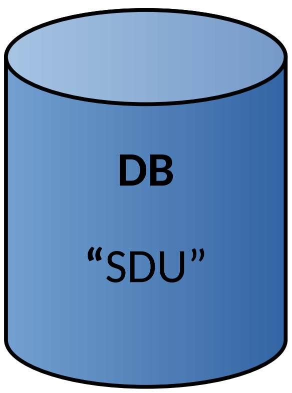
DM534: Introduction to Relational Databases

Oct 23, 2018
Christian Wiwie

UNIVERSITY OF
SOUTHERN DENMARK

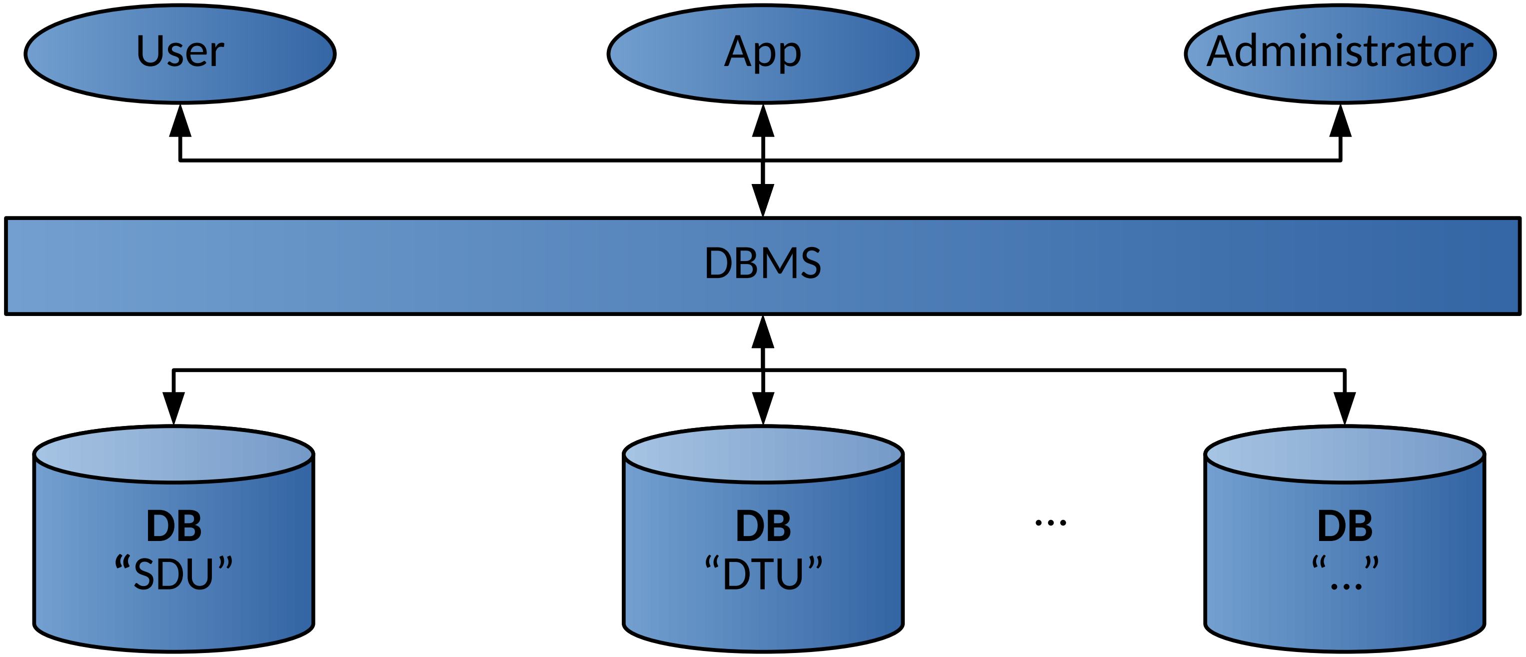
What are Databases?

- Repository for large data amounts
- Describes a logical structure of contained data
- Guarantees data integrity by enforcing constraints
- Allows for efficient access
- Consistent and safe storage



Database Management System (DBMS)

- A DBMS manages databases
- Access to database only via DBMS



Why learn about Databases?

- Used almost everywhere
- Crucial for safety & integrity of stored data
- Jobs exist dealing specifically with databases
- Increasingly relevant
  - Technical advances → More & larger data amounts

Where are Databases used?

