# CSE2DBF – CSE4DBF

# Relational Database Model

Reading:

*Elmasri and Navathe, "Fundamentals of Database Systems,  Chapters 1 & 2", Pearson, 2016.*

**Ebook**: *https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*

# Data Model

A Data Model is a set of concepts that can be used to describe the structure of a database.

By structure of a database, we mean the data types, relationships, and constraints that should hold for the data.

Existing database models:

- **Hierarchical model**: Represents the data in a database with a tree  structure consisting of progressive parent-child relations.
- **Network model**: Represents the data in a database as interconnected nodes.
- **Relational model**: represents the data in a database as a collection of relations / tables.
- **Object-Oriented model**: represents the data, as well as operations, in a database.

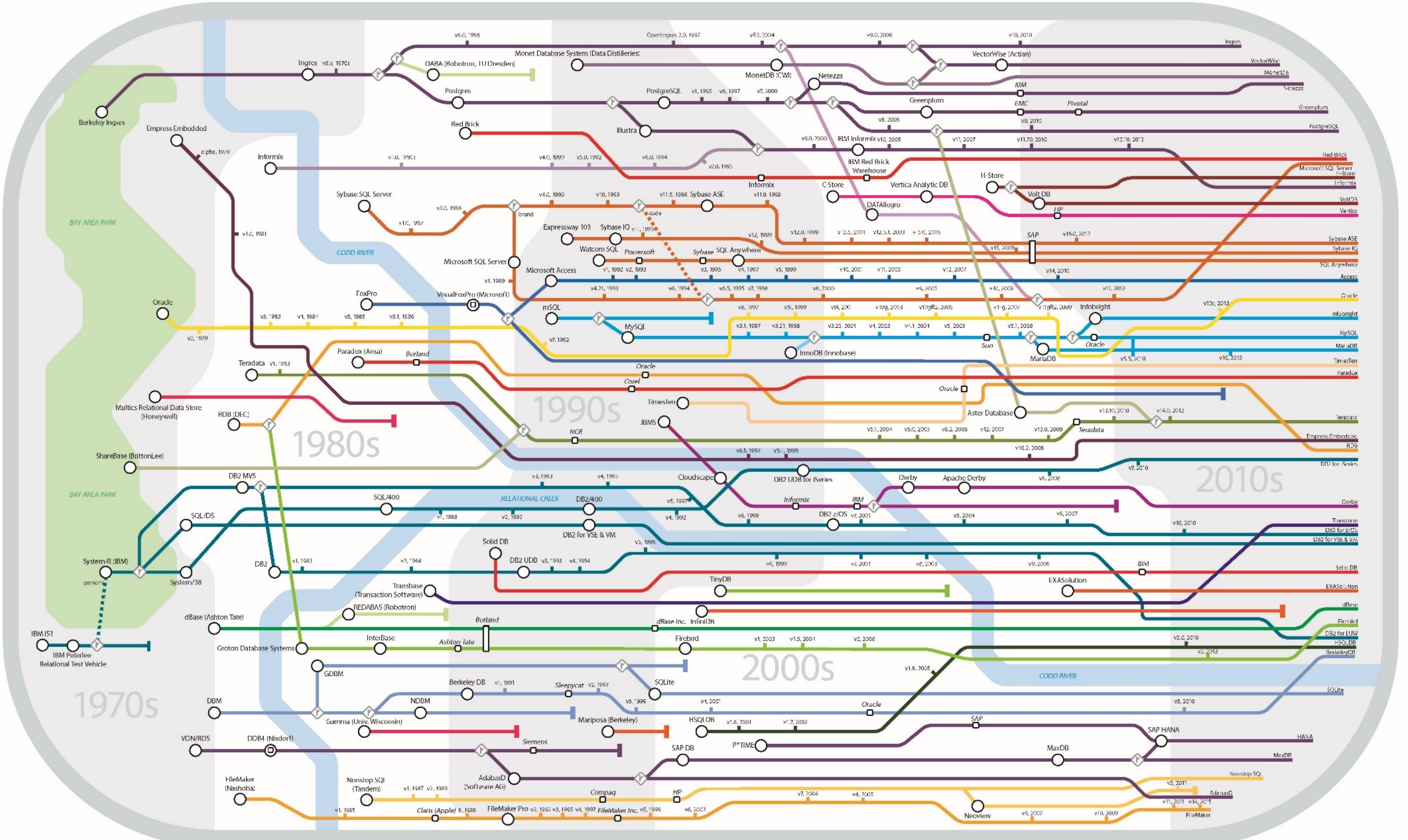
# Data Model

In this lecture, we will focus on the most prominent data model: **the Relational model**.

