<https://dbis-uibk.github.io/relax/calc.htm>

**Relational algebra**

Machine generated alternative text:
SELECT LNAME, FNAME 
FROM EMPLOYEE 
WHERE SALARY > C 

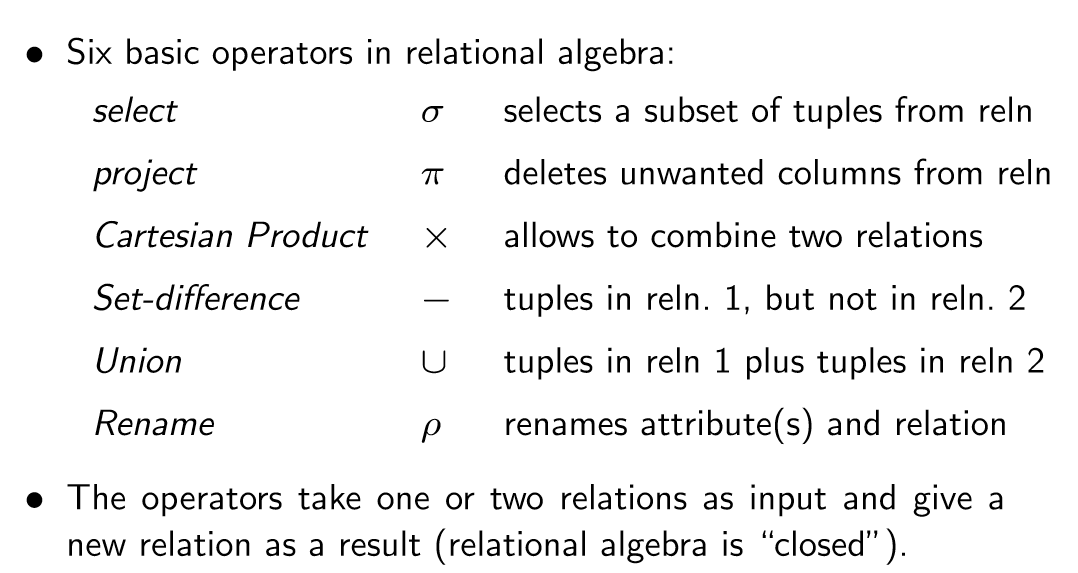
* Relational algebra, first created by [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd) while at IBM, is a family of algebras with a [well-founded semantics](https://en.wikipedia.org/wiki/Well-founded_semantics) used for modelling the data stored in relational databases, and defining queries on it. The main application of relational algebra is providing a theoretical foundation for [relational databases](https://en.wikipedia.org/wiki/Relational_database), particularly [query languages](https://en.wikipedia.org/wiki/Query_language) for such databases, chief among which is [SQL](https://en.wikipedia.org/wiki/SQL).
* Relational Algebra provides a formal foundation for relational model operations. The algebra defines a set of operations for the relational model. It very useful for representing query execution plans, and query optimization techniques. The algebra operations produce new relations. The result of a retrieval is a new relation. A sequence of relational algebra operations forms a **relational algebra expression.**

***Question****: It seems to me that formulating expressions in relational algebra is basically the same as formulating queries in SQL and that much the same thought processes underly both tasks. In particular, I can't really see that knowing relational algebra makes it*easier*to write SQL queries or vice-versa. This makes me wonder if the teaching of relational algebra is just some sort of historical hangover, or if there actually specific benefits to knowing it.*

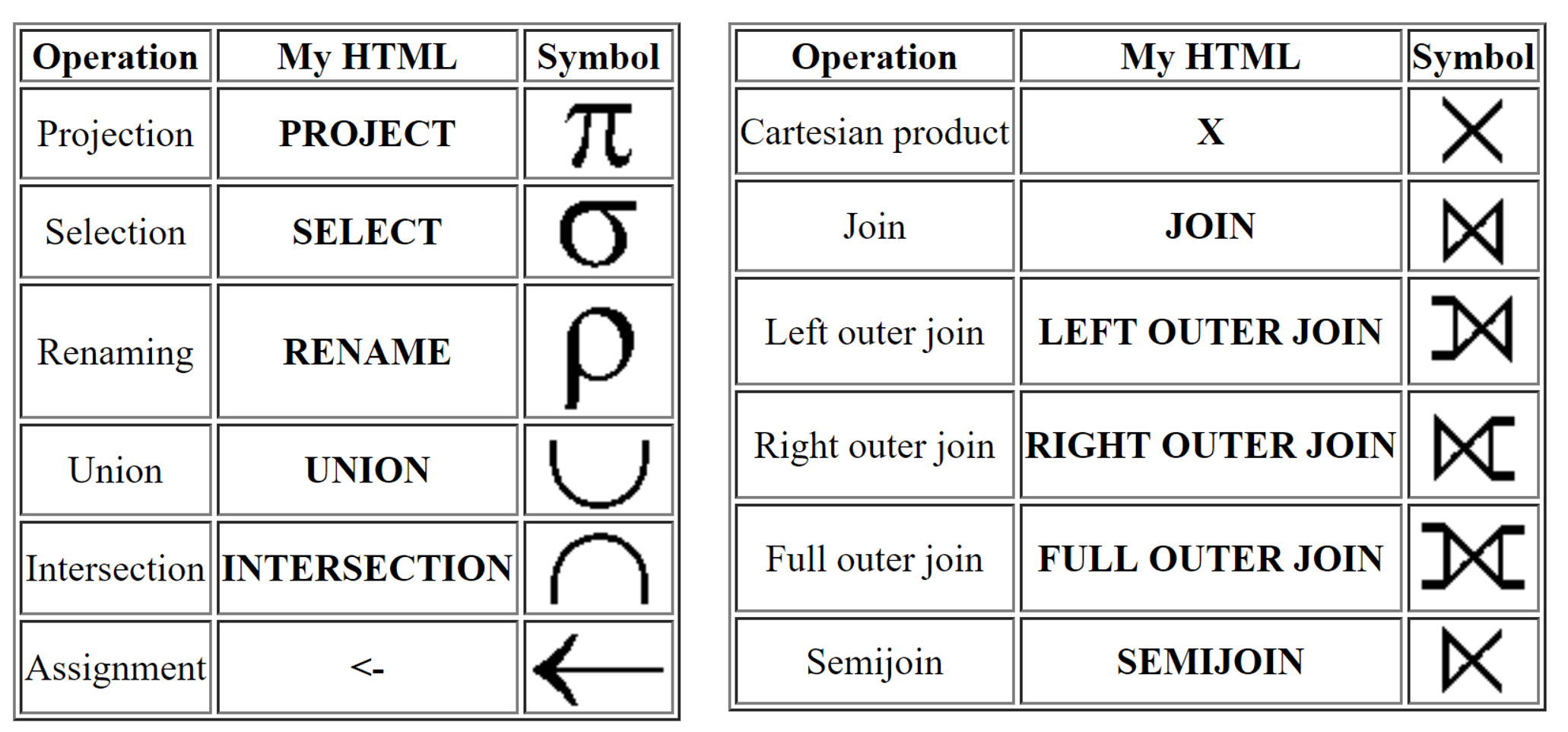
*Answer 1: Yes I would agree with that. But I think any introduction to database course needs to teach relational algebra so that students understand the foundations that led to SQL. Teaching just SQL would be like an introduction to industrial design class teaching just a CAD/CAM tool and not first exposing the students to the fundamental principles of industrial design.*