- Wherever large amounts of data are managed
- Often multiple DBMS in use that cater specific needs
- **Google** uses *Bigtable* for web indexing, Google Maps, …
- **Facebook** uses *MySQL; TAO* for graph search, …
- Other applications
  - Corporate data: payrolls, inventory, sales, customers, …
  - Web search: Google, Live, Yahoo, …
  - Social networks: Facebook, Twitter, …
  - Scientific and medical databases

Features of a modern DBMS

- Highly **efficient access** to stored data using *indexes*
- Backup/log mechanisms ensure **data safety**
- Security policies to manage access **permissions**
- Data **consistency**: Can enforce complex data constraints, including dependencies
- Flexible **searching, sorting, filtering**
- Ensures all the above with simultaneous multi-user access

Databases vs. storage in files

- File storage does not provide most of these features

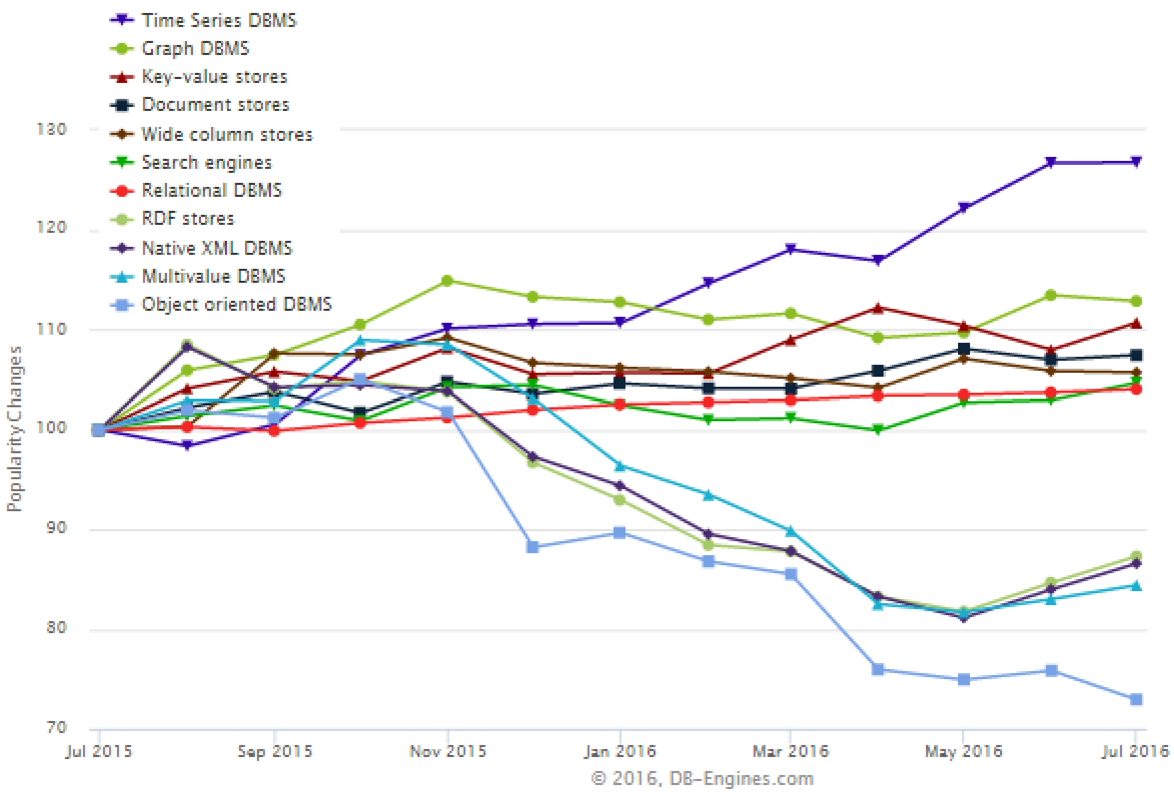
  → Structure and constraints need to be imposed manually

- Complex operations
  - not trivial to do right → Error prone
  - are slow, e.g. searching, sorting

Types of DBMS / databases

- Data can be modeled and organized differently
- Optimized for specific kinds of operations
- **Relational DBMS (RDBMS) / databases** the most wide-spread
  - Based on mathematical relations
  - Basically, a database is a collection of relations
  - e.g. MySQL, PostgreSQL, …
- **Graph DBMS / databases**
  - Data is a network, with entities and connections between them
  - e.g. neo4j

