Outline

- Overview of modeling
- SQL (Chapter 3)
  - Setting up the PostgreSQL database
  - Data Definition (3.2)
  - Basics (3.3-3.5)
  - Null values (3.6)
  - Aggregates (3.7)
- Relational Model (Chapter 2)
  - Basics
  - Keys
  - Relational operations
  - Relational algebra (basics)
  - Relational algebra (advanced)

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.
  - `avg`: average value
  - `min`: minimum value
  - `max`: maximum value
  - `sum`: sum of values
  - `count`: number of values

- **Aggregate operation** in relational algebra
  \[ G_1, G_2, ..., G_n \bar{G} F_1(A_1), F_2(A_2), ..., F_n(A_n) (E) \]

  *E* is any relational-algebra expression
  - \( G_1, G_2, ..., G_n \) is a list of attributes on which to group (can be empty)
  - Each \( F_i \) is an aggregate function
  - Each \( A_i \) is an attribute name
- Note: Some books/articles use \( \gamma \) instead of \( \bar{G} \) (Calligraphic G)
Aggregate Operation – Example

- Relation $r$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>β</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

$G_{\text{sum}(c)}(r)$

<table>
<thead>
<tr>
<th>sum(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
</tr>
</tbody>
</table>

Aggregate Operation – Example

- Find the average salary in each department

$$G_{\text{avg}(\text{salary})}(\text{instructor})$$

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>75000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>Comp. Sci.</td>
<td>65000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>Finance</td>
<td>90000</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
<td>Finance</td>
<td>80000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>60000</td>
</tr>
<tr>
<td>58583</td>
<td>Califieri</td>
<td>History</td>
<td>62000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>Physics</td>
<td>87000</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept_name</th>
<th>avg_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>Comp. Sci.</td>
<td>77333</td>
</tr>
<tr>
<td>Elec. Eng.</td>
<td>80000</td>
</tr>
<tr>
<td>Finance</td>
<td>85000</td>
</tr>
<tr>
<td>History</td>
<td>61000</td>
</tr>
<tr>
<td>Music</td>
<td>40000</td>
</tr>
<tr>
<td>Physics</td>
<td>91000</td>
</tr>
</tbody>
</table>
Aggregate Functions (Cont.)

• Result of aggregation does not have a name
  • Can use rename operation to give it a name
  • For convenience, we permit renaming as part of aggregate operation

\[ \text{dept\_name } G \text{avg(salary) as avg\_sal (instructor)} \]

Modification of the Database

• The content of the database may be modified using the following operations:
  • Deletion
  • Insertion
  • Updating

• All these operations can be expressed using the assignment operator

\[
\begin{align*}
temp1 & \leftarrow R \times S \\
temp2 & \leftarrow \sigma_{r.A_1=s.A_1 \land r.A_2=s.A_2 \land \ldots \land r.A_n=s.A_n} (temp1) \\
result & = \Pi_{R \cup S} (temp2)
\end{align*}
\]

The result of \( R \times S \) potentially has duplicated attributes. For example, \( r(A,B) \times s(B,C) \) results in tuples with attributes \( \{A, B, B, C\} \). “\( \Pi_{R \cup S} \)” gets rid of the extra B. Duplicated tuples are an entirely different thing, and are not present in the relational algebra.
Multiset Relational Algebra

- Pure relational algebra removes all duplicates
  - e.g. after projection

- Multiset relational algebra retains duplicates, to match SQL semantics
  - SQL duplicate retention was initially for efficiency, but is now a feature

- Multiset relational algebra defined as follows
  - selection: has as many duplicates of a tuple as in the input, if the tuple satisfies the selection
  - projection: one tuple per input tuple, even if it is a duplicate
  - cross product: If there are \( m \) copies of \( t_1 \) in \( r \), and \( n \) copies of \( t_2 \) in \( s \), there are \( m \times n \) copies of \( t_1.t_2 \) in \( r \times s \)
  - Other operators similarly defined
    - E.g. union: \( m + n \) copies, intersection: \( \min(m, n) \) copies
      - difference: \( \min(0, m - n) \) copies

