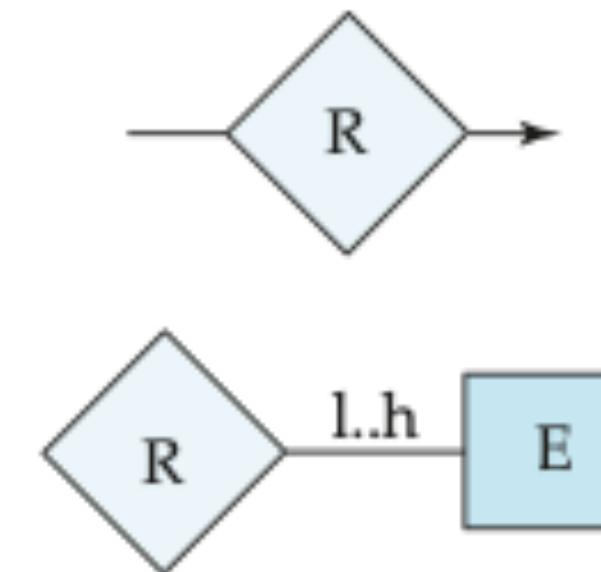
many-to-many
relationship



one-to-one
relationship

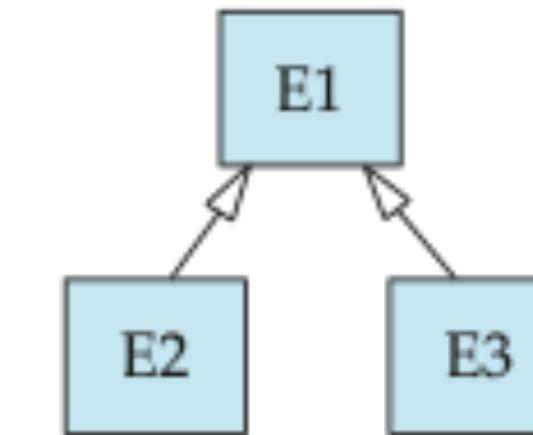


role indicator

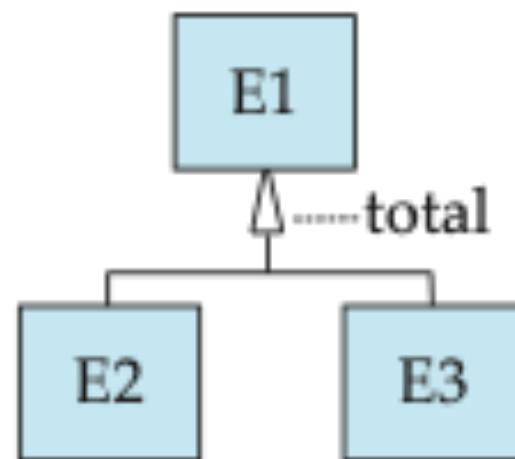


many-to-one
relationship

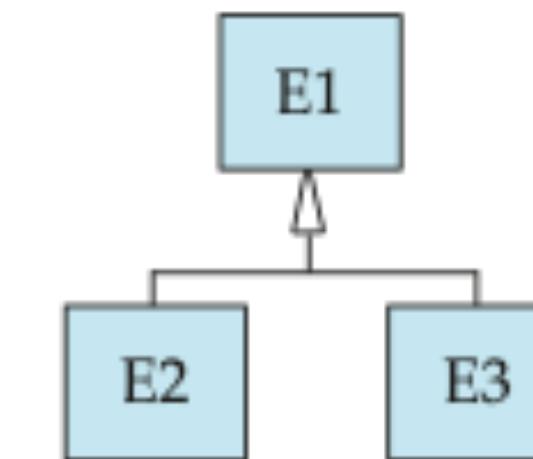
cardinality
limits



ISA: generalization
or specialization



total (disjoint)
generalization



disjoint
generalization

From ER to Relational Model

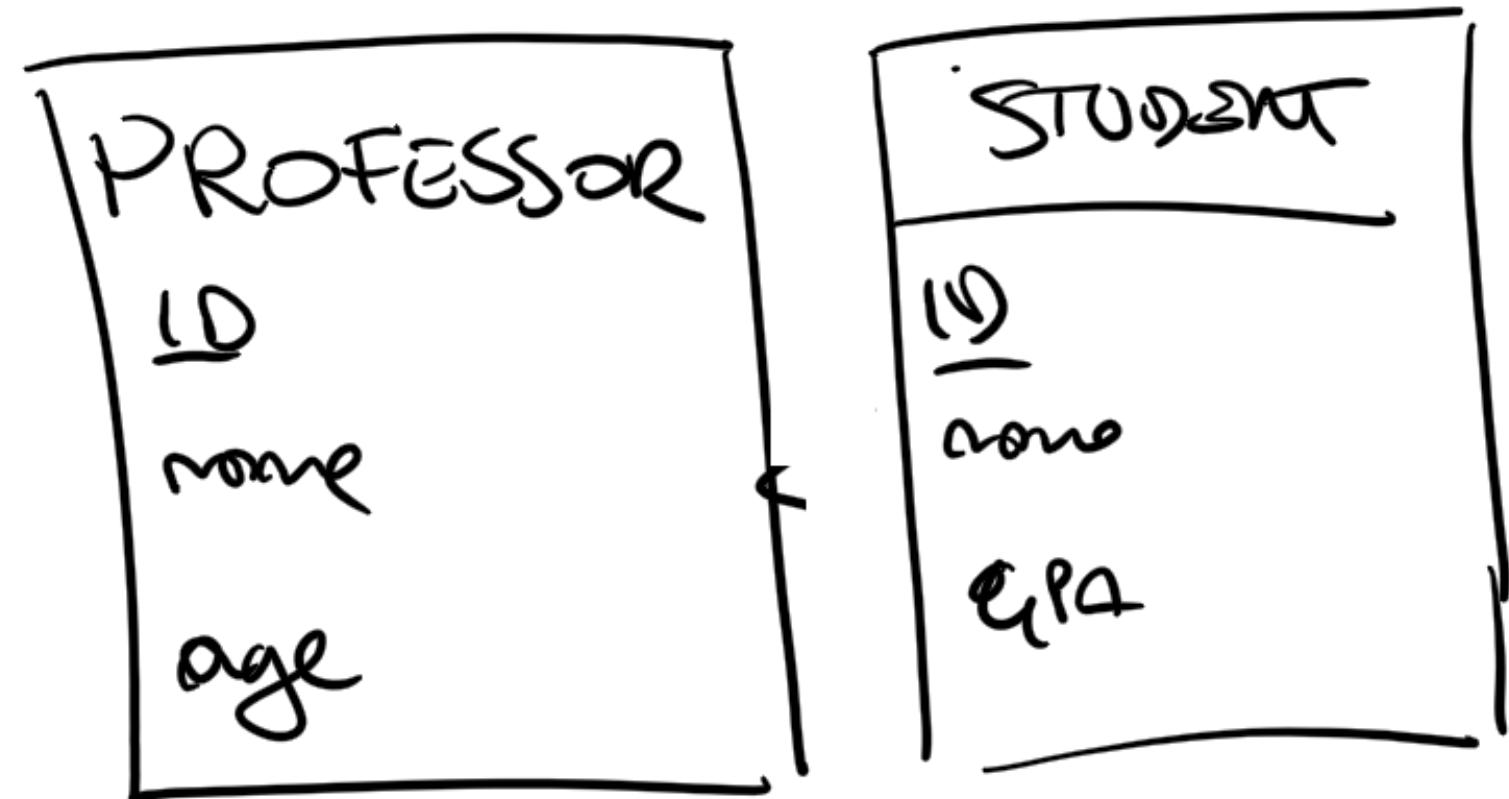
- Entity and relationship sets can be expressed as relation schemas that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.

