# Natural-Join Operation

Natural join ($\bowtie$) is a binary operation that is written as ($r \bowtie s$) where *r* and *s* are relations. The result of the natural join is the set of all combinations of tuples in r and *s* that are equal on their common attribute names.

More formally the semantics of the natural join are defined as follows:

$$r \bowtie s = \pi R \cup S\, (\sigma\, r.A1 = s.A1 \wedge r.A2 = s.A2 \wedge \ldots \wedge r.An = s.An\, (r \times s))$$

where :

*$R \cap S = \{A1, A2, \ldots, An\}$*

*R is a relation schema of r*

*S is a relation schema of s*

a relational schema is the design for the table. It includes none of the actual data, but is like a blueprint or design for the table

For an example consider the tables *Employee* and *Dept* and their natural join:

***Employee***

| Name | EmpId | DeptName |
|---|---|---|
| Harry | 3415 | Finance |
| Sally | 2241 | Sales |
| George | 3401 | Finance |
| Harriet | 2202 | Sales |

***Dept***

| DeptName | Manager |
|---|---|
| Finance | George |
| Sales | Harriet |
| Production | Charles |

***Employee ⋈ Dept***

| Name | EmpId | DeptName | Manager |
|---|---|---|---|
| Harry | 3415 | Finance | George |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | George |
| Harriet | 2202 | Sales | Harriet |

# Natural-Join Operation (cont.)

Ilustration of **Employee ⋈ Dept** operation :

$$Employee \bowtie Dept = \pi_{E \cup D}\,(\sigma_{Employee.DeptName = Dept.DeptName}\,(Employee \times Dept))$$

Where :
E ∩ D = {Deptname}, E is a relation schema of Employee, D is a relation schema of Dept

1

**Employee X Dept**

| Name | EmpId | DeptName | DeptName | Manager |
|---|---|---|---|---|
| Harry | 3415 | Finance | Finance | George |
| Harry | 3415 | Finance | Sales | Harriet |
| Harry | 3415 | Finance | Production | Charles |
| Sally | 2241 | Sales | Finance | George |
| Sally | 2241 | Sales | Sales | Harriet |
| Sally | 2241 | Sales | Production | Charles |
| George | 3401 | Finance | Finance | George |
| George | 3401 | Finance | Sales | Harriet |
| George | 3401 | Finance | Production | Charles |
| Harriet | 2202 | Sales | Finance | George |
| Harriet | 2202 | Sales | Sales | Harriet |
| Harriet | 2202 | Sales | Production | Charles |

**E ∩ D = {Deptname}**

2a

$\sigma_{Employee.DeptName = Dept.DeptName}$

2b

Steps :

1 ⇨ 2a ⇨ 2b ⇨ 3

result

3

$\pi_{E \cup D}$

| Name | EmpId | DeptName | Manager |
|---|---|---|---|
| Harry | 3415 | Finance | George |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | George |
| Harriet | 2202 | Sales | Harriet |

# Natural-Join Operation (cont.)

if we want to join three relations such as instructor, teaches, and course, then the natural join operation can be written:

$(Instructor \bowtie Teaches \bowtie \mathrm{Course})$ → the order of writing the relation will determine the order of operations on the relation

Or

$((Instructor \bowtie Teaches) \bowtie \mathrm{Course})$

Or

$(Instructor \bowtie (Teaches \bowtie \mathrm{Course}))$

**associative**

Example :

Teaches

| ID | course_id | sec_id | semester | year |
|---|---|---|---|---|
| 10101 | CS-101 | 1 | Fall | 2009 |
| 10101 | CS-315 | 1 | Spring | 2010 |
| 10101 | CS-347 | 1 | Fall | 2009 |
| 12121 | FIN-201 | 1 | Spring | 2010 |
| 15151 | MU-199 | 1 | Spring | 2010 |
| 22222 | PHY-101 | 1 | Fall | 2009 |
| 32343 | HIS-351 | 1 | Spring | 2010 |
| 45565 | CS-101 | 1 | Spring | 2010 |
| 45565 | CS-319 | 1 | Spring | 2010 |
| 76766 | BIO-101 | 1 | Summer | 2009 |
| 76766 | BIO-301 | 1 | Summer | 2010 |
| 83821 | CS-190 | 1 | Spring | 2009 |
| 83821 | CS-190 | 2 | Spring | 2009 |
| 83821 | CS-319 | 2 | Spring | 2010 |
| 98345 | EE-181 | 1 | Spring | 2009 |

