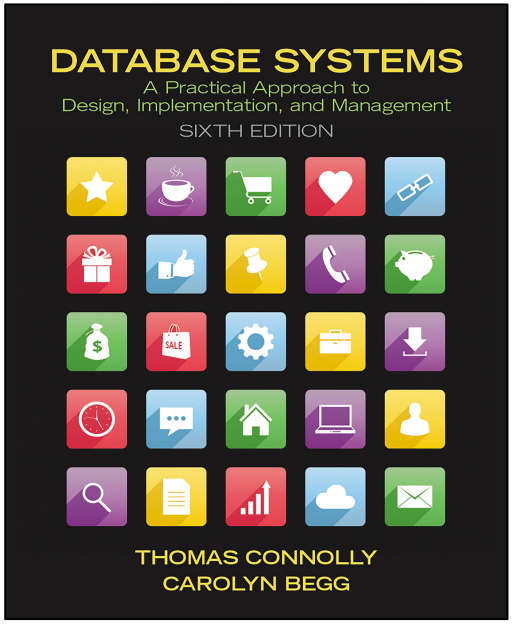
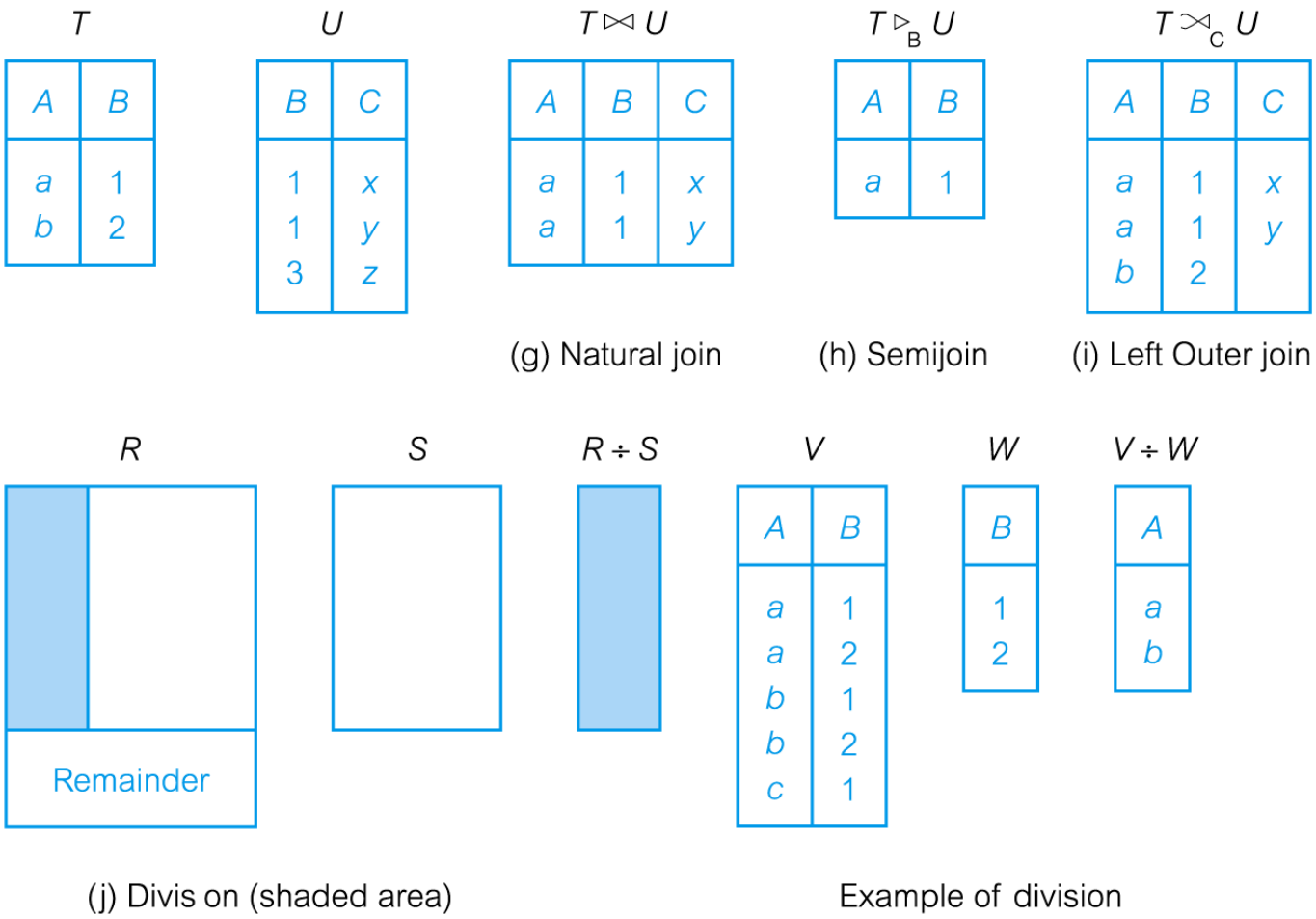
Derived Relational Algebra

