Chapter 4 - Objectives

- Terminology of relational model.
- How tables are used to represent data.
- Connection between mathematical relations and relations in the relational model.
- Properties of database relations.
- How to identify CK, PK, and FKS.
- Meaning of entity integrity and referential integrity.
- Purpose and advantages of views.
Relational Model Terminology

◆ A relation is a table with columns and rows.
  – Only applies to logical structure of the database, not the physical structure.

◆ Attribute is a named column of a relation.

◆ Domain is the set of allowable values for one or more attributes.
Relational Model Terminology

- Tuple is a row of a relation.
- Degree is the number of attributes in a relation.
- Cardinality is the number of tuples in a relation.
- Relational Database is a collection of normalized relations with distinct relation names.
Instances of Branch and Staff Relations

Branch

<table>
<thead>
<tr>
<th>branchNo</th>
<th>street</th>
<th>city</th>
<th>postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>B005</td>
<td>22 Deer Rd</td>
<td>London</td>
<td>SW1 4EH</td>
</tr>
<tr>
<td>B007</td>
<td>16 Argyll St</td>
<td>Aberdeen</td>
<td>AB2 3SU</td>
</tr>
<tr>
<td>B003</td>
<td>163 Main St</td>
<td>Glasgow</td>
<td>G11 9QX</td>
</tr>
<tr>
<td>B004</td>
<td>32 Manse Rd</td>
<td>Bristol</td>
<td>BS99 1NZ</td>
</tr>
<tr>
<td>B002</td>
<td>56 Clover Dr</td>
<td>London</td>
<td>NW10 6EU</td>
</tr>
</tbody>
</table>

Staff

<table>
<thead>
<tr>
<th>staffNo</th>
<th>fName</th>
<th>lName</th>
<th>position</th>
<th>sex</th>
<th>DOB</th>
<th>salary</th>
<th>branchNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL21</td>
<td>John</td>
<td>White</td>
<td>Manager</td>
<td>M</td>
<td>1-Oct-45</td>
<td>30000</td>
<td>B005</td>
</tr>
<tr>
<td>SG37</td>
<td>Ann</td>
<td>Beech</td>
<td>Assistant</td>
<td>F</td>
<td>10-Nov-60</td>
<td>12000</td>
<td>B003</td>
</tr>
<tr>
<td>SG14</td>
<td>David</td>
<td>Ford</td>
<td>Supervisor</td>
<td>M</td>
<td>24-Mar-58</td>
<td>18000</td>
<td>B003</td>
</tr>
<tr>
<td>SA9</td>
<td>Mary</td>
<td>Howe</td>
<td>Assistant</td>
<td>F</td>
<td>19-Feb-70</td>
<td>9000</td>
<td>B007</td>
</tr>
<tr>
<td>SG5</td>
<td>Susan</td>
<td>Brand</td>
<td>Manager</td>
<td>F</td>
<td>3-Jun-40</td>
<td>24000</td>
<td>B003</td>
</tr>
<tr>
<td>SL41</td>
<td>Julie</td>
<td>Lee</td>
<td>Assistant</td>
<td>F</td>
<td>13-Jun-65</td>
<td>9000</td>
<td>B005</td>
</tr>
</tbody>
</table>
Examples of Attribute Domains

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Domain Name</th>
<th>Meaning</th>
<th>Domain Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>branchNo</td>
<td>BranchNumbers</td>
<td>The set of all possible branch numbers</td>
<td>character: size 4, range B001–B999</td>
</tr>
<tr>
<td>street</td>
<td>StreetNames</td>
<td>The set of all street names in Britain</td>
<td>character: size 25</td>
</tr>
<tr>
<td>city</td>
<td>CityNames</td>
<td>The set of all city names in Britain</td>
<td>character: size 15</td>
</tr>
<tr>
<td>postcode</td>
<td>Postcodes</td>
<td>The set of all postcodes in Britain</td>
<td>character: size 8</td>
</tr>
<tr>
<td>sex</td>
<td>Sex</td>
<td>The sex of a person</td>
<td>character: size 1, value M or F</td>
</tr>
<tr>
<td>DOB</td>
<td>DatesOfBirth</td>
<td>Possible values of staff birth dates</td>
<td>date, range from 1-Jan-20, format dd-mmm-yy</td>
</tr>
<tr>
<td>salary</td>
<td>Salaries</td>
<td>Possible values of staff salaries</td>
<td>monetary: 7 digits, range 6000.00–40000.00</td>
</tr>
</tbody>
</table>
# Alternative Terminology for Relational Model

<table>
<thead>
<tr>
<th>Formal terms</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td>Table</td>
<td>File</td>
</tr>
<tr>
<td>Tuple</td>
<td>Row</td>
<td>Record</td>
</tr>
<tr>
<td>Attribute</td>
<td>Column</td>
<td>Field</td>
</tr>
</tbody>
</table>
Consider two sets, $D_1$ & $D_2$, where $D_1 = \{2, 4\}$ and $D_2 = \{1, 3, 5\}$.

