What is a Relation?

### Formal Definitions

- **domain** - a set of values
- **schema** - a finite set S of objects called *attribute names* or *attributes*.
- Each attribute has a domain, i.e., there is a function \( \text{dom} : S \rightarrow \{ \text{domains} \} \)

### Formal Definitions, cont’d

- **Relation on schema S** - a finite set of tuples on S
- Cardinality, degree
- Properties of relations (Date):
  - There are no duplicate tuples.
  - Tuples are unordered.
  - Attributes are unordered.
  - Attribute values are atomic.
- **A database** is a finite set of named relations.
A Sample Database, cont.

<table>
<thead>
<tr>
<th>course_off</th>
<th>cid</th>
<th>sec_no</th>
<th>semester</th>
<th>instructor</th>
<th>room</th>
<th>time_slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>201</td>
<td>f2004</td>
<td>f1</td>
<td>Olsen</td>
<td>420</td>
<td>T 5-7:30</td>
</tr>
<tr>
<td>c1</td>
<td>201</td>
<td>f2005</td>
<td>f1</td>
<td>Olsen</td>
<td>999</td>
<td>T 5-7:30</td>
</tr>
<tr>
<td>c2</td>
<td>202</td>
<td>s2003</td>
<td>f1</td>
<td>Ball</td>
<td>210</td>
<td>M 5-7:30</td>
</tr>
<tr>
<td>c2</td>
<td>202</td>
<td>s2003</td>
<td>f2</td>
<td>Ball</td>
<td>222</td>
<td>M 5-7:30</td>
</tr>
<tr>
<td>c3</td>
<td>201</td>
<td>f2003</td>
<td>f3</td>
<td>Olsen</td>
<td>115</td>
<td>M 3-4:30</td>
</tr>
<tr>
<td>c4</td>
<td>201</td>
<td>s2004</td>
<td>f4</td>
<td>Ball</td>
<td>303</td>
<td>Th 8-10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>sec_no</th>
<th>semester</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>c1</td>
<td>201</td>
<td>f2004</td>
<td>a</td>
</tr>
<tr>
<td>s1</td>
<td>c2</td>
<td>202</td>
<td>f2003</td>
<td>b</td>
</tr>
<tr>
<td>s2</td>
<td>c1</td>
<td>201</td>
<td>f2004</td>
<td>a</td>
</tr>
<tr>
<td>s2</td>
<td>c2</td>
<td>202</td>
<td>f2003</td>
<td>b</td>
</tr>
<tr>
<td>s3</td>
<td>c3</td>
<td>201</td>
<td>f2003</td>
<td>a</td>
</tr>
<tr>
<td>s3</td>
<td>c4</td>
<td>201</td>
<td>s2004</td>
<td>c</td>
</tr>
<tr>
<td>s4</td>
<td>c1</td>
<td>201</td>
<td>f2003</td>
<td>a</td>
</tr>
<tr>
<td>s4</td>
<td>c2</td>
<td>202</td>
<td>s2003</td>
<td>a</td>
</tr>
</tbody>
</table>

Candidate Keys

- **Candidate key of a relation** $R$ - a subset $K$ of $R$'s schema having the properties that
  - (Uniqueness property) No two distinct tuples ever have the same value(s) for $K$ at the same time
  - (Minimality property) No proper subset of $K$ has the uniqueness property.
- Uniqueness property is a **constraint**: true over all time!
- Every relation has at least one candidate key.

Primary & Foreign Keys

- One of a relation’s candidate key’s is designated to be a **primary key**.
- A **foreign key** in a relation refers to the primary key of some relation.

Integrity and Keys

(C. J. Date)

- **The entity integrity rule** - No component of a primary key is allowed to accept nulls.
- **The referential integrity rule** - The database must not contain any unmatched foreign key values.

Relational Algebra: Why Bother?

- Formalizing query languages allows us to prove things.
- Formal query languages can aid in the design of real query languages.
- Query plans are based on the relational algebra.