Instructor

| ID | name | dept_name | salary |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 58583 | Califieri | History | 62000 |
| 76543 | Singh | Finance | 80000 |
| 76766 | Crick | Biology | 72000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |

Course

| course_id | title | dept_name | credits |
|---|---|---|---|
| BIO-101 | Intro. to Biology | Biology | 4 |
| BIO-301 | Genetics | Biology | 4 |
| BIO-399 | Computational Biology | Biology | 3 |
| CS-101 | Intro. to Computer Science | Comp. Sci. | 4 |
| CS-190 | Game Design | Comp. Sci. | 4 |
| CS-315 | Robotics | Comp. Sci. | 3 |
| CS-319 | Image Processing | Comp. Sci. | 3 |
| CS-347 | Database System Concepts | Comp. Sci. | 3 |
| EE-181 | Intro. to Digital Systems | Elec. Eng. | 3 |
| FIN-201 | Investment Banking | Finance | 3 |
| HIS-351 | World History | History | 3 |
| MU-199 | Music Video Production | Music | 3 |
| PHY-101 | Physical Principles | Physics | 4 |

Query :

$\pi$name,title,semester ($\sigma$Instruktor.dept_name ="Comp. Sci." (instructor $\bowtie$ teaches $\bowtie$ course))

Output ??

# Theta-Join Operation

The theta join operation is a variant of the natural-join operation.

Consider relations *r (R) and s(S), and let* $\theta$ *be a predicate on attributes in the schema R ∪ S.* The theta join operation *r  s is defined as follows:*

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

Theta join produces a relation containing tuples which criteria satisfy $\theta$ of Cartesian product of R and S relation schema. Criteria can use relational operators (≤, <, =,>, ≥). Theta join operation is an extension of natural join.

Example 1 :

**Airport**

| airportId | name | city |
|---|---|---|
| LHR | Heathrow | London |
| LGW | Gatwick | London |
| CDG | Charles de Gaulle | Paris |
| ORY | Orly | Paris |

$$Airport \bowtie_{\theta\, Airport.airportId=Flight.depAirport} Flight$$

**Flight**

| flightNo | flightCompany | depAirport | arrAirport |
|---|---|---|---|
| AF1231 | Air France | LHR | CDG |
| AF1232 | Air France | CDG | LHR |
| AF1234 | Air France | LGW | CDG |
| AF1235 | Air France | CDG | LGW |
| BA2943 | British Airways | LGW | ORY |
| BA2944 | British Airways | ORY | LGW |
| BA4059 | British Airways | LHR | CDG |
| BA4060 | British Airways | CDG | LHR |

Output ??

# Theta-Join Operation (Cont.)

Example 2:

Employee

| LAST_NAME | SALARY |
|---|---|
| King | 24000 |
| Kochhar | 17000 |
| De Haan | 17000 |
| Hunold | 9000 |
| Ernst | 6000 |
| Lorentz | 4200 |
| Mourgos | 5800 |
| Rajs | 3500 |
| Davies | 3100 |
| Matos | 2600 |
| Vargas | 2500 |
| Zlotkey | 10500 |
| Abel | 11000 |
| Taylor | 8600 |

20 rows selected.

Job_grade

| GRA | LOWEST_SAL | HIGHEST_SAL |
|---|---|---|
| A | 1000 | 2999 |
| B | 3000 | 5999 |
| C | 6000 | 9999 |
| D | 10000 | 14999 |
| E | 15000 | 24999 |
| F | 25000 | 40000 |

Relational Algebra :

$\pi last\_name, salary, gra$ $(Employee \bowtie_{salary>=lowest\_sal \wedge salary<=highest\_sal} Dept)$

| LAST_NAME | SALARY | GRA |
|---|---|---|
| Matos | 2600 | A |
| Vargas | 2500 | A |
| Lorentz | 4200 | B |
| Mourgos | 5800 | B |
| Rajs | 3500 | B |
| Davies | 3100 | B |
| Whalen | 4400 | B |
| Hunold | 9000 | C |
| Ernst | 6000 | C |

20 rows selected.

