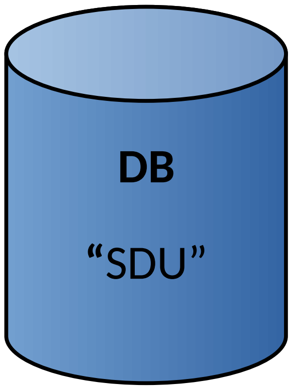
DM573: Introduction to Relational Databases

Slides by Christian Wiwie
(With edits by Rolf Fagerberg)

SDU
UNIVERSITY OF
SOUTHERN DENMARK

First Some perspective …

What are Databases?



A large software tool that:

- Stores large amounts of data
- Allows defining the structure of contained data
- Allows efficient and flexible searches and updates
- Guarantees data integrity by enforcing constraints
- Ensures consistent and safe storage
- Allows simultaneous multi-user access

Databases vs. storage in plain files

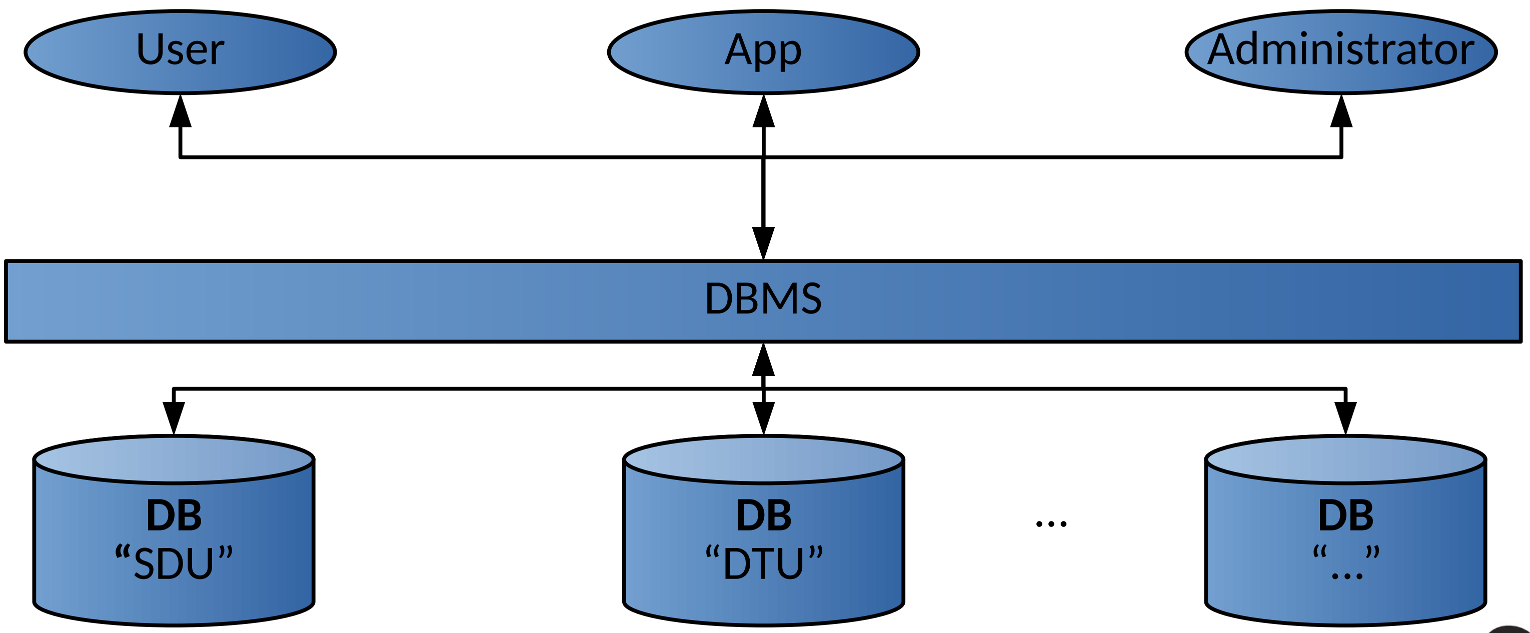
- File storage does not provide most of these features

  Structure and integrity constraints need to be imposed manually

- Complex operations
  - Not trivial to make correct
  - Not trivial to make efficient

Database Management System (DBMS)

- The DBMS is the program that manages a (set of) database(s)
- Access to database is only via the DBMS



Why learn about Databases?

