Database Lab

Dr. Andreas Geppert

Fall Term 2023

Content

- Review Exercise 1: Conceptual Design
- PostgreSQL Intro
- Exercise 2: Logical Design

Topics

- conceptual design
- logical design
- consistency constraints & data manipulation
- queries
- views
- stored procedures and user-defined functions
- triggers
- security

Logical Design

- Mapping of the conceptual schema onto a relational schema
- elementary attributes and their domains
- entity type → relation
- 1:1 relation → foreign-key added to other relation
- 1:n relation → add foreign key on the n-side
- m:n relation → separate relationship table
- specialization hierarchy → single table for entire hierarchy, or one table per entity (sub/super) type
- set-valued attributes → separate relation
- structured attributes → elementary attributes

Logical Design: Special Cases (1)

- Specialization
  - not supported in all DBMSs
  - PostgreSQL: table inheritance
- JSON documents
  - not supported by all DBMSs
  - PostgreSQL: JSON and JSONB data types

Logical Design: Special Cases (2)

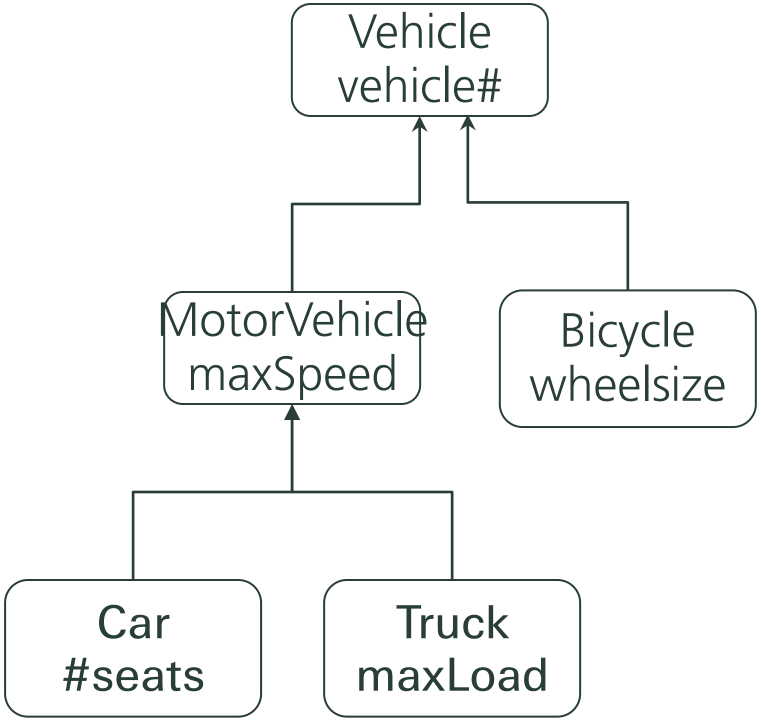
- Domains
  - not supported in all DBMSs
  - PostgreSQL: `create domain PhoneNumber as char(13)`
- Set-valued attributes
  - not supported by all DBMSs
  - PostgreSQL: array types, JSON
    `children varchar(20) array,`
  - DON'T DO ANY OF THIS:
    - `child1 varchar(20), … child9 varchar(20)`
    - `children: varchar(200) – comma-separated list of children's names`

Logical Design: Special Cases

- Enumeration data types
  - not supported by all DBMSs
  - PostgreSQL:
    ```
    create type CarType as enum (‘Limo', ‘Cabrio', ‘Van')
    ```
- Structured types
  - not supported in all DBMSs
  - violates first normal form
  - PostgreSQL: create type
    ```
    create type PhoneNumberT as (countryCode  char(3), areaCode
    char(3), ...);
    ```

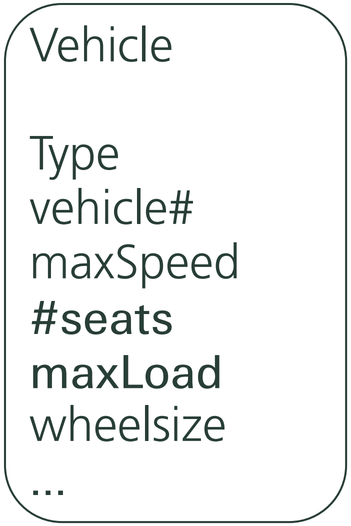
Logical Design: Specialization and Inheritance

- Conceptual models may contain specialization and inheritance between classes
- How should we map specialization onto the logical model?
  - One relation for the whole hierarchy
  - Relation per leaf class
  - Relation per class
  - Object-relational, DBMS-specific



Specialization and Inheritance: Single Relation

- The whole hierarchy is mapped onto a single relation
- All attributes defined somewhere in the hierarchy are defined for the relation
- For instances, not applicable attributes are set to null
- Additional «type» attribute

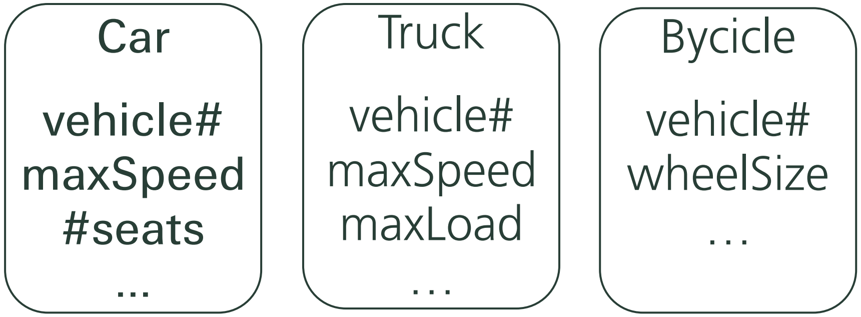


Vehicle

Type
vehicle#
maxSpeed
#seats
maxLoad
wheelsize
…

Specialization and Inheritance: Relation per Leaf

- One relation per leaf class
- All attributes defined on the path from the root to the leaf are defined for the table



Car

vehicle#
maxSpeed
#seats
...