*Answer 2: If you have formulated a query one way and are getting poor performance, you have the skills to formulate it a different way and know it has the same semantics. Another practical advantage to this is in the specification of database constraints. First, understanding the relational algebra enables you to determine the simplest way to formulate the constraint.*

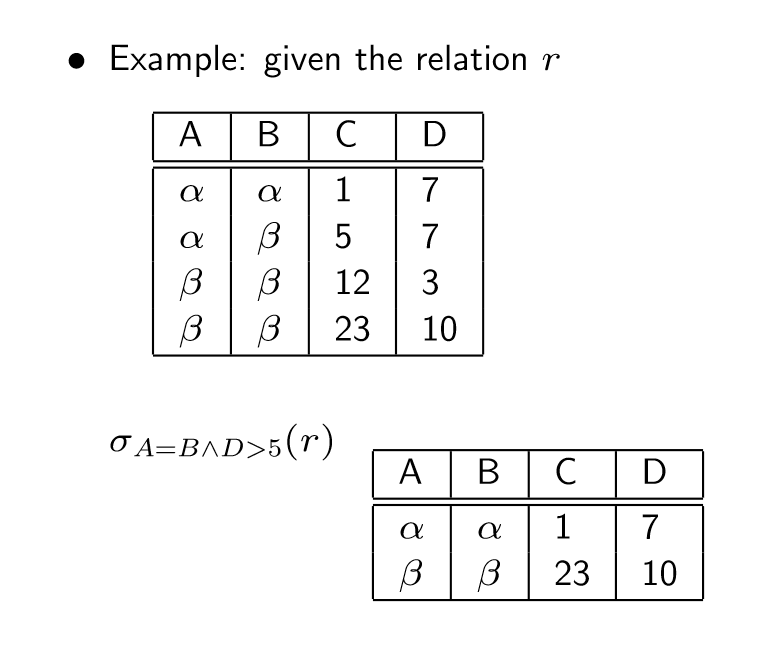
*Answer 3: I strongly feel you can be successful as a DBA without taking a formal course in relational algebra, just like you can be a successful programmer without taking a formal course in discrete math. The*need*to take a course in relational algebra would very much depend on your career path/goals. If someone wanted to get a Masters in Algorithms etc. I would say it is fairly obvious that he needs to take and master discrete math. Similarly if your goal is to write a database engine or be part of the core team working on a major relational database engine then I would strongly recommend mastering relational algebra, stats, and the like.*

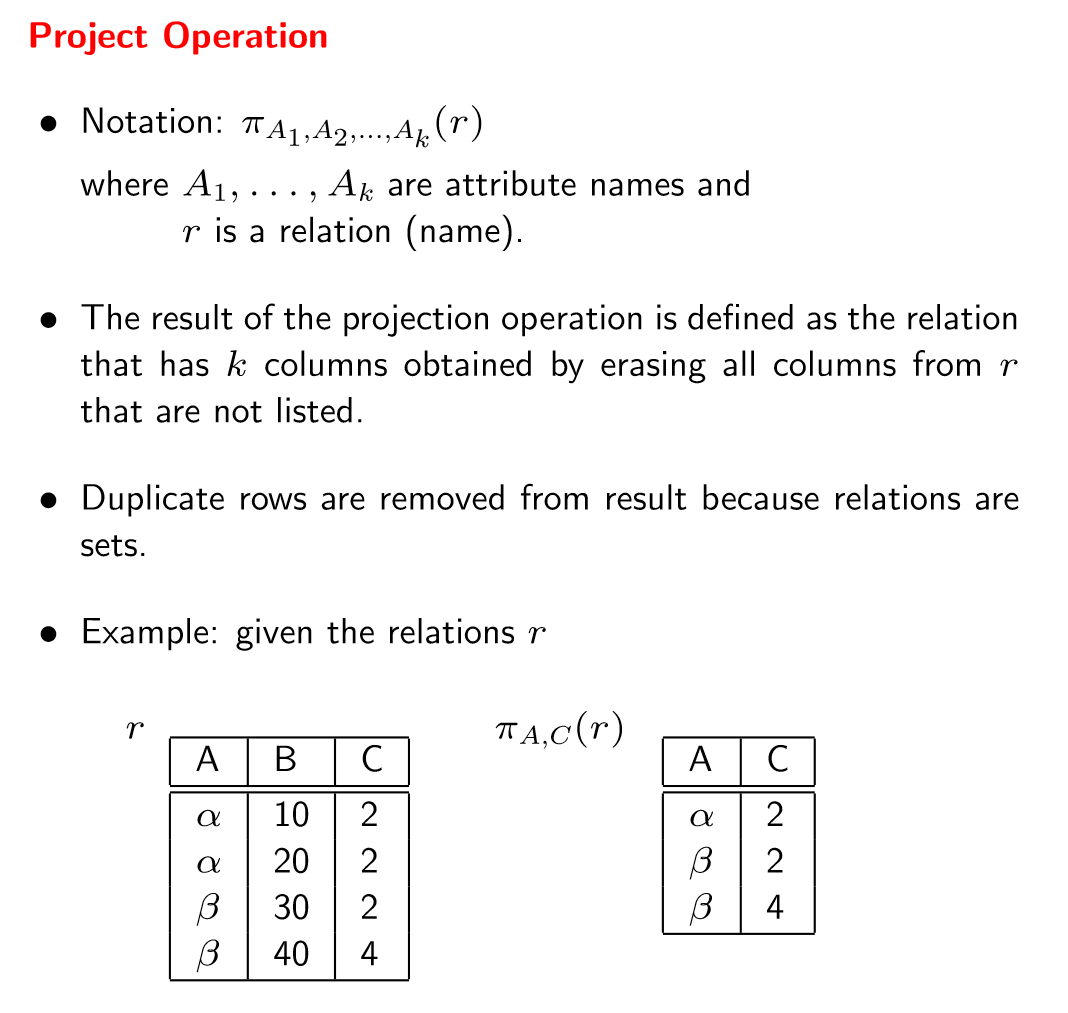


A relation is a set. In mathematics, a set is a collection of distinct objects, considered as an object in its own right.