- Used almost everywhere
- Crucial for safety & integrity of stored data
- Jobs exist dealing specifically with databases
- Increasingly relevant as we generate and store more and more data

Where are Databases used?

Wherever large amounts of data are managed:

- Corporate data: payrolls, inventory, sales, customers, …
- University records of students, grades, courses,…
- Scientific and medical databases
- Data backend behind webpages

Often different DBMS in use that cater specific needs

- **Google** uses *Bigtable* for web indexing, Google Maps, …
- **Facebook** uses *MySQL; TAO* for graph search,…

Types of DBMS / databases

- Data can be modeled and organized differently
- Optimized for specific kinds of operations
- **Relational DBMS (RDBMS) / databases** (the classic and most widely used type)
  - Based on mathematical relations
  - Basically, a database is a collection of relations
  - Example: MySQL, PostgreSQL, Oracle, ...
- **Graph DBMS / databases**
  - Data is a network, with entities and connections between them
  - Example: neo4j
- **Several other types...**

Most widely used DBMS

- Ranking of most widely used DBMS

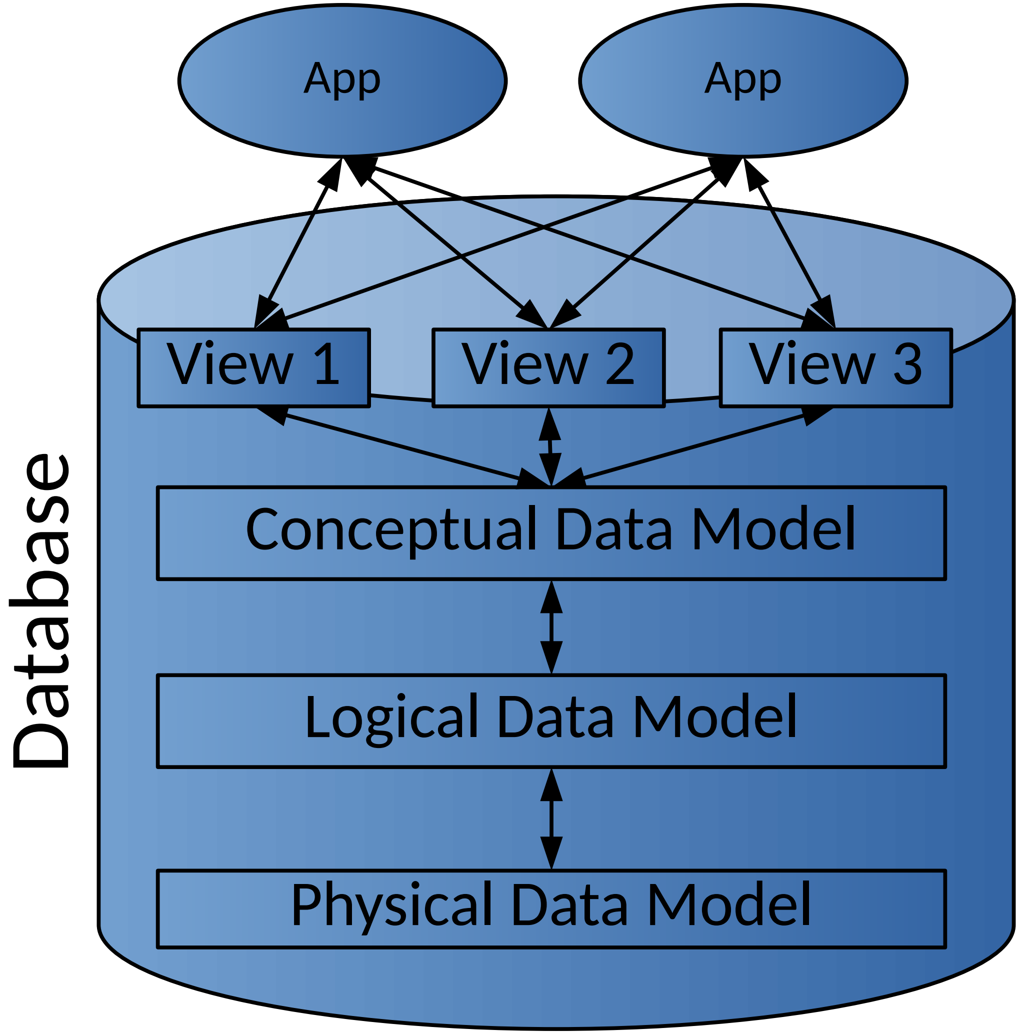
415 systems in ranking, October 2023

Rank			DBMS	Database Model	Score		
Oct 2023	Sep 2023	Oct 2022			Oct 2023	Sep 2023	Oct 2022
1.	1.	1.	Oracle	Relational, Multi-model	1261.42	+20.54	+25.05
2.	2.	2.	MySQL	Relational, Multi-model	1133.32	+21.83	-72.06
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	896.88	-5.34	-27.80
4.	4.	4.	PostgreSQL	Relational, Multi-model	638.82	+18.06	+16.10
5.	5.	5.	MongoDB	Document, Multi-model	431.42	-8.00	-54.81
6.	6.	6.	Redis	Key-value, Multi-model	162.96	-0.72	-20.41
7.	7.	7.	Elasticsearch	Search engine, Multi-model	137.15	-1.84	-13.92
8.	8.	8.	IBM Db2	Relational, Multi-model	134.87	-1.85	-14.79
9.	9.	↑ 10.	SQLite	Relational	125.14	-4.06	-12.66
10.	10.	↓ 9.	Microsoft Access	Relational	124.31	-4.25	-13.85

