CIND-110

Data Organization for Data Analysts

Lab Manual Module 5

Relational Algebra

Contents

We will be using the following Dataset:

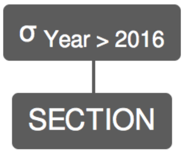
used Database

```
STUDENT = {
Name:string, Student_number:number, Class:number,
    Major:string
'Smith' , 17, 1 , 'CSS'
'Brown' , 18, 2 , 'MIE'
'Alex'  , 19, 2 , 'MIE'
}

COURSE = {
Course_name:string, Course_number:string,
   Credit_hours:number, Department:string
'Data Structures'       , 'CCPS305' , 4, 'CSS'
'Data Organization'     , 'CIND110' , 4, 'MIE'
'Data Analytics'        , 'CIND123' , 4, 'MIE'
}

SECTION = {
Section_id:string, Course_number:string, Semester
   :string, Year:number, Instructor:string
'KJ2', 'CCPS305' , 'Fall',      2016, 'Harry'
'YJ2', 'CIND110' , 'Winter',    2017, 'Larry'
'YJ3', 'CIND110' , 'Fall',      2017, 'Sally'
'KJ3', 'CIND110' , 'Winter',    2018, 'Garry'
}

Grade_REPORT = {
Student_number:number , Section_id:string, Grade:
   String
17       , 'YJ2' , 'B'
18       , 'YJ3' , 'C'
17       , 'KJ3' , 'A'
19       , 'YJ3' , 'B'
}
```

1. Unary Relational Operations: SELECT

SELECT statement

```
select distinct *
from SECTION
where Year > 2016
```

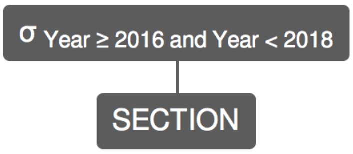


$\sigma_{\text{Year} > 2016}$ SECTION

SECTION.Section_id	SECTION.Course_number	SECTION.Semester	SECTION.Year	SECTION.Instructor
YJ2	CIND110	Winter	2017	Larry
YJ3	CIND110	Fall	2017	Sally
KJ3	CIND110	Winter	2018	Garry

SELECT statement

```
select distinct *
from SECTION
where Year >= 2016 and Year < 2018
```



$\sigma_{\text{Year} \geq 2016 \text{ and Year} < 2018}$ SECTION

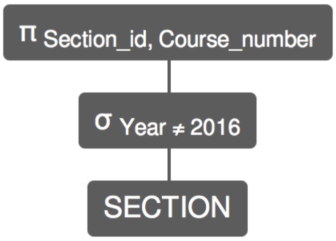
SECTION.Section_id	SECTION.Course_number	SECTION.Semester	SECTION.Year	SECTION.Instructor
KJ2	CCPS305	Fall	2016	Harry
YJ2	CIND110	Winter	2017	Larry
YJ3	CIND110	Fall	2017	Sally

Note that, if duplicates are not eliminated, the result would be a multiset or **bag** of tuples rather than a **set**. This was **NOT PERMITTED** in the formal relational model but is allowed in SQL.

2. Unary Relational Operations: PROJECT

PROJECT statement

```sql
select distinct Section_id, Course_number
from SECTION
where Year <> 2016
```

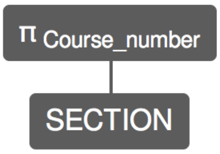


$\pi_{\text{Section\_id, Course\_number}}\ \sigma_{\text{Year} \neq 2016}$ SECTION

SECTION.Section_id	SECTION.Course_number
YJ2	CIND110
YJ3	CIND110
KJ3	CIND110

PROJECT statement

```sql
select distinct Course_number
from SECTION
```



$\pi_{\text{Course\_number}}$ SECTION

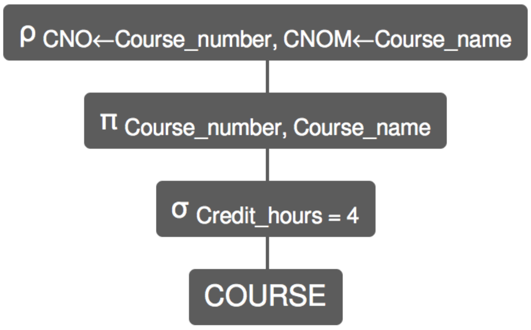
SECTION.Course_number
CCPS305
CIND110

What is the degree of the resulting relation?

