The Relational Model & Relational Algebra

CPS 352: Database Systems

Simon Miner
Gordon College
Last Revised: 9/6/12
Agenda

• Check-in
• The Relational Model
• Design Project Requirements Presentations
• Relational Algebra
• Homework 1
Check-in
The Relational Model
Historical Background

• Hierarchical and network databases came first

• First relational databases pioneered in 1970s
  • Simpler than earlier models (easier for programmers)
  • Based on mathematical theory of relations (expressed via relational algebra).
  • Had performance issues which helped other models to persist for a time
    • Extensive research (i.e. on indexing strategies) helped overcome performance bottlenecks

• Today, the relational model is dominant in the database world
  • Though other approaches are often used in tandem with it – polyglot persistence
Entities and Relationships

• All database models must implement the following two concepts
  • *Entity* – real or abstract “things”
  • *Relationships* between entities

• Relational model represents both entities and relationships via *tables*.
  • Table *attributes (columns)* must be *atomic* and *single valued*
Mathematical Terminology

• Relational database – a collection of relations

• Relation – a set of tuples of some arity
  • Tuple -- a record in the set
  • Arity – number of component attributes in a tuple
    • Tuples in any given relation have the same arity
    • Order of attributes in tuples is important
    • Order of tuples in relation is not important

• Attribute – numbered or named component of a tuple
  • Drawn from a specific domain or set of possible values

• Relation scheme – structure of tuples in a relation

• Instance – a specific relation on some scheme
  • Subset of the Cartesian product of the domains of its attributes
# Alternative Terminology

<table>
<thead>
<tr>
<th>Mathematical</th>
<th>Alternate</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Relation</td>
<td>• Table</td>
</tr>
<tr>
<td>• Tuple</td>
<td>• Row</td>
</tr>
<tr>
<td>• Attribute</td>
<td>• Column</td>
</tr>
<tr>
<td>• Relation scheme</td>
<td>• Sometimes represented by column headings</td>
</tr>
</tbody>
</table>
Keys

- The tuples comprising a relation must be unique
  - No duplicates because the relation is a set

- **Superkey** – Set of attributes which distinguish any tuple in the relation from all others

- **Candidate key** – a superkey with no proper subset of attributes that is also a superkey

- **Primary key** – a candidate key chosen to be the basis for uniquely identifying tuples

- **Foreign key** – column(s) in one table that comprise the primary key of another table
  - Represent relationships in a relational database
Nulls

• Special value NULL assigned to a field when the attribute’s value is unknown or does not exist

• NULL is not the same as:
  • String of spaces ("   ")
  • Empty string ("")
  • Zero (0)
  • NULL (NULL = NULL even returns false)

• Databases can specify not null constraints on columns which must have values
  • i.e. Candidate, primary, and foreign key columns
Schemas and Instances

- **Schema** – the logical design of a database
  - Database schema comprised of tables (relations) and their relationships with one another

- **Instance** – a snapshot of the actual data (relations) in the database at a given point in time

- Schema diagram – depicts entities and relationships in a database schema
  - Primary keys shaded or underlined
  - Foreign keys represented by arrows between related tables
Design Project
Requirements
Presentations
Librarianship Schema Diagram

Simplifying assumptions for this example:

1) author of a book is single-valued
2) there is only one copy of a book with a given call number
3) a given book can only be on reserve for a single course
4) course-id is presumably a foreign key in a table not shown
### Example Library Instance

#### borrower

<table>
<thead>
<tr>
<th>borrower_id</th>
<th>last_name</th>
<th>first_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>aardvark</td>
<td>anthony</td>
</tr>
<tr>
<td>20147</td>
<td>cat</td>
<td>charlene</td>
</tr>
<tr>
<td>89754</td>
<td>dog</td>
<td>donna</td>
</tr>
<tr>
<td>60984</td>
<td>fox</td>
<td>frederick</td>
</tr>
<tr>
<td>54872</td>
<td>zebra</td>
<td>zelda</td>
</tr>
</tbody>
</table>

#### book

<table>
<thead>
<tr>
<th>call_number</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA76.093</td>
<td>Wenham Zoo Guide</td>
<td>elephant</td>
</tr>
<tr>
<td>RZ12.905</td>
<td>Fire Hydrants I Have Known</td>
<td>dog</td>
</tr>
<tr>
<td>LM925.04</td>
<td>21 Ways to Cook a Cat</td>
<td>dog</td>
</tr>
<tr>
<td>AB123.40</td>
<td>Karate</td>
<td>koala</td>
</tr>
</tbody>
</table>