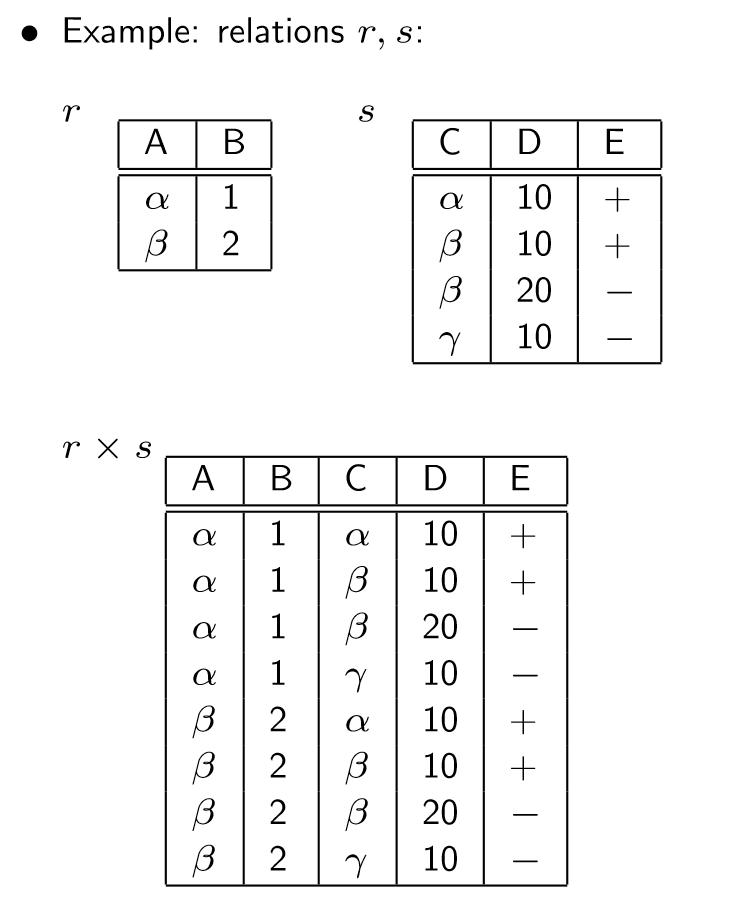




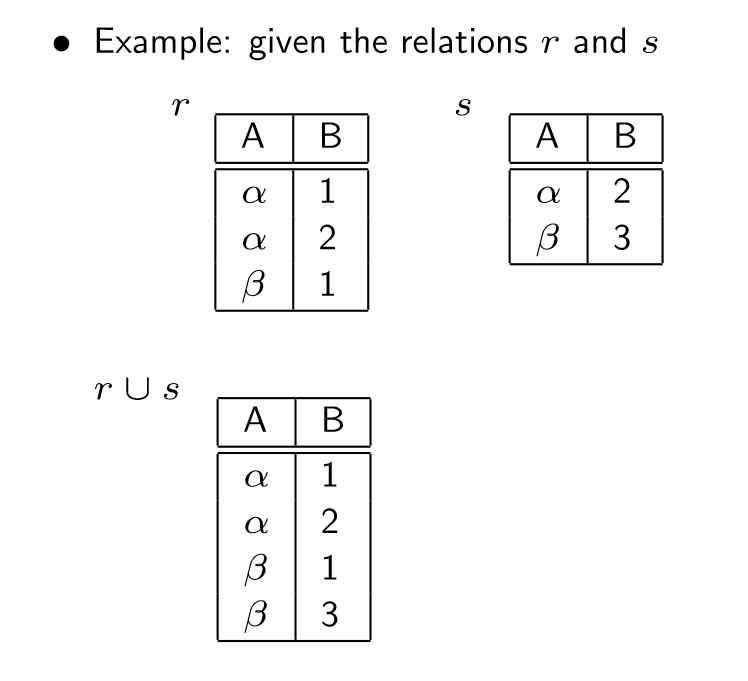


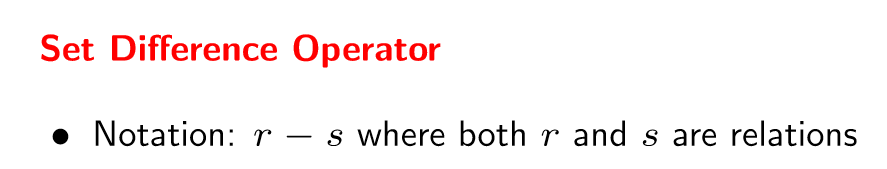


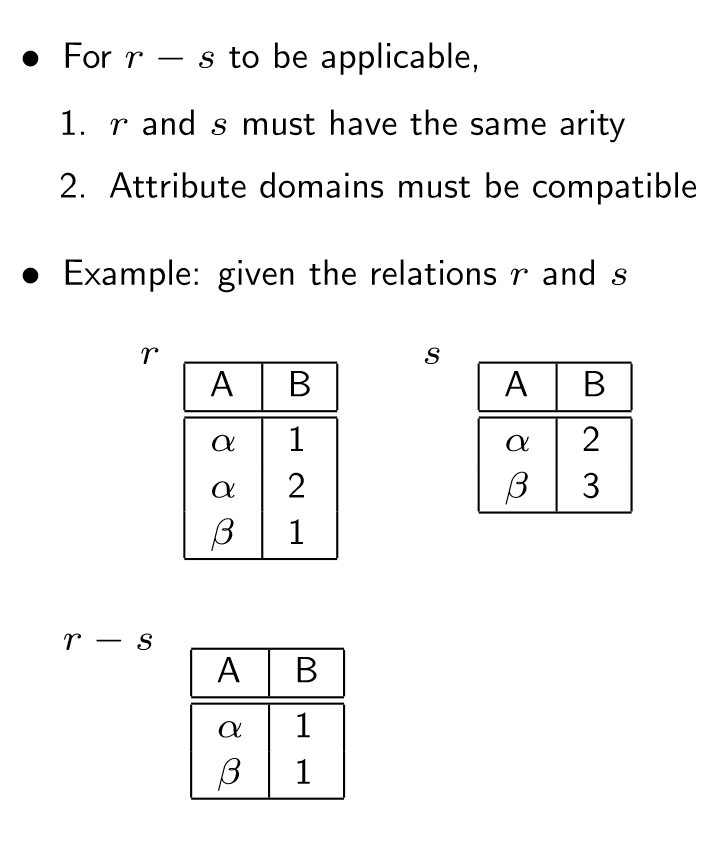


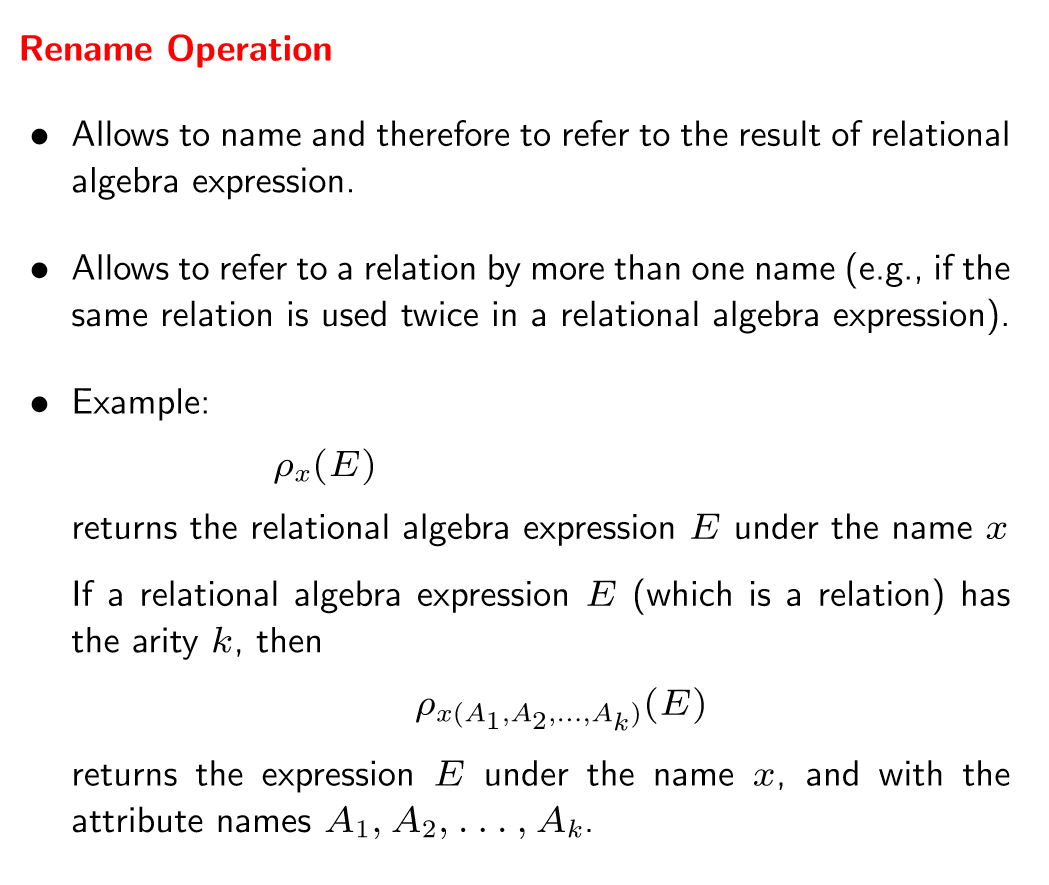












Assume the following relations:

**group:BookStore**

**BOOKS** ={ISBN:NUMBER, TITLE:STRING, PUBLISHER:STRING, YEAR:NUMBER}

**STUDENTS**= {STID:NUMBER, SNAME:STRING, MAJOR:STRING, AGE:NUMBER}

**AUTHORS**= {ANAME:STRING, ADDRESS:STRING}

**BORROWS**= {ISBN:NUMBER, STID:NUMBER, DATE:NUMBER}

**HAS\_WRITTEN**= {ISBN:NUMBER, ANAME:STRING}

**DESCRIBES**= {ISBN:NUMBER, KEYWORD:STRING}

**BOOKS** (ISBN, Title, Publisher, Year)

**STUDENTS** (StId, SName, Major, Age)

**AUTHORS** (AId, AName, Address)

**BORROWS** (ISBN, StId, Date)

**HAS**\_**WRITTEN** (ISBN, AId)

**DESCRIBES** (ISBN, Keyword)

-- 1. List the year and title of each book.

-- 2. List all information about students whose major is CS

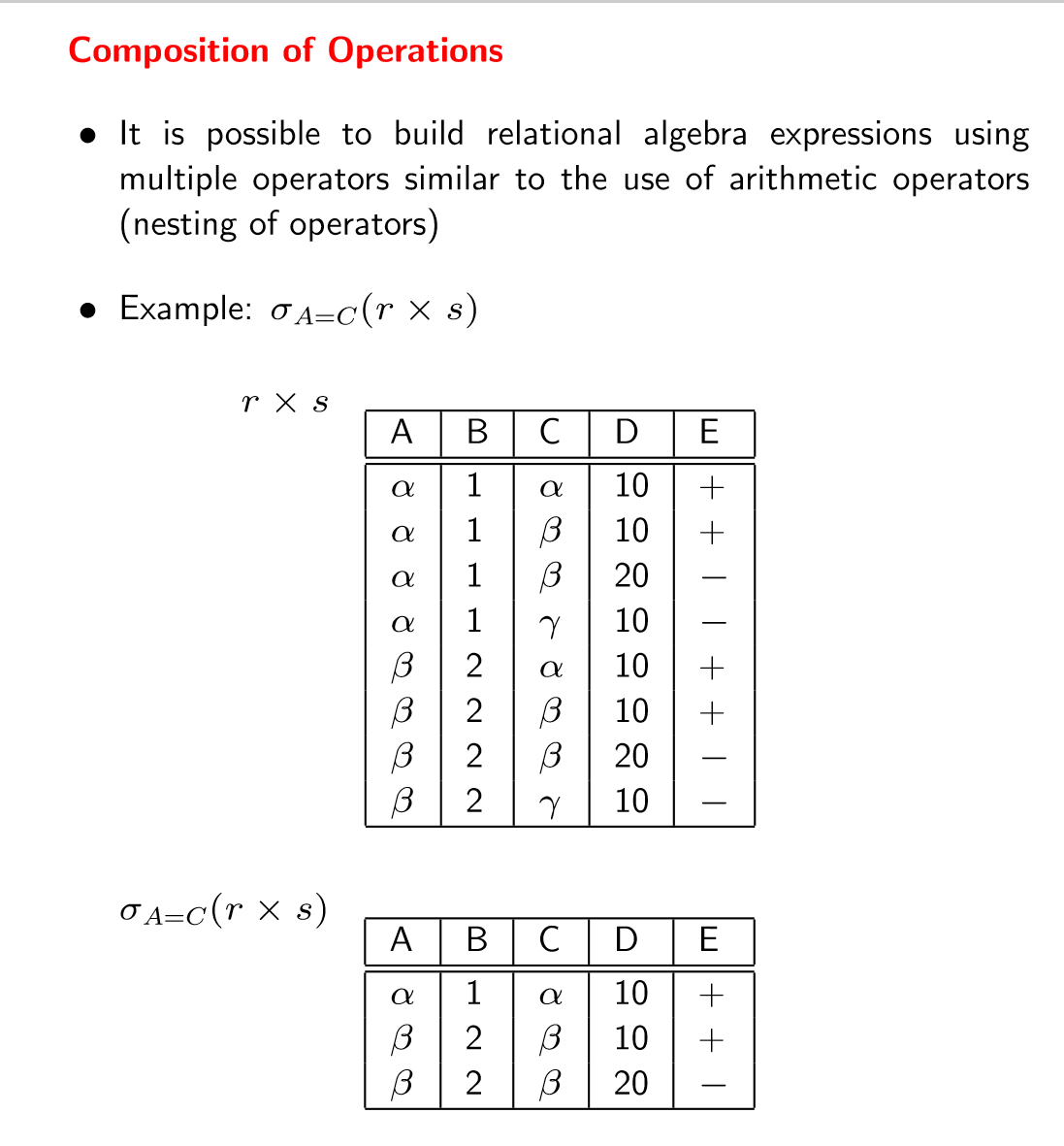
-- 3. List all students and all the possible books they can borrow

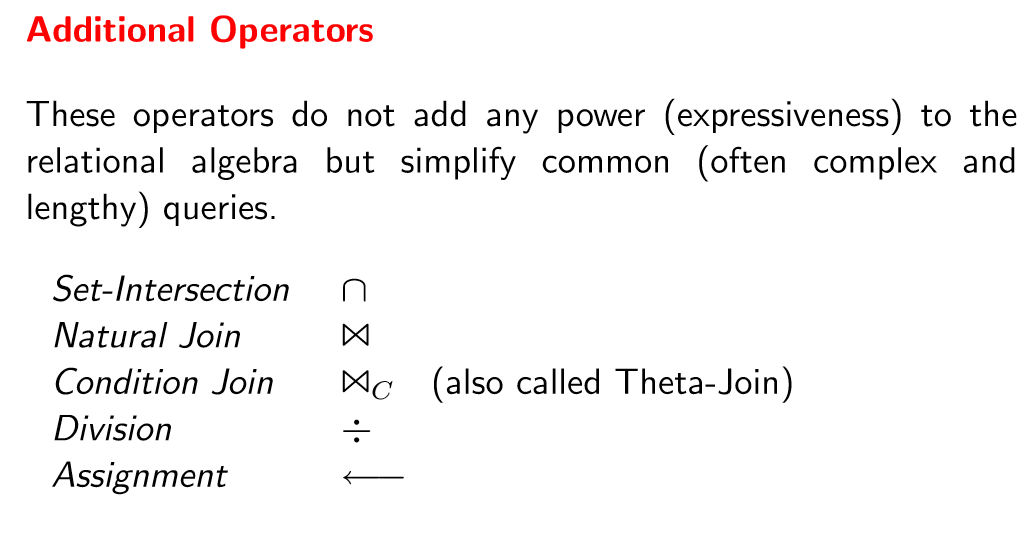
-- 4. List all books published by Pearson before 1995.