More than 90% of current database applications are built on relational database systems which utilise the relational model as their underlying data model.

- Introduced by E F Codd in 1970
- Simple and uniform data model
- Rests on a firm mathematical foundation (Relational Algebra)

# Genealogy of Relational Database Management Systems




Key to lines and symbols

- Publishing Date
- Acquisition
- v9, 2006CC Versions
- Discontinued
- Branch (intellectual and/or code)
- Crossing lines have no special semantics

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam


Felix Naumann, Jana Bauckmann, Claudia Exeler, Jan-Peer Rudolph, Fabian Tschirschnitz
Contact - Hasso Plattner Institut, Potsdam, Felix.Naumann@hpi.uni-potsdam.de
Design - Alexander Sandt Grafik-Design, Hamburg

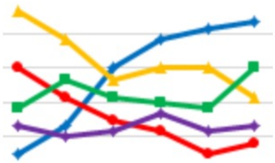

Version 5.0 - October 2015
http://www.hpi.uni-potsdam.de/naumann/projekte/rdbms_genealogy.html


http://hpi.de/fileadmin/user_upload/fachgebiete/naumann/projekte/RDBMSGenealogy/RDBMS_Genealogy_V5.jpg http://db-engines.com/en/ranking

# DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.



trend chart

410 systems in ranking, February 2023

| Rank Feb 2023 | Rank Jan 2023 | Rank Feb 2022 | DBMS | Database Model | Score Feb 2023 | Score Jan 2023 | Score Feb 2022 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1247.52 | +2.35 | -9.31 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1195.45 | -16.51 | -19.23 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 929.09 | +9.70 | -19.96 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 616.50 | +1.65 | +7.12 |
| 5. | 5. | 5. | MongoDB | Document, Multi-model | 452.77 | -2.42 | -35.88 |
| 6. | 6. | 6. | Redis | Key-value, Multi-model | 173.83 | -3.72 | -1.96 |
| 7. | 7. | 7. | IBM Db2 | Relational, Multi-model | 142.97 | -0.60 | -19.91 |
| 8. | 8. | 8. | Elasticsearch | Search engine, Multi-model | 138.60 | -2.56 | -23.70 |
| 9. | ↑ 10. | ↑ 10. | SQLite | Relational | 132.67 | +1.17 | +4.30 |
| 10. | ↓ 9. | ↓ 9. | Microsoft Access | Relational | 131.03 | -2.33 | -0.23 |
| 11. | ↑ 12. | 11. | Cassandra | Wide column | 116.22 | -0.09 | -7.76 |
| 12. | ↓ 11. | ↑ 15. | Snowflake | Relational | 115.65 | -1.60 | +32.47 |
| 13. | 13. | ↓ 12. | MariaDB | Relational, Multi-model | 96.81 | -2.55 | -10.30 |
| 14. | 14. | ↓ 13. | Splunk | Search engine | 87.08 | -1.32 | -3.73 |
| 15. | 15. | ↑ 17. | Amazon DynamoDB | Multi-model | 79.69 | -1.87 | -0.67 |
| 16. | 16. | ↓ 14. | Microsoft Azure SQL Database | Relational, Multi-model | 78.75 | -1.62 | -6.20 |