3. Unary Relational Operations: RENAME

As statement

```
select distinct Course_number as CNO, Course_name
    as CNOM
from COURSE
Where Credit_hours = 4
```



$\rho_{\text{CNO}\leftarrow\text{Course\_number},\ \text{CNOM}\leftarrow\text{Course\_name}}\ \pi_{\text{Course\_number},\ \text{Course\_name}}\ \sigma_{\text{Credit\_hours} = 4}\ \text{COURSE}$

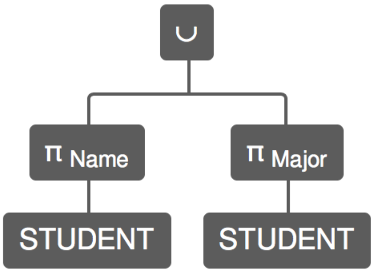
COURSE.CNO	COURSE.CNOM
CCPS305	Data Structures
CIND110	Data Organization for Data Analysts
CIND123	Data Analytics Basic Methods

In SQL, a single query typically represents a complex relational algebra expression. Renaming in SQL is accomplished by aliasing using **AS**, as in the example listed above.

4. Operations from Set Theory

Union

```
SELECT distinct Name FROM STUDENT
UNION
SELECT distinct Major FROM STUDENT
```



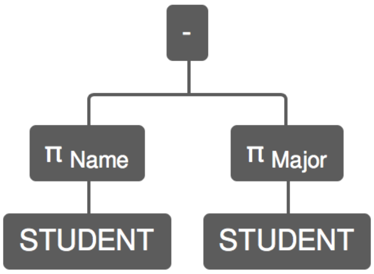
π_{Name} STUDENT $\cup$ π_{Major} STUDENT

STUDENT.Name
Smith
Brown
Alex
CSS
MIE

The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S.
What about the duplicate elimination feature?

Except

```
SELECT distinct Name FROM STUDENT
EXCEPT
SELECT distinct Major FROM STUDENT
```

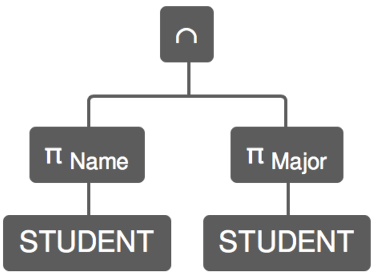


π_{Name} STUDENT - π_{Major} STUDENT

STUDENT.Name
Smith
Brown
Alex

INTERSECT

```
SELECT distinct Name FROM STUDENT
INTERSECT
SELECT distinct Major FROM STUDENT
```



π_{Name} STUDENT $\cap$ π_{Major} STUDENT

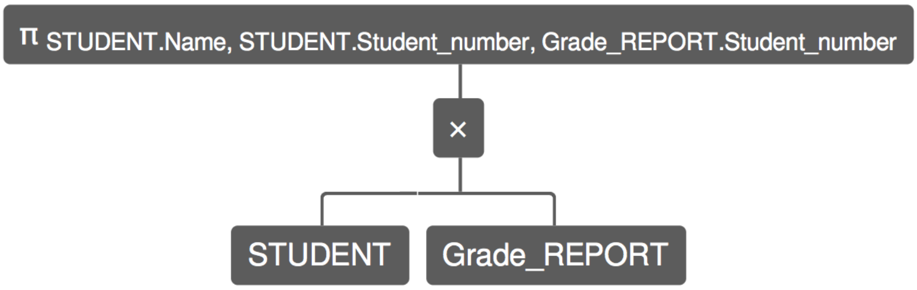
STUDENT.Name

EXCEPT returns any distinct values from the query to the left of the EXCEPT operator that are not also returned from the right query. However, **INTERSECT** returns any distinct values that are returned by both the query on the left and right sides of the INTERSECT operator.

