A data model is a collection of concepts to define the structure and operations on data. All data models do not specify operations. It has been found useful not to consider all aspects of data at the same time. Too many details. Abstraction levels filter out the non-interesting aspects. Three abstraction levels are typical in data modeling:

1. Conceptual level (real word level):
   - The meaning of data
   - How things are related
   - What are the rules in the real world that should be respected in the database
   - Conceptual level data modeling is discussed in the Introduction to Application Analysis and Design – course

2. Structural level (logical level):
   - How the programmer or the direct user of the database sees the database
   - Structures of data
   - Operations on data
   - Different users may have different views of data
   - Programming languages may also have their own views of data

3. Physical level:
   - How is the data technically stored
   - What kind of files there are
   - Distribution
   - Structures to support fast retrieval of data (indexes)

Levels of abstraction and tasks using the levels:
- Conceptual level: information analysis, determination of contents
- Structural level: queries, programming
- Physical level: tuning, distribution
Models in use

- **conceptual level**
  - Entity-Relationship models (ER)
  - Object models
  - Semantic data models

- **structural level**
  - Relational model (current)
  - Object models (future?)
  - Hierarchic model (ancient)
  - Network model (ancient)

- **physical level**
  - Supplier dependent models

Relational databases

- Origin: E.F.Codd: The relational model of data, ACM Communications, 1970
- First commercial implementations in the end of 70ies,
- Became popular by the end of 80ies
- Became dominant by the end of 90ies
- DB2, Oracle, Informix, Sybase, MS SQL Server, etc.

The Relational Model of Data

- **The basis**:
  - Database is considered as a collection of mathematical relations
  - Few simple and well defined concepts
  - Good theoretical basis
  - Easy to comprehend using tabular representation

Relation presented as a table

<table>
<thead>
<tr>
<th>Schema name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAR</td>
<td>Reg_no</td>
</tr>
<tr>
<td>ACM-256</td>
<td>black</td>
</tr>
<tr>
<td>MAC-532</td>
<td>blue</td>
</tr>
<tr>
<td>ISO-795</td>
<td>black</td>
</tr>
<tr>
<td>OSI-228</td>
<td>red</td>
</tr>
<tr>
<td>HCI-449</td>
<td>white</td>
</tr>
</tbody>
</table>

Relation presented as a table

<table>
<thead>
<tr>
<th>CAR</th>
<th>Reg_no</th>
<th>Color</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM-256</td>
<td>black</td>
<td>1988</td>
<td></td>
</tr>
<tr>
<td>MAC-532</td>
<td>blue</td>
<td>1994</td>
<td></td>
</tr>
<tr>
<td>ISO-795</td>
<td>black</td>
<td>1992</td>
<td></td>
</tr>
<tr>
<td>OSI-228</td>
<td>red</td>
<td>1987</td>
<td></td>
</tr>
<tr>
<td>HCI-449</td>
<td>white</td>
<td>1993</td>
<td></td>
</tr>
</tbody>
</table>

Mathematical definition

- Let \( D_1, D_2, ..., D_n \) be sets of values (domains). They may overlap (have common values).
- Relation \( R \) is a set of tuples, the first value of which belongs to the first domain \( D_1 \), second to the second one \( D_2 \), etc.
- Mathematically a relation is a subset of the Cartesian product \( D_1 \times D_2 \times \cdots \times D_n \)
A tuple is a sequence of values (a1, ..., an). In tabular presentation it is a row in a table.

Cartesian product:
- Cartesian product of sets A={1,2,3} and B={a,b}, denoted as A×B, is the set that contains all such pairs (binary tuples) (x,y) where the first value (x) belongs to set A and the second value (y) belongs to set B.
- {(1,a), (1,b), (2,a), (2,b), (3,a), (3,b)}

A domain
- Is a set of atomic values (value may not be broken down into pieces), for example,
  - integers
  - personal numbers
  - strings
- Some operations presuppose that there is an order specified among the values of the domain
- All values must be atomic – no sets or collections in one value position within a tuple
- There is a special value NULL (meaning unknown or not valid) that belongs to each domain!

Attribute = name of column
- Attribute is a name that identifies a value position in the tuples of a relation.
- Each attribute is associated with an interpretation that specifies the meaning of values in that position.
  - There may be many tuple positions holding integers but they all have different meanings
- Each attribute is connected to a domain of which the values are drawn.
- There is a value in each value position of a tuple.
- The domains should be defined so that their values cover all possible values needed in the corresponding position.
  - For example, all colors

Relation schema
- The structure of a relation is specified in a relation schema
  - Names the attributes
  - Connects domains to attributes
  - Specifies the interpretations for attribute values
- Each relation schema has a name.

The most simple presentations of the schema (common in research articles and text books) gives the name of the schema and lists the attributes
R(A1, ..., Am),
where R is the name of schema and A1, ..., Am are attributes
- This form of schema assumes that the attributes are descriptive enough to reveal the interpretation
Example: Car(Reg_no, Color, ModelYear)
A relational database has only one instance of each relation schema at a time. This instance is typically referred to by the name of the schema. Relation Car = the current instance of schema Car.