# Outer Join Operation

The outer-join operation is an extension of the join operation to deal with missing information. The outer join operation works in a manner similar to the natural join operation, but preserves those tuples that have no relationship with other relation (null values).

Example :

The result of operation outer of Employee and Dept relation is :

Employee

| EmpId | Name | DeptId |
|---|---|---|
| 3415 | Harry | D01 |
| 2241 | Sally | D02 |
| 3401 | Goerge | D03 |
| 2202 | Harriet | D02 |

Dept

| DeptId | DeptName |
|---|---|
| D01 | Finance |
| D02 | Sales |

Employee ⟕ Dept

| EmpId | Name | Employee.DeptId | Dept.DeptId | DeptName |
|---|---|---|---|---|
| 3415 | Harry | D01 | D01 | Finance |
| 2241 | Sally | D02 | D02 | Sales |
| 3401 | Goerge | D03 | Null | Null |
| 2202 | Harriet | D02 | D02 | Sales |

All tuples in the left relation that did not match with any tuple in the right relation

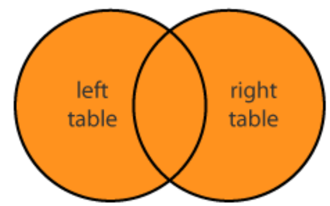
# Outer Join Operation

There are actually three forms of the outer operation namely left outer join, denoted ⟕ ; right outer join, denoted ⟖; and full outer join, denoted ⟗

Left Outer Join takes all tuples in the left relation that did not match with any tuple in the right relation

Right Outer Join takes all tuples in the right relation that did not match with any tuple in the left relation

FULL JOIN




Example of the Right Outer Join :

Employee

| EmpId | Name | DeptId |
|---|---|---|
| 3415 | Harry | D01 |
| 2241 | Sally | D02 |
| 3401 | Goerge | D03 |
| 2202 | Harriet | D02 |

Dept

| DeptId | DeptName |
|---|---|
| D01 | Finance |
| D02 | Sales |
| D03 | Production |
| D04 | Accounting |

Employee ⟖ Dept

| EmpId | Name | Employee.DeptId | Dept.DeptId | DeptName |
|---|---|---|---|---|
| 3415 | Harry | D01 | D01 | Finance |
| 2241 | Sally | D02 | D02 | Sales |
| 3401 | Goerge | D03 | D03 | Production |
| 2202 | Harriet | D02 | D02 | Sales |
| Null | Null | Null | D04 | Accounting |

All tuples in the right relation that did not match with any tuple in the left relation

# Outer Join Operation

Example of the Full Outer Join :

Customer

| **Id_cus** |
|---|
| Name |
| City_cus |
| Country |
| Phone_cus |

Supplier

| **Id_sup** |
|---|
| CompanyName |
| ContactName |
| City_sup |
| Country |
| Phone_sup |

Query :

Match all customers and suppliers by country

Relational Algebra :

$\pi$ *Name, Customer.Country, Supplier.Country, CompanyName (Customer* ⟗ *Supplier)*

| | Name | CustomerCountry | SupplierCountry | CompanyName |
|---|---|---|---|---|
| Right outer Join | NULL | NULL | Australia | Pavlova, Ltd. |
| | NULL | NULL | Australia | G'day, Mate |
| | ..... | ..... | ...... | ...... |
| Left outer Join | Simpson | Argentina | NULL | NULL |
| | Moncada | Argentina | NULL | NULL |
| | ..... | ..... | ...... | ...... |
| Join | Batista | Brazil | Brazil | Refrescos Americanas LTDA |
| | Carvalho | Brazil | Brazil | Refrescos Americanas LTDA |
| | Limeira | Brazil | Brazil | Refrescos Americanas LTDA |
| | Lincoln | Canada | Canada | Ma Maison |
| | Lincoln | Canada | Canada | Forêts d'érables |

# Assignment Operation

It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables. The assignment operation, denoted by ←,works like assignment in a programming language.

the illustrate of this operation is used to define the natural-join operation :

$$
\begin{aligned}
temp1 &\leftarrow r \times s \\
temp2 &\leftarrow \sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n} (temp1) \\
result &= \Pi_{R \cup S} (temp2)
\end{aligned}
$$

With the assignment operation, a query can bewritten as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.

