SQL (MySQL)

Gary KL Tam

Department of Computer Science
Swansea University

MySQL and Set

- SQL discussed in previous lecture are for Oracle.
- However, mysql does not support the following SQL operations!
  - minus / except
  - intersect
- What should we do?
  - Write the query in an alternative way.
  - Use our knowledge in relational algebra!

Relational algebra

- Revisit
  - Set Union ∪
  - Set difference (minus) -
  - Set Intersection ∩
  - Join
  - Division ÷

Union

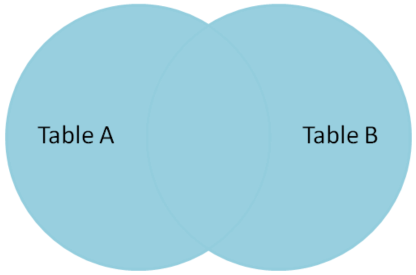
a

x	y
1	A
2	B
3	C
4	D

$\cup$

b

x	y
1	A
3	C



```
(select * from a)
union
(select * from b);
```

x	y
1	A
2	B
3	C
4	D

```
(select * from a)
union all
(select * from b);
```

x	y
1	A
2	B
3	C
4	D
1	A
3	C

Difference

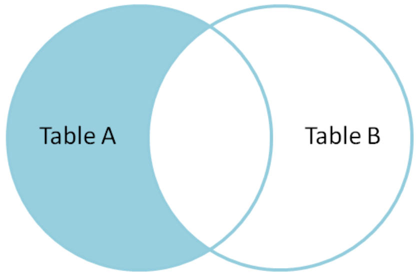
a

x	y
1	A
2	B
3	C
4	D

-

b

x	y
1	A
3	C



```
select * from a
where (x,y) not in (select *
                        from b);
```

x	y
2	B
4	D

For every tuple in a, check that it is **not** in b;

```
select * from a
where not exists (select * from b
                  where b.x=a.x and b.y=a.y)
```

x	y
2	B
4	D

For every tuple in a, check that the tuple values **does not exists** in b;

Intersection

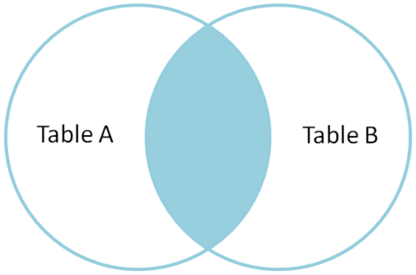
a

x	y
1	A
2	B
3	C
4	D

$\cap$

b

x	y
1	A
3	C



```
select * from a
where (x,y) in (select * from b);
```

x	y
1	A
3	C

For every tuple in a, check that it **is also** in b;

```
select * from a
where exists (select * from b
              where b.x=a.x and b.y=a.y)
```

x	y
1	A
3	C

For every tuple in a, check that the tuple values also **exists** in b;

Join (recap)

- Natural join

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

```
select distinct PROF.pid, name, dept, rank, sal, cid, year
from PROF, TEACH
where PROF.pid = TEACH.pid
```

```
+-----+---------+------+------+-------+------+------+
| pid | name    | dept | rank | sal   | cid  | year |
+-----+---------+------+------+-------+------+------+
| p1  | Adam    | CS   | asst |  6000 | c1   | 2011 |
| p1  | Adam    | CS   | asst |  6000 | c2   | 2012 |
| p2  | Bob     | EE   | asso |  8000 | c2   | 2012 |
+-----+---------+------+------+-------+------+------+
```

Join

- Join

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

select *
from PROF ***inner join*** TEACH
on PROF.pid = TEACH.pid;

```
+-----+---------+------+------+-------+------+------+------+
| pid | name    | dept | rank | sal   | pid  | cid  | year |
+-----+---------+------+------+-------+------+------+------+
| p1  | Adam    | CS   | asst |  6000 | p1   | c1   | 2011 |
| p1  | Adam    | CS   | asst |  6000 | p1   | c2   | 2012 |
| p2  | Bob     | EE   | asso |  8000 | p2   | c2   | 2012 |
+-----+---------+------+------+-------+------+------+------+
```

Join

- Left Outer Join (also check right outer join)

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012

select * from PROF ***left outer join*** TEACH ***on PROF.pid = TEACH.pid***;

```
+-----+---------+------+------+-------+------+------+------+
| pid | name    | dept | rank | sal   | pid  | cid  | year |
+-----+---------+------+------+-------+------+------+------+
| p1  | Adam    | CS   | asst |  6000 | p1   | c1   | 2011 |
| p1  | Adam    | CS   | asst |  6000 | p1   | c2   | 2012 |
| p2  | Bob     | EE   | asso |  8000 | p2   | c2   | 2012 |
| p3  | Calvin  | CS   | full | 10000 | NULL | NULL | NULL |
| p4  | Dorothy | EE   | asst |  5000 | NULL | NULL | NULL |
| p5  | Emily   | EE   | asso |  8500 | NULL | NULL | NULL |
+-----+---------+------+------+-------+------+------+------+
```

http://blog.codinghorror.com/a-visual-explanation-of-sql-joins/
http://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg

Join

- A reminder
- Left/Right outer join ***does NOT equal*** to Inner join

Make sure you know their differences

- You are warned!
- There is something called "fuller outer join" = left ∪ right outer join

Division (revisit)

