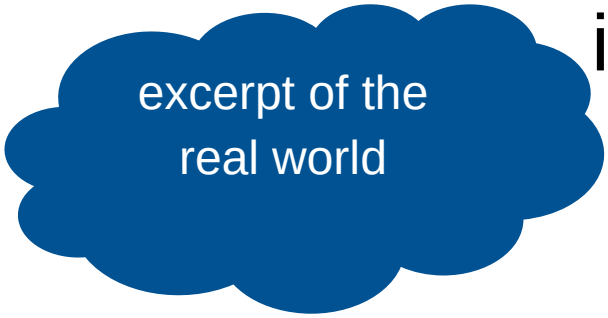
relational model in context



conceptual schema

relational schema / relational model

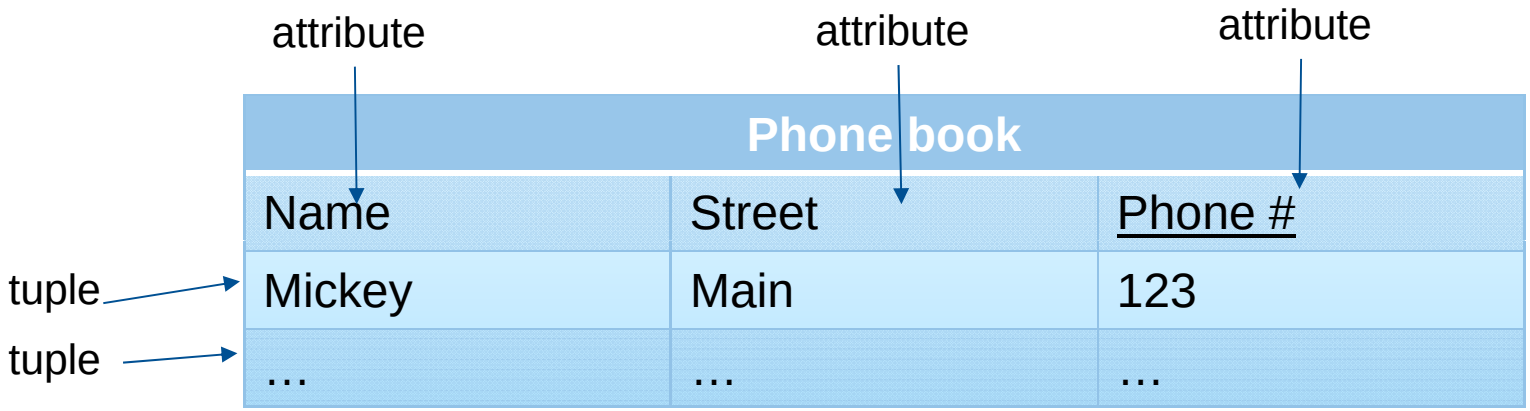
XML schema

network schema

object-oriented schema

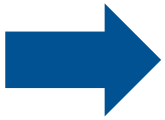
relational model - properties

- set oriented – different to network or hierarchical models
- simple structure – everything is stored in tables (called relations)
- rows of the table are tuples and columns are attributes
- primary key is underlined



transformation of conceptual schema into relations

- E / R model has two fundamental structures
  - entities
  - relations
- relational model has ONLY RELATIONS (tables)