# Aggregation Operation

Aggregate functions take a collection of values and return a single value as a result. Aggregate functions denoted by $\mathcal{G}$ (calligraphic G).

Type of the aggregate function : **sum( ), avg( ),min( ), max( ), count( )** .

Instructor

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 58583 | Califieri | History | 62000 |
| 76543 | Singh | Finance | 80000 |
| 76766 | Crick | Biology | 72000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |

Example :

$\mathcal{G}_{\text{sum(salary)}}$(Instructor) =?

$\mathcal{G}_{\text{avg(salary)}}$(Instructor) =?

$\mathcal{G}_{\text{min(salary)}}$(Instructor) =?

$\mathcal{G}_{\text{max(salary)}}$(Instructor) =?

$\mathcal{G}_{\text{count(Name)}}$(Instructor) =?

| *dept_name* | *salary* |
|---|---|
| Biology | 72000 |
| Comp. Sci. | 77333 |
| Elec. Eng. | 80000 |
| Finance | 85000 |
| History | 61000 |
| Music | 40000 |
| Physics | 91000 |

dept_name $\mathcal{G}_{\text{avg(salary}}(\pi\ \textit{dept\_name}$ (instructor))

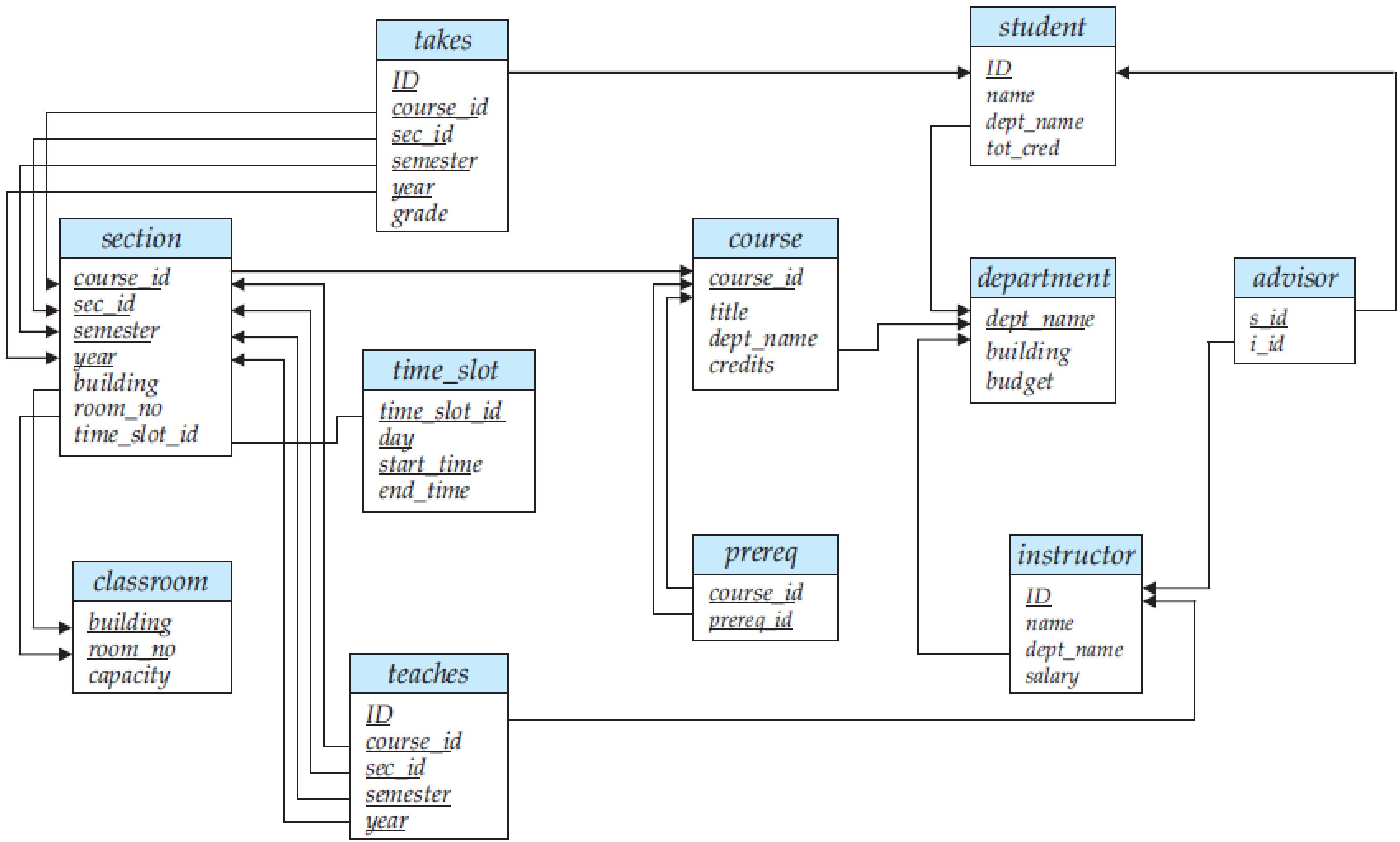
The general form of the aggregation operation G is as follows :