http://db-engines.com/en/ranking

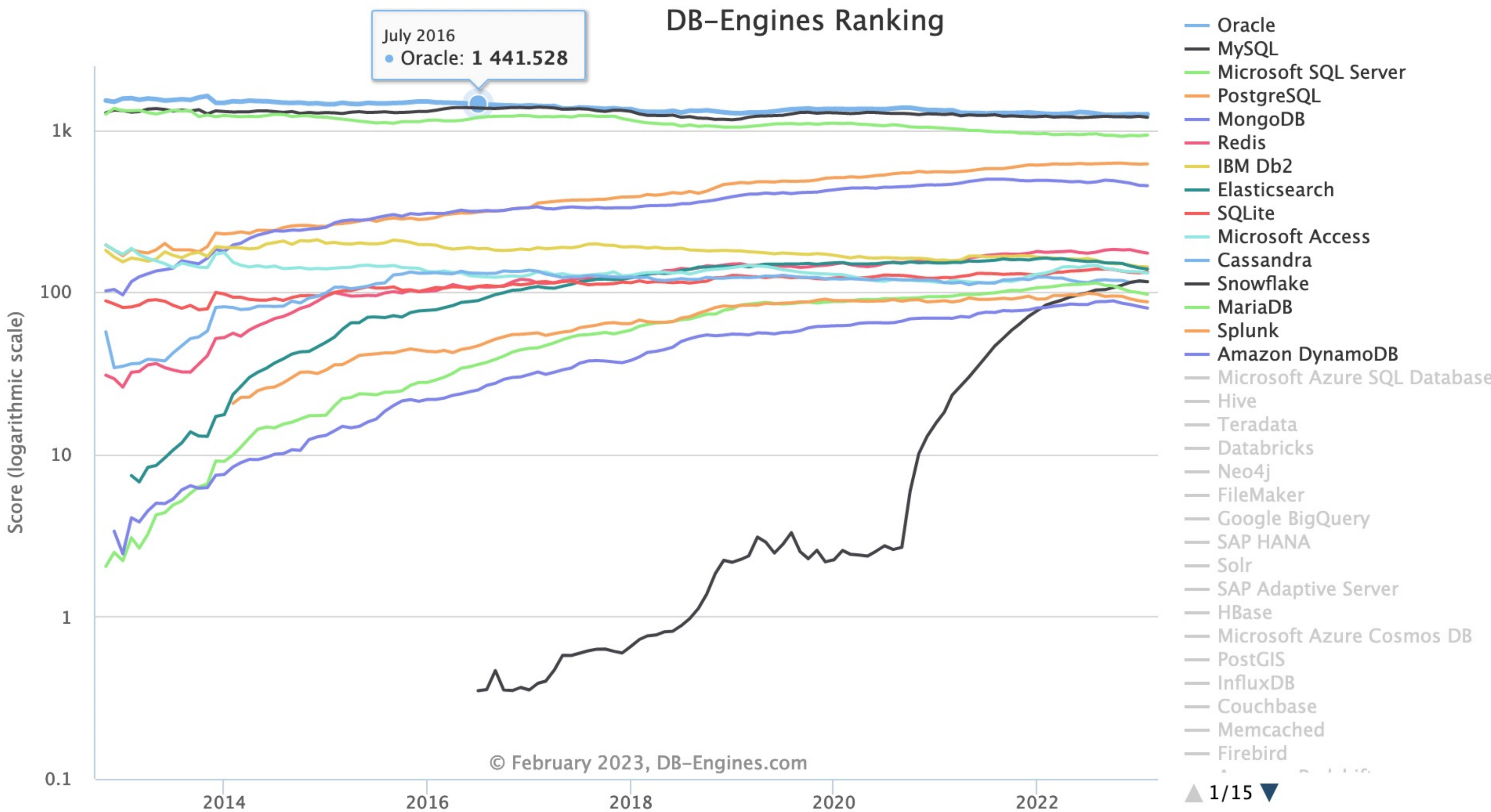
# DB-Engines Ranking - Trend Popularity

The DB-Engines Ranking ranks database management systems according to their popularity.

