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

Mathematical	Alternate
Relation	Table
Tuple	Row
Attribute	Column
Relation scheme	Sometimes represented by column headings

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

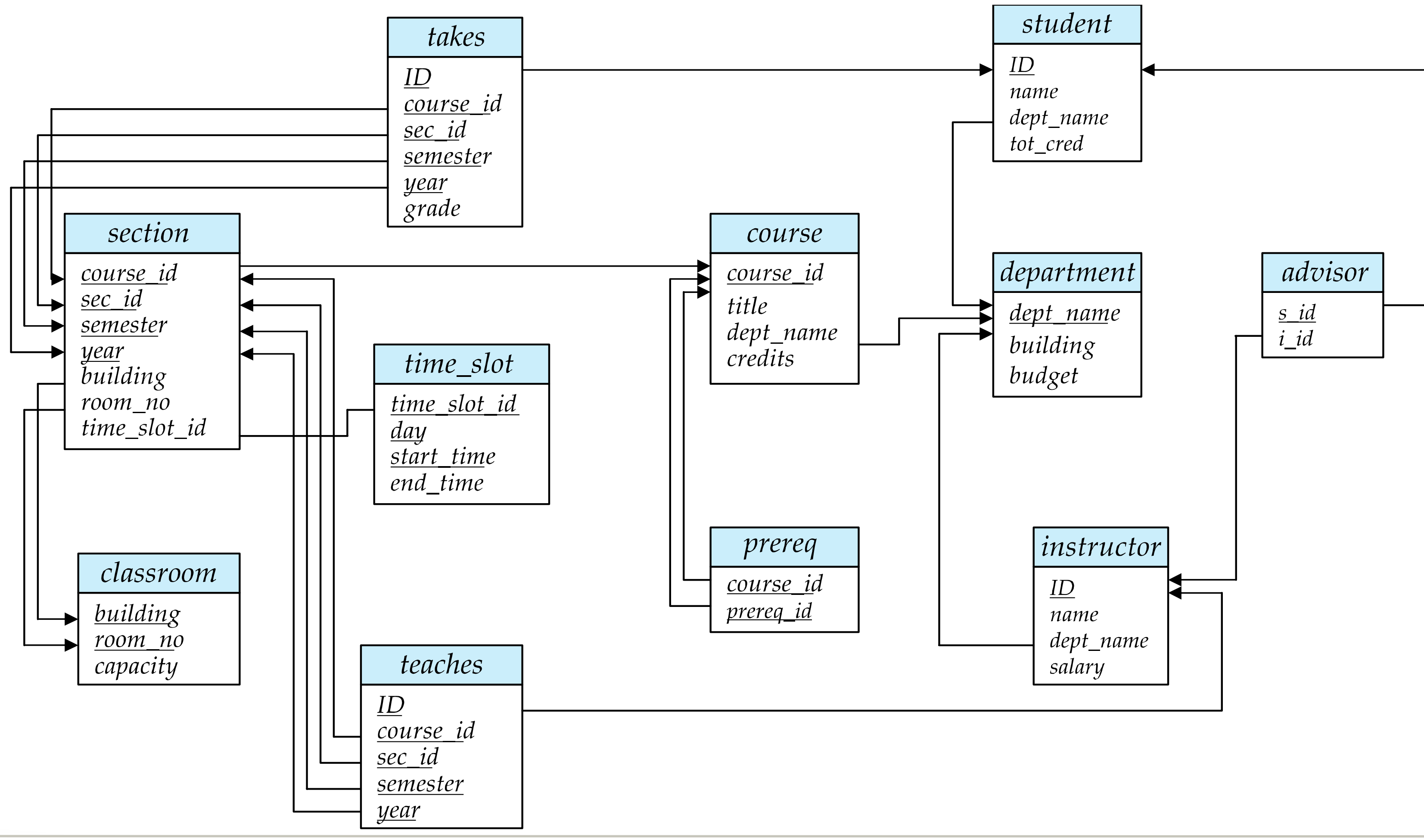
Nulls

- 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

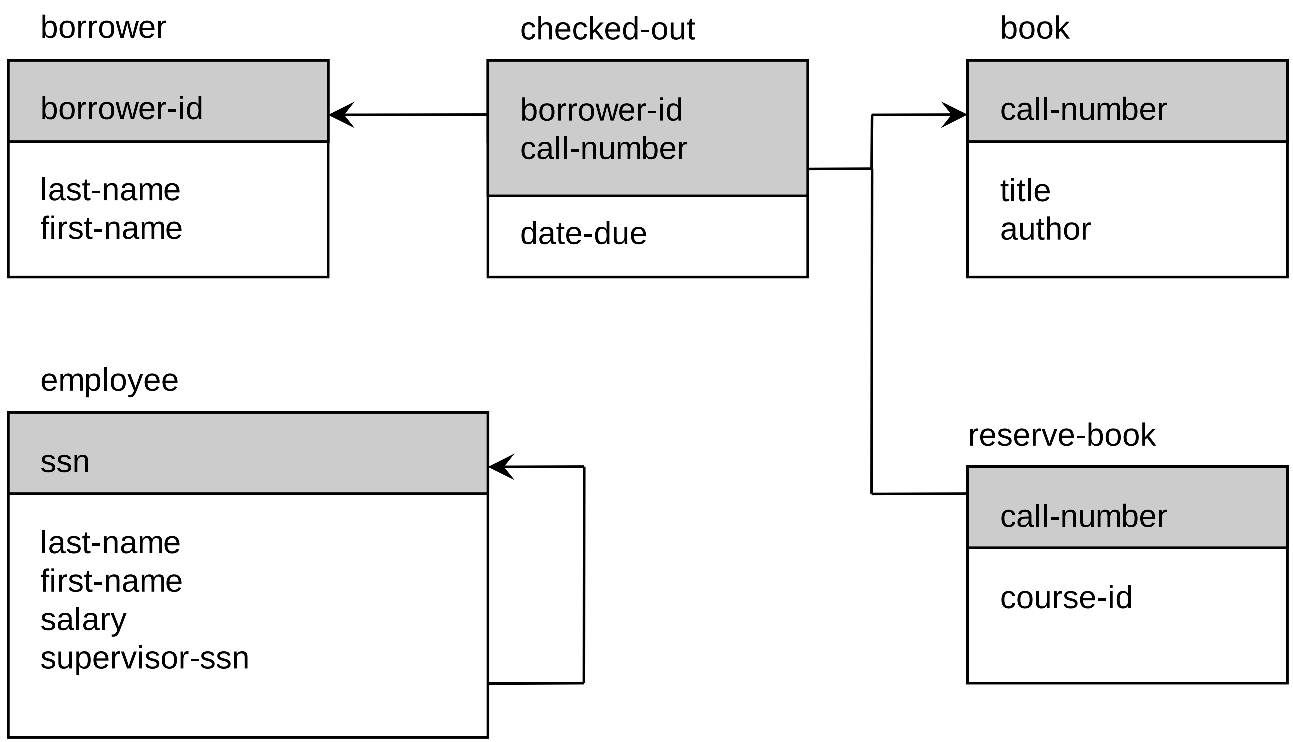
- *Schema* – the logical design of a database
  - Database schema comprised of tables (relations) and their relationships with one another
- *Instance* – a snapshot of the actual data (relations) in the database at a given point in time
- Schema diagram – depicts entities and relationships in a database schema
  - Primary keys shaded or underlined
  - Foreign keys represented by arrows between related tables

University Schema Diagram



Design Project Requirements Presentations

Library 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(borrower_id, last_name, first_name)

12345	aardvark	anthony
20147	cat	charlene
89754	dog	donna
60984	fox	frederick
54872	zebra	zelda

book(call_number, title, author)

QA76.093	Wenham Zoo Guide	elephant
RZ12.905	Fire Hydrants I Have Known	dog
LM925.04	21 Ways to Cook a Cat	dog
AB123.40	Karate	koala

checked_out(borrower_id, call_number, date_due)

89754	RZ12.905	2002-11-10
89754	LM925.04	2002-11-10
20147	AB123.40	2002-11-15

reserve_book(call_number, course_id)

QA76.093	BY123
AB123.40	PE075

employee(ssn, last_name, first_name, salary, supervisor_ssn)

123-45-6789	aardvark	anthony	40000	null
567-89-1234	buffalo	boris	30000	123-45-6789
890-12-3456	elephant	emily	50000	123-45-6789
111-11-1111	fox	frederick	45000	567-89-1234