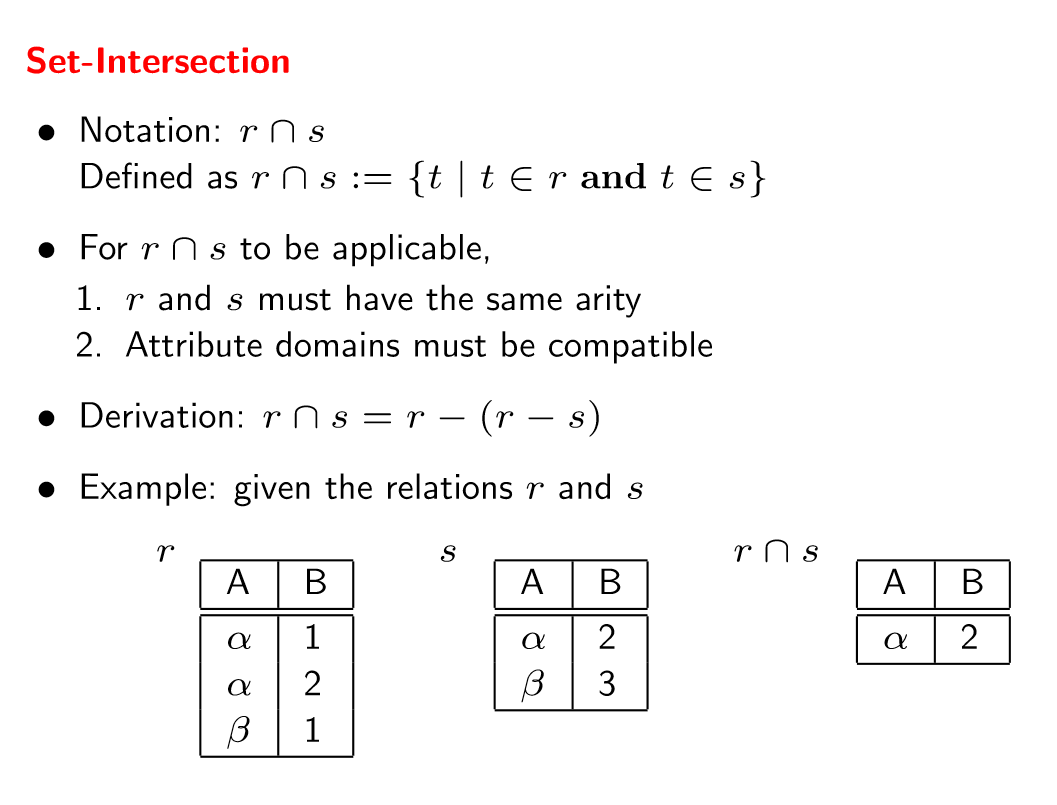
-- 5. List the name of authors who live in Sacramento.

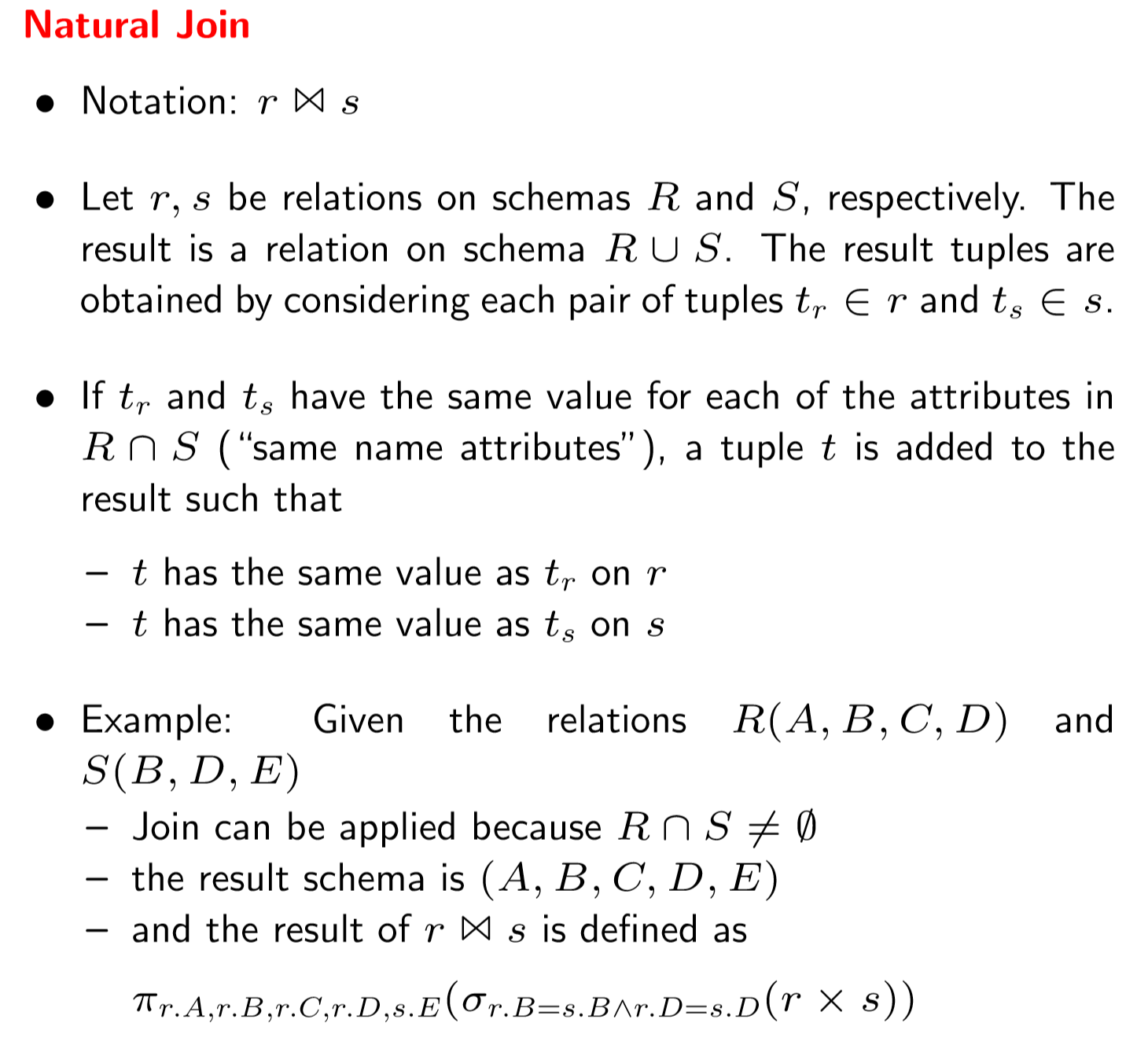
-- 6. List the name of students who are older than 25 and are not studying CS.