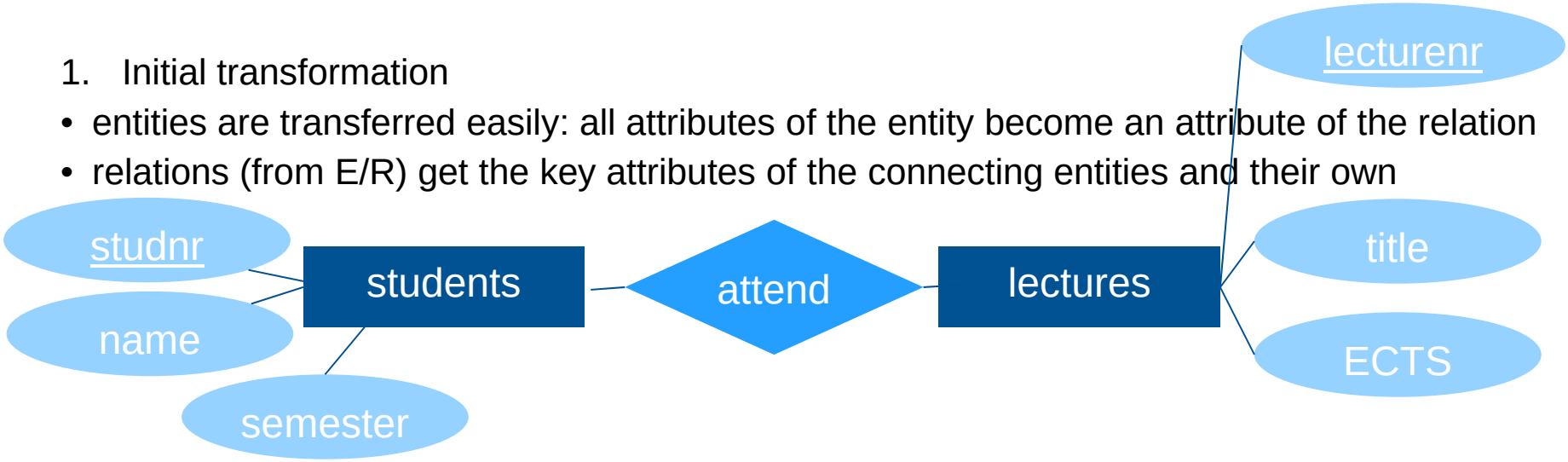


transformation of conceptual to relational is a two – step process

relational model

1. Initial transformation

- entities are transferred easily: all attributes of the entity become an attribute of the relation
- relations (from E/R) get the key attributes of the connecting entities and their own



Students: {[studnr:integer, *name*: *string*, *semester*: *integer*]}

attend (N:M): {[studnr: integer, lecturenr: integer]}

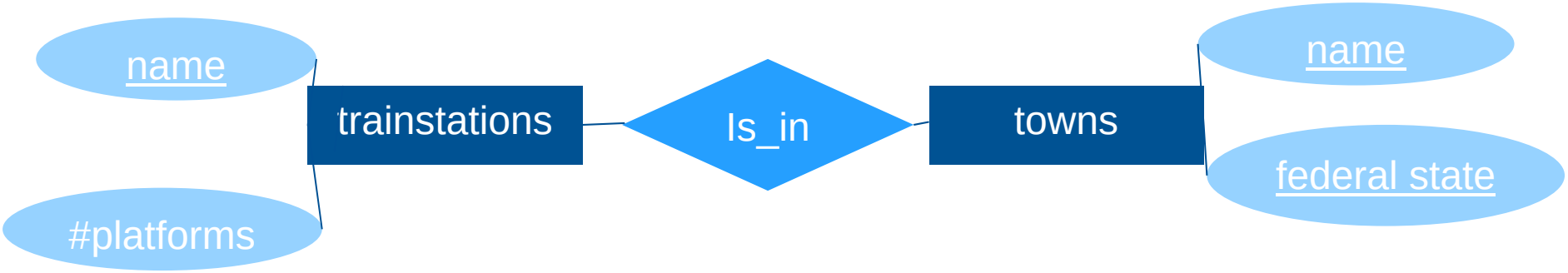
- for (1:N) or (N:1) relations only the key attribute of the N part becomes key
- **give (1:N)**: {[PersNr: integer, lecturenr: integer]}

attend	
studnr	lecturenr
123	0010
…	…

2. Refinement for (1:N) or (N:1) and (1:1)

- relations with the same primary key can be condensed

relational model – example



trainstation: {[name: string, #platforms:integer]}
towns: {[name: string, federal state: string]}
Is_in: {[name: string, federal state: string, name: string]}

(1:N) relation: one train station is in only one city.

trainstation: {[name, #platforms, is_in]}
towns: {[name,federal state]}

DDL: Data Definition Language

Part of SQL that is used for Data Definition to:

- Define the schema
- Controll access to the DB

Typical statements:

- Create/drop tables
- Create/drop views
- Create/drop indexes

Supported data types: numbers, strings, dates etc.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Integrity Constraints

Guarantees the consistency of the data.

Typical integrity constraints:

- Primary key constraint
- NOT NULL
- UNIQUE
- DEFAULT
- CHECK clauses

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

Referential Integrity

- Foreign key constraint
- Constraint that links tables with each other
- Prevent actions that destroy that link
- Variants: SET NULL, CASCADE

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID) ON DELETE SET NULL
);
```

Example exercises

- Provide SQL for:
  - New table **lectures** with constraint that weekly hours are between 2 and 6
  - New table **students** with semester set to 1 by default
- The following code is given:

Professors

PersNr	Name	Level	Room
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Students

StudNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Lectures

Lecture Nr	Title	Weekly Hours	Given_by
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

attend

StudNr	LectureNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
25403	5022
29555	5022
29555	5001

require

Predecessor	Successor
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Assistants

PersNr	Name	Area	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

test

StudNr	LectureNr	PersNr	Grade
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

```
CREATE TABLE Assistants (
PersNr int NOT NULL PRIMARY KI
Name varchar(255) NOT NULL,
Area varchar(255),
Boss int,
FOREIGN KEY (Boss) REFERENCES Professors(PersNr) ON DELETE CASCADE );
```

```
DELETE FROM Professors
WHERE Name = 'Kopernikus';

SELECT Name FROM Assistants;
```