5. The Cartesian Product (CROSS PRODUCT)

Union

```sql
SELECT distinct STUDENT.Name, STUDENT.
   Student_number, Grade_REPORT.Student_number
FROM STUDENT
CROSS JOIN
Grade_REPORT
```



π STUDENT.Name, STUDENT.Student_number, Grade_REPORT.Student_number STUDENT × Grade_REPORT

STUDENT.Name	STUDENT.Student_number	Grade_REPORT.Student_number
Smith	17	17
Smith	17	18
Smith	17	19
Brown	18	17
Brown	18	18
Brown	18	19
Alex	19	17
Alex	19	18
Alex	19	19

The CARTESIAN PRODUCT creates tuples with the combined attributes of two relations.

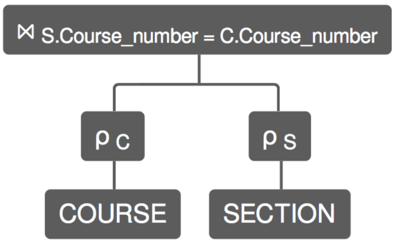
We can SELECT related tuples only from the two relations by specifying an appropriate selection condition after the Cartesian product, as we did in the preceding example.

The operation produces a new element by combining every member (tuple) from one relation with every member (tuple) from the other relation.

6. Binary Relational Operations

JOIN

```
SELECT distinct *
FROM COURSE as C
INNER JOIN SECTION as S
ON S.Course_number = C.Course_number
```

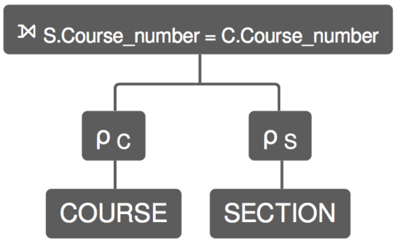


$\rho_{C}\, \text{COURSE} \bowtie_{\text{S.Course\_number = C.Course\_number}} \rho_{S}\, \text{SECTION}$

C.Course_name	C.Course_number	C.Credit_hours	C.Deprtment	S.Section_id	S.Course_number	S.Semester	S.Year	S.Instructor
Data Structures	CCPS305	4	CSS	KJ2	CCPS305	Fall	2016	Harry
Data Organization for Data Analysts	CIND110	4	MIE	YJ2	CIND110	Winter	2017	Larry
Data Organization for Data Analysts	CIND110	4	MIE	YJ3	CIND110	Fall	2017	Sally
Data Organization for Data Analysts	CIND110	4	MIE	KJ3	CIND110	Winter	2018	Garry

LEFT JOIN

```
SELECT distinct *
FROM COURSE as C
LEFT JOIN SECTION as S
ON S.Course_number = C.Course_number
```



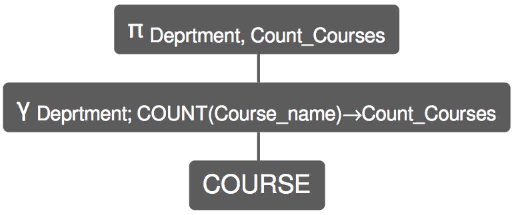
ρ_{C} COURSE $\bowtie_{S.Course\_number = C.Course\_number}$ ρ_{S} SECTION

C.Course_name	C.Course_number	C.Credit_hours	C.Deprtment	S.Section_id	S.Course_number	S.Semester	S.Year	S.Instructor
Data Structures	CCPS305	4	CSS	KJ2	CCPS305	Fall	2016	Harry
Data Organization for Data Analysts	CIND110	4	MIE	YJ2	CIND110	Winter	2017	Larry
Data Organization for Data Analysts	CIND110	4	MIE	YJ3	CIND110	Fall	2017	Sally
Data Organization for Data Analysts	CIND110	4	MIE	KJ3	CIND110	Winter	2018	Garry
Data Analytics Basic Methods	CIND123	4	MIE	*null*	*null*	*null*	*null*	*null*

7. Aggregate Functions and Grouping

COUNT

```sql
SELECT Department, COUNT(Course_name) As
   Count_Courses
FROM COURSE
Group by Department
```



$\pi_{\text{Deprtment, Count\_Courses}}\ \gamma_{\text{Deprtment; COUNT(Course\_name)} \rightarrow \text{Count\_Courses}}$ COURSE

COURSE.Deprtment	Count_Courses
CSS	1
MIE	2

Common functions applied to collections of numeric values include **SUM**, **AVERAGE**, **MAXIMUM**, and **MINIMUM**.

The **COUNT** function is used for counting tuples or values.