Reduction of Entities

- For each **entity** set there is a unique schema with the same name
- Each schema has a number of columns (generally corresponding to attributes), which have unique names

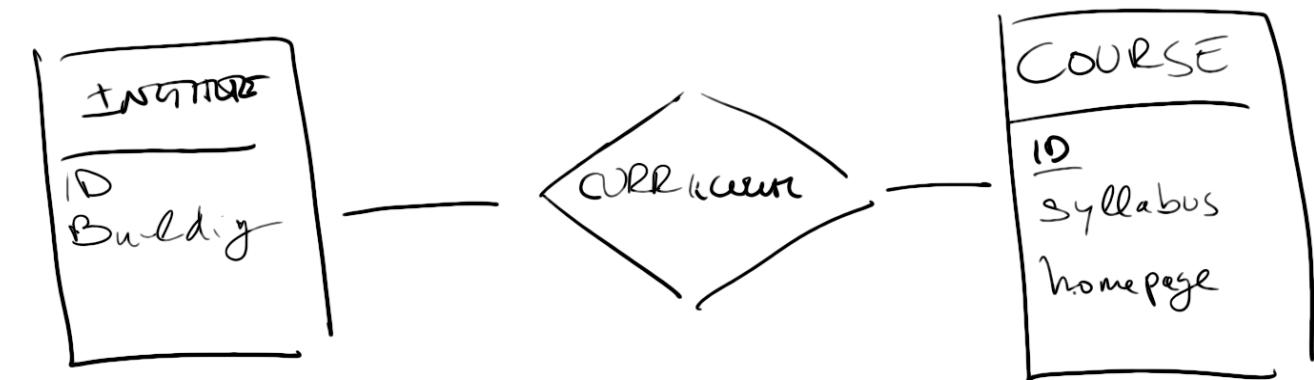
Professor(<u>ID</u>,Name,Age)

Student(<u>ID</u>,Name,GPA)



Reduction of Relationships

- For each **relationship** set there is a unique schema with the same name
- A **many-to-many** relationship (figure) is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.



Curriculum(<u>Institute_ID</u>,<u>Course_ID</u>)

Reduction of Relationships

- **Many-to-one** and one-to-many** relationship can be represented by adding an extra attribute to the "many" side
- For **one-to-one** relationship, either side can be chosen to act as the "many" side

Normalisation

- Typically decomposes tables to avoid redundancy
- Spans both logical and physical database design
- Aims at **improving** the database design

Goals

- Make the schema informative
- Minimize information duplication
- Avoid modification anomalies
- Disallow spurious tuples

<i>ID</i>	<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
:				

<i>ID</i>	<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:				
57766	<i>Kim</i>	<i>Main</i>	<i>Perryridge</i>	75000
98776	<i>Kim</i>	<i>North</i>	<i>Hampton</i>	67000
:				

employee

<i>ID</i>	<i>name</i>
:	
57766	<i>Kim</i>
98776	<i>Kim</i>
:	

<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:			
<i>Kim</i>	<i>Main</i>	<i>Perryridge</i>	75000
<i>Kim</i>	<i>North</i>	<i>Hampton</i>	67000
:			

<i>ID</i>	<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
:				

employee

<i>ID</i>	<i>name</i>
:	
57766	Kim
98776	Kim
:	

<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
:			

natural join

<i>ID</i>	<i>name</i>	<i>street</i>	<i>city</i>	<i>salary</i>
:				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
:				

Normal Forms (Refresh)

- First Normal Form (1NF)
 - A table has only atomic valued columns.
 - Values stored in a column should be of the same domain
 - All the columns in a table should have unique names.
 - And the order in which data is stored, does not matter.
- Second Normal Form (2NF)
 - A table is in the First Normal form and every non-prime attribute is fully functional dependent²⁰ on the primary key
- Third Normal Form (3NF)
 - A table is in the Second Normal form and every non-prime attribute is non-transitively dependent[^21] on every key

²⁰ $X \rightarrow Y, \forall A \in X((X - A) \not\rightarrow Y)$

[^21]:

Modeling for Database: A note on Storage

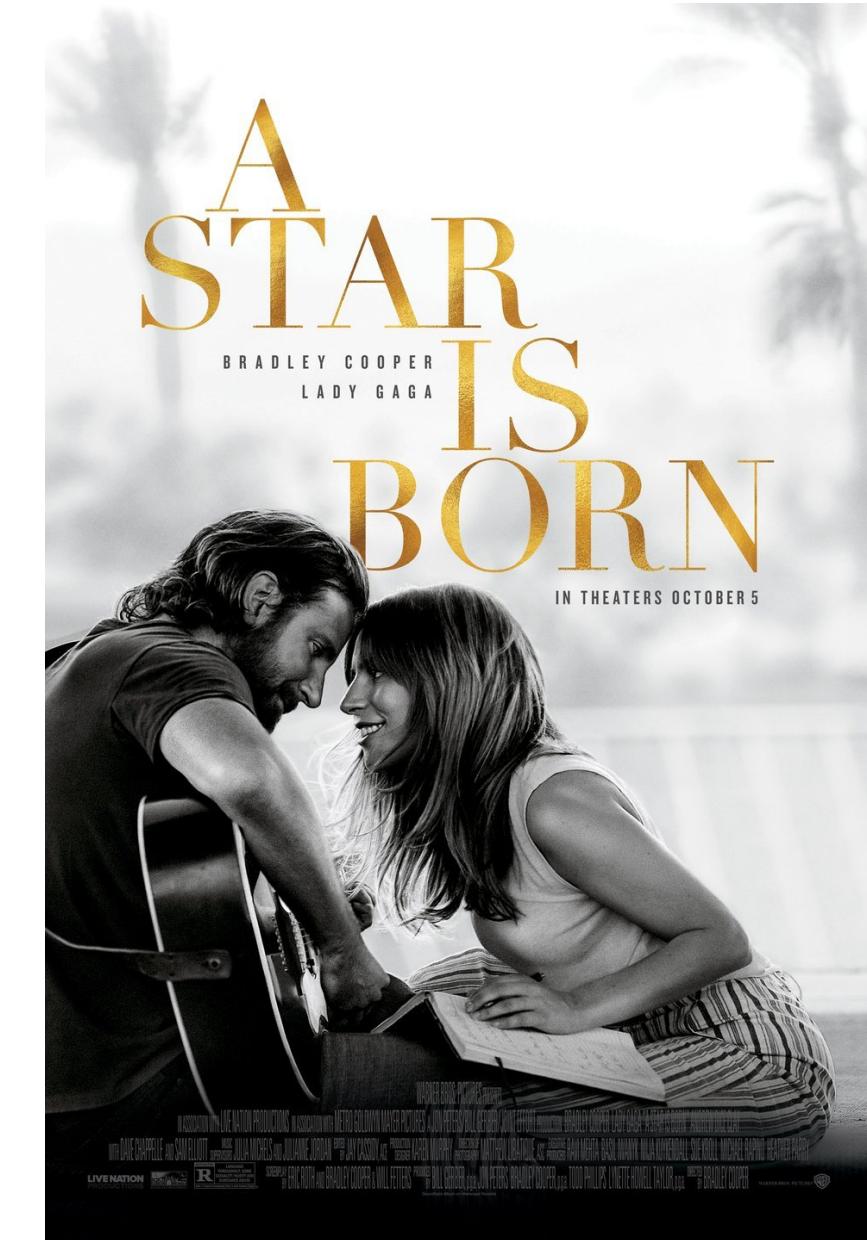
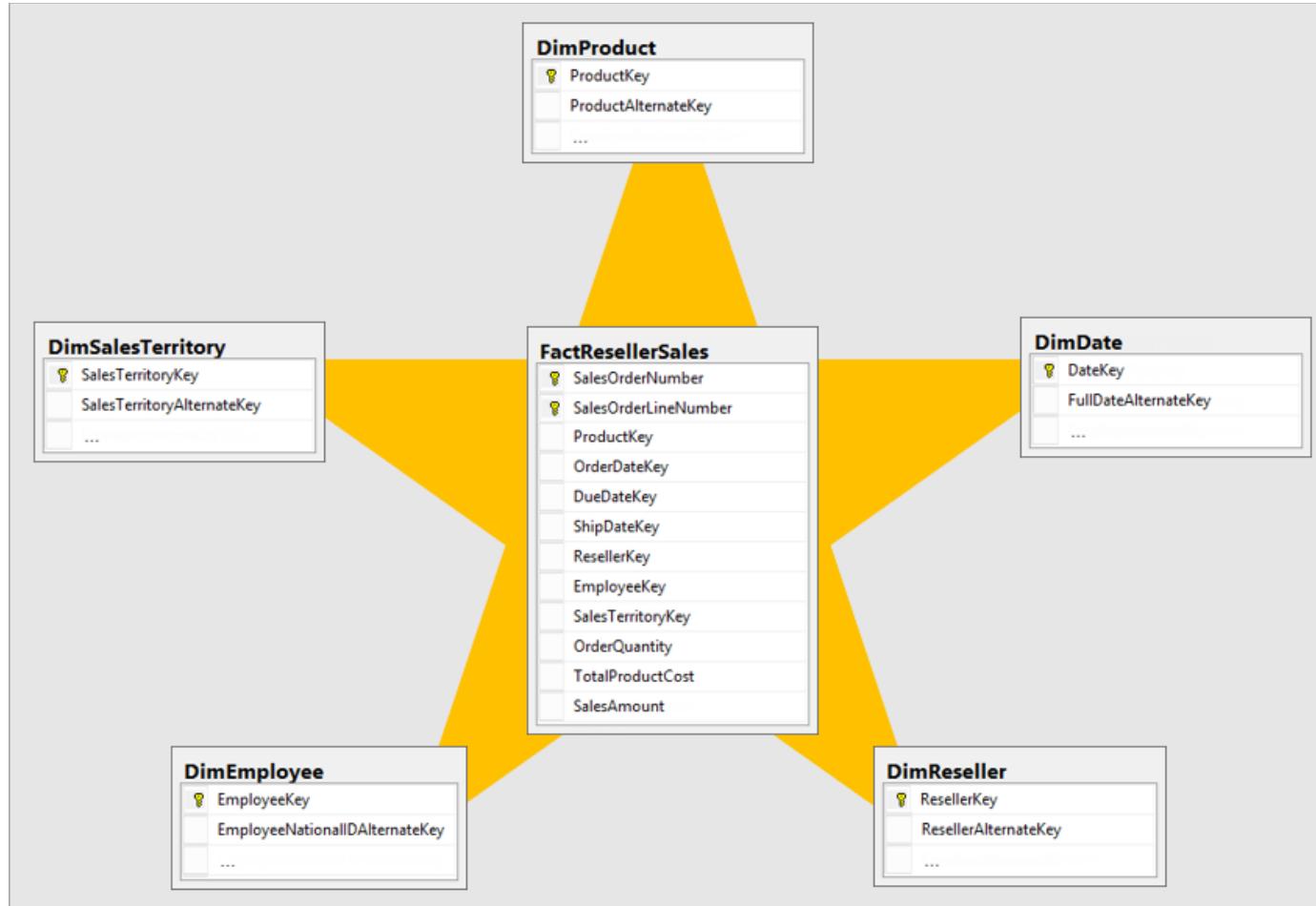
- Storage is laid out in a row-oriented fashion
- For relational this is as close as the the tabular representation
- All the values from one row of a table are stored next to each other.
- This is true also for some NoSQL (we will see it again)
 - Document databases stores documents a contiguous bit sequence