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

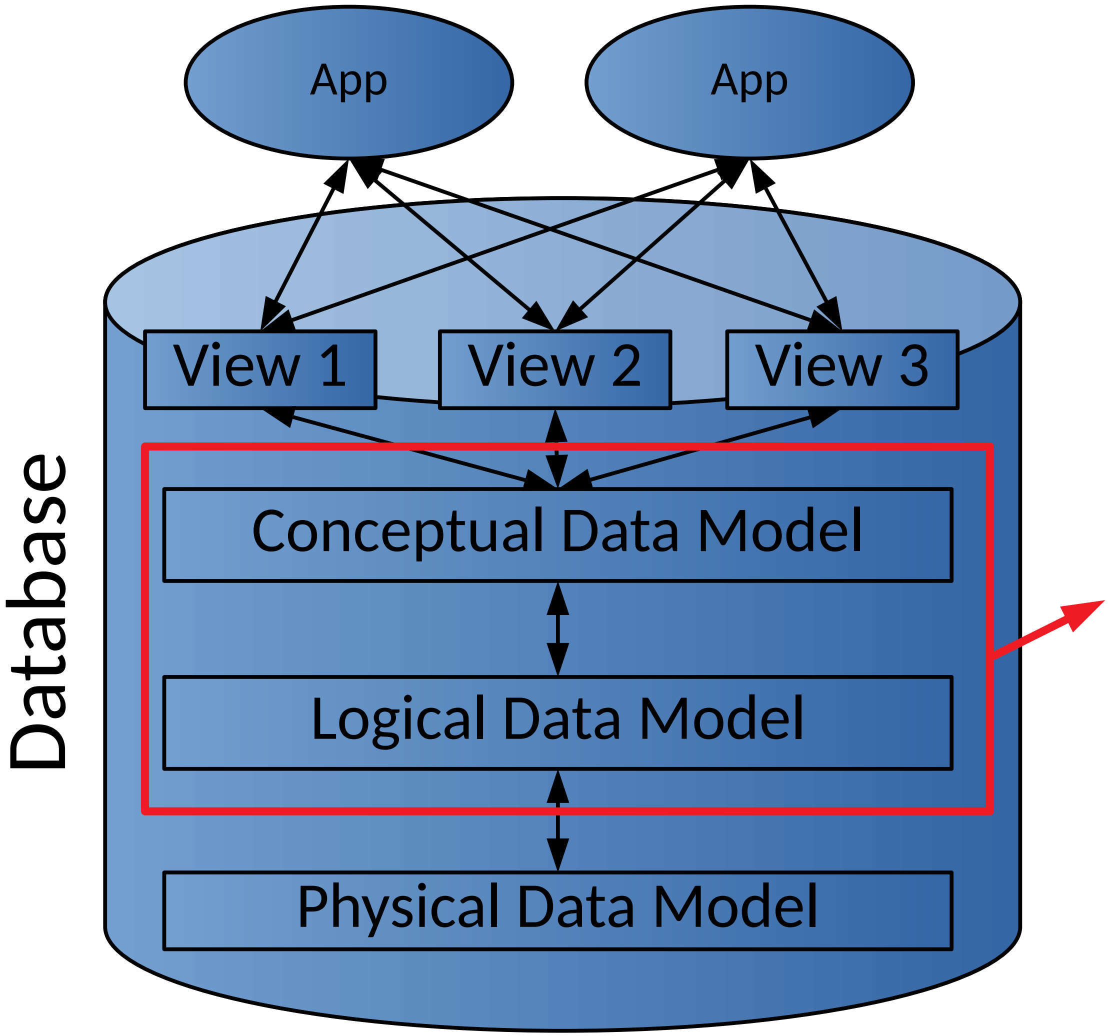
Internal Structure of a Database

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



SDU UNIVERSITY OF SOUTHERN DENMARK

Internal Structure of a Database



We will be looking at this part

From here on the material is exam relevant

Conceptual Data Model

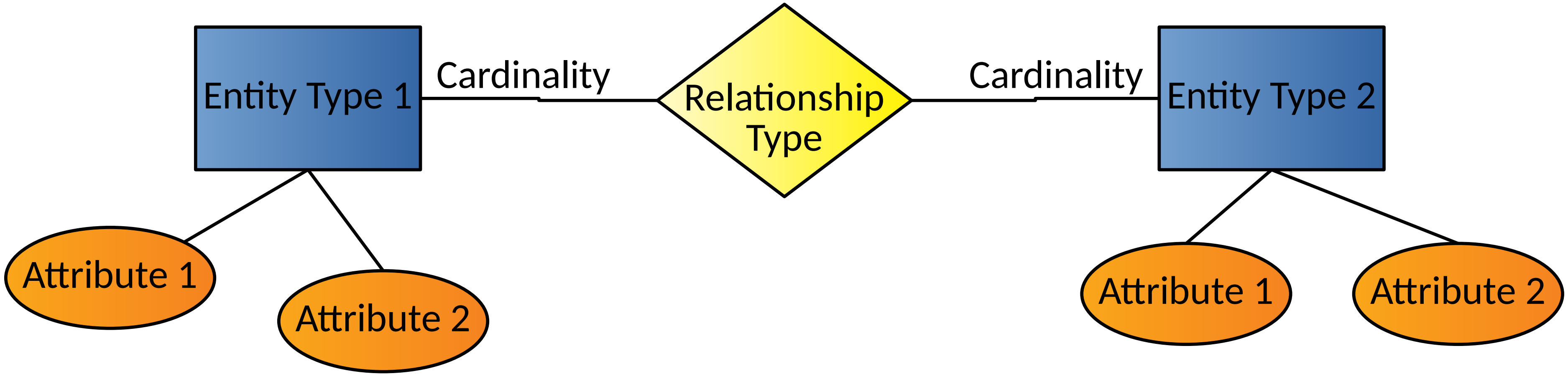
Used for modeling/defining the structure of the data that can be stored.

