A relational database is a collection of relations.
A relation is a 2-dimensional table, in which each row represents a collection of related data values.
Rows of a relation are called tuples.
The actual values in a relation can be interpreted as the facts describing an instance of an entity or a relationship.

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>Maria Macarena</td>
<td>111-22-3333</td>
</tr>
<tr>
<td>Juan Valdez</td>
<td>123-45-6789</td>
</tr>
<tr>
<td>V. Sundarabatharan</td>
<td>999-88-7777</td>
</tr>
<tr>
<td>Shi-Wua Yan</td>
<td>881-99-0101</td>
</tr>
<tr>
<td>Bart Simpson</td>
<td>777-12-3456</td>
</tr>
</tbody>
</table>

Characteristics of Relations

- Relations are defined as a (mathematical) set of tuples.
- Duplicate tuples are not allowed.
- Ordering of tuples inside a relation is immaterial.
- Ordering of values within a tuple is irrelevant, therefore column ordering is not important.
- Each value in a tuple is atomic (not divisible) and matches a simple data type (numbers, strings, dates, etc.).
- Recent research is oriented to removing the atomicity of First Normal Form databases to accommodate complex objects.

Domains, Tuples, Attributes and Relations

- A domain D is a set of atomic values. A domain is given a name, data type and format.
- A relational schema R, denoted by R(A₁, A₂, … Aₙ), is a set of attributes (column names). The degree of a relation is the number of attributes of its relation scheme.
- Each attribute Aᵢ is the name of a role played by some domain D in R(A₁, A₂, … Aₙ). D is the domain of Aᵢ and is denoted by dom(Aᵢ).
- A relation r defined on a schema R(A₁, A₂, … Aₙ), also denoted by r(R), is a set of n-tuples r={t₁, t₂, … tₘ}.
- Each n-tuple t is an ordered list of n values t=<v₁, v₂, … vₙ>, where each value vᵢ is an element of dom(Aᵢ) or a special null value.
- A relation r(R) is a subset of the cartesian product of the domains dom(A₁) that define R. Therefore r(R) ⊆ dom(A₁) x dom(A₂) x … x dom(Aₙ).

Key Attributes of a Relation

- A superkey SK of a relation r(R) is a group of attributes which uniquely identifies all the other attributes of r(R).
- A key K of a relational schema R is a minimal superkey of R.
- A relational schema R may have more than one key. Each is called a candidate key.
- It is common to select one of the candidate keys and elevate it to primary key.
- Convention: The attributes in the primary key of schema R are underlined.
Examples: EMPLOYEE(SSN, Name, Address, Salary)
STOCK(PartNum, SupNum, Quantity)
Integrity Constraints

- *Integrity constraints* are rules specified on the database and are expected to hold on *every* instance of that schema.
- *Key constraints* specify the candidate keys of each relation scheme R.
- *Entity integrity constraints* state that no primary key value can be null.
- *Referential integrity constraints* are specified between two tables and are used to maintain the consistency among tuples of the two relations. *Foreign key(s)* of one relation are used to refer to the primary key values in the other relation.

Relational Algebra

- Collection of operators which are used to manipulate entire relations.
- The result of each operation is a new relation.
- Consists of two groups: operations on sets and operations specifically designed to manipulate relational databases.

<table>
<thead>
<tr>
<th>Set Operations</th>
<th>Database Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>SELECT</td>
</tr>
<tr>
<td>DIFFERENCE</td>
<td>PROJECT</td>
</tr>
<tr>
<td>INTERSECTION</td>
<td>JOIN</td>
</tr>
<tr>
<td>CARTESION PRODUCT</td>
<td>AGGREGATE</td>
</tr>
<tr>
<td></td>
<td>DIVISION</td>
</tr>
<tr>
<td></td>
<td>RENAME</td>
</tr>
</tbody>
</table>

Union Compatibility

Two relations r(R) and r(S) in a relational database D are Union Compatible and so are schemes R and S iff there exists a bijective mapping f: R → S (f is one-to-one and onto) such that:

\[
\text{Dom}(A_j) = \text{Dom}(B_k) \text{ for some } [B_k = f(A_j)] \text{ and } [A_j = f^{-1}(B_k)]
\]

where \( A_j \in R, B_k \in S, f \circ f^{-1} = \text{Identity} \)

Example:
Consider the following two relations:
OHIO_SCHOOL (Teacher, Course, School)
CALIF_SCHOOL (Faculty, Class, Institution)

They are union compatible iff there exists a mapping such as:
Teacher ↔ Faculty
Course ↔ Class
School ↔ Institution
Union

The result of this operation, denoted \((r \cup s)\) or \((r + s)\), is a relation that includes all tuples that \textit{either are in \(r\) or \(s\) or in both \(r\) and \(s\)}.

