This Week

Intro to SQL and MySQL

Mapping Relational Algebra to SQL queries

We will focus on queries to start – assume tables and database exist.

Time permitting:

creating tables, more involved queries…

Projection

Symbol is Π

Selection of attributes.

$\Pi_{\text{ID, salary}}(\textit{instructor})$

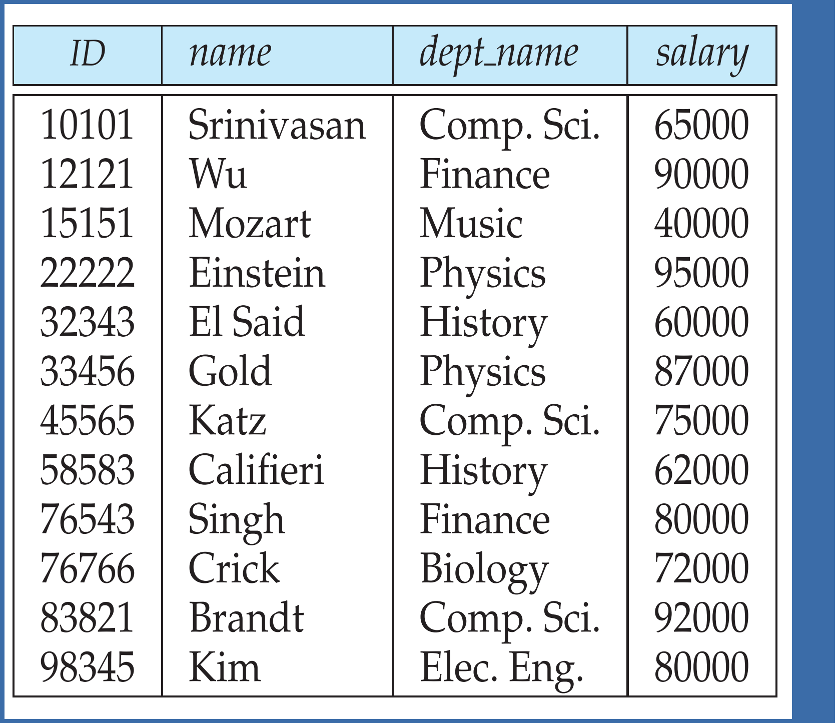
SQL Notation:

SELECT col_1,…, col_N FROM instructor

Or

SELECT * FROM instructor (means select all columns)