Independent of the specific DBMS used.

Conceptual Data Model

Most widely used conceptual model:

Entity-Relationship (ER) diagrams

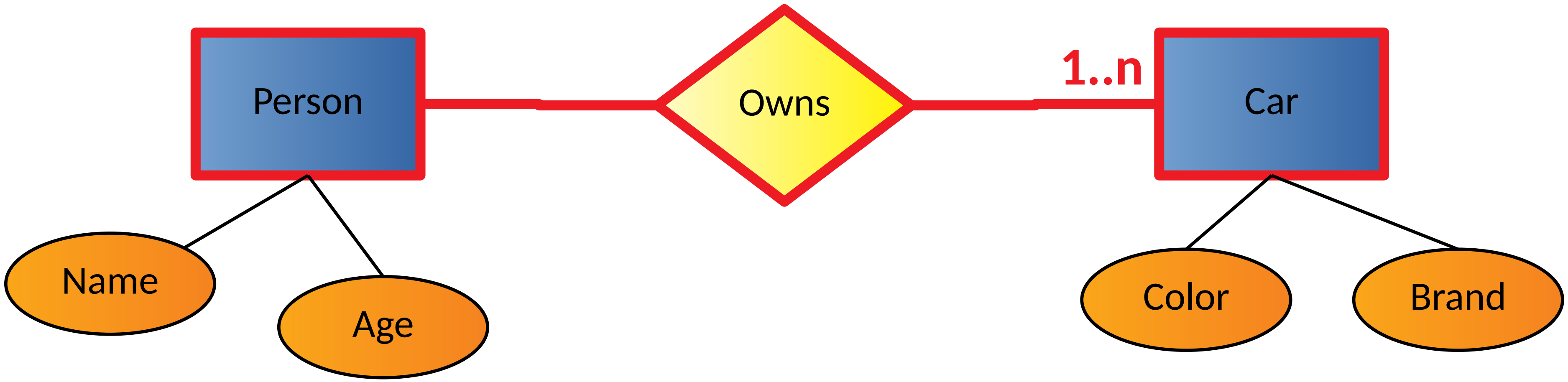


Cardinality: How many entities are involved in a relationship?

