Additional Operators: Joins

- Natural join (⋈)
  - A Cartesian product with equality condition on common attributes
  - Example:
    - if r has schema R(A, B, C, D), and if s has schema S(E, B, D)
    - Common attributes: B and D
    - Then:
      \[
      r \bowtie s = \Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))
      \]

- SQL Equivalent:
  - select r.A, r.B, r.C, r.D, s.E from r, s where r.B = s.B and r.D = s.D, OR
  - select * from r natural join s
Additional Operators: Joins

- **Equi-join**
  - A join that only has equality conditions

- **Theta-join** \( (\bowtie_\theta) \)
  - \( r \bowtie_\theta s = \sigma_\theta(r \times s) \)

- **Left outer join** \( (\bowhook) \)
  - Say \( r(A, B), s(B, C) \)
  - We need to somehow find the tuples in \( r \) that have no match in \( s \)
  - Consider: \( (r - \pi_{r.A, r.B}(r \bowhook s)) \)
  - We are done:
    \[
    (r \bowhook s) \cup \rho_{\text{temp}} (A, B, C) \ ( (r - \pi_{r.A, r.B}(r \bowhook s)) \times \{\text{NULL}\} )
    \]

---

Additional Operators: Join Variations

- **Tables:** \( r(A, B), s(B, C) \)

<table>
<thead>
<tr>
<th>name</th>
<th>Symbol</th>
<th>SQL Equivalent</th>
<th>RA expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross product</td>
<td>( \times )</td>
<td>cross join</td>
<td>( r \times s )</td>
</tr>
<tr>
<td>natural join</td>
<td>( \bowcirc )</td>
<td>natural join</td>
<td>( \pi_{r.A, r.B, s.C}(r \times s) )</td>
</tr>
<tr>
<td>equi-join</td>
<td>( \bowtie_\theta )</td>
<td>(theta must be equality)</td>
<td>( \sigma_\theta(r \times s) )</td>
</tr>
<tr>
<td>theta join</td>
<td>( \bowtie_{\theta} )</td>
<td>from .. where ( \theta; )</td>
<td>( \sigma_\theta(r \times s) )</td>
</tr>
<tr>
<td>left outer join</td>
<td>( r \bowhook s )</td>
<td>left outer join (with “on”)</td>
<td>(see previous slide)</td>
</tr>
<tr>
<td>full outer join</td>
<td>( r \bowhook s )</td>
<td>full outer join (with “on”)</td>
<td>-</td>
</tr>
<tr>
<td>(left) semijoin</td>
<td>( r \bowtimes s )</td>
<td>subquery</td>
<td>( \pi_{r.A, r.B}(r \bowtimes s) )</td>
</tr>
<tr>
<td>(left) antijoin</td>
<td>( r \bowminus s )</td>
<td>subquery</td>
<td>( r - \pi_{r.A, r.B}(r \bowminus s) )</td>
</tr>
</tbody>
</table>
Additional Operators: Division

- Suitable for queries that have “for all”
  - $r \div s$
- Think of it as “opposite of Cartesian product”
  - $r \div s = t \iff t \times s \subseteq r$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td>α</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>α</td>
<td>1</td>
<td>β</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>α</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>β</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>β</td>
<td>2</td>
<td>γ</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cc}
A & B \\
\alpha & 1 \\
\beta & 2 \\
\end{array} \div \begin{array}{cc}
C & D & E \\
\alpha & 10 & a \\
\beta & 10 & a \\
\beta & 20 & b \\
\gamma & 10 & b \\
\end{array}
\]

Relational Algebra Examples

Find all loans of over $1200$

$\sigma_{\text{amount} > 1200} (\text{loan})$

Find the loan number for each loan of an amount greater than $1200$

$\Pi_{\text{loan-number}} (\sigma_{\text{amount} > 1200} (\text{loan}))$

Find the names of all customers who have a loan, an account, or both, from the bank

$\Pi_{\text{customer-name}} (\text{borrower}) \cup \Pi_{\text{customer-name}} (\text{depositor})$
Relational Algebra Examples

Find the names of all customers who have a loan and an account at bank.
\[ \Pi_{\text{customer-name}} (\text{borrower}) \cap \Pi_{\text{customer-name}} (\text{depositor}) \]

Find the names of all deposit customers who have a loan at the Perryridge branch.
\[ \Pi_{\text{customer-name}} (\sigma_{\text{branch-name} = \text{"Perryridge"}} ( \sigma_{\text{borrower. customer-name} = \text{loan. customer-name}} (\text{borrower} \times \text{loan}))) \]

Find the largest account balance
1. Rename the account relation to \( d \)
\[ \Pi_{\text{balance}} (\text{account}) - \Pi_{\text{account.balance}} ( \sigma_{\text{account.balance} < d.\text{balance}} (\text{account} \times \rho_d (\text{account}))) \]

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

\[ \Pi_{F_1, F_2, \ldots, F_n} (E) \]