Data Modeling for Data Warehouse

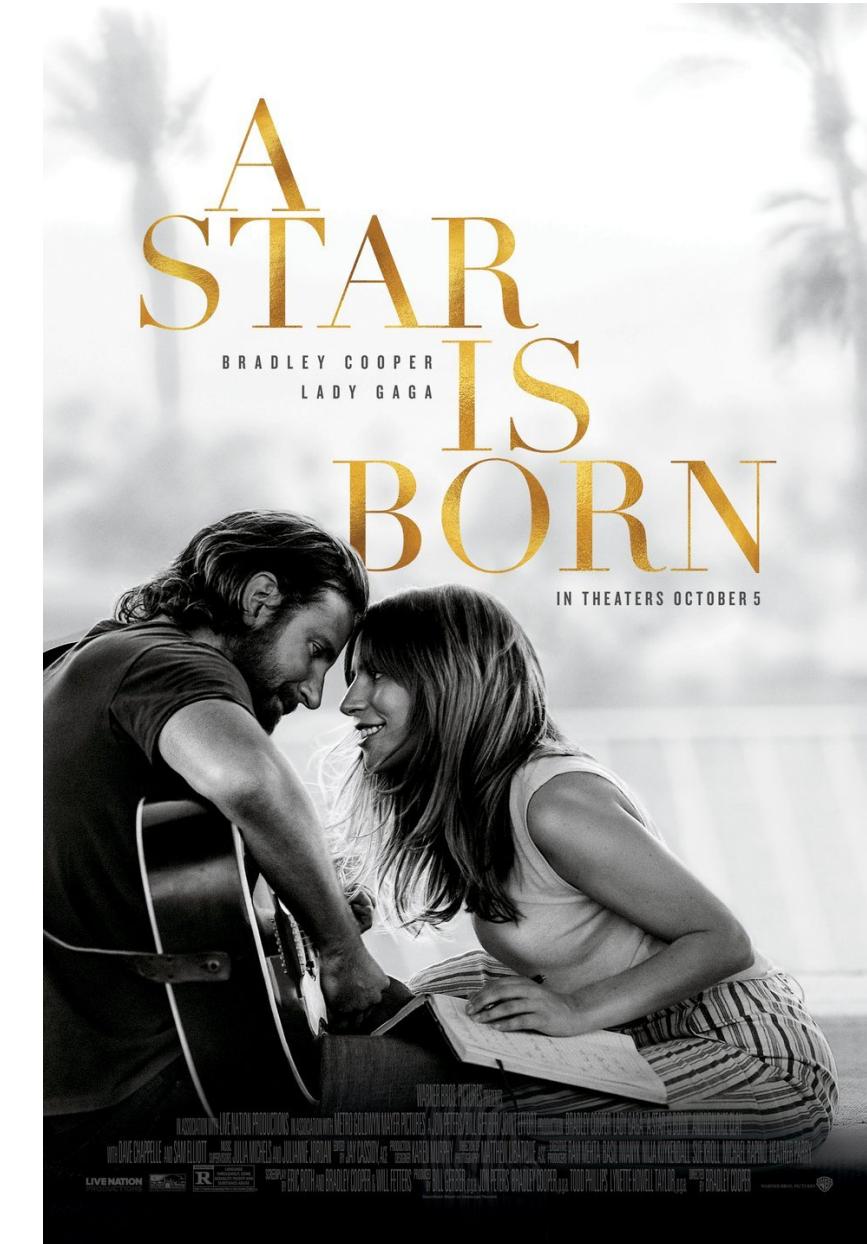
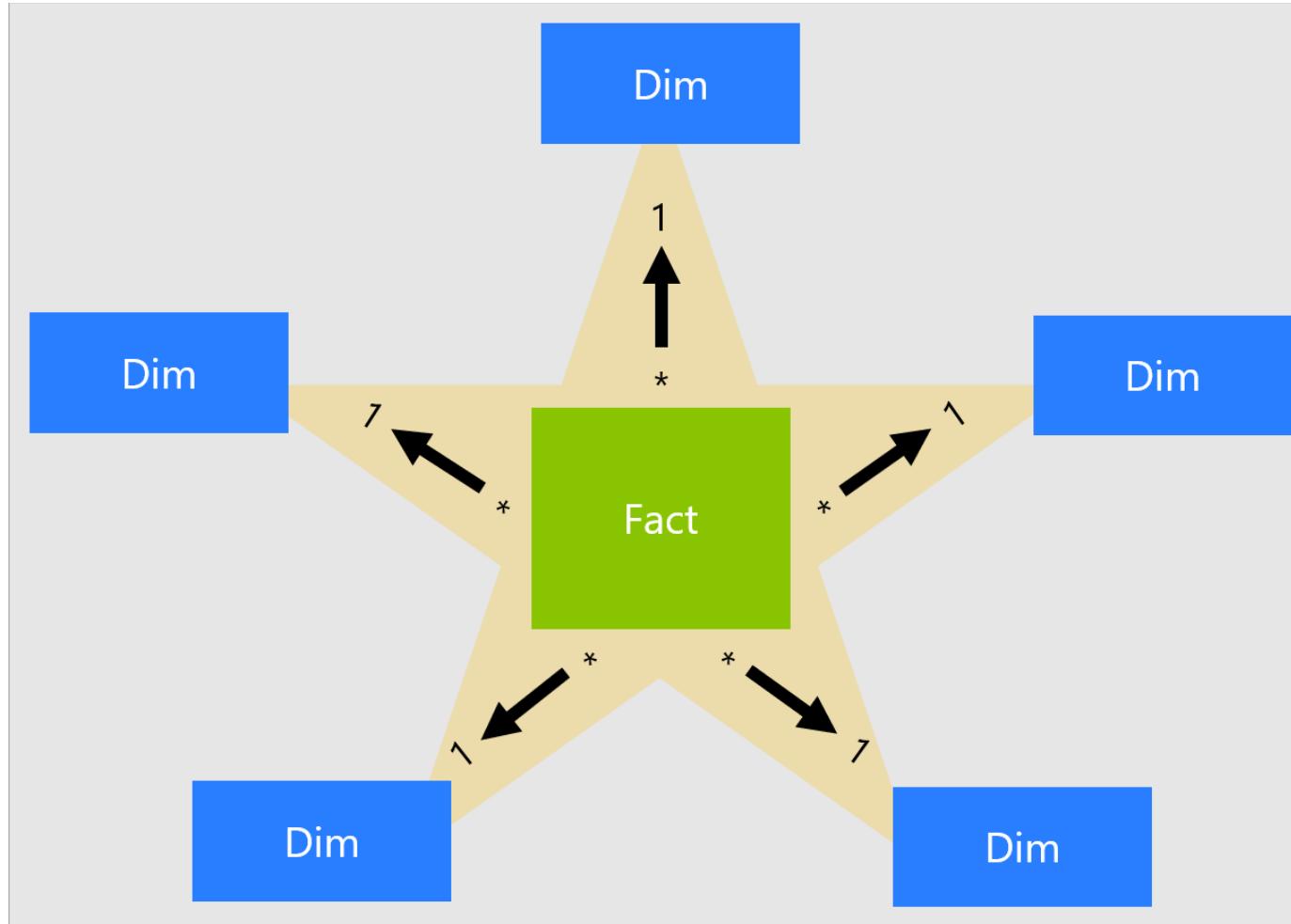
- Works in phases related to the aforementioned levels of abstractions
- Less diversity in the data model, usually relational in the form of a star schema (also known as dimensional modeling⁷).
- Redundancy and incompleteness are not avoided, fact tables often have over 100 columns, sometimes several hundreds.
- Optimized for OLAP

⁷ Ralph Kimball and Margy Ross: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd edition. John Wiley & Sons, July 2013. ISBN: 978-1-118-53080-1

A Star is Born



A Star is Born



Dimensional Modelling

Four-Step Dimensional Design Process

1. Select the business process.
2. Declare the grain.
3. Identify the dimensions.
4. Identify the facts.

Mandatory Read

The Data Warehouse Toolkit

Second Edition

The Complete
Guide to
Dimensional
Modeling

Ralph Kimball
Margy Ross



Dimensional Modelling: Fact Table

A **fact table** contains the numeric measures produced by an operational measurement event in the real world.

A **single fact** table row has a one-to-one relationship to a measurement event as described by the fact table's grain.

