Database Languages (Chapter 2)

- Structure-oriented:
  - Data definition language (DDL): “A language that allows … the description and naming of the entities, attributes, and relationships required for the application, with any associated integrity and security constraints.”
- Data-oriented:
  - Data manipulation language (DM): “A language that provides a set of operations to support the basic data manipulation operations on the data held in the database.”
  - Four essential operations: Insert, Modify, Delete, Retrieve

Database Languages (Chapter 2)

- SQL (Structured Query Language) actually acts as both a DDL and a DML
  - DDL: Chapter 7
  - DDL: Chapter 6

Creating Databases: Defining Tables

Key DDL Functionality To Start:

- Create database (namedcollection of relations/tables)
- Create domains for attributes
- Create named tables, specifying
  - Attributes and associated domains
  - Primary keys
  - Foreign keys and references
  - Integrity constraints decisions
- Can also remove (“Drop”) databases, tables, domains
- Can modify (“Alter”) tables and domains

Database Virtual Machine

- Will be provided access to a virtual machine supporting MySQL next week
- Has a private WFU address
  - Only available from on-campus or using VPN (think about what that means for your work/study habits)
- Ubuntu Linux, terminal access
  - Let me know if uncomfortable in Linux

Relational Algebra

(Dealing with Tuples Themselves: Data Manipulation)

Relational Algebra

- Think in terms of *operators* and *operands*
  - *Relations* are the *operands*
  - Need to define *operators*
    - *Unary:* apply to one relation
    - *Binary:* apply to two relations

- *Closure:* Applying a relational operator to its relation operands results in another relation

Example Relations

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

Course

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191

Enrollment

studentID	courseID
1123	06902
1129	06902
1145	06902
1123	06903
1145	06903
1123	06904
1129	06904

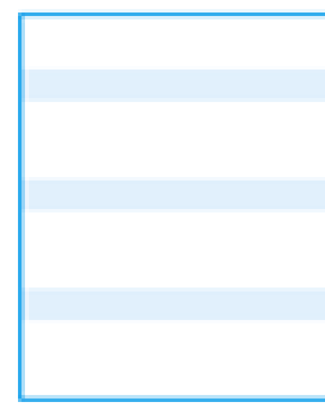
Rename Operator

- Assume there is a *rename* operator, which allows the *name of a rela*tion and *names of the attributes of a relation* to be modified without changing the actual structure of the

- Allows us to deal with some small issues down the road…

Fundamental Operators: Unary

- *Selection (restriction):* $R2 = \sigma_p(R1)$
  - Parameterized by predicate *p*
    - *p* is a predicate defined over the attributes of *R* which returns *true* or *false* for each tuple in the relation depending on tuple values
  - Takes an input relation R
  - Returns a new relation of all tuples from R that satisfy the predicate (make predicate true)
  - Closest to our simple idea of querying
  - Essentially, row selection



Fundamental Operators: Selection

- Selection example:
  - Select all students with a GPA > 3.0 $\sigma_{GPA>3.0}(Student)$

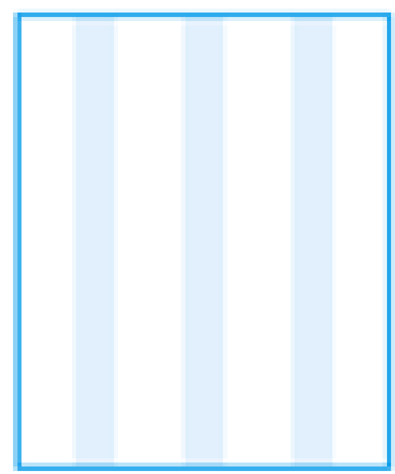
studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1145	Brady	Susan	4	CSC	3.8

  - Select all students with year < 4 and major = CSC $\sigma_{(year<4)\ \&\&\ GPA)\ \&\&\ (major=\text{“CSC”})}(Student)$

studentID	lastName	firstName	year	major	GPA

Fundamental Operators: Unary

- *Projection:* $R2 = \Pi_{col1, \ldots, coln}(R1)$
  - Parameterized by a set of attribute names
  - Takes an input relation R
  - Returns a new relation of values from all tuples, but only containing the attributes of interest *and with duplicates removed*
  - Essentially, column selection



Fundamental Operators: Projection

- Projection example:
  - Show all majors and GPAs of students: $\Pi_{major,GPA}(Student)$

major	GPA
CSC	3.5
MTH	2.9
CSC	3.8

  - Show all departments offering courses: $\Pi_{dept}(Course)$

dept.
CSC

Nesting

- *Nesting:* Ability to apply output relation from one operator as an input to another operator (due to closure: all operators return relations)
  - Can abstract multiple applications to a higher-level operator: $h(x) = g(f(x))$