- \textit{Duplicate tuples are eliminated}
- Combined relations must be union compatible
- \(r + s = \{ t / t \in r \text{ or } t \in s \}\)

\textbf{Example}

\begin{array}{c|c|c|c}
\text{r} & A & B & C \\
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
\hline
\text{s} & A & B & C \\
1 & 2 & 3 & 3 \\
1 & 1 & 1 & 1 \\
3 & 2 & 1 & 1 \\
\hline
\text{r + s} & A & B & C \\
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
1 & 2 & 3 & 3 \\
3 & 2 & 1 & 1 \\
\end{array}

Difference

The result of this operation, denoted \((r - s)\), is a relation that includes all tuples in \(r\), but not in \(s\).

- Participating relations must be union compatible
- \(r - s = \{ t / t \in r \text{ and } t \notin s \}\)

\textbf{Example}

\begin{array}{c|c|c|c}
\text{r} & A & B & C \\
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
\hline
\text{s} & A & B & C \\
1 & 2 & 3 & 3 \\
1 & 1 & 1 & 1 \\
3 & 2 & 1 & 1 \\
\hline
\text{r - s} & A & B & C \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
\end{array}

\textbf{NOTE}: The \textit{difference} operator is \textit{not commutative}, that is, in general \(r - s \neq s - r\)
Intersection

The result of this operation, denoted \((r \cap s)\), is a relation that includes all tuples that are in both \(r\) and \(s\):

- Participating relations must be union compatible
- \(r \cap s = \{ t / t \in r \text{ and } t \in s \}\)

**Example**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ r \cap s \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Cartesian Product

This operation, denoted \((r \times s)\), is also known as the *cross product* or *cross join*. The purpose of the operator is to concatenate rows from two relations, making all possible combination of rows.

Consider relation schemas \(r(A_1, A_2, \ldots A_n)\) and \(s(B_1, B_2, \ldots B_m)\):

- Relations \(r\) and \(s\) do not have to be union compatible
- If \(r\) has \(n\) tuples and \(s\) has \(m\) tuples, then \((r \times s)\) will have a total of \((n \times m)\) tuples
- The resulting relation schema is \((A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_m)\)

**Example**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>b</td>
</tr>
</tbody>
</table>

\[ r \times s \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>r \times s</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>20</td>
<td>b</td>
</tr>
</tbody>
</table>
Selection

Retrieves all tuples that satisfy a condition from a relation r. The condition is expressed in terms of formulas. A formula could be in the form:
1. (Column \(_i\) \(\theta\) Column \(_j\)) or (Column \(_i\) \(\theta\) Value) where \(\theta = \{=, \geq, \leq, <>, <, >\}\)
2. If \(F_1\) and \(F_2\) are formulas \(\Rightarrow F_1\ and\ F_2\ or\ F_1\ or\ F_2\, not\ F_1\ are\ also\ valid\ formulas\)
3. Nothing else is a valid formula

Definition: \(\sigma_F(r) = \{ t / t \in r, F(t) = \text{true}\}\)

Properties
1. If \(F = \Phi\), then \(\sigma_F(r) = r\)
2. If \(r = \Phi\), then \(\sigma_F(t) = \Phi\)
3. \(\sigma_{F_1}(r)(\sigma_{F_2}(t))) = \sigma_{F_2}(t)(\sigma_{F_1}(t))\)
4. \(\sigma_{F_1}(r) + \sigma_{F_2}(r) = \sigma_{F_1+F_2}(t)\)

Example

Find \(S = \sigma_F(r)\), where \(F\) is defined as: (A=1) or (C=3)

\[
\begin{array}{|c|c|c|}
\hline
\text{r} & \text{A} & \text{B} & \text{C} \\
\hline
1 & 1 & 1 & 1 \\
2 & 2 & 2 & 2 \\
3 & 3 & 3 & 3 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{s} & \text{A} & \text{B} & \text{C} \\
\hline
1 & 1 & 1 & 1 \\
3 & 3 & 3 & 3 \\
\hline
\end{array}
\]
Join

The join operation, denoted by \((\text{Tab}_1 \otimes \text{Tab}_2)\), is used to combine related tuples of two relations.