SELECT ID, salary FROM instructor

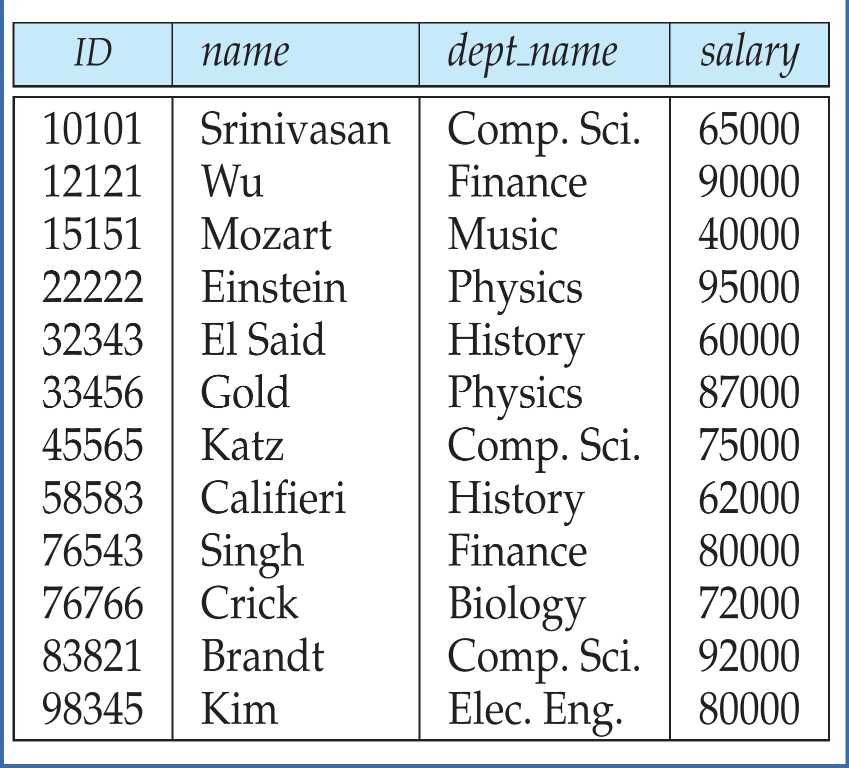


ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000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

Selection

Notation is $\sigma_p(x)$.

$\sigma_{salary >= 85000}(\mathit{instructor})$



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

SQL Notation:

```
SELECT * FROM instructor WHERE salary >= 85000
```

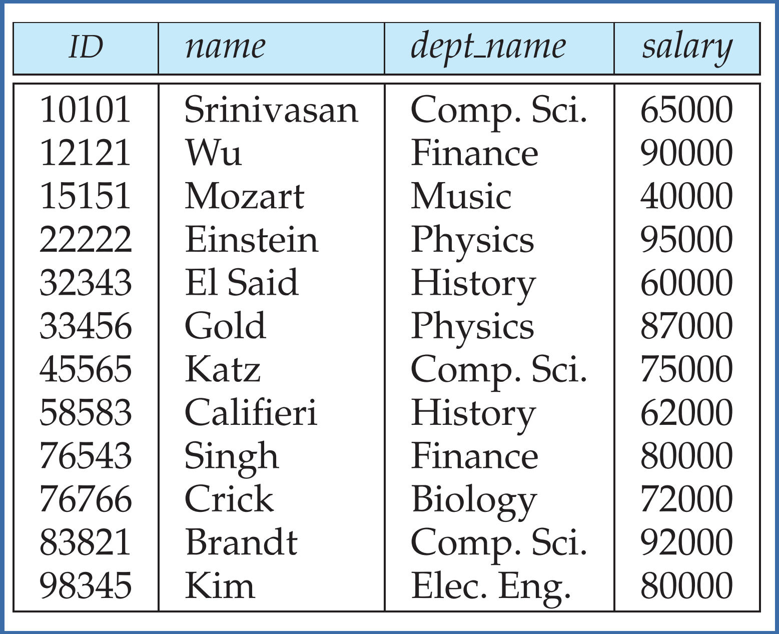
```
SELECT col_1,…, col_N FROM instructor WHERE salary >= 85000
```

Natural Join

Recall we combine two relations into a single relation.

The tuples are joined if the attributes common to both relations are equal.