A relation is a mathematical set:
- Each value in a mathematical set is unique (occurs only once). Thus the tuples of a relation are unique.
- A mathematical set is unordered.

The order of attributes in a relation schema is not significant.

A relation is a mathematical set. Each value in a mathematical set is unique (occurs only once). Thus, the tuples of a relation are unique. A mathematical set is unordered. The order of attributes in a relation schema is not significant.

Relation Car = the current instance of schema Car.

- Cardinality of a relation:
  - Number of tuples (rows)
- Degree of a relation:
  - Number of attributes
- Relational database:
  - Collection of relations
- Relational database schema:
  - Collection of the relation schemas that specify the relations of the database

Key

- The tuples in a relation are unique:
  - A tuple may be uniquely identified by its contents
  - Usually we don’t need the whole tuple for the identification, only some attribute values
  - A key is an attribute (or attribute combination) that
    - has a unique value (or value combination) in each tuple of whatever instance of the relation schema
    - is minimal so that no attribute may be taken out of the combination and the remaining attributes still satisfy the above condition (i.e. identify tuples)

- Relations:
  - Number of tuples (rows)
  - Number of attributes

- Tuple:
  - Row

- Attribute:
  - Column name

<table>
<thead>
<tr>
<th>CAR</th>
<th>Reg_no</th>
<th>Color</th>
<th>ModelYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM-256</td>
<td>black</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>MAC-352</td>
<td>blue</td>
<td>1994</td>
<td></td>
</tr>
<tr>
<td>ISO-295</td>
<td>black</td>
<td>1992</td>
<td></td>
</tr>
<tr>
<td>OSI-228</td>
<td>red</td>
<td>1987</td>
<td></td>
</tr>
<tr>
<td>HCI-449</td>
<td>white</td>
<td>1993</td>
<td></td>
</tr>
</tbody>
</table>

A tuple may be uniquely identified by its contents. Usually, we don’t need the whole tuple for the identification, only some attribute values. A key is an attribute (or attribute combination) that:
- has a unique value (or value combination) in each tuple of whatever instance of the relation schema
- is minimal so that no attribute may be taken out of the combination and the remaining attributes still satisfy the above condition (i.e. identify tuples)

Reg_no and ModelYear, both alone, identify rows in this instance. ModelYear would not anyhow identify rows in whatever valid instance – that would mean that there can be only one car for model year.
Key

- An attribute that belongs to a key may not have null values
- Sometimes we may find out many keys for a relation schema
- One of these keys must be selected as a primary key (the primary way to refer to the tuples of this relation schema)
- Employee(personal_number,..., employee_number)
  - Both employee_number and personal_number are keys – we select as the primary key the one that suits better for the purpose. There are restrictions for the use of personal number, thus employee number is a better choice.

Key

- The primary key is expressed in the relation schema by underlining its attributes

- Car (RegNo, Color, ModelYear)
- Employee (EmployeeNumber,...)
- TennisCourtReservation (CourtID, StartingTime, Duration, PlayerName)

Foreign key

- A database consists of many relations
- Typically there are connections between tuples of different relations
  - A tuple in Employee relation is connected to a tuple in Department relation indicating that the employee works on that department
  - A tuple in a library's Loan relation is connected to a tuple in relation Book and a tuple in relation Customer.
- In relational databases tuples are connected by including the primary key attribute values of the tuple to be connected into the tuple that establishes the connection.

Foreign key

- An attribute or a combination of attributes that establishes a connection is called a foreign key.
- Connections are established using the values of the primary key attributes of the tuple to be connected as the values of the foreign key attributes.
- Null values may be allowed for foreign keys – then the tuples need not be connected to any other tuples

© Harri Laine
Foreign key values are used in referring from one tuple to another. However, the existence of the connection may be utilized also in the other direction:
- Which department is connected to a certain employee
- Which employees are connected to a certain department

Tuples with foreign key
Tuples referred to

Foreign key
- In this course we show the foreign keys in a relation schema using an arrow in the following way:
  Ownership(PersonID \(\rightarrow\) Person, CarID \(\rightarrow\) Car)
  - PersonID refers to Person, CarID refers to Car
  - In this relation the foreign keys are part of the primary key. Thus they may not have null values.
  - Foreign keys may be outside the primary keys.
  - Then they may or may not have null values
    EMP(EmpNo, ..., DeptNo \(\rightarrow\) DEPT)
    - DeptNo refers to a tuple of DEPT

Foreign key
- The foreign key that refers to ExerciseGroup consists of two attributes. This indicates that the primary key of ExerciseGroup also consists of two attributes.

Foreign key
- Referential integrity:
  - It is not allowed to refer to non-existing tuples, i.e., the foreign keys may have as their values only values that exist as primary key values in their own relation.
  - If NULL is used as the foreign key value the tuple is not connected to anything.
The relational model

A graphical database schema

- Customer (customerID, name, address)
- Collateral (receiptNo, type, age, value)
- Loan (loanID, amount, customer)
- Payment plan (planID, planNo, interest, acceptedBy)
- Evaluation (receiptNo, officerID)
- Loan (loanID, installment)
- Installment (installment)
- Disbursement (datePaid)