Projection

"Give the names of all students"

\[ \Pi_{\text{name}}(\text{student}) \]

<table>
<thead>
<tr>
<th>sname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Moe</td>
</tr>
<tr>
<td>Curly</td>
</tr>
<tr>
<td>Shemp</td>
</tr>
</tbody>
</table>
Selection

"Tell me about CS majors"

\[ \sigma_{\text{major = 'cs'}} (\text{student}) \]

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
<th>sex</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Larry</td>
<td>grad 3.5</td>
<td>M</td>
<td>cs</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>Curly</td>
<td>2004 4</td>
<td>M</td>
<td>cs</td>
<td></td>
</tr>
</tbody>
</table>

Cartesian Product

"Pair all students and all courses"

\[ \text{student} \times \text{course} \]

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
<th>sex</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Larry</td>
<td>grad 3.5</td>
<td>M</td>
<td>cs</td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>Moe</td>
<td>2005 3.2</td>
<td>M</td>
<td>math</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>Curly</td>
<td>2004 4</td>
<td>M</td>
<td>cs</td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>Shemp</td>
<td>2000 3</td>
<td>M</td>
<td>physics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>course</th>
<th>id</th>
<th>course</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Database I</td>
<td>best...</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>Database II</td>
<td>best...</td>
<td></td>
</tr>
<tr>
<td>c3</td>
<td>Database III</td>
<td>best...</td>
<td></td>
</tr>
</tbody>
</table>

Union

\[ \text{longhair} \cup \text{cs\_major} \]

<table>
<thead>
<tr>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larry</td>
<td>grad 3.5</td>
<td>Shemp</td>
</tr>
<tr>
<td>Moe</td>
<td>2005 3.2</td>
<td>2000 3.2</td>
</tr>
<tr>
<td>Curly</td>
<td>2004 4</td>
<td></td>
</tr>
<tr>
<td>Shemp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Difference

\[ \text{longhair} - \text{cs\_major} \]

<table>
<thead>
<tr>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moe</td>
<td>2005 3.2</td>
<td></td>
</tr>
<tr>
<td>Curly</td>
<td>2004 4</td>
<td></td>
</tr>
<tr>
<td>Shemp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schemas

(Assume schema of r and s are R and S, resp.)

- \[ \Pi_s (r) \rightarrow S \ (R \subseteq S) \]
- \[ \sigma_s (r) \rightarrow R \]
- \[ r \times s \rightarrow R \cup S \ (R \cap S = \emptyset) \]
- \[ r \cup s \rightarrow R \ (R \text{ and } S \text{ must be union compatible}) \]
- \[ r - s \rightarrow R \ (R \text{ and } S \text{ must be union compatible}) \]
Definition of Intersection

- \( r \cap s = r - (r - s) \)
- R and S must be union compatible.
- Schema is R (or S)

Definition of Theta-Join

- \( r \theta s = \sigma_\theta (r \times s) \)
- Schema is \( R \cup S \), with attribute renaming to disambiguate duplicate attributes

Definition of Natural Join

- \( r \bowtie s = \pi_A (r \bowtimes s) \), where
  - \( \theta \) declares that attributes with the same names are equal
  - A is \( R \cup S \), with duplicate attributes removed rather than renamed.
- Schema is \( R \cup S \) with duplicate attributes removed

Natural Join

- \( \bowtie \)

Division

- \( A \)
Renaming Operator (for relations)
\( \rho(\text{csmajor}, \sigma_{\text{major}=\text{cs}}(\text{student})) \)

- Value is same as \( \sigma_{\text{major}=\text{cs}}(\text{student}) \)
- The name “csmajor” can be used as a temporary or for disambiguating.

Semijoin
\[
\begin{array}{c}
\text{student} \times \text{enr} \\
= \pi_{\text{sid}, \text{sname}, \text{yog}, \text{gpa}, \text{sex}, \text{major}}(\text{student} \times \text{enr})
\end{array}
\]

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>yog</th>
<th>gpa</th>
<th>sex</th>
<th>major</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Larry</td>
<td>grad</td>
<td>3.5</td>
<td>M</td>
<td>cs</td>
</tr>
<tr>
<td>s2</td>
<td>Moe</td>
<td>1999</td>
<td>3.2</td>
<td>M</td>
<td>math</td>
</tr>
<tr>
<td>s3</td>
<td>Curly</td>
<td>1998</td>
<td>4</td>
<td>M</td>
<td>cs</td>
</tr>
</tbody>
</table>

Example 1
English:
“Find the names of students who have earned an A in some course.”
Relational Algebra:
\[
\pi_{\text{sname}}(\text{student} \times\times \sigma_{\text{grade} = 'A'}(\text{enr}))
\]