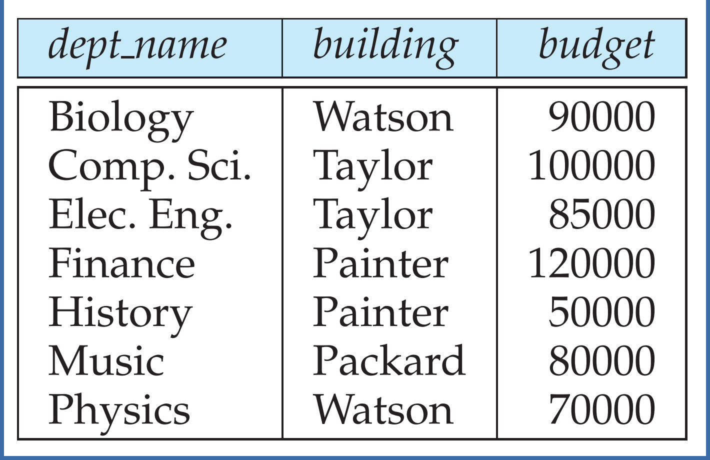
instructor



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

$\bowtie$

department

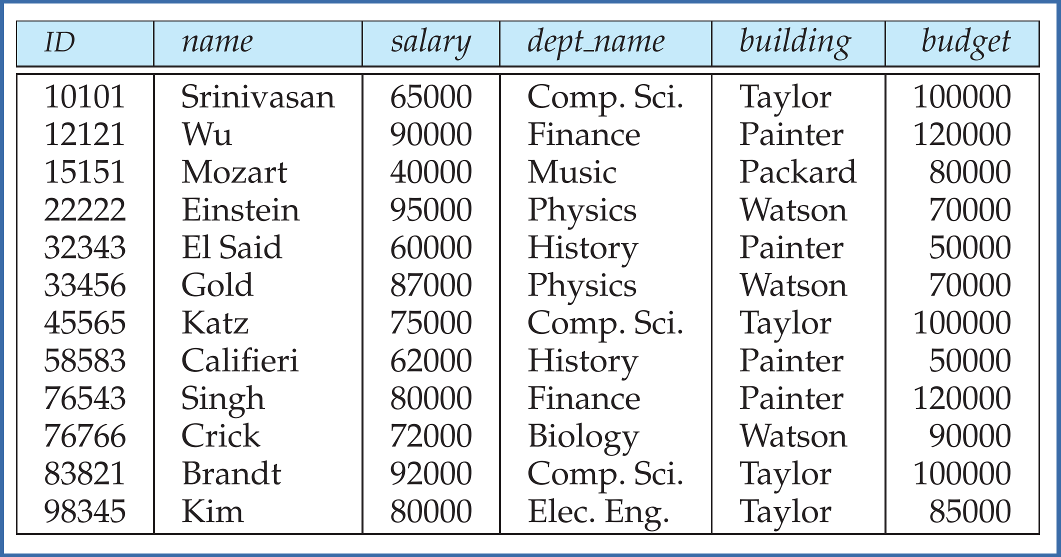


dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Natural Join

instructor ⋈ department

The tuples are joined if the attributes common to both relations are equal.



ID	*name*	*salary*	*dept_name*	*building*	*budget*
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

SQL Notation:

SELECT * FROM instructor NATURAL JOIN department

Cartesian Product Example

Relations *r, s*:

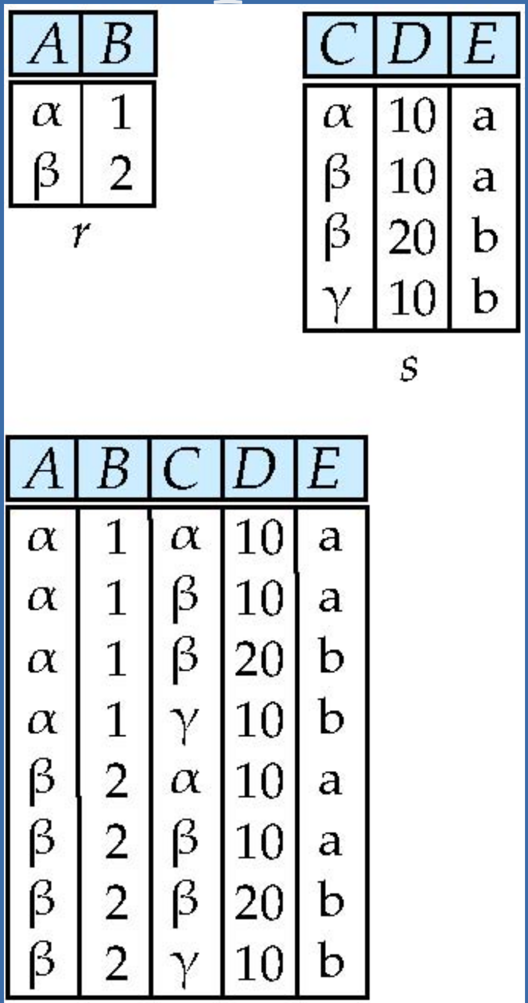
r x *s*:

SQL Notation:

SELECT * FROM r INNER JOIN s

or

SELECT * FROM r, s



A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

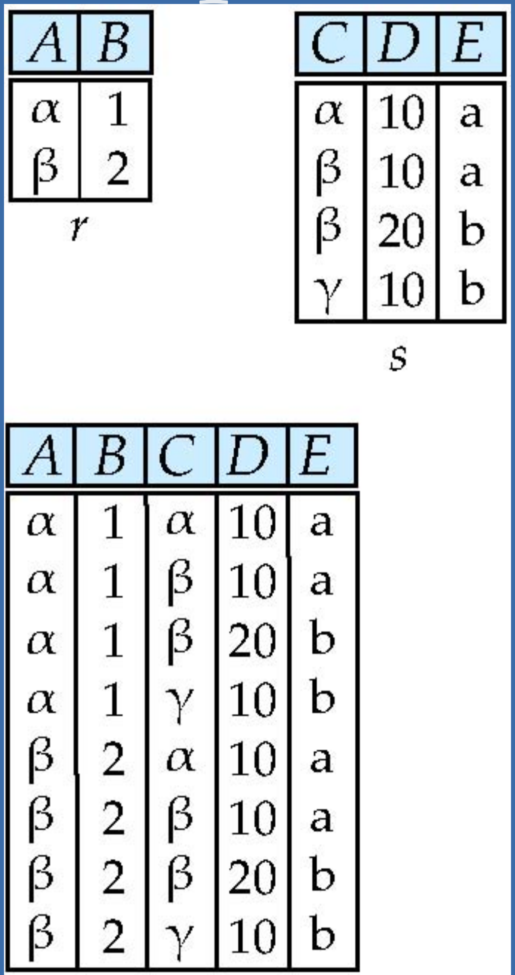
A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Note: can have as many relations as needed…but what may be a concern?

Cartesian Product Example

Relations *r, s*:

r x *s*:



A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

SQL Notation:

SELECT * FROM r INNER JOIN s

What if we don't want ALL rows?

For example, we want rows where A's value and C's value are equal?

SELECT * FROM r INNER JOIN s ON A = C

Inner Join

SQL Notation:

```
SELECT Column1, Column2, …, ColumnK FROM
            TableA INNER JOIN TableB
            ON join_constraints
            WHERE contraints
            ORDER BY  ColumnX
```

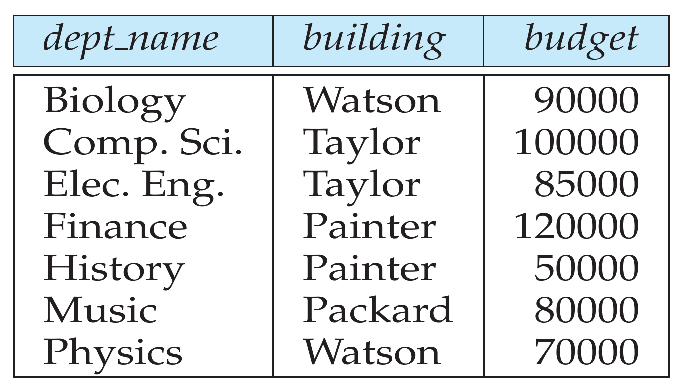
There are many other options, we will see these later…

Self Join

Suppose we want to join a table to itself.