Nesting Example

- Show the <u>lastname</u> of <u>students</u> with a <u>GPA > 3.0</u>

$$\Pi_{lastName}(\sigma_{GPA>3.0}(Student))$$

$$\sigma_{GPA>3.0}(Student)$$

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1145	Brady	Susan	4	CSC	3.8

$$\Pi_{lastName}$$

lastName
Smith
Brady

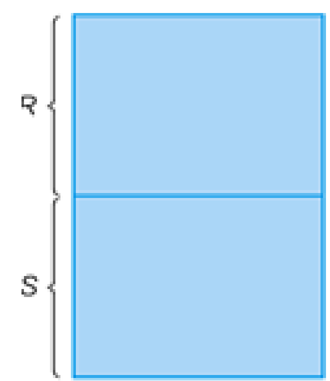
Is this the only way we could have nested the operators to get this answer?

Fundamental Operators: Binary

- A key idea: *union-compatible*
  - Two relations are union-compatible if:
    - They have the same number of attributes
    - Corresponding attributes (position-wise) have the same domain
  - Note, no rules concerning:
    - Attribute names

Fundamental Operators: Binary

- Union: R ∪ S
  - Returns a new relation that contains all of the tuples from R and all of the tuples from S, but without duplicates
  - Resulting union relation could be of cardinality:
    - |R| (S was a duplicate of R) to
    - |R|+|S| (S had nothing in common with R)



Fundamental Operators: Union

- Union example:

CSCourses

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191
06906	CSC	355

MathCourses

courseID	dept.	number
06801	MTH	112
06805	MTH	113
06807	MTH	121
06808	MTH	205
06906	MTH	355

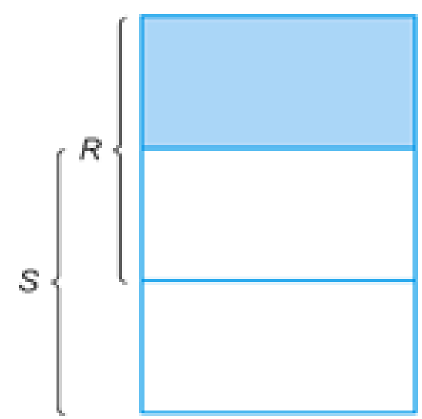
$\Pi_{courseID} R \cup \Pi_{courseID} S$

courseID
06801
06805
06807
06808
06906
06902
06903
06904
06905

"All courses offered by CS and all courses offered by Math"

Fundamental Operators: Binary

- Difference: R - S
  - Returns a new relation that contains the tuples that are in relation R but not in S.
    - Essentially, the tuples unique to R
    - Note ordering of R and S is important here
  - Resulting difference relation could be of cardinality:
    - 0 (S was a duplicate of R) to
    - |R| (S had nothing in common with R)



Fundamental Operators: Difference

- Difference example:

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191
06906	CSC	355

courseID	dept.	number
06801	MTH	112
06805	MTH	113
06807	MTH	121
06808	MTH	205
06906	MTH	355

$\Pi_{courseID}\mathrm{R} - \Pi_{courseID}\mathrm{S}$

courseID
06902
06903
06904
06905

How could we describe the output of this operator?

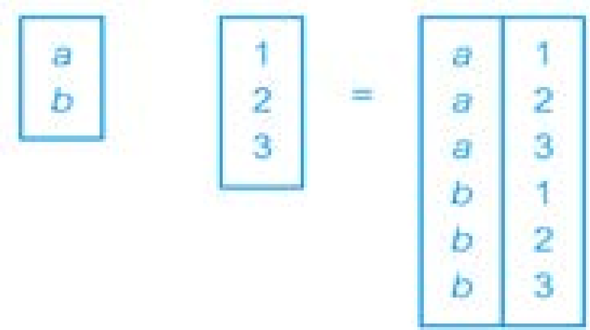
"All courses… "

Fundamental Operators: Binary

- Cartesian Product: R X S
  – Returns a new relation that is the concatenation of each tuple in relation R with each tuple in relation S

  – Resulting union relation will be of cardinality:
    - |R|*|S|

  – Attributes with same name in R and S should be tagged with Relation ID (R,S) to differentiate