A **surrogate key** is a unique identifier that you add to a table to support star schema modeling. By definition, it's not defined or stored in the source data

Dimensional Modelling: Dimension Table

Dimension tables contain the descriptive attributes used by BI applications for filtering and grouping the facts.

Every dimension table has a single **primary key** column , which is embedded as a foreign key in any associated fact table.

The 10 Essential Rules of Dimensional Modeling (Read)²³

1. Load detailed atomic data into dimensional structures.
2. Structure dimensional models around business processes.
3. Ensure that every fact table has an associated date dimension table.
4. Ensure that all facts in a single fact table are at the same grain or level of detail.
5. Resolve many-to-many relationships in fact tables.
6. Resolve many-to-one relationships in dimension tables.
7. Store report labels and filter domain values in dimension tables.
8. Make certain that dimension tables use a surrogate key.
9. Create conformed dimensions to integrate data across the enterprise.
10. Continuously balance requirements and realities to deliver a DW/BI solution that's accepted by business users and that supports their decision-making.

²³ <https://www.kimballgroup.com/2009/05/the-10-essential-rules-of-dimensional-modeling/>

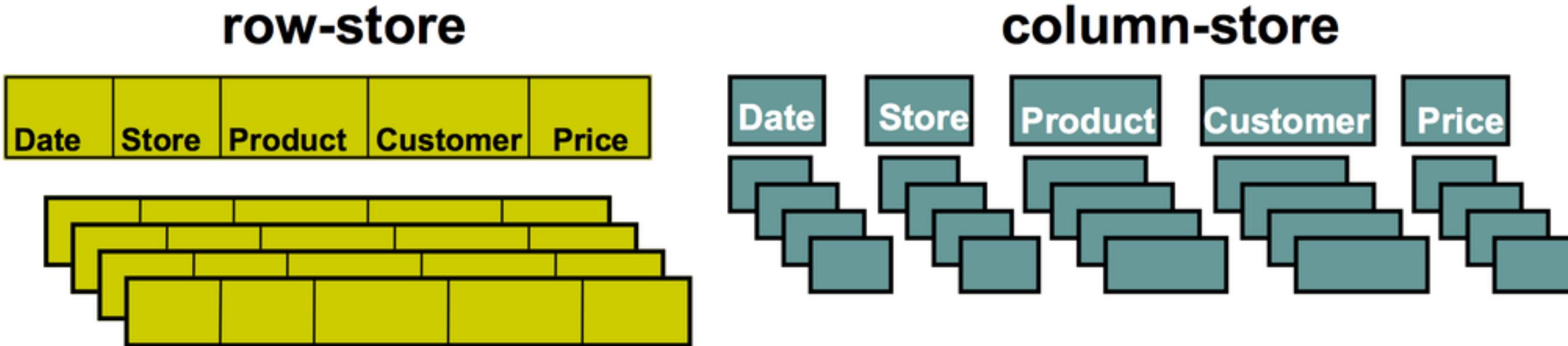
The Traditional RDBMS Wisdom Is (Almost Certainly) All Wrong²²



²² Source with slides: [The Traditional RDBMS Wisdom Is \(Almost Certainly\) All Wrong,](#) presentation at EPFL, May 2013

A note on Storage

- Data warehouse typically interact with OLTP database to expose one or more OLAP system.
- Such OLAP system adopt storage optimized for analytics, i.e., Column Oriented
- The column-oriented storage layout relies on each column file containing the rows in the same order.
- Not just relational data, e.g., Apache Parquet



Data Modeling for Big Data

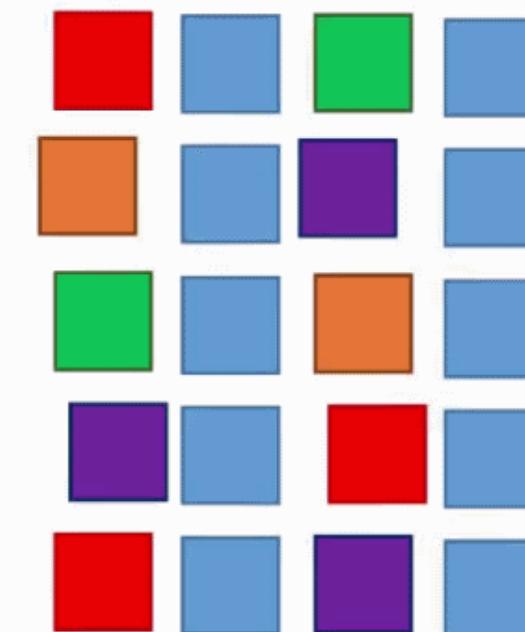
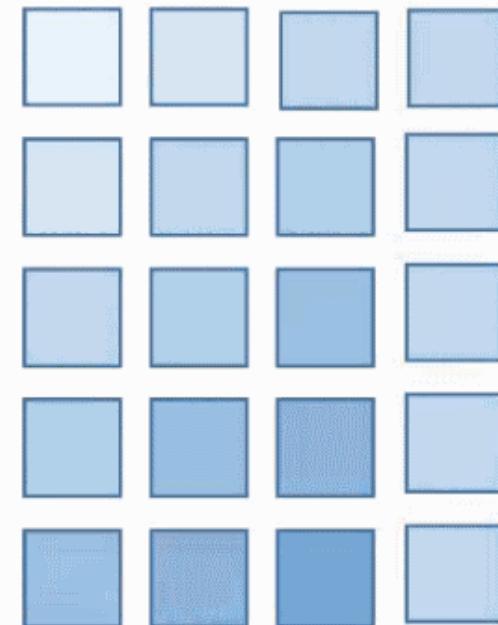


The Data Landscape

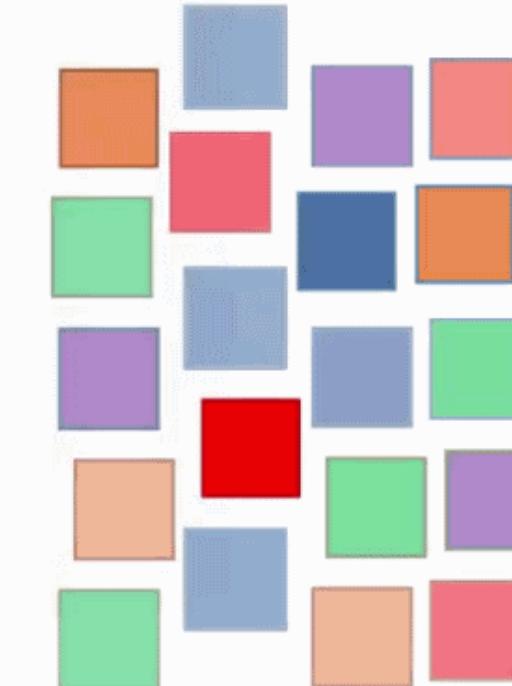
Structured, Unstructured and Semi-Structured