$$_{G_1, G_2, \ldots, G_n}\mathcal{G}_{F_1(A_1),\ F_2(A_2), \ldots,\ F_m(A_m)}(E)$$

where *E* is any relational-algebra expression; $G_1$, $G_2$, . . . , $G_n$ constitute a list of attributes on which to group; each $F_i$ is an aggregate function; and each $A_i$ is an attribute name.

# Exercise

Schema diagram for the university database



SQL data definition for part of the university database :

**create table *department*** (*dept_name* ***varchar (20),*** *building* ***varchar (15),*** *budget* ***numeric (12,2),*** **primary key (*dept_name));***

**create table *course*** (*course_id* ***varchar (7),*** *title* ***varchar (50),*** *dept_name* ***varchar (20),*** *credits* ***numeric (2,0),*** **primary key (*course id),*** **foreign key (*dept_name) references department);***

**create table *instructor*** (*ID* ***varchar (5),*** *name* ***varchar (20) not null,*** *dept_name* ***varchar (20),*** *salary* ***numeric (8,2),*** **primary key (*ID),*** **foreign key (*dept_name) references department(dept_name));***

**create table *section*** (*course_id* ***varchar (8),*** *sec_id* ***varchar (8),*** *semester* ***varchar (6),*** *year* ***numeric (4,0),*** *building* ***varchar (15),*** *room_number* ***varchar (7),*** *time_slot_id* ***varchar (4),*** **primary key (*course_id, sec_id, semester, year),*** **foreign key (*course_id) references course*** **(*course_id));***

**create table *teaches*** (*ID* ***varchar (5),*** *course_id* ***varchar (8),*** *sec_id* ***varchar (8),*** *semester* ***varchar (6),*** *year* ***numeric (4,0),*** **primary key (*ID, course_id, sec_id, semester, year),*** **foreign key (*course_id, sec_id, semester, year) references section*** **(*course_id, sec_id, semester, year),*** **foreign key (*ID) references instructor*** **(*ID));***

***etc....***

# The Tuples of some tables of the university database :

Department

| *dept_name* | *building* | *budget* |
|---|---|---|
| Comp. Sci. | Taylor | 100000 |
| Biology | Watson | 90000 |
| Elec. Eng. | Taylor | 85000 |
| Music | Packard | 80000 |
| Finance | Painter | 120000 |
| History | Painter | 50000 |
| Physics | Watson | 70000 |

Instructor

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 58583 | Califieri | History | 62000 |
| 76543 | Singh | Finance | 80000 |
| 76766 | Crick | Biology | 72000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |

Teaches

| *ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|
| 10101 | CS-101 | 1 | Fall | 2009 |
| 10101 | CS-315 | 1 | Spring | 2010 |
| 10101 | CS-347 | 1 | Fall | 2009 |
| 12121 | FIN-201 | 1 | Spring | 2010 |
| 15151 | MU-199 | 1 | Spring | 2010 |
| 22222 | PHY-101 | 1 | Fall | 2009 |
| 32343 | HIS-351 | 1 | Spring | 2010 |
| 45565 | CS-101 | 1 | Spring | 2010 |
| 45565 | CS-319 | 1 | Spring | 2010 |
| 76766 | BIO-101 | 1 | Summer | 2009 |
| 76766 | BIO-301 | 1 | Summer | 2010 |
| 83821 | CS-190 | 1 | Spring | 2009 |
| 83821 | CS-190 | 2 | Spring | 2009 |
| 83821 | CS-319 | 2 | Spring | 2010 |
| 98345 | EE-181 | 1 | Spring | 2009 |