Truck

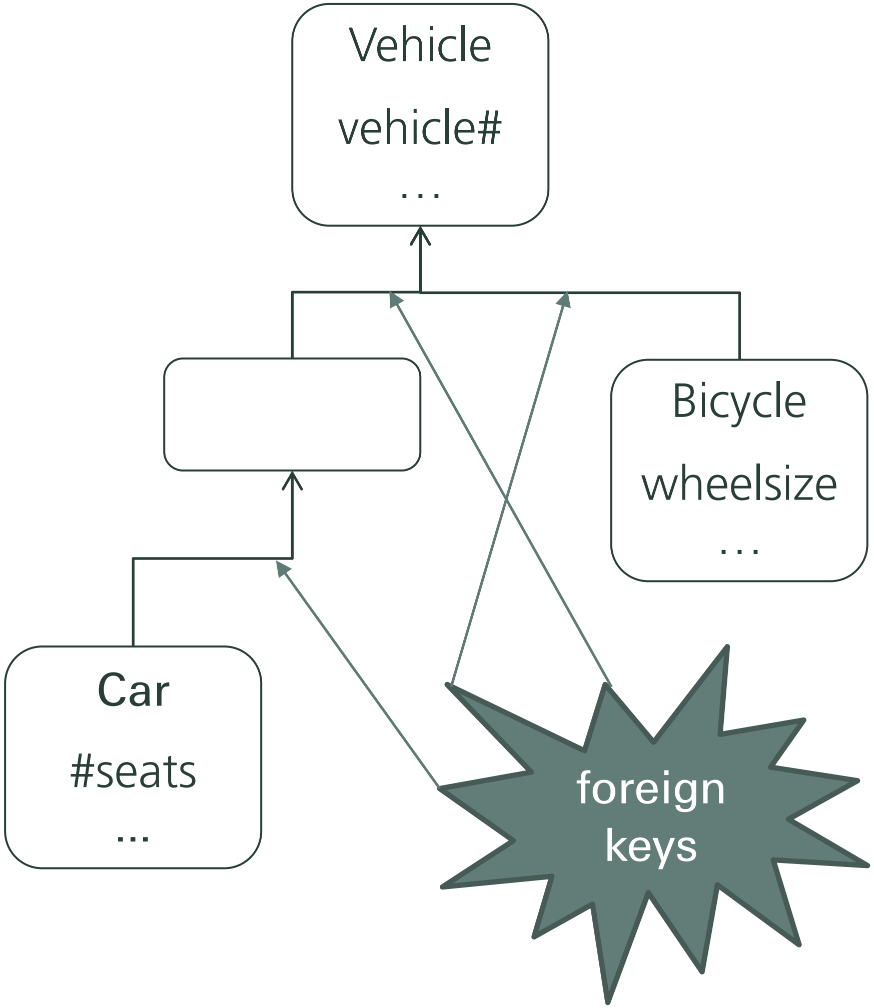
vehicle#
maxSpeed
maxLoad
...

Bycicle

vehicle#
wheelSize
...

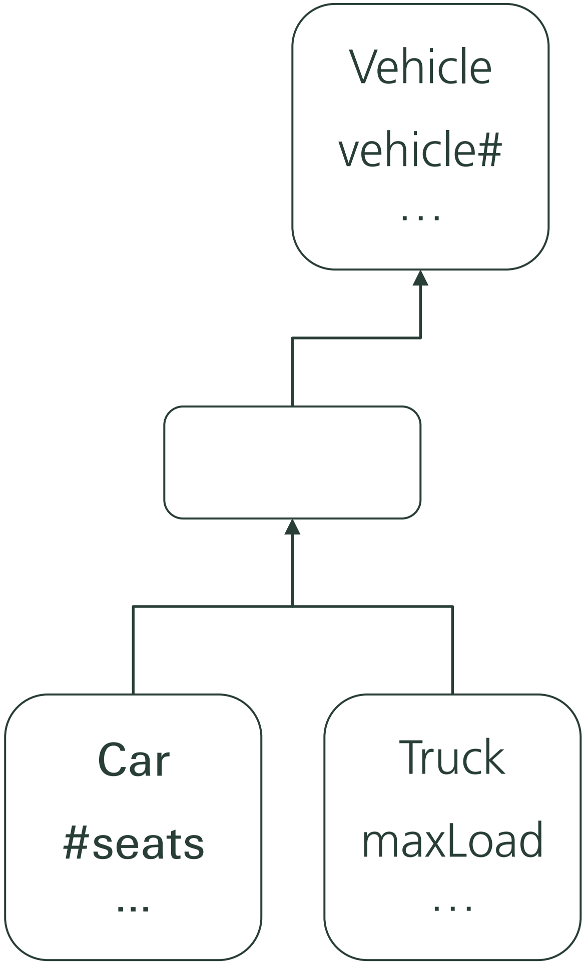
Specialization and Inheritance: Relation per Class

- One table per class in the hierarchy
- Specialization relationship is implemented using foreign key/primary key relationship



Specialization and Inheritance: Object-Relational

- Object-relational database systems (DB2, Oracle, Postgres) support type and/or table inheritance
- PostgreSQL: table inheritance
- Using «inherits» keyword
  ```
  create table Car ( … )
  inherits (MotorCar)
  ```
- very similar to inheritance in OO programming
- tables can inherit from multiple supertables


Vehicle
vehicle#
…
Car
#seats
…
Truck
maxLoad
…