Topic 2
Lesson 10 – Relational Algebra

Chapter 5 section 1 Connolly and Begg



Relational Algebra Derived Operations



Other Relational Algebra Operations

Rename – provide a name to an expression
Join – many different versions of JOINs
Aggregation – method to perform a function across tuples – also be able to group tuples before applying the aggregation
Division operation – inverse function to cartesian product

Derived operations

Theta join or a conditional JOIN INNER JOIN : the inner join of A AND B is the cartesian product with a filter on the rows

NATURAL JOIN: the natural join of A AND B is the cartesian product with an equality filter on all the common named columns between the 2 tables

SEMI-JOIN: the semi-join of A and B is the tuples in A are filtered by values from table B

LEFT OUTER JOIN: the left outer join of A and B, is an INNER JOIN result, plus the tuples in A that do not have a corresponding value in B

RIGHT OUTER JOIN: the right outer join of A and B, is an INNER JOIN result, plus the tuples in B that do not have a corresponding value in A

DIVISION: division of A/B, A and B must have a field in common, call the field c. For all tuples in B, there must be a tuple in A for each value in B (for the common field).

JOIN Variations

Various forms of join operation

- Theta join, an INNER JOIN with a conditional
- Equijoin (a particular type of Theta join, conditional is equality)
- Natural join – no conditional is described, the JOIN conditional is determined by the structure of the tables
- Outer join – identify one specific table that must be represented in the results
- Semi-join – all field results from one table in the JOIN

JOIN operation

Join is a derivative of the Cartesian product.

It is equivalent to performing a Selection, using join predicate as the selection criteria, over the cartesian product of the two operand relations.

JOIN is one of the most difficult operations to implement efficiently in a RDBMS and is one reason why RDBMSs have intrinsic performance problems.

Theta JOIN θ-Join

Provides a filter to the cross product of two tables. The filter $R \bowtie_F S$ contain one of the following operations: $(<, \leq, >, \geq, =, \neq)$.

Filters may contain conjunction or disjunction, AND ($\wedge$) and OR (V)

Syntax: $R \bowtie_F S$

Example: Student $\bowtie_{\text{student.id=available\_major.id}}$ available_major

If the filter involves equality, then it is called an **equijoin**

Natural JOIN

A natural join performs an equijoin over all the field names that the two tables have in common. No filter needs to be provided since the common fields determine the filter. The result does NOT contain two copies of the common fields. It removes one copy of the common fields in the result.

Syntax: $R \bowtie S$

Example: Student ⋈ student_major

OUTER JOIN

For the result to contain tuples that do not have matching values in the join column, use Outer join.

There is LEFT OUTER and RIGHT OUTER. (Left) outer join is join in which tuples from R that do not have matching values in common columns of S are also included in result relation.

Syntax: R ⟕ S

Example: Student ⟕ student_major (LEFT)

Includes students who do not have a major

Student ⋈ student_major ⟖ available_major (RIGHT)

Included majors that a student has not declared

Semi-join

Defines a relation that contains the tuples of R that participate in the join of R with S. S provides a filter for the R tuples.

Syntax: $R \triangleright_F S$

Example: student $\triangleright_{\text{student.sid = student\_major.sid}}$ student_major

Applying Aggregate Operations

Applies a list of aggregate operations to a collection of tuples.
The AL (aggregate list) list contains pairs of operations and attribute names:
(<aggregate_function>, <attribute>) pairs.