DBMS type popularity



Most widely used DBMS

- Ranking of most widely used DBMS

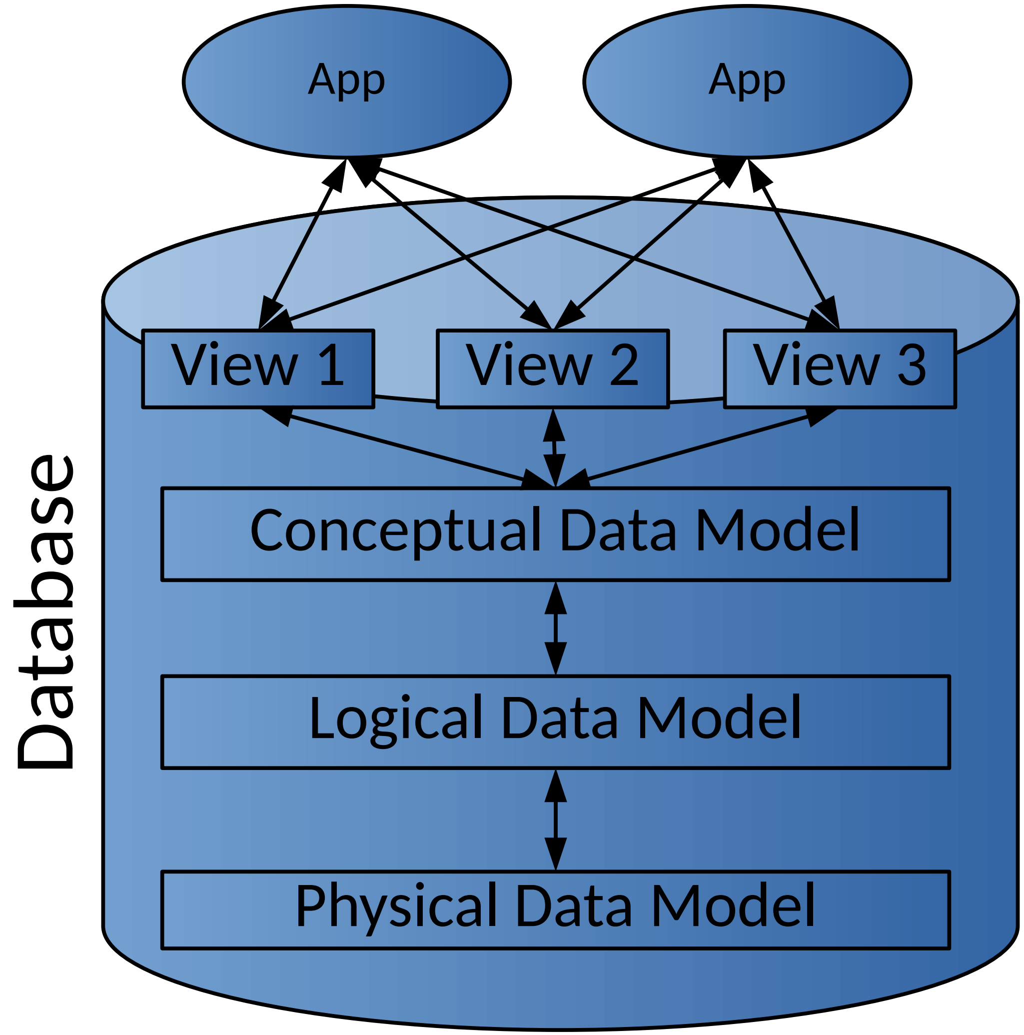
346 systems in ranking, October 2018

Rank Oct 2018	Rank Sep 2018	Rank Oct 2017	DBMS	Database Model	Score Oct 2018	Score Sep 2018	Score Oct 2017
1.	1.	1.	Oracle	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2	Relational DBMS	179.69	-1.38	-14.90
7.	↑ 8.	↑ 9.	Redis	Key-value store	145.29	+4.35	+23.24
8.	↓ 7.	↑ 10.	Elasticsearch	Search engine	142.33	-0.28	+22.09
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	↓ 8.	Cassandra	Wide column store	123.39	+3.83	-1.40