#### checked_out

<table>
<thead>
<tr>
<th>borrower_id</th>
<th>call_number</th>
<th>date_due</th>
</tr>
</thead>
<tbody>
<tr>
<td>89754</td>
<td>RZ12.905</td>
<td>2002-11-10</td>
</tr>
<tr>
<td>89754</td>
<td>LM925.04</td>
<td>2002-11-10</td>
</tr>
<tr>
<td>20147</td>
<td>AB123.40</td>
<td>2002-11-15</td>
</tr>
</tbody>
</table>

#### reserve_book

<table>
<thead>
<tr>
<th>call_number</th>
<th>course_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA76.093</td>
<td>BY123</td>
</tr>
<tr>
<td>AB123.40</td>
<td>PE075</td>
</tr>
</tbody>
</table>

#### employee

<table>
<thead>
<tr>
<th>ssn</th>
<th>last_name</th>
<th>first_name</th>
<th>salary</th>
<th>supervisor_ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>aardvark</td>
<td>anthony</td>
<td>40000</td>
<td>null</td>
</tr>
<tr>
<td>567-89-1234</td>
<td>buffalo</td>
<td>boris</td>
<td>30000</td>
<td>123-45-6789</td>
</tr>
<tr>
<td>890-12-3456</td>
<td>elephant</td>
<td>emily</td>
<td>50000</td>
<td>123-45-6789</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>fox</td>
<td>frederick</td>
<td>45000</td>
<td>567-89-1234</td>
</tr>
</tbody>
</table>
Example Queries Against the Library Database

- Who is the borrower whose borrower id is 12345?
- List the names of all borrowers.
- What is the title of the book whose call number is QA76.093?
- List the titles of all books that are currently checked out.
- List the names of all borrowers having one or more books overdue.
- List the names of all employees who earn more than their supervisor.
- List the names of all people connected with the library - whether borrowers, employees, or both.
- List the names of all borrowers who are not employees.
- List all books needed as course reserves that are currently checked out to someone.
- List the names of employees together with their supervisor’s name.
- List the call numbers of all overdue books, together with the number of days they are overdue.
- What is the average salary of all employees?
- Print a list of borrower id's and the number of books each has out
- List the titles of all books, together with the borrower id of the person (if any) who has the book out.
Query Languages

• All DBMS’s support at least one query language which allow for the following
  • Interactive usage
  • Embedded within applications in programming languages

• Classifications
  • Formal query language – uses mathematical notation and concepts useful for research (i.e. proving theorems)
    • Relational algebra
  • Commercial languages – built on top of mathematical language principles for easier usage
    • SQL
Relational Algebra
Relational Algebra Operations

- Involve either on one or two relations
  - Unary and binary operations
- Each operation returns a new relation
  - Enables composing or “chaining” of relations
- Operation Types
  - Primitive Operations
  - Composite Operations
    - Built with primitive operations, but common enough to warrant their own operations
  - Extended Relational Algebra
# Primitive Operations

- Selection
- Projection
- Join – a.k.a. Cartesian Product or Simple Join
- Rename
- Union
- Difference
Selection

• Select rows/tuples from a table/relation which meet certain criteria

• Denoted by lowercase Greek letter sigma -- \( \sigma \)

• Example: Who is the borrower whose borrower id is 12345?

\( \sigma_{\text{borrower_id} = 12345} \text{borrower} \)

• Returns: 12345, aardvark, anthony

• A subset of rows/tuples in a table/relation

• Multiple criteria can be specified by logical operators
  • \( \land \) – and
  • \( \lor \) – or
  • \( \neg \) – negation (not)
Projection

• Choose only specific columns/attributes from all rows/tuples in a table/relation

• Denoted by the lowercase Greek letter pi - π

• Example: List the names of all borrowers

  • \( \pi \) last_name, first_name borrower

• Returns the following rows/tuples
  • aardvark, anthony
  • cat, charlene
  • dog, donna
  • fox, frederick
  • zebra, zelda
Composing Operations

- Relational algebra operations can be combined
- Example: What is the title of the book whose call number is QA76.093?
  \[ \pi \text{title} \ \sigma \text{call_number} = \text{QA76.093} \ \text{book} \]
- Returns: Wenham Zoo Guide
A projection could produce duplicate rows by suppressing the column(s)/attribute(s) which distinguish rows.