SDU
UNIVERSITY OF SOUTHERN DENMARK

Conceptual Data Model

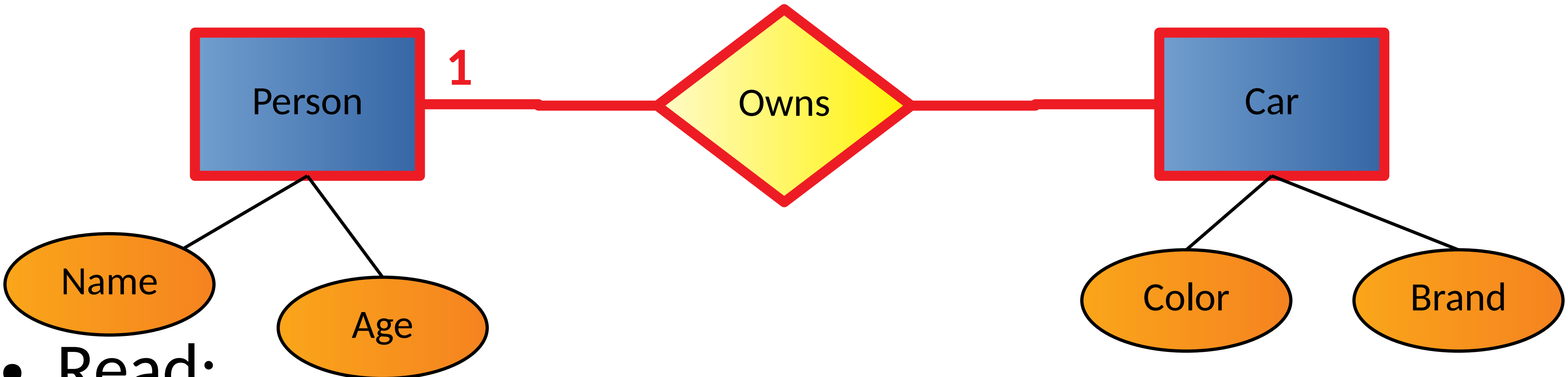
- Example Cardinalities



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

Conceptual Data Model

- Example Cardinalities



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

    Observe: Constraints do not necessarily hold true in all of reality (joint ownership). We are **modeling** what we want to hold in our database, i.e., what type of data we can store.

Logical Data Model

- Usually derived from conceptual data model
- Expressed in terms of data structures specific to *type* of DBMS
  - Relational DBMS: relational data model
  - Graph DBMS: a graph structure
  - Etc…

- But: Still independent of the specific details of the DBMS used (different DBMS of the same type may differ in details).

SDU UNIVERSITY OF SOUTHERN DENMARK

We focus on relational DBMS

Relational (Logical) Data Model

- Main structural concept: **relations**
  - Is basically a fancy name for tables
- A relation has a **relation schema**
  - Specifies structure of data that *can be stored* in relation. A fancy name for table header.