Cartesian product, $D_1 \times D_2$, is set of all ordered pairs, where first element is member of $D_1$ and second element is member of $D_2$.

$$D_1 \times D_2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$$

Alternative way is to find all combinations of elements with first from $D_1$ and second from $D_2$. 
Mathematical Definition of Relation

- Any subset of Cartesian product is a relation; e.g.
  \[ R = \{(2, 1), (4, 1)\} \]

- May specify which pairs are in relation using some condition for selection; e.g.
  - second element is 1:
    \[ R = \{(x, y) \mid x \in D_1, y \in D_2, \text{ and } y = 1\} \]
  - first element is always twice the second:
    \[ S = \{(x, y) \mid x \in D_1, y \in D_2, \text{ and } x = 2y\} \]
Mathematical Definition of Relation

Consider three sets $D_1$, $D_2$, $D_3$ with Cartesian Product $D_1 \times D_2 \times D_3$; e.g.

\[
D_1 = \{1, 3\} \quad D_2 = \{2, 4\} \quad D_3 = \{5, 6\}
\]

\[
D_1 \times D_2 \times D_3 = \{(1,2,5), (1,2,6), (1,4,5), (1,4,6), (3,2,5), (3,2,6), (3,4,5), (3,4,6)\}
\]

Any subset of these ordered triples is a relation.
Mathematical Definition of Relation

- Cartesian product of $n$ sets $(D_1, D_2, \ldots, D_n)$ is:

$$D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$$

usually written as:

$$\prod_{i=1}^{n} D_i$$

- Any set of $n$-tuples from this Cartesian product is a relation on the $n$ sets.
Database Relations

◆ Relation schema
  – Named relation defined by a set of attribute and domain name pairs.

◆ Relational database schema
  – Set of relation schemas, each with a distinct name.
Properties of Relations

- Relation name is distinct from all other relation names in relational schema.

- Each cell of relation contains exactly one atomic (single) value.

- Each attribute has a distinct name.

- Values of an attribute are all from the same domain.
Properties of Relations

- Each tuple is distinct; there are no duplicate tuples.

- Order of attributes has no significance.

- Order of tuples has no significance, theoretically.
Relational Keys

◆ Superkey
  – An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

◆ Candidate Key
  – Superkey (K) such that no proper subset is a superkey within the relation.
  – In each tuple of R, values of K uniquely identify that tuple (uniqueness).
  – No proper subset of K has the uniqueness property (irreducibility).
Relational Keys

◆ Primary Key
  – Candidate key selected to identify tuples uniquely within relation.

◆ Alternate Keys
  – Candidate keys that are not selected to be primary key.

◆ Foreign Key
  – Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.
Integrity Constraints

◆ Null

- Represents value for an attribute that is currently unknown or not applicable for tuple.
- Deals with incomplete or exceptional data.
- Represents the absence of a value and is not the same as zero or spaces, which are values.
Integrity Constraints

◆ Entity Integrity
  – In a base relation, no attribute of a primary key can be null.

◆ Referential Integrity
  – If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.
Integrity Constraints

- **General Constraints**
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.
Views

◆ Base Relation
  – Named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.

◆ View
  – Dynamic result of one or more relational operations operating on base relations to produce another relation.
Views

◆ A virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.

◆ Contents of a view are defined as a query on one or more base relations.

◆ Views are dynamic, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.
Purpose of Views

- Provides powerful and flexible security mechanism by hiding parts of database from certain users.

- Permits users to access data in a customized way, so that same data can be seen by different users in different ways, at same time.

- Can simplify complex operations on base relations.
Updating Views

- All updates to a base relation should be immediately reflected in all views that reference that base relation.

- If view is updated, underlying base relation should reflect change.
Updating Views

There are restrictions on types of modifications that can be made through views:

- Updates are allowed if query involves a single base relation and contains a candidate key of base relation.
- Updates are not allowed involving multiple base relations.
- Updates are not allowed involving aggregation or grouping operations.
Updating Views

- Classes of views are defined as:
  - theoretically not updateable;
  - theoretically updateable;
  - partially updateable.