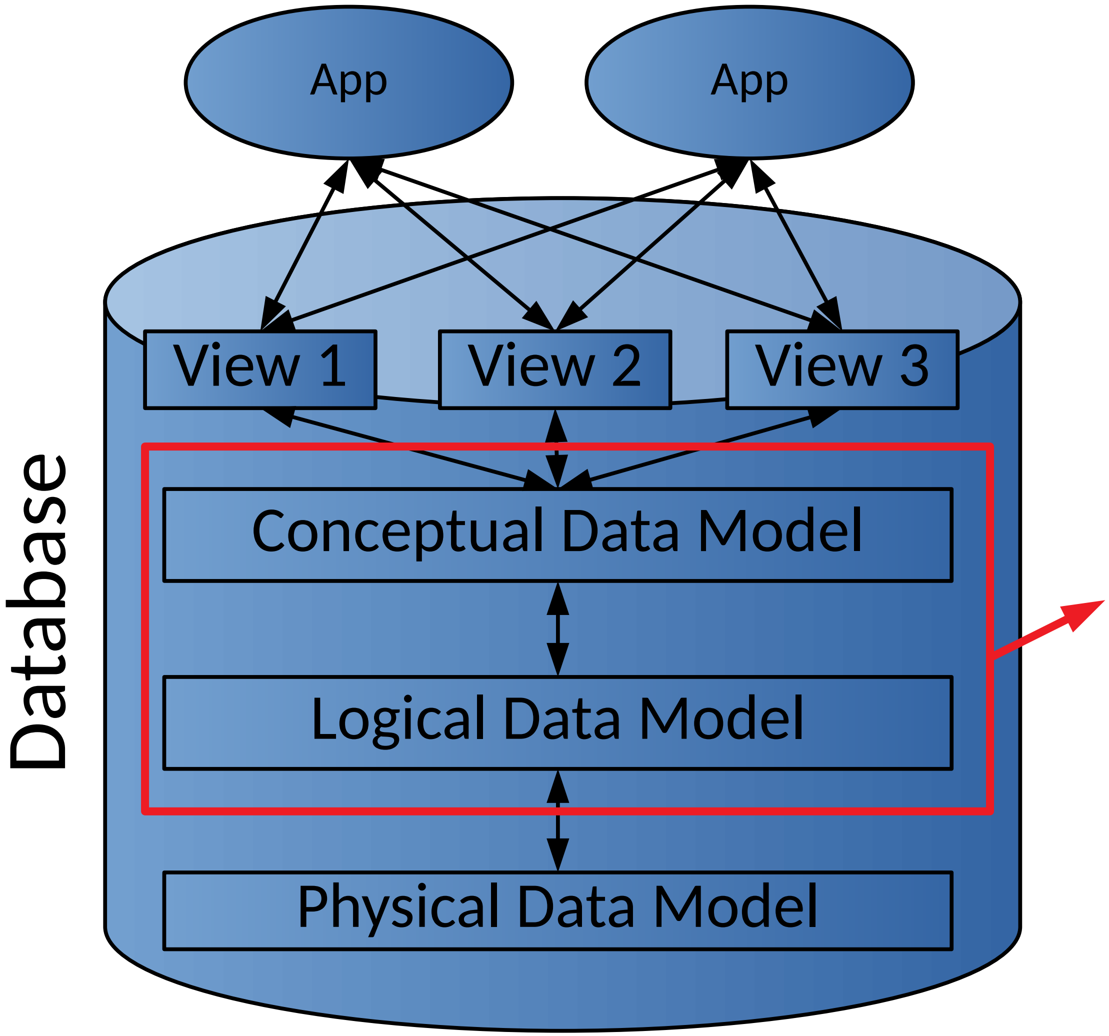
Source: https://db-engines.com/en/ranking

Internal Structure of a Database

- Multiple levels of abstraction
- Higher levels independent of lower levels
- Software independent of how data is logically and physically structured and stored