Course

| *course_id* | *title* | *dept_name* | *credits* |
|---|---|---|---|
| BIO-101 | Intro. to Biology | Biology | 4 |
| BIO-301 | Genetics | Biology | 4 |
| BIO-399 | Computational Biology | Biology | 3 |
| CS-101 | Intro. to Computer Science | Comp. Sci. | 4 |
| CS-190 | Game Design | Comp. Sci. | 4 |
| CS-315 | Robotics | Comp. Sci. | 3 |
| CS-319 | Image Processing | Comp. Sci. | 3 |
| CS-347 | Database System Concepts | Comp. Sci. | 3 |
| EE-181 | Intro. to Digital Systems | Elec. Eng. | 3 |
| FIN-201 | Investment Banking | Finance | 3 |
| HIS-351 | World History | History | 3 |
| MU-199 | Music Video Production | Music | 3 |
| PHY-101 | Physical Principles | Physics | 4 |

Section

| *course_id* | *sec_id* | *semester* | *year* | *building* | *room_number* | *time_slot_id* |
|---|---|---|---|---|---|---|
| BIO-101 | 1 | Summer | 2009 | Painter | 514 | B |
| BIO-301 | 1 | Summer | 2010 | Painter | 514 | A |
| CS-101 | 1 | Fall | 2009 | Packard | 101 | H |
| CS-101 | 1 | Spring | 2010 | Packard | 101 | F |
| CS-190 | 1 | Spring | 2009 | Taylor | 3128 | E |
| CS-190 | 2 | Spring | 2009 | Taylor | 3128 | A |
| CS-315 | 1 | Spring | 2010 | Watson | 120 | D |
| CS-319 | 1 | Spring | 2010 | Watson | 100 | B |
| CS-319 | 2 | Spring | 2010 | Taylor | 3128 | C |
| CS-347 | 1 | Fall | 2009 | Taylor | 3128 | A |
| EE-181 | 1 | Spring | 2009 | Taylor | 3128 | C |
| FIN-201 | 1 | Spring | 2010 | Packard | 101 | B |
| HIS-351 | 1 | Spring | 2010 | Painter | 514 | C |
| MU-199 | 1 | Spring | 2010 | Packard | 101 | D |
| PHY-101 | 1 | Fall | 2009 | Watson | 100 | A |

Student

| *ID* | *name* | *dept_name* | *tot_cred* |
|---|---|---|---|
| 00128 | Zhang | Comp. Sci. | 102 |
| 12345 | Shankar | Comp. Sci. | 32 |
| 19991 | Brandt | History | 80 |
| 23121 | Chavez | Finance | 110 |
| 44553 | Peltier | Physics | 56 |
| 45678 | Levy | Physics | 46 |
| 54321 | Williams | Comp. Sci. | 54 |
| 55739 | Sanchez | Music | 38 |
| 70557 | Snow | Physics | 0 |
| 76543 | Brown | Comp. Sci. | 58 |
| 76653 | Aoi | Elec. Eng. | 60 |
| 98765 | Bourikas | Elec. Eng. | 98 |
| 98988 | Tanaka | Biology | 120 |

**Practical Task :**

1. Create a university database that consists of tables such as the schema diagram above (SQL data definition and tuples of some tables as shown above)
2. Please complete SQL data definition and tuples of some tables others
3. Fill the tuple of each table at least 10 tuples
4. Write the following queries in Relational Algebra and SQL :
    1. Finds the names of all instructors in the History department
    2. Finds the instructor ID and department name of all instructors associated with a department with budget of greater than $95,000
    3. Finds the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach
    4. Find the names of all students who have taken the course title of "Game Design".
    5. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.
    6. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.
    7. Find the ID and names of all students who do not have an advisor.

**the practical task will evaluate at next week, Thursday, 09.30-12.00 WIB, H.6.12**