Read more about the method of calculating the scores.

| Rank | Trend | System | Score | Change |
|---|---|---|---|---|
| 1 | | Oracle | 1560 | + 27 |
| 2 | | MySQL | 1342 | + 47 |
| 3 | | SQL Server | 1278 | - 40 |
| 4 | | PostgreSQL | 174 | - 3 |
| 5 | | MS Access | 161 | - 8 |
| 6 | | DB2 | 155 | - 4 |

ranking table
February 2023




Click at a system in the legend
to hide or show its trend line

# Relational Model

There are 3 major aspects of a relational data model

a) Data Structure (structural part)

Consists of n-ary two dimensional (attributes, tuples) tables/relations whose attributes' domains are of **atomic types.**

b) Data Integrity (integrity part):

Consists of two general integrity rules, namely entity integrity and referential integrity.

c) Data Manipulation (manipulative part):

Consists of a set of algebraic operators for data manipulations. (This will be discussed in Topic 5)

# Relational Model

- To explain each of the components mentioned earlier, we use an example of a Tram Database

  Many different types of data are involved in running trams.

  - Vehicle: number, type, capacity, status, maintenance record, current depot, . . .
  - Route: number, start, end, tracks, . . .
  - Track: streets, sections, . . .
  - Driver: name, rating, personal details, . . .
  - Shift: driver, tram, route, times, . . .
  - Timetable: routes and times

# Relational Model

Information about each kind of thing is presented as a table. These tables need more columns to be realistic.

**VEHICLE**

| No | Type | Seats |
|---|---|---|
| 181 | W | 66 |
| 182 | W | 64 |
| 622 | LR | 102 |

**ROUTE**

| No | Start | End |
|---|---|---|
| 69 | Hawthorn | St Kilda |
| 75 | City | East Burwood |
| 48 | Pt Melbourne | North Balwyn |

**DRIVER**

| Name | Rating |
|---|---|
| Adams | A1 |
| Bernard | C1 |

**SHIFT**

| Driver | Route | Tram | Day |
|---|---|---|---|
| Bernard | 69 | 181 | Wed |
| Bernard | 69 | 181 | Thu |
| Adams | 48 | 622 | Thu |

# Relational Model

- Some consistency rules must apply to represent the real-world situations. for example:
  - Every vehicle must have a number
  - It is implicit that driver names are unique, or the data won't make sense.
  - A tram entry in the Shift table should correspond to a row in the Vehicle table.
  - Suburb entries should be valid suburbs, Day entries should be days of the week, . . .

# A. Data Structure

# Data Structure

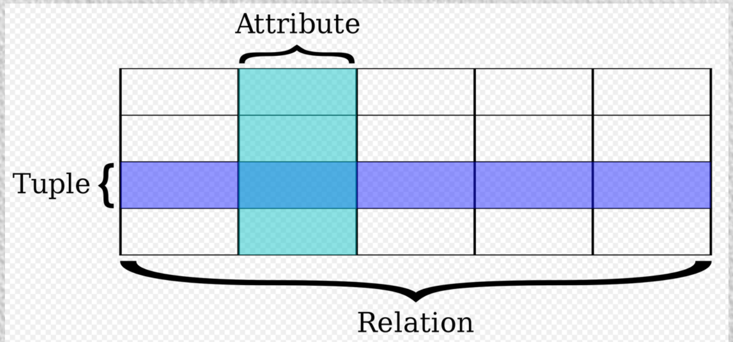
- The relational model is based on three fundamental concepts derived from set theory.

1. **Domains**

   To define the original sets of data values used to model data.

   E.g. The domain of day shift is the set of all possible days : {Mon, Tue, Wed …}, the domain of salary is the set of all floating-point numbers greater than 0 and less than 200,000 (say).

# Data Structure




2. **Relation**

   A subset of the Cartesian product of a list of domains characterised by a name.
   A relation is composed of tuples. These are represented by the rows in a table.
   Each tuple must have a primary key which can uniquely identify the tuple.

3. **Attribute**

   A column of a relation designated by name. The name associated should be meaningful.

# Data Structure

## Keys

- It is often desirable to extract a particular tuple from a relation.
  For example, with a “bank database” each client wants the balance of a particular account.

- That is, there must be ***some way of uniquely identifying each tuple***. This is usually achieved by defining some attribute to be a key, or unique label.

    - In the “tram database”, **the route number** is a key - different routes have different numbers.

    - In banks, each **account number** is a key. Name and address is not a key - there can be more than one person of the same name at the same address.