Entity-Relationship Model
Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
- Formal Semantics of SQL

Entity-Relationship Model

This is how we start a database project

- Two key concepts
  - **Entities**:
    - An object that *exists* and is *distinguishable* from other objects
    - Examples: Bob Smith, BofA, CMSC424
    - Have **attributes** (people have names and addresses)
    - Form **entity sets** with other entities of the same type that share the same properties
      - Set of all people, set of all classes
    - Entity sets may overlap
      - Customers and Employees
  - ...**Relationships**
Entity-Relationship Model

- Two key concepts
  - **Entities**...
  - **Relationships**:
    - Relate 2 or more entities
      - E.g. Bob Smith *has account at* College Park Branch
    - Form *relationship sets* with other relationships of the same type that share the same properties
      - Customers *have accounts at* Branches
    - Can have attributes:
      - *has account at* may have an attribute *start-date*
    - Can involve more than 2 entities
      - Employee *works at Branch at Job*

Next: Relationship Cardinalities

- We may know:
  - One customer can only open one account
  - OR
  - One customer can open multiple accounts

- Representing this is important

- Why?
  - Better manipulation of data
    - If former, can store the account info in the customer table
  - Can enforce such a constraint
    - “Application logic will handle it” NOT GOOD
  - If not represented in conceptual model, domain knowledge can easily be lost
Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set

- Most useful in describing binary relationship sets

- N-ary relationships?
  - More complicated
  - Details in the book

---

**Cardinality**: how many entities mapped to each other

- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many
Next: Types of Attributes

- Simple vs Composite
  - Single value per attribute?
    - Are parts accessed separately?
    - E.g. accessing first and last names from name

- Single-valued vs Multi-valued
  - E.g. Phone numbers are multi-valued

- Derived
  - If date-of-birth is present, age can be derived
  - Can help in avoiding redundancy, enforcing constraints etc...

Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
- Relational Query Languages
- SQL Basics
- Formal Semantics of SQL
E-R Diagram for a University Enterprise

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes
Composite, Multivalued, and Derived

**instructor**

- ID
- name
- first_name
- middle_initial
- last_name
- address
- street
  - street_number
  - street_name
  - apt_number
- city
- state
- zip
  { phone_number }
- date_of_birth
- age ( )

**Relationship Sets with Attributes**

**instructor**

- ID
- name
- salary

**advisor**

**student**

- ID
- name
- tot_cred
Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “course_id” and “prereq_id” are called roles.

Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.
- One-to-one relationship:
  - A student is associated with at most one instructor via the relationship advisor
  - A student is associated with at most one department via stud_dept
One-to-One Relationship

- one-to-one relationship between an instructor and a student
  - an instructor is associated with at most one student via advisor
  - and a student is associated with at most one instructor via advisor

One-to-Many Relationship

- one-to-many relationship between an instructor and a student
  - an instructor is associated with any number (including 0) of students via advisor
  - a student is associated with at most one instructor via advisor,
Many-to-One Relationships

- In a many-to-one relationship between an instructor and a student,
  - an instructor is associated with at most one student via advisor,
  - and a student is associated with several (including 0) instructors via advisor.

Many-to-Many Relationship

- An instructor is associated with any number (possibly 0) of students via advisor.
- A student is associated with any number (possibly 0) of instructors via advisor.
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

**Participation of an Entity Set in a Relationship Set**

- Total participation (indicated by **double line**): every entity in the entity set participates in at least one relationship in the relationship set
- E.g., participation of section in sec_course is total
  - every section must have an associated course
- Partial participation: some entities may not participate in any relationship in the relationship set
- Example: participation of course in section is partial
Relationship Set Keys

- What attributes are needed to describe a relationship completely and uniquely?
  - Union of primary entities keys and relationship attributes
    - Example: One specific relationship might encode:
      - keleher became bengfort’s advisor Nov 1, 2018

- \{i.ID, s.ID, date\} describes the relationship completely

- However! date is not included in the key.