Semi-Structured Data

Structured Data



Unstructured Data



Big Data Storage

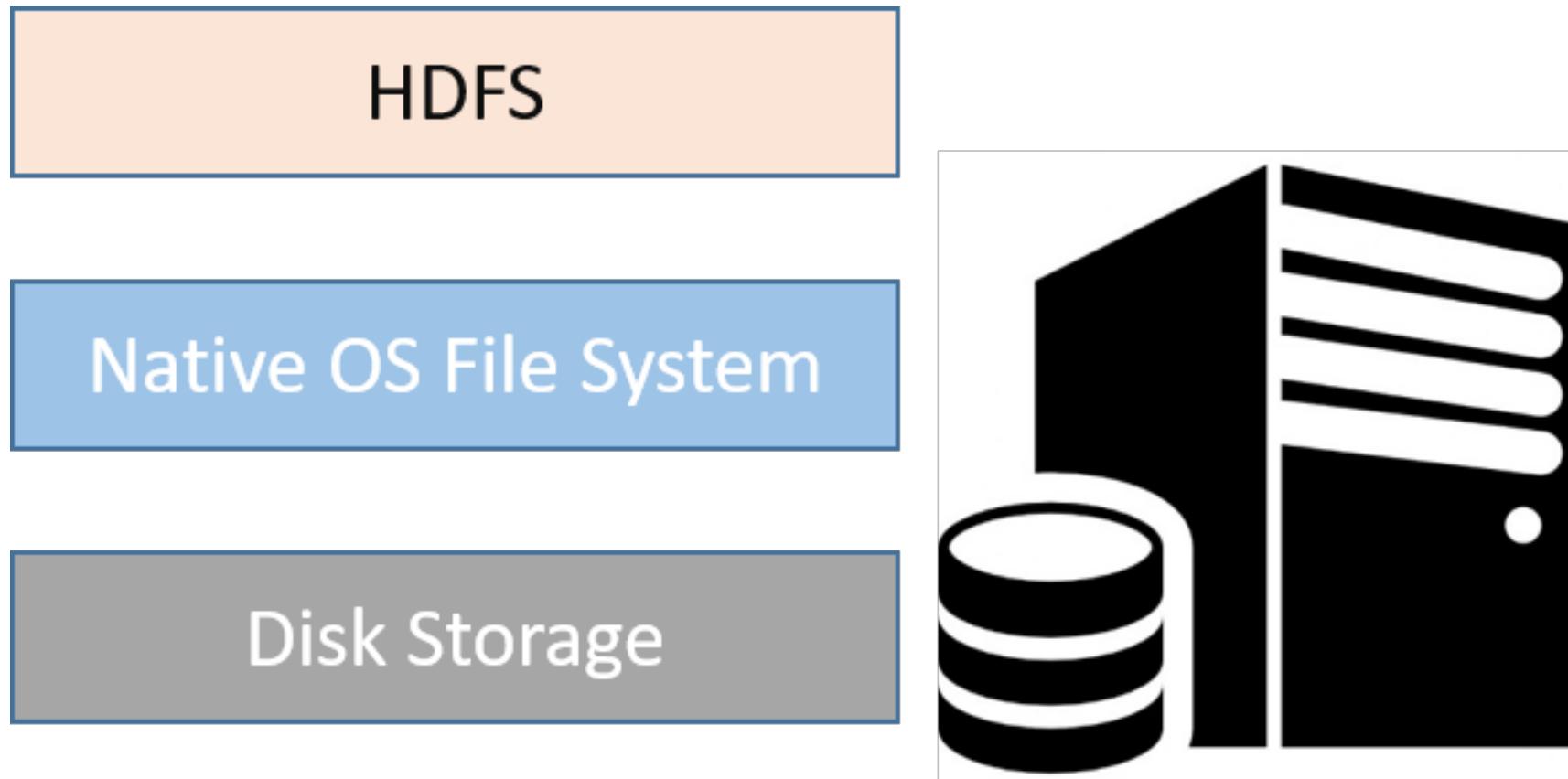
- Distributed File Systems, e.g., HDFS
- Distributed Databases, e.g., VoltDB
- Key-Value Storage Systems, e.g., Redis or Cassandra
- Queues, e.g., Pulsar or Kafka

Let's Talk Distributed File Systems

A distributed file system stores files across a large collection of machines while giving a single-file-system view to clients.

Hadoop Distributed File System (HDFS)¹²

- Abstracts physical location (Which node in the cluster) from the application
- Partition at ingestion time
- Replicate for high-availability and fault tolerance



¹² Inspired by [Google File System](#)

