The data structures of the relational model

- Attributes and domains
- Relation schemas and database schemas
- Relations and databases
- First normal form (1NF)

Running Example Database

Pubs-Drinkers-DB:

- Pubs (name, location)
- Drinkers (name, location)
- Sells (pub, beer, price)
- Visits (drinker, pub)

- each pub has a name and location
- each drinker has a name and location where they live
- pubs sell beers at various prices
- drinkers visit various pubs

Running Example Tables

<table>
<thead>
<tr>
<th>Pub</th>
<th>Beer</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse and Hound</td>
<td>Bad Habit</td>
<td>1.50</td>
</tr>
<tr>
<td>Horse and Hound</td>
<td>Rampant Ram</td>
<td>2.00</td>
</tr>
<tr>
<td>Hound and Hare</td>
<td>Shining Wit</td>
<td>2.75</td>
</tr>
<tr>
<td>Hound and Hare</td>
<td>Rampant Ram</td>
<td>2.50</td>
</tr>
<tr>
<td>March Hare</td>
<td>Bad Habit</td>
<td>1.75</td>
</tr>
<tr>
<td>March Hare</td>
<td>Rampant Ram</td>
<td>2.50</td>
</tr>
<tr>
<td>Black Horse</td>
<td>Bad Habit</td>
<td>2.50</td>
</tr>
<tr>
<td>Black Horse</td>
<td>Shining Wit</td>
<td>2.25</td>
</tr>
<tr>
<td>Black Horse</td>
<td>Rampant Ram</td>
<td>2.50</td>
</tr>
<tr>
<td>White Horse</td>
<td>Rampant Ram</td>
<td>2.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drinker</th>
<th>Pub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Black Horse</td>
</tr>
<tr>
<td>Bob</td>
<td>Hound and Hare</td>
</tr>
<tr>
<td>Carol</td>
<td>Horse and Hound</td>
</tr>
<tr>
<td>Dave</td>
<td>White Horse</td>
</tr>
<tr>
<td>Eve</td>
<td>March Hare</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visits:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Carol</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Eve</td>
</tr>
</tbody>
</table>
Attributes and Domains

- Names such as “beer” and “price” are known as attributes.
- Each attribute \( A \) has values drawn from a domain, denoted \( \text{DOM}(A) \).
- In practice, this domain will be specified by a type, such as string or integer.

Relation Schemas

A relation schema (or table header) \( R \) has the following components:

- A relation symbol \( R \), which is the name of the schema.
- A set of attributes (or column headers), denoted by \( \text{schema}(R) \).

E.g. \( \text{Visits} \) is a relation symbol, \( \text{schema(Visits)} = \{ \text{drinker, pub} \} \).

Database Schemas

A database schema \( R \) is a collection \( \{ R_1, \ldots, R_n \} \) of relation schemas (table headers).

E.g. The database schema of the Pubs-Drinkers-DB is \( \{ \text{Drinkers, Pubs, Sells, Visits} \} \).

Notation.
\( \text{schema}(R) \) is the union of all \( \text{schema}(R_i) \) such that \( R_i \) is in \( R \).

E.g. \( \text{schema(Pubs-Drinkers-DB)} = \{ \text{name, location, pub, beer, price, drinker} \} \).

First Normal Form Relation Schema

- A relation schema \( R \) is in First Normal Form (1NF) if all the domains of attributes \( A_i \) in \( \text{schema}(R) \) are atomic.
  (i.e. non-decomposable by the DBMS.)
- A database schema \( R \) is in 1NF if all the relation schemas \( R_i \) in \( R \) are in 1NF.
From now on we will assume that database schemas are in 1NF.

The justification for this assumption is:

1. The semantics of 1NF are easy to understand (e.g. ST_NO, ST_NAME and CITY vs ADDRESS).
2. 1NF makes it easier to formalise the relational model; flat relations (Visits table) vs. nested relations (Visits non-1NF table).
3. 1NF makes querying simpler too.

Relations and Databases

A tuple (or row) over \( R \), with schema(\( R \)) = \{A\_1, \ldots, A\_m\} is a sequence of values

\[(a_1, a_2, \ldots, a_m)\]

where each \( a_i \) is a value from \( \text{DOM}(A_i) \).

A relation (or table) over \( R \) is a finite set of tuples over \( R \).

A database \( d \) over \( R \) is a collection \( \{r_1, \ldots, r_n\} \) of relations \( r_i \) over \( R_i \).

Synonymous terminology

- relation schema = table header
- database schema = database headers
- relation = table
- database = database tables
- attribute = column header
- attribute value = table cell
- tuple = row
Summary of the properties of relations

- Relation names in a database are distinct.
- Attribute names in a relation are distinct.
- The order of attributes and tuples in a relation is not important.
- No two tuples in a relation are the same, i.e. a relation does not contain duplicate rows.
- Attribute values are atomic.

Null Values

We must allow for missing or incomplete information by allowing null values as place holders for non-null constants.

E.g. An Employee’s address is unknown.
E.g. An Employee’s spouse does not exist.

The special place holder null will be used as a null value.
E.g. if we want to represent Alice in the Drinkers table but do not know where she lives, we can store the row (Alice, null).

Projection

Let \( R \) be a relation schema, \( X \) be a subset of \( \text{schema}(R) \) (i.e., a set of attributes) and \( t \) be a tuple over \( R \).

The projection of \( t \) onto \( X \), denoted by \( t[X] \), is the collection of values of \( t \) for the attributes in \( X \), i.e. the restriction of \( t \) to \( X \).