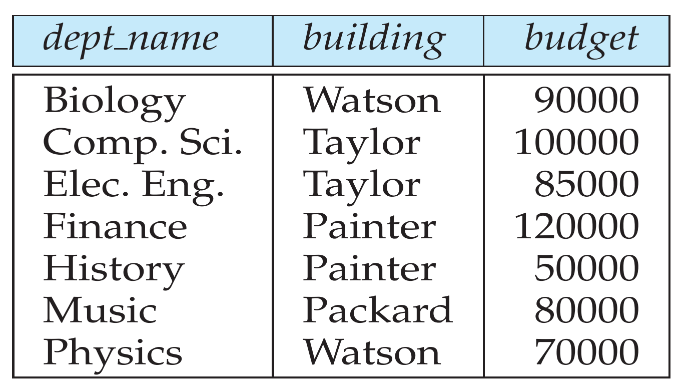
We want to find those departments that are in the same building.

department A



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

department B



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

SQL Notation:

```
SELECT A.dept_name, B.dept_name FROM
       department A INNER JOIN department B
       ON A.building = B.building
```

Union

Relations *r, s:*

For *r* ∪ *s* to be valid.

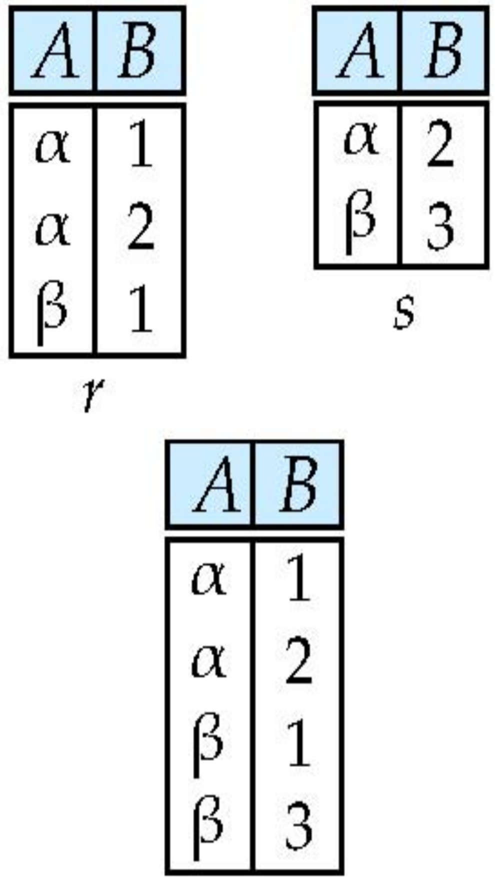
1. *r, s* must have the *same arity* (same number of attributes)

2. The attribute domains must be *compatible*

MySQL Notation:

```
(SELECT * FROM r)
UNION
(SELECT * FROM s)
```

Use UNION ALL to keep duplicates.



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

r ∪ s:

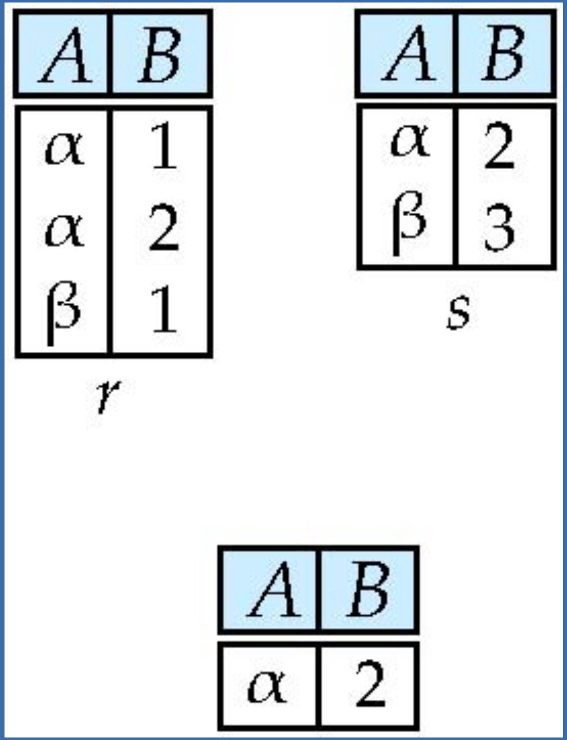
Intersection

Relation *r, s*:

$r \cap s = r - (r - s)$

SQL Notation:

(SELECT * FROM r)
INTERSECT
(SELECT * FROM s)



A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

A	*B*
α	2

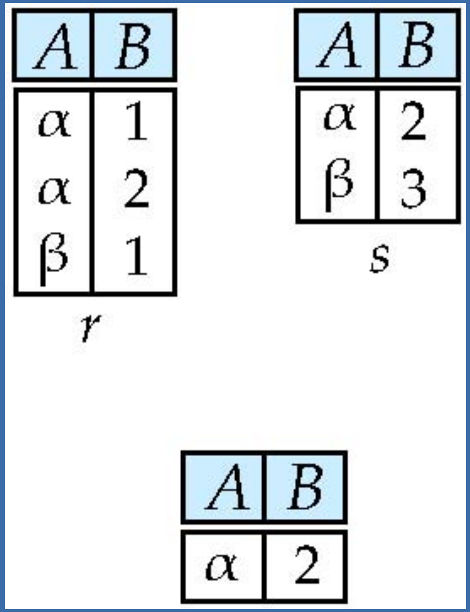
Intersection

SQL Notation:

```
(SELECT * FROM r)
INTERSECT   Does NOT Work in MySQL
(SELECT * FROM s)
```

MySQL Options:

LEFT AS EXERCISE



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

Difference

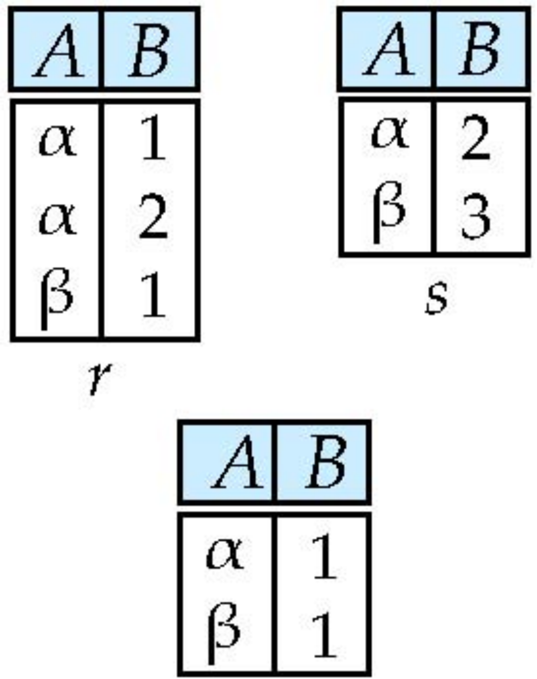
What would you expect them to be?

Relations *r*, *s*:

r – *s*

SQL Notation:

```
(SELECT * FROM r)
EXCEPT
(SELECT * FROM s)
```



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

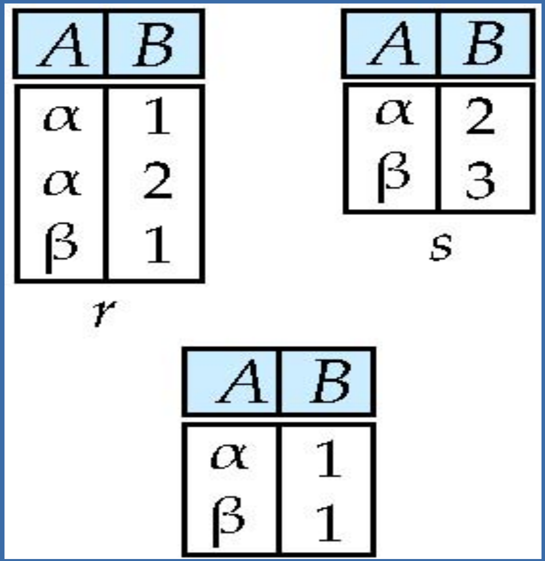
Difference

SQL Notation:

```
(SELECT * FROM r)
EXCEPT          Does NOT Work in MySQL
(SELECT * FROM s)
```

MySQL Options:

LEFT AS EXERCISE



A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

SQL Types

char(*n*): A fixed-length character string.

varchar(*n*): A variable-length character string with max length *n*.

int: An integer.

numeric(*p, d*): A fixed-point number with *p* digits of which *d* of the digits are to the right of the decimal point.

real, double precision: Floating point and double precision floating point.

float(*n*): A floating point with at least *n* digits of precision.