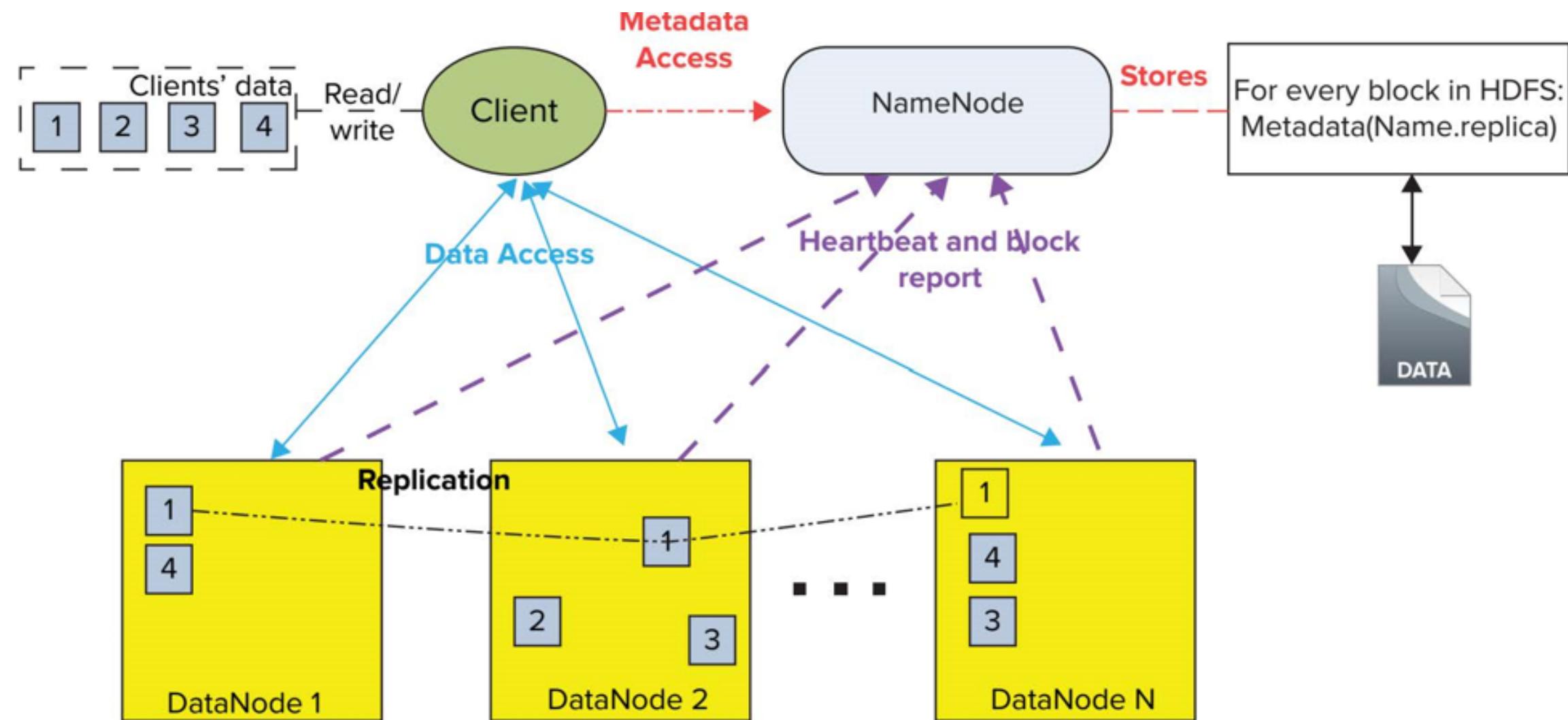
Design Objectives

- Partition and distribute a single file across different machines
- Favor larger partition sizes
- Data replication
- Local processing (as much as possible)

Optimizations

- Reading sequentially versus (random access and writing)
- No updates on files
- No local caching

HDFS Architecture¹³



¹³ Figure 2-1 in book Professional Hadoop Solutions

HDFS Files

- A single large file is partitioned into several blocks
 - Size of either 64 MB or 128MB
 - Compare that to block sizes on ordinary file systems
 - This is why sequential access is much better as the disk will make less numbers of seeks

Data Node

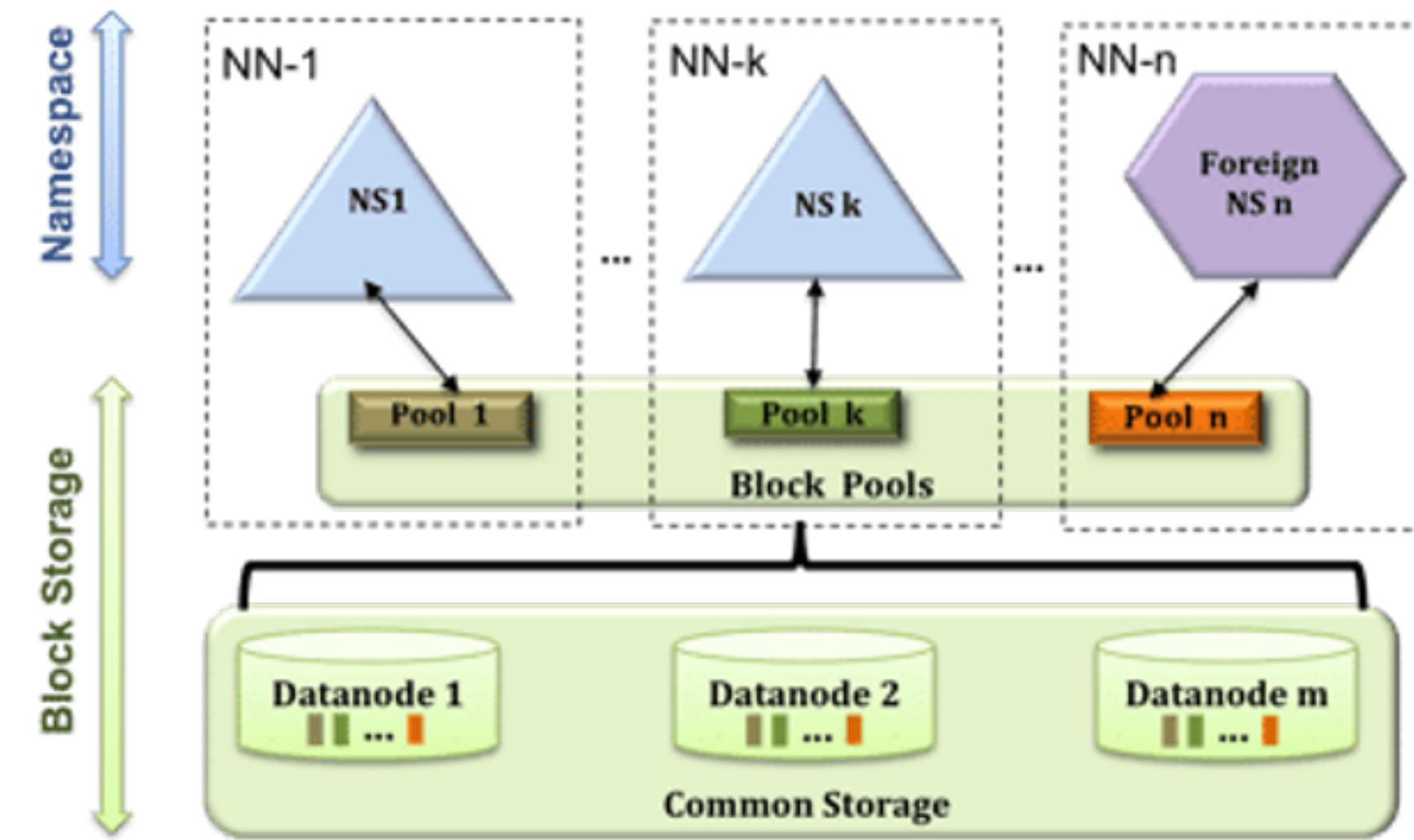
- It stores the received blocks in a local file system;
- It forwards that portion of data to the next DataNode in the list.
- The operation is repeated by the next receiving DataNode until the last node in the replica set receives data.

Name Node

- A single node that keeps the metadata of HDFS
 - Keeps the metedata in memory for fast access
 - Periodically flushes to the disk (FsImage file) for durability
 - Name node maintains a daemon process to handle the requests and to receive heartbeats from other data nodes

HDFS Federation

- By default, HDFS has a single NameNode. What is wrong with that? If NameNode daemon process goes down, the cluster is inaccessible
- A solution: HDFS Federation
 - Namespace Scalability: Horizontal scalability to access meta data as to access the data itself
 - Performance: Higher throughput as NameNodes can be queried concurrently
 - Isolation: Serve blocking applications by different NameNodes
- Is it more reliable?



Writing to HDFS

- When a client is writing data to an HDFS file, this data is first written to a local file.
- When the local file accumulates a full block of data, the client consults the NameNode to get a list of DataNodes that are assigned to host replicas of that block.
- The client then writes the data block from its local storage to the first DataNode in 4K portions.

Writing a File to HDFS Cont.