Internal Structure of a Database



We will be looking at this part

From here on we are exam relevant

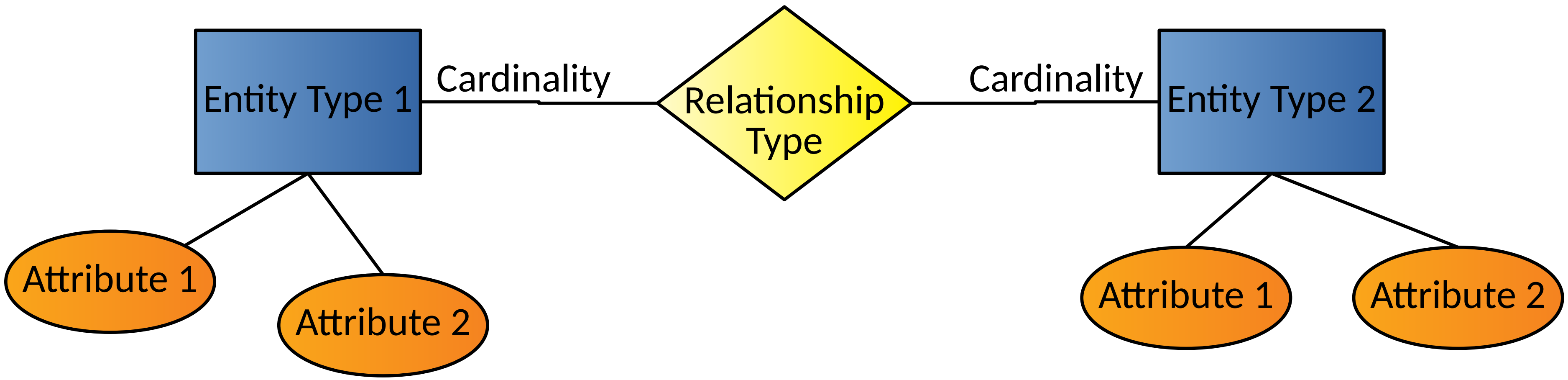
Conceptual Data Model

- Semantics of stored data
- Which entities (concepts) are stored?
- Which relationships exist between entities?

- Independent of DBM type and specific DBMS used

Conceptual Data Model

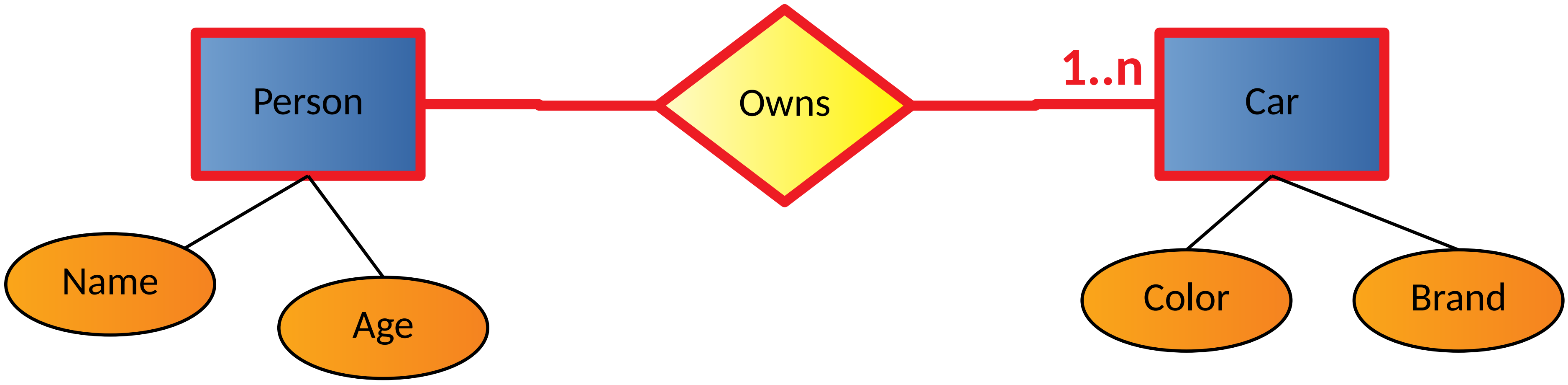
- Visualized with Entity-Relationship (ER) diagrams:



- Cardinality: How many entities are involved in a relationship?