Example 2
English:
“Find the names of students who have enrolled in all courses.”
Relational Algebra:
\[
\pi_{\text{sname}}(\text{student} \times\times (\pi_{\text{sid}, \text{cid}}(\text{enr}) \div \pi_{\text{cid}}(\text{course}))
\]

Example 3
English:
“Find the names of students who have earned A’s in all their courses.”
Relational Algebra:
\[
\pi_{\text{sname}}(\text{student} \times\times (\pi_{\text{sid}}(\text{student}) \div \pi_{\text{sid}}(\sigma_{\text{grade} = 'A'}(\text{enr})))
\]

Example 4
English:
“Find the SIDs of students who have enrolled in at least two courses.”
Relational Algebra:
\[
\pi_{\text{E1.sid}}(\rho(\text{E1, enr}) \times \rho(\text{E2, enr}))
\]

E1.sid = E2.sid and E1.sid ≠ E2.sid
Example 5

**English:**
"Find the names of students who have enrolled in at least one course."

**Relational Algebra:**
\[ \pi_{\text{name}}(\text{student} \bowtie \text{enr}) \]

Examples for You

- Find the IDs of faculty of faculty who have taught a course named “Database”.
- Find the names of students who have enrolled in a course named “Database”.
- Find the IDs of instructors who have taught every course.
- Find the SIDs of students who have never taken a section numbered 201.

Relational Calculi

- Based on Predicate Calculus
- Less procedural than Relational Algebra
- Tuple Calculus - Variables range over the tuples of relations.
- Domain Calculus - Variables range over domains.

Tuple Calculus - Example 1

**English:**
"Find the SIDs of students who have earned at least one A."

**Tuple Calculus:**
\[ \{ t \mid \exists e ( e \in \text{enr} \land e.\text{grade} = 'A' \land t.\text{sid} = e.\text{sid}) \} \]

Tuple Calculus - Example 2

**English:**
"Find the names of students who have earned at least one A."

**Tuple Calculus:**
\[ \{ t \mid \exists s \exists e ( s \in \text{student} \land e \in \text{enr} \land s.\text{sid} = e.\text{sid} \land e.\text{grade} = 'A' \land t.\text{name} = s.\text{name}) \} \]

Tuple Calculus - Example 3

**English:**
"Find the names of students who have earned A’s in all their courses."

**Tuple Calculus:**
\[ \{ t \mid \exists s \forall e ( s \in \text{student} \land e \in \text{enr} \land t.\text{name} = s.\text{name} \land s.\text{sid} = e.\text{sid} \land e.\text{grade} = 'A') \rightarrow e.\text{grade} = 'A') \} \]
Domain Calculus - Example

**English:**
"Find the names of students who have earned at least one A."

**Domain Calculus:**
\[
\{s \mid \exists \text{sid} \exists \text{name} \exists \text{Gpa} \exists \text{Sex} \exists \text{Maj} \\
\quad \exists \text{sid} \exists \text{Cid} \exists \text{Sec} \exists \text{Sem} \exists \text{Gr} \\
\quad < \text{Sid}, \text{Nam}, \text{Gpa}, \text{Sex}, \text{Maj} > \in \text{student} \\
\quad \land < \text{Sid}2, \text{Cid}, \text{Sec}, \text{Sem}, \text{Gr} > \in \text{enr} \\
\quad \land s = \text{Name} \\
\quad \land \text{Sid1} = \text{Sid2} \\
\quad \land \text{Gr} = \text{A}'\}
\]

Relational Completeness

- A query language is *relationally complete* if it is as powerful as the relational algebra.
- The relational algebra and the relational calculi are all equivalent in computational power.

What You Cannot Do In Relational Algebra

- Transitive Closures
- Aggregate Functions
- Grouping
- Sorting

Other Data Models

- Ancient ones (pre-1990s): hierarchical and network
- ER model
- Object-oriented model – C++ with persistence and sharing
- Object relational model – relational model whose domains can be complex types or abstract data types, with inheritance, object identity, and other “object-oriented” features
- Semi-structured (XML)

Relational Model

- Data model – set of primitives for modeling a miniworld, including data types and language
- Relational model - data represented entirely by relations (tables), languages “like” relational algebra, relational calculi, SQL

Conclusions

- The relational model is an astonishing and unexpected success story.
- The relational algebra and relational calculi are integral parts of the (theoretical) relational model.
- They are also the basis of some real query languages.
- The relational algebra plays a central role in most query optimizers.