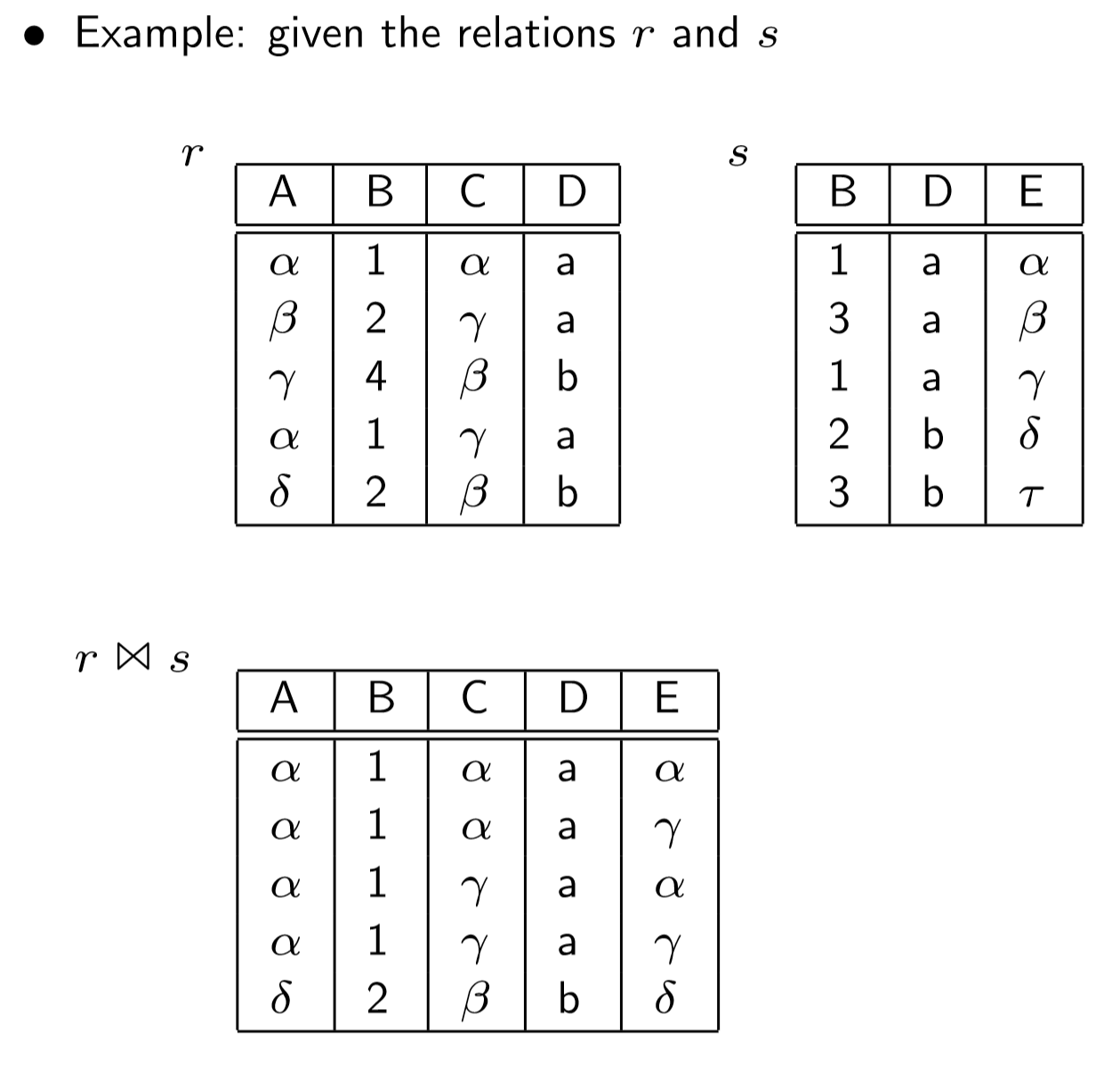
-- 7. Rename AName in the relation AUTHORS to Name.