Conceptual Data Model

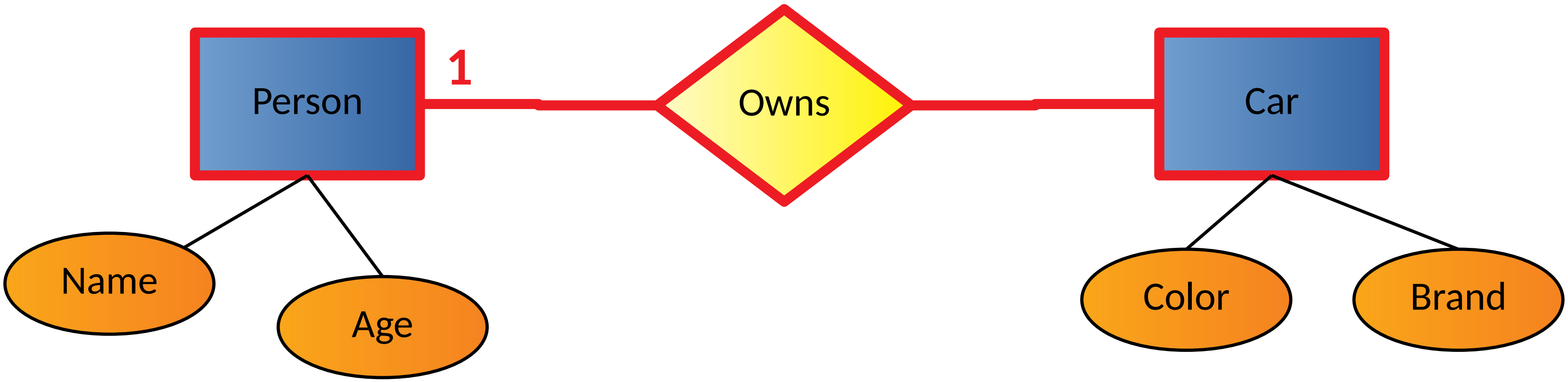
- Example Cardinalities



- Read:
  - **One person** owns **one or more cars**

Conceptual Data Model

- Example Cardinalities



- Read:
  - **One car** is owned by **exactly one person**

    → Constraints do not necessarily hold in reality (joint ownership)

Logical Data Model

- Usually derived from conceptual data model
- Expressed in terms of data structures specific to type of DBMS
  - Relational DBMS: relational (logical) data model
  - Graph DBMS: a graph structure

- But: Still independent of specific DBMS used

We focus on relational DBMS

Relational (Logical) Data Model

- Main structural concept: **relations**
  - Basically a table with rows and columns
- A relation has a **relation schema**
  - Specifies structure of data that *can be stored* in relation

- relations != relationship
  - Relationship is part of conceptual data model
  - A relation can hold data for entities or relationships

Relational (Logical) Data Model

- A **relation schema** consists of:
  - a name
  - a set of attribute names
  - Optionally: attribute types

relation_name(attribute$_1$, attribute$_2$, …) or
relation_name(attribute$_1$: type$_1$, attribute$_2$: type$_2$, …)

Relation Schemas

- A **relation** usually corresponds to
  - Real world entity types (e.g. car, person, ...)
  - Real world relationship types (e.g. person owns car)
- Example **relation schemas**:
  - *Car(color, brand)*
  - *Person(name: CHAR(20),age: INTEGER)*
  - *Owns(name, age, color, brand)*

Relation Schemas

- Example **relation schema**:
  - *Car(color, brand)*

- Reads:
  - Relation ‘Car’ contains/describes cars with attributes color and brand

Relation Instances

- A relation or relation schema does not specify which data is stored
- A **relation instance** is a realization of a relation with data
  - Data must conform to relation's schema
- Many relation instances can exist for the same relation

Tuples

- A data entry in a relation instance is called **tuple**
- A **tuple** is a realization of the relation’s schema
  - Assigns values to the attributes of the relation
  - Must conform to relation schema