SYNTAX:

$$\Im_{AL}(R)$$

Supported aggregate functions are: COUNT, SUM, AVG, MIN, and MAX.

Example: Aggregate Function

Count the number of schools in the student table.

sid	Name	School	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
2	Shah	D'Amore McKim	64	128	2019
3	Li	Khoury	50	120	2020

$\mathfrak{F}_{(COUNT,\ school)}$ (student)

Select count(school)
FROM student;

COUNT(school)
2

Grouping Operation

- Groups tuples of R by grouping attributes, GA, and then applies aggregate function list, AL, to define a new relation.
- AL contains one or more (<aggregate_function>, <attribute>) pairs.
- Resulting relation contains the grouping attributes, GA, along with results of each of the aggregate functions.

– SYNTAX: $_{GA}\Im_{AL}(R)$

Example: Aggregate with GROUP BY

Count the number of students per school.

sid	Name	School	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
2	Shah	D'Amore McKim	64	128	2019
3	Li	Khoury	50	120	2020

school $\mathfrak{F}$(COUNT, sid) (student)

school	COUNT(sid)
Khoury	2
D'Amore McKim	1

Rename Operation

For many operations, an alias or an attribute name can be assigned to either a relation or an attribute the rename operation, also known as the rho operation (ρ) provides this functionality.

Rename a relation:

Syntax $\rho_{new\_name}(old\_name)$

Rename an attribute:

Syntax $\rho_{old \rightarrow new1,..,oln \rightarrow newn}(old\_name)$

Rename a Relation result

Syntax $\rho_{new\_name}(old\_name)$ or

$\rho(new\_name, old\_name)$

$\rho_{khoury\_students}(\sigma_{school = \text{'Khoury'}}(student))$

khoury_students

sid	Name	School	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
3	Li	Khoury	50	120	2020

Rename Fields

Syntax $\rho_{\text{old} \rightarrow \text{new1,..,oln} \rightarrow \text{newn}}(\text{old\_name})$

sid	Name	School	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
3	Li	Khoury	50	120	2020

$\rho_{\text{id} \rightarrow \text{student\_id, school} \rightarrow \text{college}}(\text{khoury\_student})$

Student_id	Name	college	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
3	Li	Khoury	50	120	2020

Rename Fields and Table

Syntax $\rho_{new\_name(old \rightarrow new1,..,oln \rightarrow newn)}$ (old_name)

New_name(f1, f2, f3) ← (old_expression)

sid	Name	School	Credits_Earned	Credits_Req	Yr_grad
1	Smith	Khoury	32	120	2019
3	Li	Khoury	50	120	2020

khoury_student (student_id, name, college) ←

$\Pi_{id, name, school}$ ($\sigma_{school = 'Khoury'}$ (student)))

Student_id	Name	college
1	Smith	Khoury
3	Li	Khoury

Division Operation

Division is the inverse operation for the cross product. Defines a relation over the attributes C that consists of set of tuples from R that match a combination of **every** tuple in S.

SYNTAX: $R \div S$

Definition of division using basic operations:

$$T_1 \leftarrow \prod_c (R)$$

$$T_2 \leftarrow \prod_c ((S \times T_1) - R)$$

$$T \leftarrow T_1 - T_2$$

More on this operation later.

Student, Register and Lecturer relations

1. Name of people who registered for ‘History 101’

2. Find the lecturers teaching History 101, whose Students GPA >3.2

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55516	Jones	jones@hist	Feb 11, 1992	2.98
55517	Ali	ali@math	Sep 22, 1989	3.11
55518	Smith	smith@math	Nov 30, 1991	3.32

Sid	CId	LID	Grade
55515	History 101	45	C
5516	History 101	47	a
5515	Music 101	48	B
5516	Biology 220	46	C
55515	Biology 220	46	A
55517	History 101	45	B
55518	Music 101	48	A

LID	Name	CID
45	Fisk	History 101
46	Alder	Biology 220
47	Wong	History 101
48	Foster	Music 101

Summary

In this module you learned:

Relational operators: selection, projection, union, intersect, set difference, cross product, natural join, theta join, equijoin, semi-join, aggregate operations, grouping operations, division.