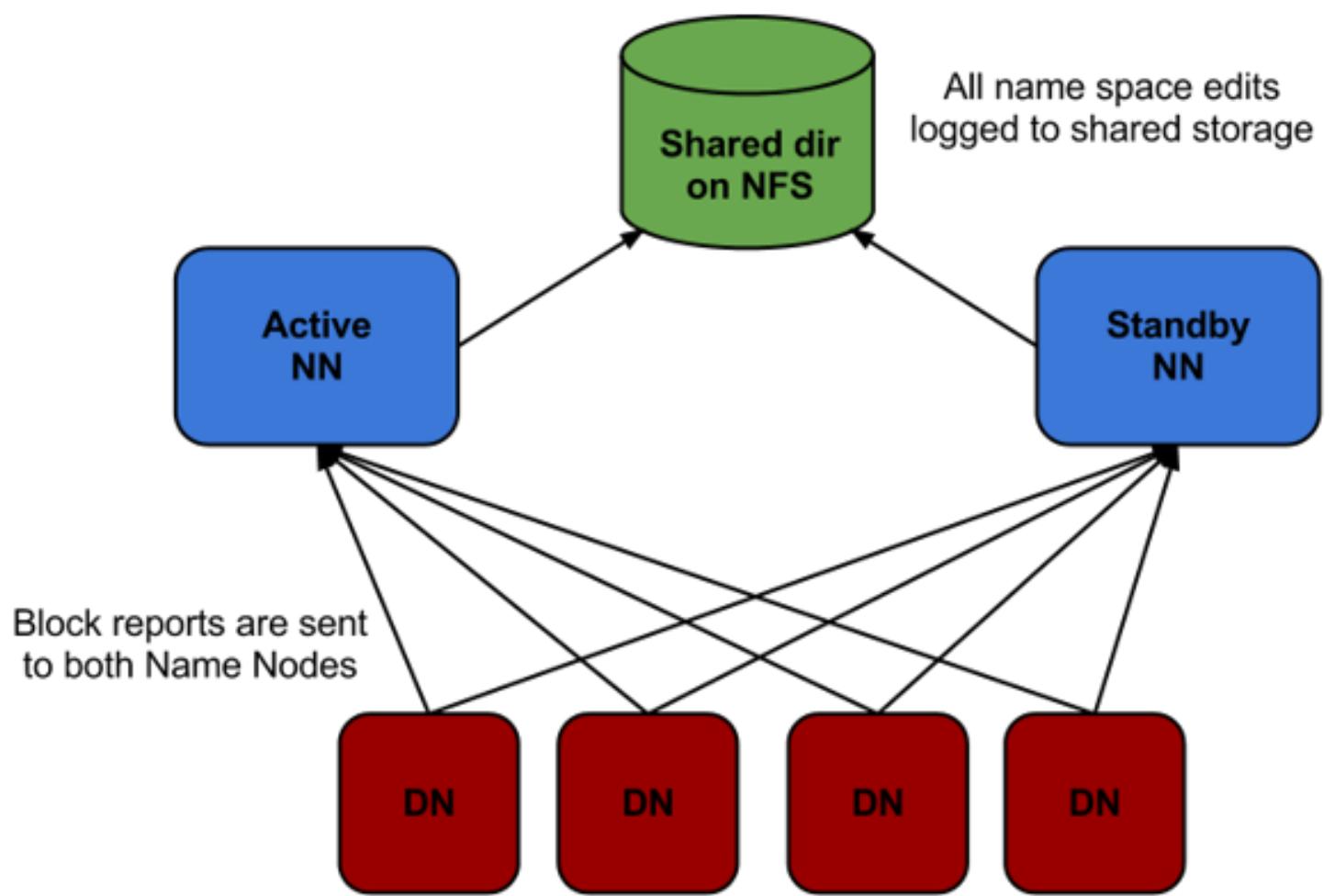
- This DataNode stores data locally without sending it any further
- If one of the DataNodes fails while the block is being written, it is removed from the pipeline
- The NameNode re-replicates it to make up for the missing replica caused by the failed DataNode
- When a file is closed, the remaining data in the temporary local file is pipelined to the DataNodes
- If the NameNode dies before the file is closed, the file is lost.

Replica Placement

- Replica placement is crucial for reliability of HDFS
 - Should not place the replicas on the same rack
- All decisions about placement of partitions/replicas are made by the NameNode
- NameNode tracks the availability of Data Nodes by means of Heartbeats
 - Every 3 seconds, NameNode should receive a heartbeat and a block report from each data node
 - Block report allows verifying the list of stored blocks on the data node
 - Data node with a missing heartbeat is declared dead, based on the catalog, replicas missing on this node are made up for through NameNode sending replicas to other available data nodes

HDFS High-availability

- Each NameNode is backed up with a slave other NameNode that keeps a copy of the catalog
- The slave node provides a failover replacement of the primary NameNode
- Both nodes must have access to a shared storage area
- Data nodes have to send heartbeats and block reports to both the master and slave NameNodes.



Data Modeling Techniques

According to Len Silverston (1997) only two modeling methodologies stand out, top-down and bottom-up[^14].



Data Modeling Techniques[^19]

- **Entity-Relationship (ER) Modeling**¹⁵ prescribes to design model encompassing the whole company and describe enterprise business through Entities and the relationships between them
 - it complies with 3rd normal form
 - tailored for OLTP
- **Dimensional Modeling (DM)**[^16] focuses on enabling complete requirement analysis while maintaining high performance when handling large and complex (analytical) queries
 - The star model and the snowflake model are examples of DM
 - tailored for OLAP
- **Data Vault (DV) Modeling**¹⁷ focuses on data integration trying to take the best of ER 3NF and DM
 - emphasizes establishment of an auditable basic data layer focusing on data history, traceability, and atomicity
 - one cannot use it directly for data analysis and decision making

¹⁵ by Bill Inmon

[^16]: Ralph Kimball, book 'The Data Warehouse Toolkit – The Complete Guide to Dimensional Modeling"

[^19]: [source](#)

¹⁷ Extra Read [Codd, Edgar F. "A relational model of data for large shared data banks." Communications of the ACM 13.6 \(1970\): 377-38z](#)

Event Sourcing¹⁸

The fundamental idea of Event Sourcing is ensuring that every change to the state of an application is captured in an event object, Event objects are immutable and stored in the sequence they were applied for the same lifetime as the application state itself.

¹⁸ Martin Fowler, [link](#)

Events

Events are both a fact and a notification.

They represent something that happened in the real world but include no expectation of any future action.

They travel in only one direction and expect no response (sometimes called “fire and forget”), but one may be “synthesized” from a subsequent event.



Hey, I've seen this one!



***What do you mean you've seen it?
It's brand new!***

It has
always been



So it is all dimensional modeling?



History of Data Models⁵

⁵ by Ilya Katsov



Key-Value



Ordered Key-Value



Big Table



Document,
Full-Text Search



Graph



SQL

The World's most simple database

```
#!/bin/bash

db_set () {
    echo "$1,$2" >> db
}

db_get () {
    grep "^\$1," db | sed -e "s/^$1,//" | tail -n 1
}
```

The Log

A log is an append-only sequence of records. It doesn't have to be human-readable; it might be binary and intended only for other programs to read.

Commit log

Offset	0	1	2	3	4	5	6	7	8	
key	k_1	k_2	k_1	k_3	k_4	k_5	k_5	k_2	k_6	
Value	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	