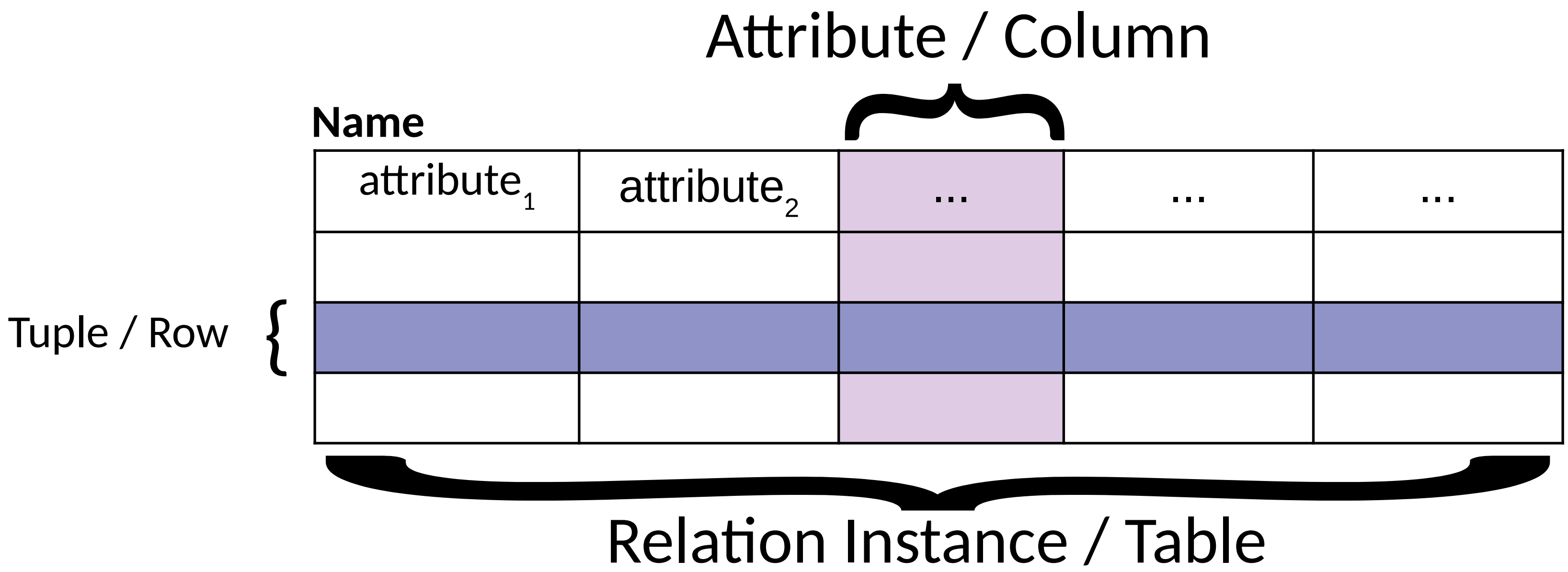
Tuples

- Example tuples of the relation *Car(color*, *brand)*:
  - (‘red’, ‘Ford’)
  - (‘blue’, ‘Mercedes’)

- Example tuples of the relation *Person(name*, *age)*:
  - (‘Henry’, 36)
  - (‘Thomas’, 22)

Relation Instances

- Can be visualized by a table:



Relation Instance

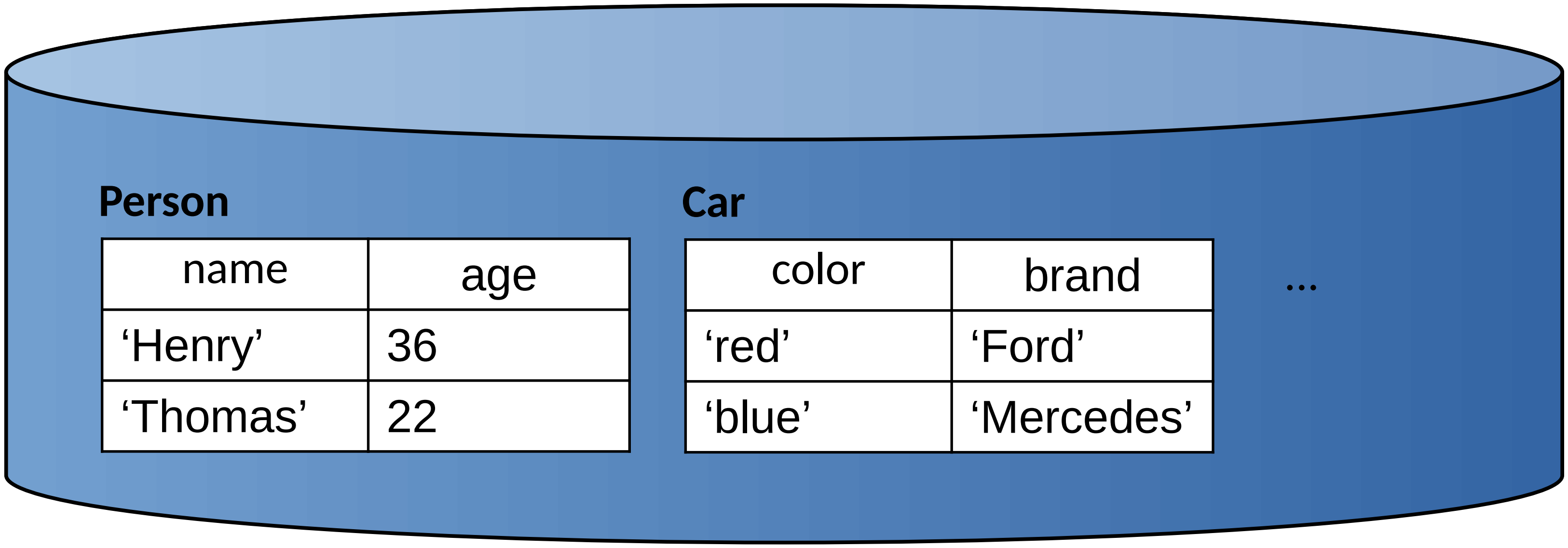
- Example **relation instance** of the person relation