Relationship Set Keys

- Is \{i.ID, s.ID, date\} a candidate key for advisor?

- No. Attribute “date” can be removed from this set without losing key-ness

- An individual relationship is completely identified by the two endpoints,
  - Relationship attributes might be needed to completely describe a relationship.
  - “date” might be when the advisor relationship started

- In fact, union of primary keys of associated entities is always a superkey
Relationship Set Keys

- If one-to-many relationship, \{s.ID\} is a candidate key for advisor
  - Assumes an instructor can advise many students, but students each have at most one advisor.

Relationship Set Keys

- If one-to-one relationship, \{s.ID\} is a candidate key
  - But so it is \{i.ID\}
  - If one-to-one relationship, either \{i.ID\} or \{s.ID\} sufficient
    - Since a given advisor can only advise a single student, she can only participate in one relationship
    - Same for student
Relationship Set Keys

- General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the entity set on the many side
  - many-to-many: union of primary keys of the associate entity sets

- n-ary relationships
  - More complicated rules

Summary: Keys for Relationship Sets

- Primary keys of entity sets together form super key of a relationship set.
  - \((s_{id}, i_{id})\) is the super key of \textit{advisor}
  - \textit{NOTE: this means an entity pair have at most one relationship in a relationship set.}
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting.
    - Fix: use a \textit{multivalued attribute}.

- Must consider the mapping cardinality of the relationship set when identifying candidate keys.

- Consider relationship set semantics in selecting the \textit{primary key} if more than one candidate key.
E-R Diagram with a Ternary Relationship

Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint.
- E.g., an arrow from `proj_guide` to `instructor` indicates each student has at most one guide for a project.
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g., a ternary relationship `R` between `A`, `B` and `C` with arrows to `B` and `C` could mean
    1. each `A` entity is associated with a unique entity from `B` and `C` or
    2. each pair of entities from `(A, B)` is associated with a unique `C` entity, and each pair `(A, C)` is associated with a unique `B`
  - Each alternative has been used in different formalisms.
  - To avoid confusion we outlaw more than one arrow.
Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.

  - The existence of a weak entity set depends on the existence of an *identifying entity set*
    - It must relate to the identifying entity set via a one-to-many relationship set from the identifying to the weak entity set
    - Weak side of relationship set must be *total*
    - **Identifying relationship** depicted using a double diamond

  - The **discriminator** *(or partial key)* of a weak entity set is the set of attributes that help distinguish among all the entities of a weak entity set.

  - Primary key of a weak entity set is:
    - primary key of the strong entity set on which the weak entity set is existent-dependent,
    - plus the weak entity set’s discriminator.

Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.

- We put the identifying relationship of a weak entity in a double diamond.

- Primary key for **section** – *(course_id, sec_id, semester, year)*
Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

- If course_id were explicitly stored, section could be made a strong entity,
  - but then the relationship between section and course would be duplicated by an implicit relationship defined by the attribute course_id common to course and section

- Duplication is bad

Summary of E-R Notation

- **E** entity set
- **R** relationship set
- **R** identifying relationship set for weak entity set
- **R to E** total participation of entity set in relationship

attributes:
- simple (A1)
- composite (A2) and multivalued (A3)
- derived (A4)

primary key

discriminating attribute of weak entity set
Symbols Used in E-R Notation (Cont.)

- `R` for many-to-many relationship
- `R` for many-to-one relationship
- `E` for cardinality limits
- Role indicator

E-R Diagram for a University Enterprise

- course_dept
- instructor
- teaches
- sec_course
- section
- takes
- sec_slot
- classroom
- department
- inst_dept
- stud_dept
- advisor
- student
- grade
- time_slot
- sec_class