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

Pubs:

<table>
<thead>
<tr>
<th>name</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse and Hound</td>
<td>Bloomsbury</td>
</tr>
<tr>
<td>Hound and Hare</td>
<td>Islington</td>
</tr>
<tr>
<td>March Hare</td>
<td>Bloomsbury</td>
</tr>
<tr>
<td>Black Horse</td>
<td>Islington</td>
</tr>
<tr>
<td>White Horse</td>
<td>Bloomsbury</td>
</tr>
</tbody>
</table>

<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>

Sells:

<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>
</tbody>
</table>

Visits:

<table>
<thead>
<tr>
<th>drinker</th>
<th>pub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Hound and Hare</td>
</tr>
<tr>
<td>Bob</td>
<td>White Horse</td>
</tr>
<tr>
<td>Carol</td>
<td>March Hare</td>
</tr>
<tr>
<td>Dave</td>
<td>Hound and Hare</td>
</tr>
<tr>
<td>Eve</td>
<td>March Hare</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 $DOM(A)$
- In practice, this domain will be specified by a type, such as em 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 $schema(R)$.

E.g. Visits is a relation symbol, schema(Visits) = { 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 $\{Drinkers, Pubs, Sells, Visits\}$.

Notation. $schema(R)$ is the union of all $schema(R_i)$ such that $R_i$ is in $R$.

E.g. schema(Pubs-Drinkers-DB) = { 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 $schema(R)$ 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.
A non-1NF version of the Visits table

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

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. ADDRESS vs. ST_NO, ST_NAME and CITY).
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 member of

$$\text{DOM}(A_1) \times \ldots \times \text{DOM}(A_m),$$

where $\times$ is the Cartesian product operator.

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

A database $d$ over $R$ is a collection $\{n_1, \ldots, r_n\}$ of relations $r_i$ over $R_i$.

Cartesian Product Example

All possible Drinker and Beer values:

<table>
<thead>
<tr>
<th>Drinker</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bad Habit</td>
</tr>
<tr>
<td>Bob</td>
<td>Rampant Ram</td>
</tr>
<tr>
<td>Carol</td>
<td>Shining Wit</td>
</tr>
</tbody>
</table>

Drinker $\times$ Beer
**Drinker** × **Beer** represents all possible combinations of **Drinker** and **Beer** values, i.e., all possible tuples (rows).

A relation called **Likes**, representing drinkers and the beers they like, will be a subset of **Drinker** × **Beer**, e.g.:

<table>
<thead>
<tr>
<th>Drinker</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Shining Wit</td>
</tr>
<tr>
<td>Bob</td>
<td>Bad Habit</td>
</tr>
<tr>
<td>Bob</td>
<td>Shining Wit</td>
</tr>
<tr>
<td>Carol</td>
<td>Rampant Ram</td>
</tr>
</tbody>
</table>

The contents of the relation could change over time.

**First Normal Form (1NF)**

Relations over 1NF relation schemas are called **1NF relations** (or **flat relations** or simply relations).

1NF relations are advantageous since they have

1. A simple tabular representation.
2. Simple query languages.
3. A simple set of fundamental integrity constraints.

### 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 $t = \text{\{March Hare, Bad Habit, 1.75\}}$ is a tuple over Sells, then $t[\text{pub}] = \text{\{March Hare\}}$ and $t[\text{beer, price}] = \text{\{Bad Habit, 1.75\}}$.

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 $r[S]$ 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.
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?

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.

Translating from ER Diagrams to Database Schemas

Essentially

- each entity type becomes a relation schema
- each relationship type becomes a relation schema but
- 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 $E$.

Question. What happens if a relationship type is many-to-one or one-to-one?

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 $(K_1, \ldots, K_m)$.

Answer. No new relation schemas need to 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 $(K, D)$.

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].

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 $E_2$. 