- Join condition format is: \((\text{Table}_1.\text{Col}_1 \theta \text{Table}_2.\text{Col}_2)\)
- Restrictions: \((\text{Table}_1.\text{Col}_1 \theta \text{Value})\) can also be included (and-ed, or-ed) in the joining condition

Example
Consider relation schema \(r(A, B, C)\) and \(s(D, E)\) and the expression

\[\text{Temp1} = r \otimes s\]

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Temp1</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>c</td>
</tr>
</tbody>
</table>
Natural Join

The natural join operation, denoted by \((Tab1 \ast Tab2)\), is used to combine tuples of two relations under an equi-join.

- Related columns must have the same name and domain
- Implicit Join-Condition is \((Table_1.Col_1 = Table_2.Col_2)\)

Example
Consider relation schema \(r(A, B, C)\) and \(s(B, C, D)\) and the expression \(Temp1 = (r \ast s)\)

\[
\begin{array}{|c|c|c|}
\hline
r & A & B & C \\
\hline
1 & 1 & 1 & |
2 & 1 & 0 & |
4 & 3 & 2 & |
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
s & B & C & D \\
\hline
1 & 1 & a & \\
1 & 2 & b & \\
3 & 2 & c & \\
4 & 3 & d & \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
Temp1 & A & B & C & D \\
\hline
1 & 1 & 1 & A & \\
4 & 3 & 2 & C & \\
\hline
\end{array}
\]
Left Outer Join

The *left outer join* operation, denoted by \((r \Join_{<\text{join condition}>} s)\), is a special case of the general join.

- LOJ keeps in the resulting table every tuple from the first or left relation
- If no matching value is found in \(s\), then the attributes of \(s\) in the result are "padded" with \texttt{null} values

**Example**

Consider relation schema \(r(A, B, C)\) and \(s(D, E)\) and the expression

\[
\text{Temp1} = r \Join_{<r.C = s.D>} s
\]

<table>
<thead>
<tr>
<th>Temp1</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>\text{null}</td>
<td>\text{null}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(r)</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(s)</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>
Aggregate Functions

Originally proposed by A. Klug (1982) to extend the scope of relational algebra allowing mathematical computations of summary functions.

- Syntax: `<grouping attribute> F <function list> (<relation name>)`
- Common functions are MAX, MIN, AVG, SUM, COUNT
- Grouping attributes force a partitioning of the relation, the function is computed in each independent group
- Output consists of the grouping attributes and the result of the summary functions
- If no grouping field(s) is/are given the function(s) applies on the entire table

Example

Compute \( Temp = A \ F \ \text{SUM}(B), \text{MAX}(C) \ (r) \)

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 10 & 1 \\
1 & 2 & 5 \\
2 & 3 & 3 \\
3 & 6 & 10 \\
3 & 5 & 7 \\
\hline
\end{array}
\]

<table>
<thead>
<tr>
<th>Group-by field</th>
<th>Summary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Sum_B</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Temp1
Division

The division operator, denoted by \((r \div s)\), is useful in situations where the problem is phrased as:

\[
\text{for all tuples of } s \text{ some existential condition must hold}
\]

Consider the database schemas \(r(A, B)\) and \(s(B)\)
- A given value \(a\) from \(r.A\) is chosen to the output if there are tuples \(<a, b>\) in \(r\) whose \(b\) value matches each one of the \(b\) values of \(B\) in \(s\)

**Example:**
Consider relation schemas \(r(A, B)\) and \(s(B)\). The expression \(\text{Temp1} = (r \div s)\) produces the following output:

<table>
<thead>
<tr>
<th>(r)</th>
<th>(A)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>610</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>650</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>610</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>634</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>610</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>620</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(s)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td></td>
</tr>
<tr>
<td>610</td>
<td></td>
</tr>
<tr>
<td>620</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\text{Temp1})</th>
<th>(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>