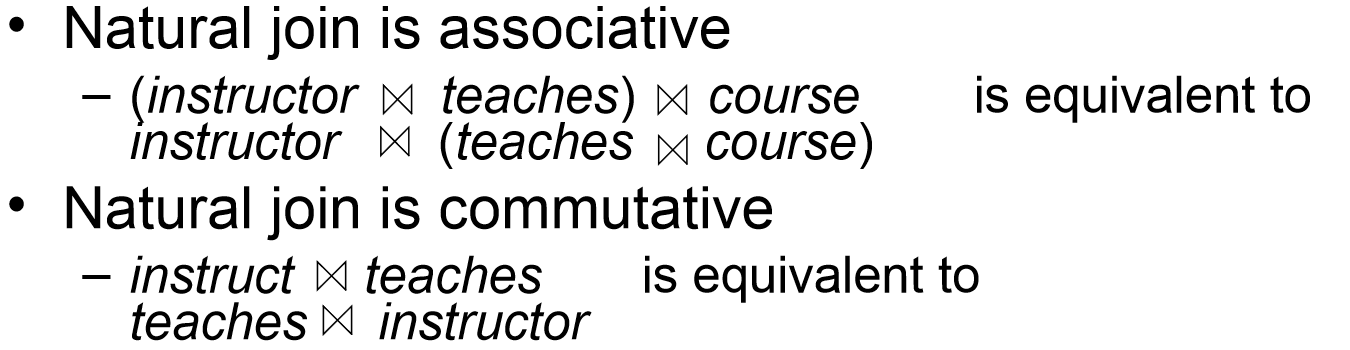


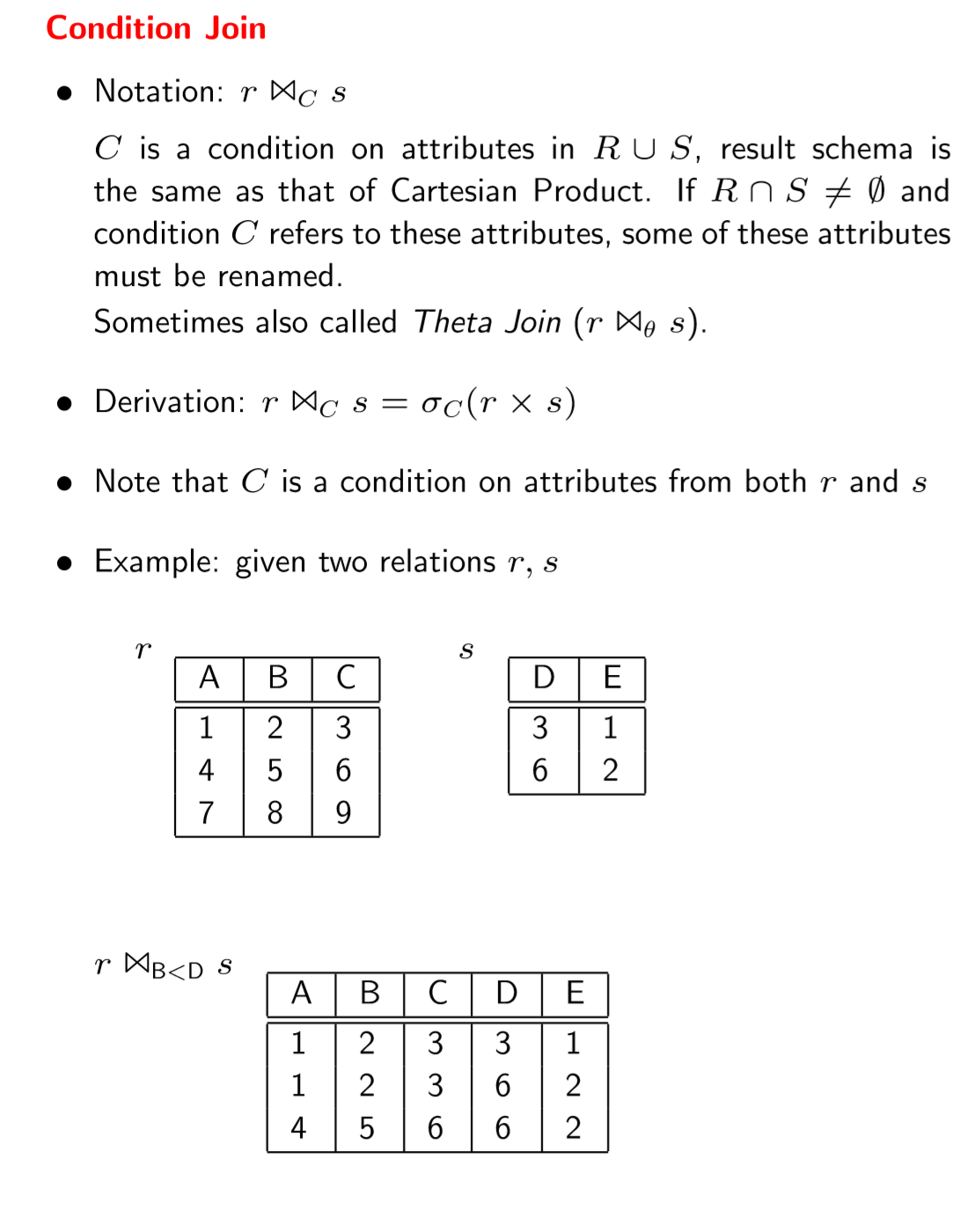


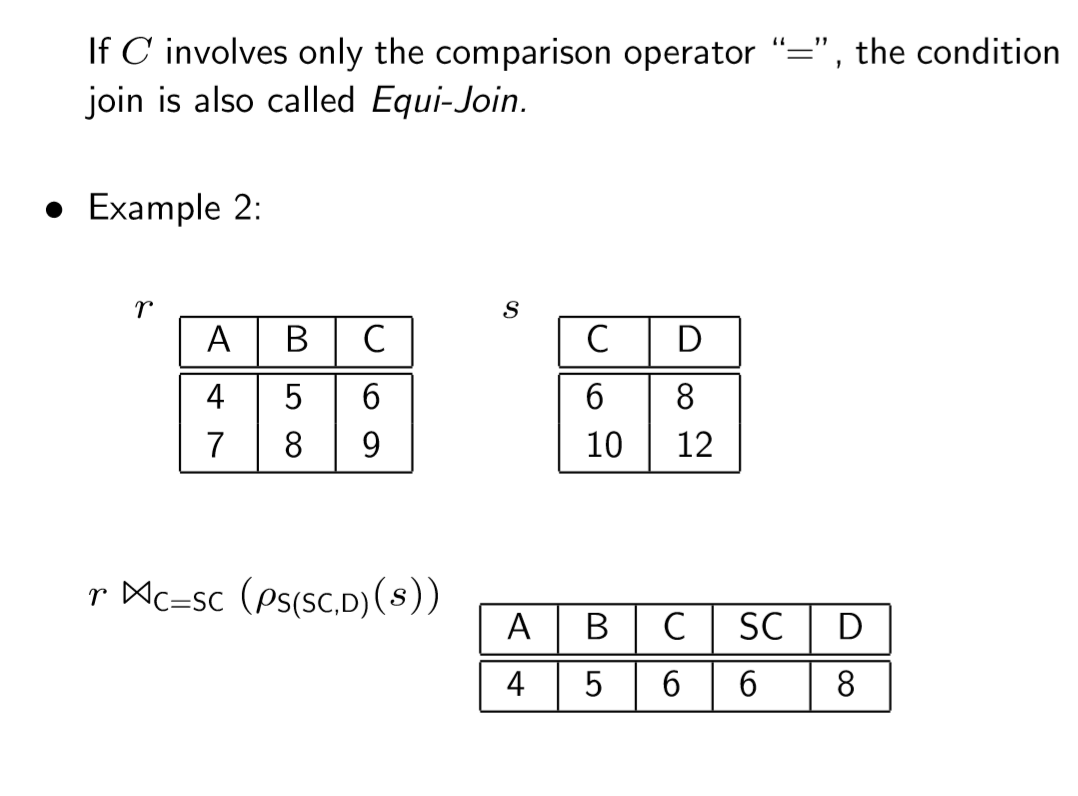


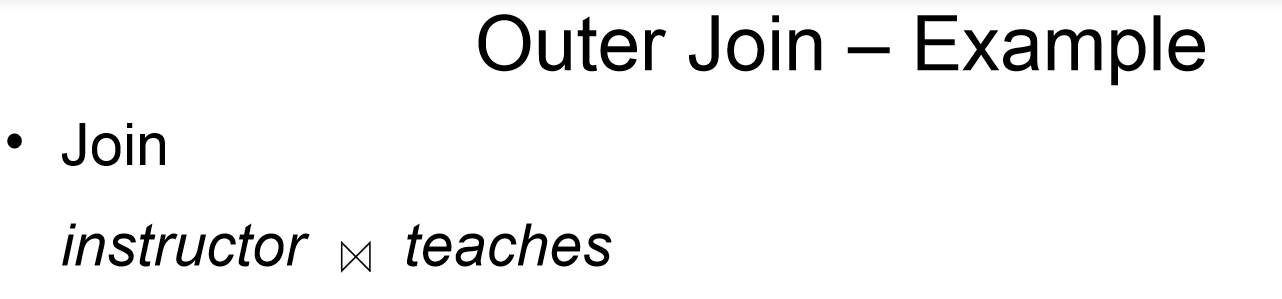




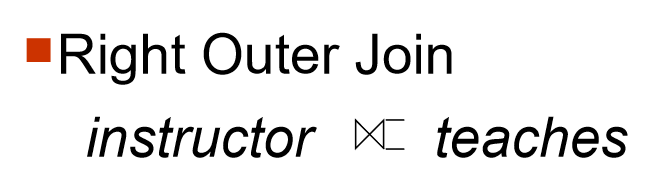




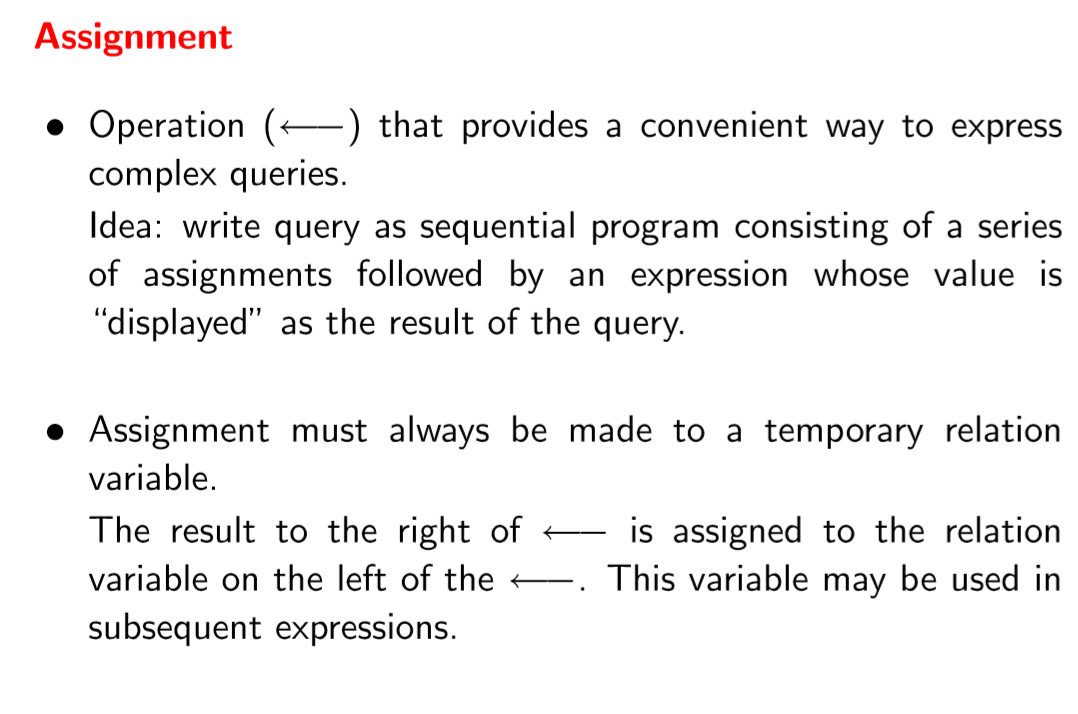












**BOOKS** (ISBN, Title, Publisher, Year)

**STUDENTS** (StId, SName, Major, Age)

**AUTHORS** (AId, AName, Address)

**BORROWS** (ISBN, StId, Date)

**HAS\_WRITTEN** (ISBN, AId)

**DESCRIBES** (ISBN, Keyword)

(1, ‘database’)