- \( E \) is any relational-algebra expression
- Each of \( F_1, F_2, \ldots, F_n \) are are arithmetic expressions involving constants and attributes in the schema of \( E \).
- Given relation instructor\((ID, name, dept\_name, salary)\) where salary is annual salary, get the same information but with monthly salary
\[ \Pi_{ID, name, dept\_name, salary/12} (\text{instructor}) \]
Aggregate Functions and Operations

- An **aggregation function** takes a collection of values and returns a single value.
  - **avg**: average value
  - **min**: minimum value
  - **max**: maximum value
  - **sum**: sum of values
  - **count**: number of values

- **Aggregate operations** in relational algebra

  \[
  G_{G_1}^G_{G_2}^{\cdots}G_{G_n}^{F_1(A_1),F_2(A_2),\ldots,F_n(A_n)}(E)
  \]

  - \(E\) is any relational-algebra expression
  - \(G_1, G_2, \ldots, G_n\) 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 \(G\) (Calligraphic G)

Aggregate Operation – Example

- Relation \(r\):

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

\[ \text{dept\_name} \overline{\text{avg(salary)}} (\text{instructor}) \]

<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

\[ \text{dept\_name} \overline{\text{avg(salary)}} \text{ as avg\_sal} (\text{instructor}) \]
Modification of the Database

- The content of the database may be modified using the following operations:
  - Assignment
    
    \[
    \text{temp}1 \leftarrow R \times S \\
    \text{temp}2 \leftarrow \sigma_{r.A_1 = s.A_1 \land r.A_2 = s.A_2 \land \ldots \land r.A_n = s.A_n} (\text{temp}1) \\
    \text{result} = \Pi_{R \cup S} (\text{temp}2)
    \]
  - Updating
    - same
  - Deletion
    - \( R \rightarrow R - \sigma_{\text{amount} > 1200} (R) \)
    - deletes rows w/ amount > 1200 from R

Multiset Relational Algebra

- Pure relational algebra removes all duplicates
  - e.g. after projection
- Multiset relational algebra retains duplicates to match SQL
  - SQL duplicate retention was initially for efficiency, but is now a feature
- Multiset relational algebra defined as follows
  - selection:
    - result has as many duplicates as 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 \( t1 \) in \( r \), and \( n \) copies of \( t2 \) in \( s \), there are \( m \times n \) copies of \( t1.t2 \) in \( r \times s \)
  - Other operators similar
    - E.g. union: \( m + n \) copies, intersection: \( \min(m, n) \) copies
      difference: \( \min(0, m - n) \) copies
Outline

- Overview of modeling
- Relational Model
- Relational Query Languages
- SQL Basics
- Relational Algebra
- E/R Model

Entity-Relationship (E-R) Diagrams
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:**
  - 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
Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes
Next: Types of Attributes

- Simple vs Composite
  - Single value per attribute?

- 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...

Composite, Multivalued, and Derived Attributes

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
</tbody>
</table>
{ phone_number }
| date_of_birth|
| age ( )    |
Entity Keys

- **Superkey**
  - any attribute set that can distinguish entities

- **Candidate key**
  - a minimal superkey
    - Can't remove any attribute and preserve key-ness
    - \{cust-id, age\} not a candidate key
    - \{cust-name, cust-city, cust-street\} is
      - assuming cust-name is not unique

- **Primary key**
  - Candidate key chosen as the key by DBA
  - Underlined in the ER Diagram

Relationship Set Keys

- What attributes are needed to represent a relationship completely and uniquely?
  - Union of primary entities keys, and relationship attributes

- \{cust-id, access-date, account number\} describes a relationship completely
Relationship Set Keys

- Is \{cust-id, access-date, account number\} a candidate key?
  - No. Attribute \textit{access-date} can be removed without losing key-ness
  - In fact, union of primary keys of associated entities is always a superkey

\[\text{customer} \rightarrow \text{account} \text{ has}\]

not a real E/R diagram

Relationship Set Keys

- Is \{cust-id, account-number\} a candidate key?
  - Depends…

\[\text{customer} \rightarrow \text{account} \text{ has}\]

not a real E/R diagram
Is \{cust-id, account-number\} a candidate key?

- Depends…

- If one-to-many relationship, \{account-number\} is a candidate key
  - A given customer can have many accounts, but at most one account holder per account allowed

- If one-to-one, either \{cust-id\} or \{account-number\} sufficient
  - Since a given customer can only have one account, she can only participate in one relationship
  - Ditto account
Relationship Set Keys

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

- n-ary relationships
  - More complicated rules

Summary: Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (s_id, i_id) is the super key of advisor
  - NOTE: this means a pair of entities can have at most one relationship in a particular 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 multivalued attribute.
- Must consider the mapping cardinality of the relationship set when identifying candidate keys.
- Need to consider semantics of relationship set in selecting the primary key in case of more than one candidate key.
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.

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 have to do it; NOT GOOD
  - Remember: If not represented in conceptual model, the domain knowledge may 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

- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many

Arrow side is the “one”!
One-to-One Relationship

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

```
instructor
   ID
   name
   salary

advisor

student
   ID
   name
   tot_cred
```

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

```
instructor
   ID
   name
   salary

advisor

many

student
   ID
   name
   tot_cred
```
Many-to-One Relationships

- Many-to-one relationship between instructor and 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