E.g. if \( t = (\text{March Hare}, \text{Bad Habit}, 1.75) \) is a tuple over \text{Sells}, then

\[
\begin{align*}
\text{t[pub]} &= (\text{March Hare}) \\
\text{t[beer, price]} &= (\text{Bad Habit}, 1.75).
\end{align*}
\]

Superkeys

Definition of a superkey for \( R \).

A subset \( S \) of \( \text{schema}(R) \) is a superkey for \( R \) if for all relation instances \( r \) of \( R \), the projection \( t[S] \) of any tuple \( t \) over \( R \) uniquely identifies a single tuple in \( r \).

This must hold for all time.
**Keys**

**Definition of a key for R.** A (candidate) key for a relation schema \( R \) is a superkey for \( R \) having a minimal number of attributes.

**Definition of primary key of R.** A primary key for \( R \) is one of the candidate keys, which is designated by the database designer as being primary.

**Question.** What are the candidate keys for the Pubs-Drinkers DB tables?

Using the value of a primary key for a table ensures that only one row (entity) will be retrieved from the table, i.e., a single row is identified.

---

**Foreign Keys**

Let \( R \) be a database schema, \( R_1, R_2 \) be relation schemas in \( R \) and assume that \( K \) is the primary key of \( R_2 \).

**Definition of a foreign key.**

\( F \), a subset of \( R_1 \), is a foreign key for \( R_1 \) referencing the primary key \( K \) of \( R_2 \) if the following condition holds:

for all database instances \( d = \{ r_1, r_2, \ldots, r_n \} \) of \( R \) and for all tuples \( t_1 \) in \( r_1 \), if \( t_1[F] \) does not contain any null values, then there exists a tuple \( t_2 \) in \( r_2 \) such that \( t_1[F] = t_2[K] \).

Foreign keys are declared by the database designer or DBA.

---

**First Fundamental Integrity Constraint**

(of the relational model)

Let \( K \) be the primary key of some relation schema \( R \).

**Definition of entity integrity.** Primary key values \( t[K] \) of tuples \( t \) in relations over \( R \) should not contain null values.

This is because, since the value of \( null \) is unknown, it could be equal to some other key value which would mean the key was no longer unique.

---

**Second Fundamental Integrity Constraint**

(of the relational model)

Let \( F \) be a foreign key for \( R_2 \) referencing \( K \) in \( R_1 \).

**Definition of referential integrity.** If the foreign key values \( t[F] \) of a tuple \( t \) in a relation over \( R_2 \) are all non-null, then \( t[F] \) are primary key values for \( K \) in the referenced relation over \( R_1 \).

Referential integrity ensures that references to values do not become “dangling”.

**Question.** What are the foreign keys for the Pubs-Drinkers DB tables?
Translating from ER Diagrams to Database Schemas

Essentially

- each entity type becomes a relation schema
- each relationship type becomes a relation schema
- weak entity types need special treatment
- ISA relationship types need special treatment
- sometimes it makes sense to combine relations

Modelling Entity Types in the Relational Model

An entity type $E$, having attributes $A_1, \ldots, A_m$, is modelled by a relation schema $R$, with $\text{schema}(R) = \{A_1, \ldots, A_m\}$.

- The primary key of $R$ is the primary key of $E$.

Modelling Relationship Types in the Relational Model

A many-to-many relationship type $M$ involving entity types $E_1, \ldots, E_m$ is modelled by a relation schema $R$, with $\text{schema}(R) = (K_1, \ldots, K_m)$, where $K_1, \ldots, K_m$ are the primary keys of the entity types $E_1, \ldots, E_m$.

- If $M$ has attributes, these are added to $\text{schema}(R)$.
- The primary key of $R$ is $(K_1, \ldots, K_m)$.

Question. What happens if a relationship type is many-to-one or one-to-one?
**Answer.** No new relation schemas need to be defined but foreign keys need be present in the appropriate relation schemas.

Assume that we have
- a many-to-one relationship type from $E_2$ to $E_1$
- the primary key of $E_1$ is $K_1$
- relation schemas $R_1$ modelling $E_1$ and $R_2$ modelling $E_2$

Then $K_1$ must be included in $\text{schema}(R_2)$.

- $K_1$ is a foreign key in $R_2$ referencing $K_1$ in $R_1$.

---

**Modelling Weak Entities in the Relational Model**

Assume that, in an ERD, we have
- a weak entity type $E$
- $E$ has discriminator $D$
- the other attributes of $E$ are $A_1, \ldots, A_m$
- the owner entity type of $E$ has primary key $K$

Then $E$ is modelled by a relation schema $R$, with $\text{schema}(R) = \{K, D, A_1, \ldots, A_m\}$.

- The primary key of $R$ is $(K, D)$.

---

**Modelling ISA Relationship Types in the Relational Model**

Assume that, in an ERD, we have
- entity type $E_1$ with attributes $A_1, \ldots, A_m$
- entity type $E_2$ with attributes $B_1, \ldots, B_n$
- an ISA relationship type from $E_1$ to $E_2$

Then $E_1$ can be modelled by a relation schema $R$, with $\text{schema}(R) = \{A_1, \ldots, A_m, B_1, \ldots, B_n\}$.

- The primary key of $R$ is the same as that of $E_2$.

---

**For more information**

See, e.g.,
- Chapter 4 of [Connolly and Begg].
- Chapters 2 and 7 of [Silberschatz et al.].
- Chapters 2 and 4 of [Ullman and Widom].