Example: List authors of books

\[ \pi_{\text{author}} \text{ book} \]

This is a problem

Duplicates eliminated because relations are sets

Returns the following

- dog
- elephant
- koala
Cartesian Product / Simple Join

- Select every combination of rows/tuples from two tables/relations
  - Result has as many rows as the product of the number of rows/tuples in the two tables/relations being joined
  - Result has as many columns/attributes as the sum of the columns in each table/relation involved in the join
- Denoted by a capital X
Cartesian Product Example

- **Example:** List the titles of all books that are currently checked out
  - Requires an initial Cartesian product

- **checked_out X book**

<table>
<thead>
<tr>
<th>Borrower call id</th>
<th>Borrower call number</th>
<th>date-due</th>
<th>call number</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>89754</td>
<td>RZ12.905</td>
<td>11-10-02</td>
<td>QA76.093</td>
<td>Wenham Zoo Guide</td>
<td>elephant</td>
</tr>
<tr>
<td>89754</td>
<td>RZ12.905</td>
<td>11-10-02</td>
<td>RZ12.905</td>
<td>Fire Hydrants I Have Known</td>
<td>dog</td>
</tr>
<tr>
<td>89754</td>
<td>RZ12.905</td>
<td>11-10-02</td>
<td>LM925.04</td>
<td>21 Ways to Cook a Cat</td>
<td>dog</td>
</tr>
<tr>
<td>89754</td>
<td>RZ12.905</td>
<td>11-10-02</td>
<td>AB123.40</td>
<td>Karate</td>
<td>koala</td>
</tr>
<tr>
<td>89754</td>
<td>LM925.04</td>
<td>11-10-02</td>
<td>QA76.093</td>
<td>Wenham Zoo Guide</td>
<td>elephant</td>
</tr>
<tr>
<td>89754</td>
<td>LM925.04</td>
<td>11-10-02</td>
<td>RZ12.905</td>
<td>Fire Hydrants I Have Known</td>
<td>dog</td>
</tr>
<tr>
<td>89754</td>
<td>LM925.04</td>
<td>11-10-02</td>
<td>LM925.04</td>
<td>21 Ways to Cook a Cat</td>
<td>dog</td>
</tr>
<tr>
<td>89754</td>
<td>LM925.04</td>
<td>11-10-02</td>
<td>AB123.40</td>
<td>Karate</td>
<td>koala</td>
</tr>
<tr>
<td>20147</td>
<td>AB123.40</td>
<td>11-15-02</td>
<td>QA76.093</td>
<td>Wenham Zoo Guide</td>
<td>elephant</td>
</tr>
<tr>
<td>20147</td>
<td>AB123.40</td>
<td>11-15-02</td>
<td>RZ12.905</td>
<td>Fire Hydrants I Have Known</td>
<td>dog</td>
</tr>
<tr>
<td>20147</td>
<td>AB123.40</td>
<td>11-15-02</td>
<td>LM925.04</td>
<td>21 Ways to Cook a Cat</td>
<td>dog</td>
</tr>
<tr>
<td>20147</td>
<td>AB123.40</td>
<td>11-15-02</td>
<td>AB123.40</td>
<td>Karate</td>
<td>koala</td>
</tr>
</tbody>
</table>
Cartesian Product Example
(continued)

- Apply selection to limit results to meaningful rows/tuples
  - \( \sigma \text{ checked\_out.call\_number} = \text{book.call\_number} \) ( \( \text{checked\_out X book} \) )

- Yields the following:
  - 89754 RZ12.905 11-10-02 RZ12.905 Fire Hydrants I Have Known dog
  - 89754 LM925.04 11-10-02 LM925.04 21 Ways to Cook a Cat dog
  - 20147 AB123.40 11-15-02 AB123.40 Karate koala

- Use a projection to return only book titles
  - \( \pi \text{ title} \) \( \sigma \text{ checked\_out.call\_number} = \text{book.call\_number} \) ( \( \text{checked\_out X book} \) )

- Which in turn yields:
  - Fire Hydrants I Have Known
  - 21 Ways to Cook a Cat
  - Karate
Rename

- Renames a given table/relation and potentially its attributes as well
- Denoted by the lowercase Greek letter rho – ρ
- Useful in conjunction with joins
  - Especially when joining a table with itself
- Example: List the names of all employees who earn more than their supervisor

\[
p \pi \text{employee.last_name, employee.first_name} \\
\sigma \text{employee.supervisor_ssn} = \text{supervisor.ssn} \land \text{employee.salary} > \text{supervisor.salary} \\
\left( \text{employee} \times \rho_{\text{supervisor}} \text{employee} \right)
\]
- Returns
  - elephant, emily
  - fox, frederick
Union

• Combine two tables/relations in the same scheme into one
  • Eliminates duplicate rows/tuples