Person

name	age
‘Henry’	36
‘Thomas’	22

Database Instance

- A **database instance** is the collection of all its relation instances
  - i.e. all relation schemas and their corresponding tuples



Person

name	age
'Henry'	36
'Thomas'	22

Car

color	brand
'red'	'Ford'
'blue'	'Mercedes'

Integrity Constraints (ICs)

Integrity Constraints (ICs)

- Condition that must be true for any database instance
- Specified when relation schemas are defined
- Checked whenever relation instances are modified
  - i.e., when tuple is added, deleted, or modified

Domain constraints

- Domain of valid values for an attribute
  - e.g., INTEGER, FLOAT, CHAR(20), …
  - correspond to data types in programming languages
- Example relation schema:

  Person(name: CHAR(20),age: INTEGER)

name	age
'Henry'	36
'Mads'	'Doe'

Domain constraint violation

→ DBMS will not allow insertion of this tuple

Semantic integrity constraints

- Semantic restrictions on the data
  - e.g., age >= 18
- Example relation schema:

  Person(name: CHAR(20),age: INTEGER)

name	age
'Henry'	36
'Mads'	16

Constraint violation

→ DBMS will not allow insertion of this tuple

Primary Keys

- Set of relation attributes
  - that uniquely identifies tuples of relation
  - all tuples need to have unique values for these attributes
- Example: CPR is primary key of relation **Person**

  → There cannot be two tuples with same CPR number

<u>CPR</u>	Name	Birthday	Address
...	...	...	...
1904651243	Svensson	19.04.1965	...
...	...	...	...
1904651243	...	...	...
...	...	...	...

Not allowed

Primary Keys

- Primary key “points” to exactly one tuple
  - → can be used to lookup corresponding tuple
  - → e.g., person can be looked up using CPR

What is the name of the person with CPR=1904651243 ?

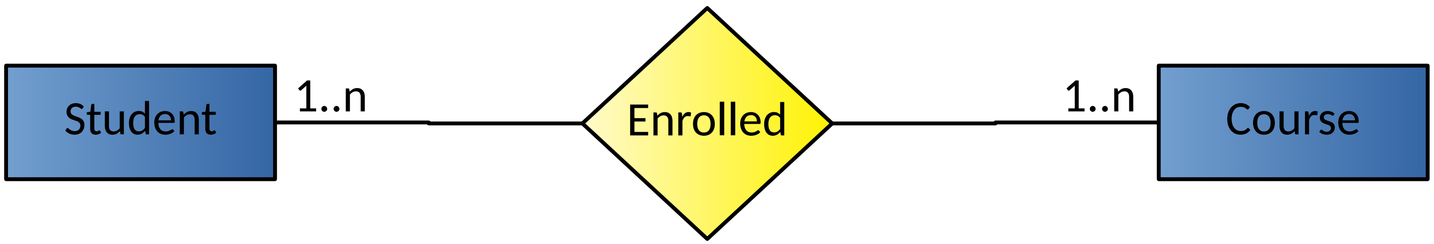
<u>**CPR**</u>	Name	Birthday	Address
...	...	...	...
1904651243 →	Svensson	19.04.1965	...
...	...	...	...

UNIVERSITY OF SOUTHERN DENMARK

Foreign Keys

- Allow to associate tuples in different relations
- Tuple of source relation → tuple of target relation
  - Source and target relation can be the same
  - Can only point to a primary key in the target relation

Example: University Database



- Conceptual schema:
  - Students(sid: string, name: string, login: string, age: integer, gpa:real)
  - Courses(cid: string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)

Example: Foreign Keys

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

FOREIGN Key

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

PRIMARY Key

Query Languages

Query Languages

- Allow manipulation and retrieval of data from a database
- Query languages != programming languages
- not expected to be “turing complete”
  - → i.e., not every operation can be expressed
- not intended to be used for complex calculations
- support easy, efficient access to large data sets

Relational Query Languages

- Based on relational algebra
- For relational databases, i.e. relational data model
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization
- **SQL**: Most widely used relational query language

→ Understanding Relational Algebra is key to understanding SQL, query processing!

Relational Query Languages

- More on relational query languages and relational algebra on Thursday, 10-11