name	age
'Henry'	36
'Thomas'	22

NOTE: "relation" != "relationship"

- Relationships are part of the ER-model (conceptual level). Relations are part of the relational model (logical level) .
- There is a standard translation (see later slides) from the ER-model model to the relational model. In this translation, one relation will represent either an entity or a relationship.

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

- Example **relation schemas**:
  - *Car(color, brand)*
  - *Person(name: CHAR(20),age: INTEGER)*
  - *Owns(name, age, color, brand)*

Relation Instances

- A relation or relation schema does not specify which data is stored
- A **relation instance** is a set of data entries each conforming to the relation's schema. A data entry is also called a "table row" or a "tuple".
- Many relation instances can exist for the same relation (similarly to classes vs. objects).

Examples of 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)

Example of a relation Instance

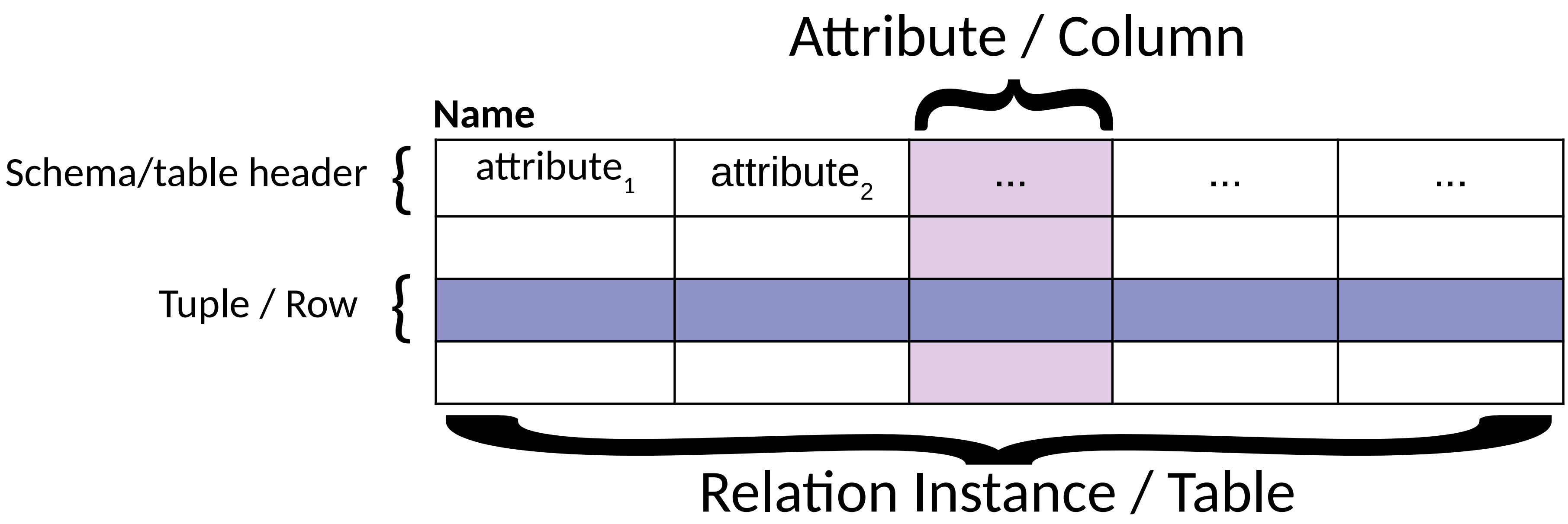
An instance of the person relation:

Person

name	age
‘Henry’	36
‘Thomas’	22

Relation Instances

Summing up: the relational model is a set of fancy new names for the various parts of tables:



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

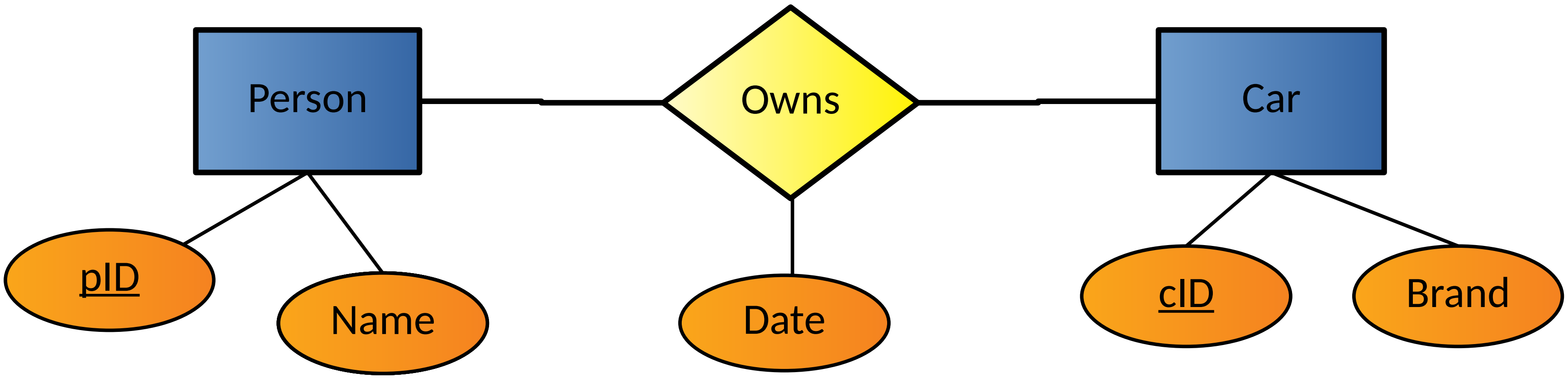
Car

color	brand
‘red’	‘Ford’
‘blue’	‘Mercedes’