Fundamental Operators: Cartesian Product

Course X Enrollment

Course ID	dept.	number	student ID	Course ID
06902	CSC	221	1123	06902
06903	CSC	231	1123	06902
06904	CSC	241	1123	06902
06902	CSC	221	1129	06902
06903	CSC	231	1129	06902
06904	CSC	241	1129	06902
06902	CSC	221	1145	06902
06903	CSC	231	1145	06902
06904	CSC	241	1145	06902
06902	CSC	221	1123	06903
06903	CSC	231	1123	06903
06904	CSC	241	1123	06903

courseID	dept.	number	student ID	Course ID
06902	CSC	221	1145	06903
06903	CSC	231	1145	06903
06904	CSC	241	1145	06903
06902	CSC	221	1123	06904
06903	CSC	231	1123	06904
06904	CSC	241	1123	06904
06902	CSC	221	1129	06904
06903	CSC	231	1129	06904
06904	CSC	241	1129	06904

courseID	dept.	number	student ID	Course ID
06905	CSC	191	1123	06902
06905	CSC	191	1129	06902
06905	CSC	191	1145	06902
06905	CSC	191	1123	06903
06905	CSC	191	1145	06903
06905	CSC	191	1123	06904
06905	CSC	191	1129	06904

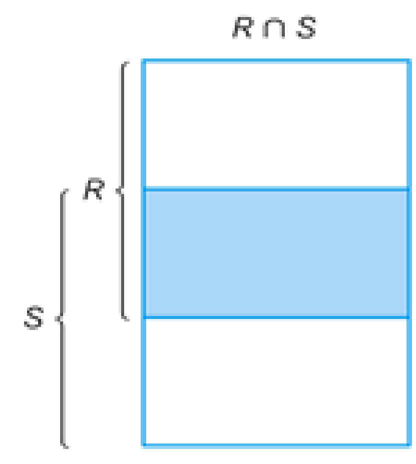
Forgive the differences in capitalization
and the weird ordering
Also, the courseIDs would need to be renamed
One way, tag with relation they were in

Higher Order Operators

- All other operators of interest can be constructed from these fundamental operators
  - Intersection
  - Joins

Higher Order Operators: Binary

- Intersection: R ∩ S
  - Returns a new relation that contains all of duplicate tuples from R and S
  - Resulting intersection relation could be of cardinality:
    - 0 (S had nothing in common with R)   to
    - |R| (S and R were duplicates to begin with)



  - Could define employing *difference*:

    R – (R – S)

    →R minus the things that are unique to R

Higher Order Operators: Intersection

- Intersection example:

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191
06906	CSC	355

courseID	dept.	number
06801	MTH	112
06805	MTH	113
06807	MTH	121
06808	MTH	205
06906	MTH	355

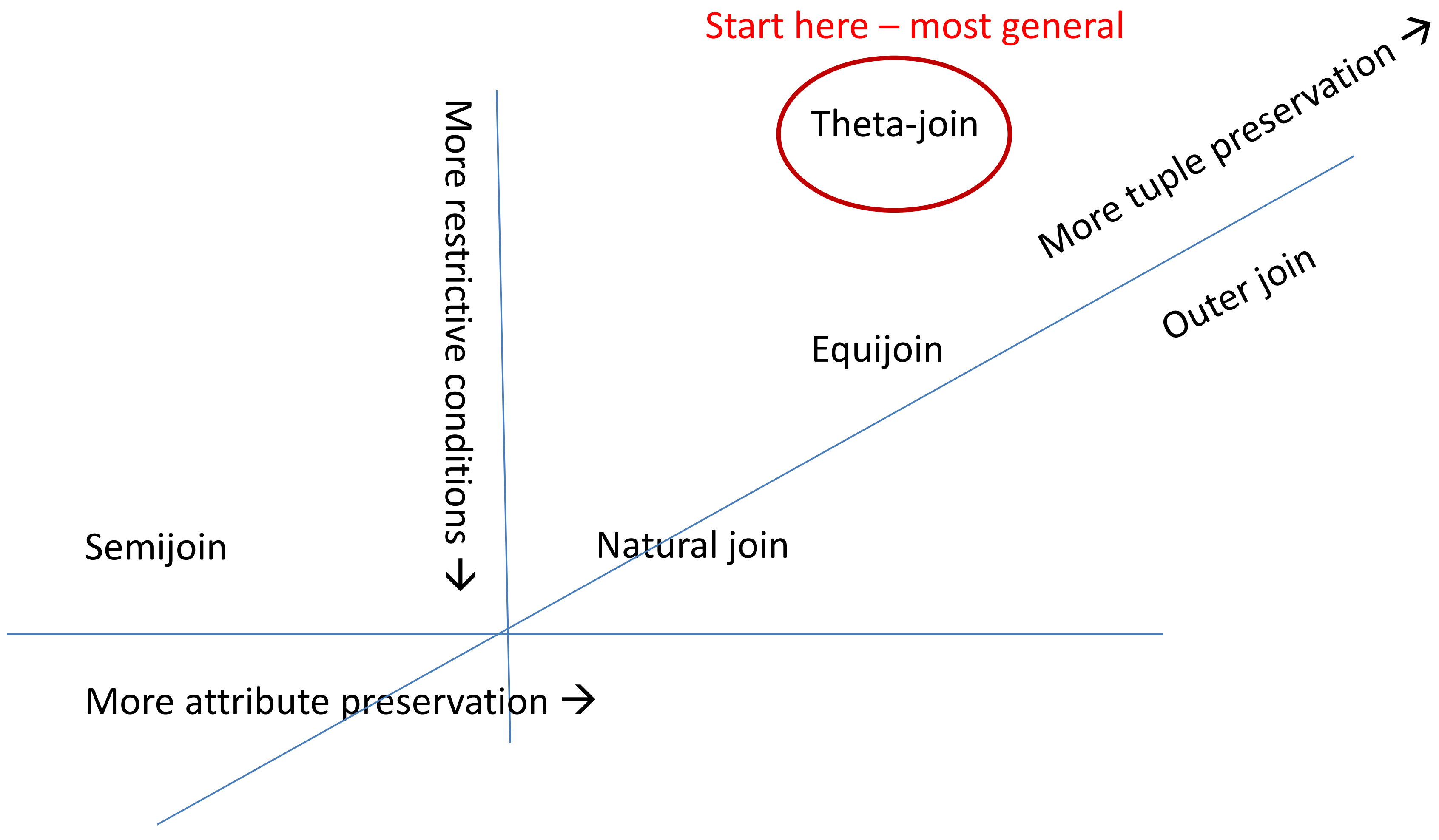
$$\Pi_{courseID} R \cap \Pi_{courseID} S$$