• Denoted by $\cup$ set algebra operator

• Example: List the names of all people connected with the library - whether borrowers, employees, or both

  $\left( \pi_{\text{last\_name, first\_name}} \text{borrower} \right) \cup \left( \pi_{\text{last\_name, first\_name}} \text{employee} \right)$

• Preparing similar tables/relations for union operation
  • Projecting columns/attributes common to both relations
  • Renaming attributes
Difference

- Takes rows/tuples from two tables/relations with the same scheme, and returns only those rows present in the first table, but not the second.
- Denoted by \(-\) set algebra operator.
- Example: List the names of all borrowers who are not employees.
- \((\pi_{\text{last\_name, first\_name}} \text{borrower}) - (\pi_{\text{last\_name, first\_name}} \text{employee})\)
Composite Operators

• Intersection
• Natural Join
• Theta Join
Intersection

- Returns rows/tuples from two tables/relations with the same scheme which occur in both of them
- Denoted by $\cap$ set algebra operator
- Example: List all books (call numbers only) needed as course reserves that are currently checked out to someone

\[
( \pi_{\text{call}\_\text{number}} \text{ reserve}\_\text{book} ) \cap ( \pi_{\text{call}\_\text{number}} \text{ checked}\_\text{out} )
\]

- Can be computed via primitive relational operations
  - Given relations $R$ and $S$:
  - Intersection $= R - (R - S)$
Natural Join

• Special join which returns only those rows/tuples from two tables/relations which have the same values in one or more columns/attributes in a selection
  • Natural join removes duplicate join key values

• Denoted by the |X| (bowtie) operator

• Example: List all data for books that are checked out
  • \( \pi \) checked_out.call_number, borrower_id, date_due, title, author
    \( \sigma \) checked_out.call_number = book.call_number (checked_out X book)
  • Similar to Cartesian product example, except for more data in projection

• This could also be done as a natural join:
  • checked_out |X| book
Theta Join (\(\theta\)-join)

- Join allowing for any arithmetic comparison operator (\(<, \leq, =, \geq, \text{ or } >\)), not just strict equality of values of columns/attributes
  - Natural join (which does an equality comparison is actually a subset of theta join)

- Example: List the names of all employees together with their supervisor’s name
  - Can be done as follows:
    - Cartesian product of the table against itself (renamed appropriately)
    - Selection comparing the employee’s and supervisor’s SSN values
    - Projection of the desired name data

\[
\pi_{e.\text{last\_name}, e.\text{first\_name}, s.\text{last\_name}, s.\text{first\_name}} \\
\sigma_{e.\text{supervisor\_ssn} = s.\text{ssn}} \left( \rho_{e \text{employee}} X \rho_{s \text{employee}} \right)
\]

- The selection can be “injected” into the Cartesian product as its join criteria

\[
\pi_{e.\text{last\_name}, e.\text{first\_name}, s.\text{last\_name}, s.\text{first\_name}} \text{employee} \\
\left( \rho_{e \text{employee}} X \theta_{e..\text{supervisor\_ssn} = s.\text{ssn}} \rho_{s \text{employee}} \right)
\]
Extended Relational Algebra

- Generalized Projection
- Aggregate Functions
- Outer Join
Generalized Projection

• Allow projections to include computations based on column/attribute values in addition to column values themselves

• Example: List the call numbers of all overdue books, together with the number of days they are overdue.

• $\pi_{\text{call\_number}, \text{today} - \text{date\_due}} \sigma_{\text{date\_due} < \text{today}} \text{checked\_out}$
Aggregate Functions

- Allow the use of functions which return summary data from a set of rows/tuples
  - min, max, sum, average to a column/attribute
  - count to an entire table/relation
- Denoted by the fancy capital $G$
- Example: What is the average salary of all employees?
  - $G_{\text{average( salary )}}\text{ employee}$
- Can produce summaries by groups
- Example: Print a list of borrower id's and the number of books each has out
  - borrower_id $G_{\text{count( call_number )}}\text{ checked_out}$
Outer Join

- Variant of natural or theta join which will include rows/tuples in one table/relation, even if there is no match in the other
  - Includes a dummy relation of all nulls in the result row for the unmatched relation
- Variants
  - Left outer join – denoted by $\bowtie$ -- no match in right table OK
  - Right outer join – denoted by $\triangleright$ -- no match in left table OK
  - Full outer join – denoted by $\bowtie$ -- no match in either table OK
- Example: List the titles of all books, together with the borrower id of the person (if any) who has the book checked out.
  - $\pi$ borrower_id, title book $\bowtie$ checked_out
Homework 1