…

From ER Diagrams to Relations

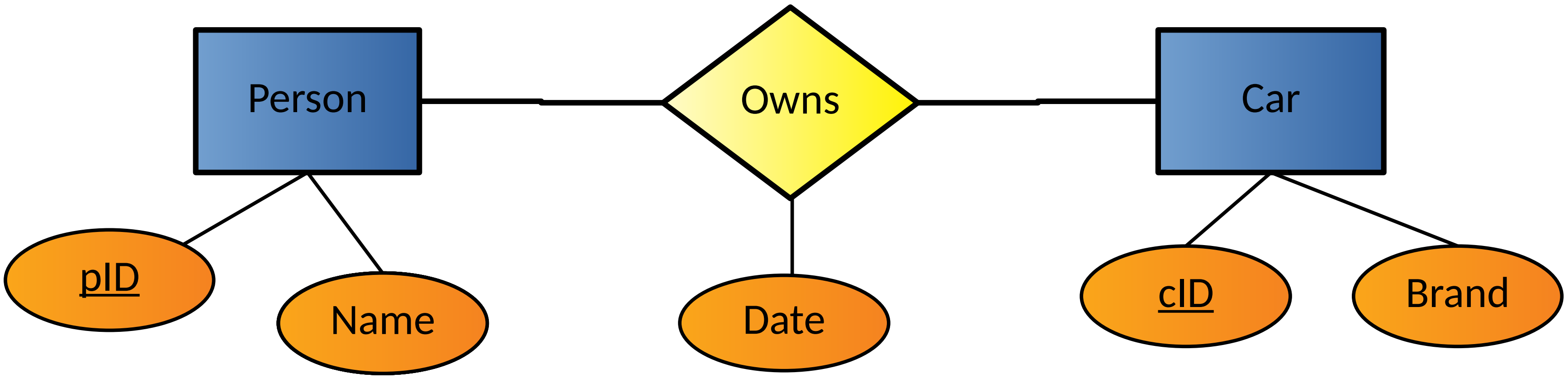
Standard translation:



- Each **entity** is converted directly to a relation (same attributes and keys).
- Each **relationship** is converted to a relation with attributes consisting of the keys of its related entities plus its own attributes (if any). More on keys later.

From ER Diagrams to Relations

Example:



Person(pID: INTEGER, Name: CHAR(20))

Car(cID: INTEGER, Brand: CHAR(20))

Owns(pID: INTEGER, cID: INTEGER, Date: CHAR(10))

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

CPR	Name	Birthday	Address
...	...	...	...
1904651243	Svensson	19.04.1965	...
...	...	...	...
1904651243	...	...	...
...	...	...	...

Not allowed

UNIVERSITY OF SOUTHERN DENMARK

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

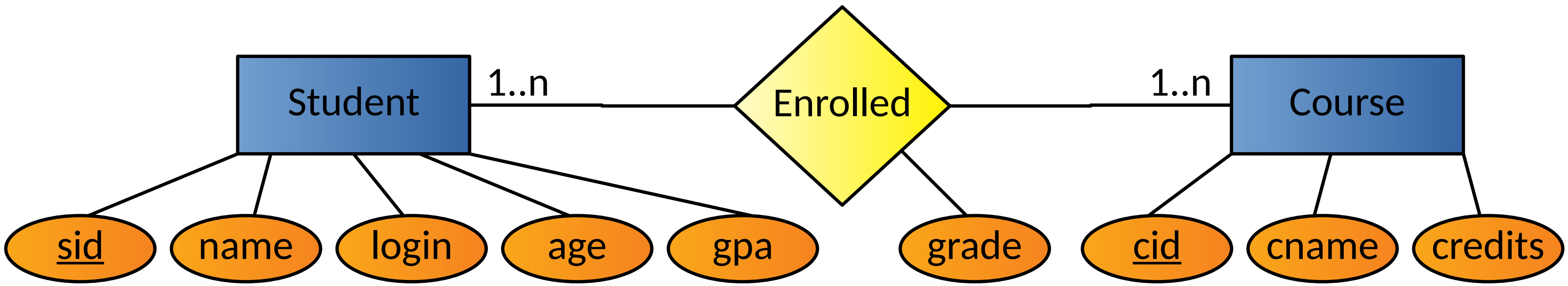
What is the name of the person with CPR=1904651243 ?

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

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
  - Existence of a tuple with that primary key in target relation can be enforced by DBMS

Example: University Database



Relation schemas:

- 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

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

FOREIGN Key

PRIMARY Key

SDU UNIVERSITY OF SOUTHERN DENMARK

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 algorithm 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 query languages:
  - 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 at next lecture/slide set.