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 -- σ
- Example: Who is the borrower whose borrower id is 12345?
- $\sigma_{\text{borrower\_id} = 12345}$ borrower
- Returns: 12345, aardvark, anthony
  - A subset of rows/tuples in a table/relation
- Multiple criteria can be specified by logical operators
  - ∧ – and
  - ∨ - or
  - – - 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
- $\pi_{\text{last\_name, first\_name}}$ 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?
- $\pi_{\text{title}}\ \sigma_{\text{call\_number = QA76.093}}\ \text{book}$
- Returns: Wenham Zoo Guide

Projection and Duplicate Results

- A projection could produce duplicate rows by suppressing the column(s)/attribute(s) which distinguish rows.
- Example: List authors of books
- π_{author} 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 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

Borrower id	call number	date-due	call number	title		author
89754	RZ12.905	11-10-02	QA76.093	Wenham Zoo Guide		elephant
89754	RZ12.905	11-10-02	RZ12.905	Fire Hydrants I Have Known	dog	
89754	RZ12.905	11-10-02	LM925.04	21 Ways to Cook a Cat	dog	
89754	RZ12.905	11-10-02	AB123.40	Karate		koala
89754	LM925.04	11-10-02	QA76.093	Wenham Zoo Guide		elephant
89754	LM925.04	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	
89754	LM925.04	11-10-02	AB123.40	Karate		koala
20147	AB123.40	11-15-02	QA76.093	Wenham Zoo Guide		elephant
20147	AB123.40	11-15-02	RZ12.905	Fire Hydrants I Have Known	dog	
20147	AB123.40	11-15-02	LM925.04	21 Ways to Cook a Cat	dog	
20147	AB123.40	11-15-02	AB123.40	Karate		koala

Cartesian Product Example (continued)

- Apply selection to limit results to meaningful rows/tuples
  - $\sigma_{\text{checked\_out.call\_number = book.call\_number}}$ (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
  - $\pi_{\text{title}}\ \sigma_{\text{checked\_out.call\_number = book.call\_number}}$ (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
- $\pi_{\text{employee.last\_name, employee.first\_name}}$
  $\sigma_{\text{employee.supervisor\_ssn = supervisor.ssn} \wedge \text{employee.salary > supervisor.salary}}$
  $(\text{employee X } \rho_{\text{supervisor}} \text{ employee})$
- Returns
  - elephant, emily
  - fox, frederick

Union

- Combine two tables/relations in the same scheme into one
  - Eliminates duplicate rows/tuples
- Denoted by ∪ set algebra operator
- Example: List the names of all people connected with the library - whether borrowers, employees, or both
- $(\pi_{last\_name,\ first\_name}\ borrower) \cup (\pi_{last\_name,\ first\_name}\ employee)$
- 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 ∩ set algebra operator
- Example: List all books (call numbers only) needed as course reserves that are currently checked out to someone
- $(\pi_{\text{call\_number}} \text{ reserve\_book}) \cap (\pi_{\text{call\_number}} \text{ checked\_out})$
- 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
  - $\pi_{\text{checked\_out.call\_number, borrower\_id, date\_due, title, author}}\ \sigma_{\text{checked\_out.call\_number = book.call\_number}}$ (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_cout |X| book

Theta Join (θ-join)

- Join allowing for any arithmetic comparison operator (<, ≤, =, ≥, 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
  - $\pi_{e.last\_name,\ e.first\_name,\ s.last\_name,\ s.first\_name}\ \sigma_{e.supervisor\_ssn\ =\ s.ssn}\ (\ \rho_e\ \text{employee}\ X\ \rho_s\ \text{employee}\ )$
- The selection can be "injected" into the Cartesian product as its join criteria
  - $\pi_{e.last\_name,\ e.first\_name,\ s.last\_name,\ s.first\_name}\ \text{employee}\ (\ \rho_e\ \text{employee}\ X_{\ \theta\ e..supervisor\_ssn\ =\ s.ssn}\ \rho_s\ \text{employee}\ )$

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.
- $\pi_{\text{call\_number, today – date\_due}} \sigma_{\text{date\_due < today}}$ 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?
  - $G_{\text{average(salary)}}$ employee
- Can produce summaries by groups
- Example: Print a list of borrower id's and the number of books each has out
  - $_{\text{borrower\_id}}G_{\text{count(call\_number)}}$ 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 ⟕ -- no match in right table OK
    - Right outer join – denoted by ⟖ -- no match in left table OK
    - Full outer join – denoted by ⟗ -- 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.
  - $\pi_{\text{borrower\_id, title}}$ book ⟕ checked_out

Homework 1