courseID
06906

How could we describe the output of this operator?

"All courses… "

Joins in 3-dimensions



Join Notation

- Everything is based off of join symbol $\bowtie$
  - Join with a predicate: $\bowtie_F$
  - Natural join: $\bowtie$ predicate assumed to be equality
  - Left outer join: $⟕$ want to preserve left relation tuples (branches left)
  - Right outer join: $⟖$ want to preserve right relation tuples (branches right)
  - Full outer join: $⟗$ preserve left and right tuples (branches both ways)
  - Semijoin – $\ltimes$ throw away attributes on right (discard right connector – book has just left triangle)

Doesn’t exactly match book, but close

Higher Order Operators: Join Operators

- Joins: Employed when want to create new relations with data from two relations, but only want those tuples containing certain properties

- Essentially all joins are fundamentally:
  - Cartesian product, followed by Selection
    $R3 = \sigma_p(R1 \times R2)$

- Multiple different applications lead to definition of several special variations of join operators

Higher Order Operators: Join Operators

- Theta-join (θ-join): $\mathbf{R} \bowtie_{\mathbf{F}} \mathbf{S}$
  – Returns a new relation that contains the tuples out of the Cartesian product of R and S satisfying predicate F, where F is defined over comparisons R.a θ S.b and θ is limited to { <, >, <=, >=, =, !=)

- Equivalent to σ_F (R X S)

- Note resulting relation could be empty, if predicate not satisfied by any tuple

Higher Order Operators: Theta-Join

- θ -join example:

Assume have a table of “Honors” based on GPA, want list of students and their possible Honors

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

Honors

Name	gpaReq
Dean’s List	3.0
President’s List	3.5
Superstar	4.0

$$\text{Student} \bowtie_{\text{Student.GPA} > \text{Honors.gpaReq}} \text{Honors}$$

Higher Order Operators: Theta-Join

Output Relation

Student ID	last Name	first Name	year	major	GPA	name	gpaReq
1123	Smith	Robert	4	CSC	3.5	Dean’s List	3.0
1145	Brady	Susan	4	CSC	3.8	Dean’s List	3.0
1145	Brady	Susan	4	CSC	3.8	President’s List	3.5

Think about the Cartesian product that we would have seen before the selection was employed

Higher Order Operators: Theta-Join

Cartesian Product – look for GPA > gpaReq

Student | Honors

Student ID	last Name	first Name	year	major	GPA	name	gpaReq
1123	Smith	Robert	4	CSC	3.5	Dean's List	3.0
1123	Smith	Robert	4	CSC	3.5	President's List	3.5
1123	Smith	Robert	4	CSC	3.5	Superstar	4.0
1129	Jones	Douglas	3	MTH	2.9	Dean's List	3.0
1129	Jones	Douglas	3	MTH	2.9	President's List	3.5
1129	Jones	Douglas	3	MTH	2.9	Superstar	4.0
1145	Brady	Susan	4	CSC	3.8	Dean's List	3.0
1145	Brady	Susan	4	CSC	3.8	President's List	3.5
1145	Brady	Susan	4	CSC	3.8	Superstar	4.0