Null Value

Every type can have the special value null.

A value of null indicates the value is unknown or that it may not exist at all.

Sometimes we do not want a null value at all – we can add such a constraint.

Creating a Table

SQL Notation:

CREATE TABLE table_name
(col_name$_1$ type$_1$,
col_name$_2$ type$_2$,
… ,
col_name$_n$ type$_n$,
<integrity-constraint$_1$>,
… ,
<integrity-constraint$_k$>);

Integrity Constraints

Primary key(list of attributes) :

These attributes form the primary keys for the relation. Primary keys must be *non-null* and *unique*.

Foreign key(list of attributes) references *s* :

The values of these attributes for any tuple in the relation must correspond to values of the *primary key attributes* of some tuple in relation *s*.

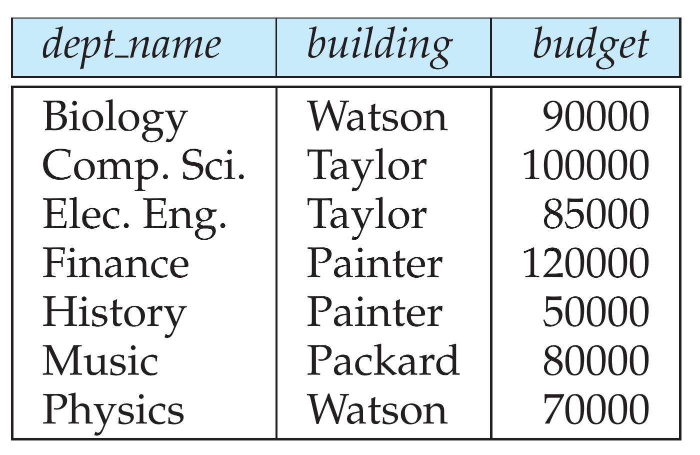
not null:

Specifies that this attribute may not have the *null value*. We list this constraint when defining the type of the attribute.

Examples

```
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),
         credit         NUMERIC(2,0),
         PRIMARY KEY (course_id),
         FOREIGN KEY (dept_name) REFERENCES department );
```



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

Editing Tables

DROP TABLE table_name;	remove the table

DELETE FROM table_name
WHERE predicate;
	delete tuples satisfying the predicate

ALTER TABLE table_name ADD column type;	add a column
ALTER TABLE table_name DROP column;	remove a column

Inserting

In MySQL we can insert into a table with the command:

INSERT INTO table_name
VALUES (value$_1$, value$_2$, …, value$_n$,);

OR

INSERT INTO table_name (col$_1$, col$_2$, …, col$_n$)
VALUES (value$_1$, value$_2$, …, value$_n$,);

OR

INSERT INTO table_name
SELECT QUERY

For example:

```
INSERT INTO instructor
        SELECT ID, name, dept_name, 18000
        FROM student
        WHERE dept_name = ‘Music’ AND tot_cred > 144;
```

Updating

In MySQL we can update a table with the command:

```
        UPDATE table_name
                SET attribute = new_value
```

OR

```
        UPDATE table_name
                SET attribute = new_value
                WHERE predicate or select statement;
```

OR

```
        UPDATE table_name
                SET attribute = CASE
                                WHEN predicate_1 THEN result_1
                                WHEN predicate_2 THEN result_2
                                        …
                                WHEN predicate_n THEN result_n
                                ELSE    result_0
                              END
```