- Keys are by definition unique, so no two tuples have the same key. **More generally, a set of attributes can be a key**. For example, {Start, End} is a key for Route.

# Data Structure

## Primary Keys

The “tram” model can be now be written:

Vehicle(No., Type, Seats)

Route(No., Start, End)

Driver(Name, Rating)

Shift(Driver, Route, Tram, Day)

The attributes that make up the keys are underlined.

Primary Key

**VEHICLE**

| No | Type | Seats |
|---|---|---|
| 181 | W | 66 |
| 182 | W | 64 |
| 622 | LR | 102 |

# Data Structure

## Foreign Keys

Attribute values for one relation can be drawn from another relation.

For example, **Shift:Tram** values must be from **Vehicle:No**. (but not vice versa – vehicles are not necessarily scheduled to a shift).

Foreign Key

**VEHICLE**

| No | Type | Seats |
|---|---|---|
| 181 | W | 66 |
| 182 | W | 64 |
| 622 | LR | 102 |

**SHIFT**

| Driver | Route | Tram | Day |
|---|---|---|---|
| Bernard | 69 | 181 | Wed |
| Bernard | 69 | 181 | Thu |
| Adams | 48 | 622 | Thu |

Shift:Tram is an example of a foreign key.

# B. Data Integrity

# Data Integrity

- The data models in application systems need to include certain integrity rules with the purpose of **reflecting the reality of certain constraints** in the real world.

- Most of the integrity rules are specific, in the sense that they apply to one specific application only.

  For example, in a University Database,
  a **Student ID** must be an integer and not null.

- Two integrity rules are applicable to any application:

  **The Entity and Referential integrity constraints**.

# Data Integrity

- Entity Integrity constraints:
  - no primary key value can be null
  - the primary key must not be duplicated

Primary Key

**VEHICLE**

| No | Type | Seats |
|---|---|---|
| 181 | W | 66 |
| 182 | W | 64 |
| 622 | LR | 102 |

# Data Integrity

- Referential Integrity constraints:
  - A constraint that is specified between two relations and is used to maintain consistency among tuples of the two relations.
  - Ensuring Referential Integrity is done by defining the columns that uniquely identify table rows (primary key) and columns within the table which refer to other tables (foreign key). Appropriate actions should be taken by the system if the referred row is deleted/updated.

**VEHICLE**

| No | Type | Seats |
|---|---|---|
| 181 | W | 66 |
| 182 | W | 64 |
| 622 | LR | 102 |

**SHIFT**

Foreign Key

| Driver | Route | Tram | Day |
|---|---|---|---|
| Bernard | 69 | 181 | Wed |
| Bernard | 69 | 181 | Thu |
| Adams | 48 | 622 | Thu |

# Data Integrity

- Referential Integrity constraints (cont..):
  - On update or delete of the target of a foreign key reference (using the vehicle-shift example) which there exists at least one matching shift, the referential integrity constraints can be maintained in the following ways:
    - **RESTRICTED**: The update / delete operation is restricted to the case where there are no such matching shifts (it is rejected otherwise).
    - **CASCADE**: The update / delete operation cascades to those matching shifts also.
    - **NULLIFIES**: The foreign key is set to null in all such matching shifts and the vehicle is then updated / deleted.

# Data Integrity

Referential Integrity Constraint (Example) – when to Cascade, Restrict or Nullify?

**LECTURER**

| Lecturer ID | First Name | Last Name | Dept Number |
|---|---|---|---|
| L1 | Eric | Pardede | D1 |
| L2 | Daniel | Hodges | D2 |
| L3 | Shaskia | Ramanthan | null |

**DEPARTMENT**

| Dept Number | Dept Name | Mail Number |
|---|---|---|
| D1 | Computer Science | 39 |
| D2 | Information Science | 30 |
| D3 | Physics | 37 |
| D4 | Chemistry | 35 |

# Database Modelling using Entity-Relationship Model (E-R Model)

Reading:

*Elmasri and Navathe, "Fundamentals of Database Systems,  Chapters 1 & 2", Pearson, 2016.*

***Ebook**: https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*