(1,’programming’)

(2,’database’)

-- 1. List each book with its keywords

-- 2. List each student with the books s/he has borrowed

-- 3. List the title of books written by the author ’Ullman’

-- 4. List the names of all students who have borrowed a book and who are CS majors

-- 5. Redo query 4 using assignments

-- 6. List the authors of the books the student ’Smith’ has borrowed

-- 7. Which books have both keywords ’database’ and ‘Programming?

-- 8. List the title of books written by the author ‘Expert’ but not books that have the

-- keyword ’database’.

**COLLEGE** = {CID:NUMBER, CNAME:STRING, STATE:STRING, ENROLLMENT:STRING}

**STUDENT** = {SID:NUMBER, SNAME:STRING, GPA:NUMBER}

**APPLY** = {SID:NUMBER, CID:NUMBER, MAJOR:STRING, DECISION:STRING}

-- 1) SID and sName of all Students with GPA > 2.7

-- 2) All students who applied to CSUS and have been accepted

-- 3) Name and GPA of all students who are computer science majors, applied to colleges in

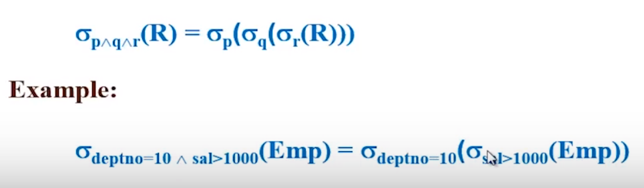
-- California and were rejected (Cross product, Natural Join and Theta join)

-- 4) ID and name of students who did not apply anywhere

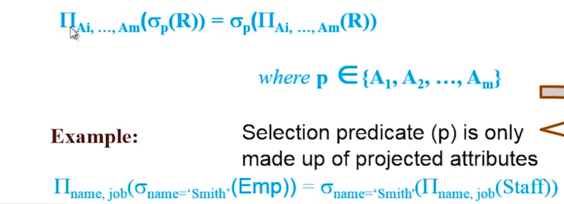
-- 5) Names that are both college names and student names

-- 6) Pairs of colleges in the same state

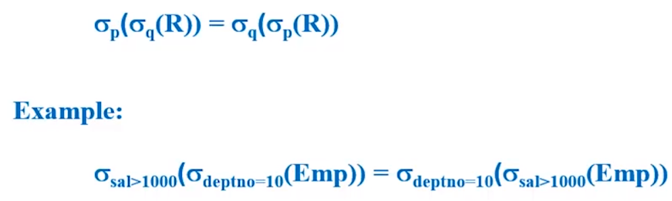
**Some Relational Algebraic Equivalence Transformation rules**

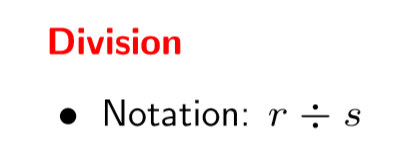


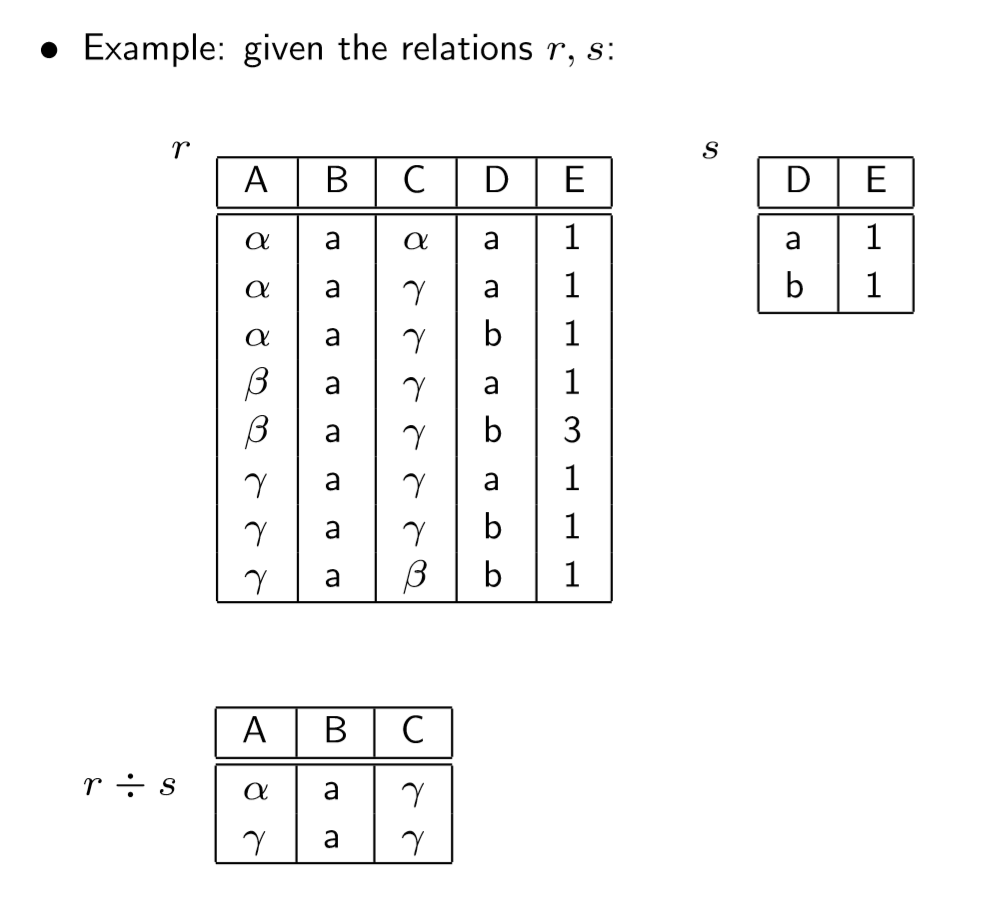
===============================================



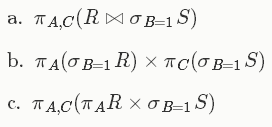
================================================









Consider a schema with two relations, *R*(A, B) and *S*(B, C), where all values are integers. Make no assumptions about keys. Consider the following three relational algebra expressions. Which two are the same?  


Consider a relation *R*(A, B) that contains r tuples, and a relation *S*(B, C) that contains s tuples; assume r > 0 and s > 0. For each of the following relational algebra expressions, state in terms of r and s the minimum and maximum number of tuples that could be in the result of the expression.  