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2

x
1
2
3

$=$

y
A

$S_1 - S_2 = \{y, x\} - \{x\} = \{y\}$

$$T_1 \div T_2 = \Pi_{S_1 - S_2}(T_1) - \Pi_{S_1 - S_2}\left(\Pi_{S_1 - S_2}(T_1) \times T_2 - T_1\right)$$

```
(select distinct y from T1)
minus
select distinct y from (
        (select * from (select distinct y from T1), T2)
         minus
        (select * from T1)
)
```

Division (MySQL)

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2

x
1
2
3

$=$

y
A

$S_1 - S_2 = \{y, x\} - \{x\} = \{y\}$

$$T_1 \div T_2 \quad = \quad \Pi_{S_1 - S_2}(T_1) - \Pi_{S_1 - S_2}\left(\Pi_{S_1 - S_2}(T_1) \times T_2 - T_1\right)$$

MySQL
- does not support minus!
- But we can use **not in** / **not exists**.

Division (MySQL: no minus)

T_1, S_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2, S_2

x
1
2
3

$=$

y
A

$S_1 - S_2 = \{y, x\} - \{x\} = \{y\}$

$$T_1 \div T_2 \quad = \quad \Pi_{S_1-S_2}(T_1) - \Pi_{S_1-S_2}\left(\Pi_{S_1-S_2}(T_1) \times T_2 - T_1\right)$$

```
select distinct y
from T1
where y not in (select distinct y
                from ((select distinct y from T1) as Tt, T2)
                where (y,x) not in (select * from T1)
               );
```

MySQL: Every derived table must have its own alias

Division (MySQL: no minus)

```
(select distinct y from T_1)
minus
select distinct y from (
          (select * from (select distinct y from T_1), T_2)
           minus
          (select * from T_1)
)
```

```
select distinct y
from T_1
where y not in (select distinct y
                from ((select distinct y from T_1) as T_t, T_2)
                where (y,x) not in (select * from T_1)
               );
```

Same colour shows same block

Rewrite using **not in**

If you understand the relational algebra, you understand the above.

Division (not exists)

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

=

y
A

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists ( select * from T1 as s
                                       where s.y = T1.y and
                                             s.x = T2.x
                                     )
                 );
```

Division – analysis (Past Muddiest Points)

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

=

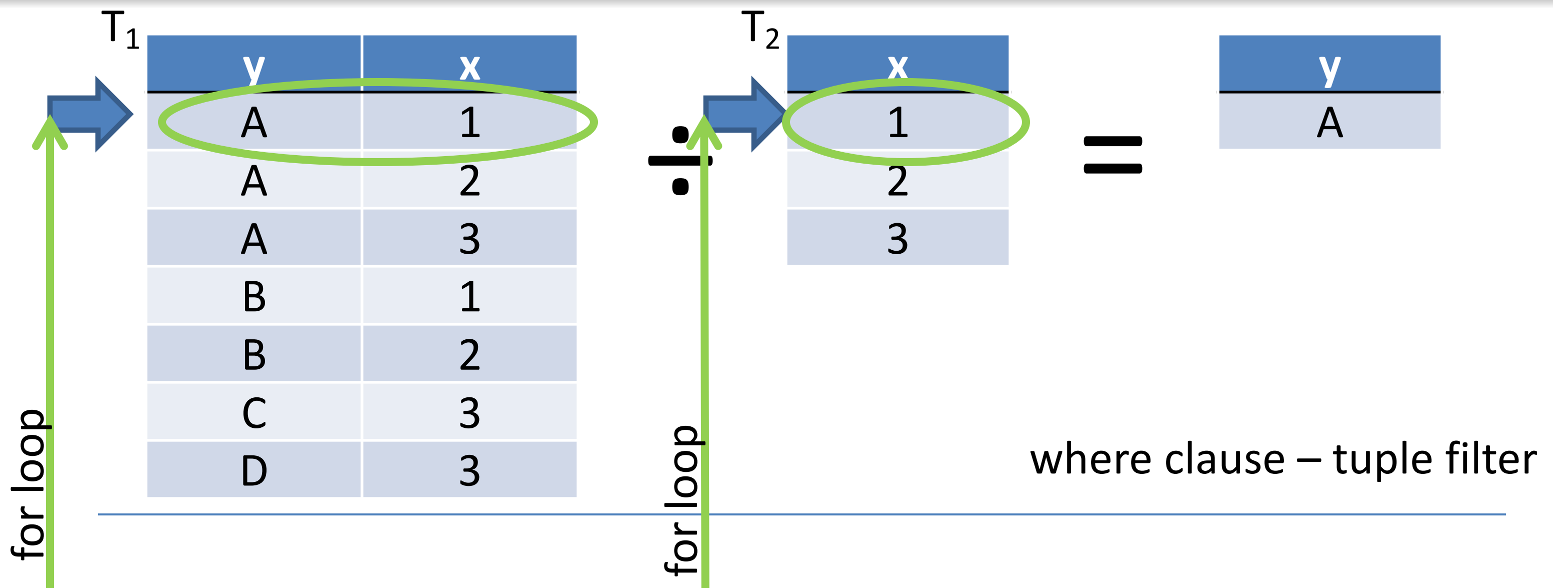
y
A

for loop

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists ( select * from T1 as s
                                       where s.y = T1.y and
                                             s.x = T2.x
                                     )
                 );
```

Division - analysis



T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

T_2

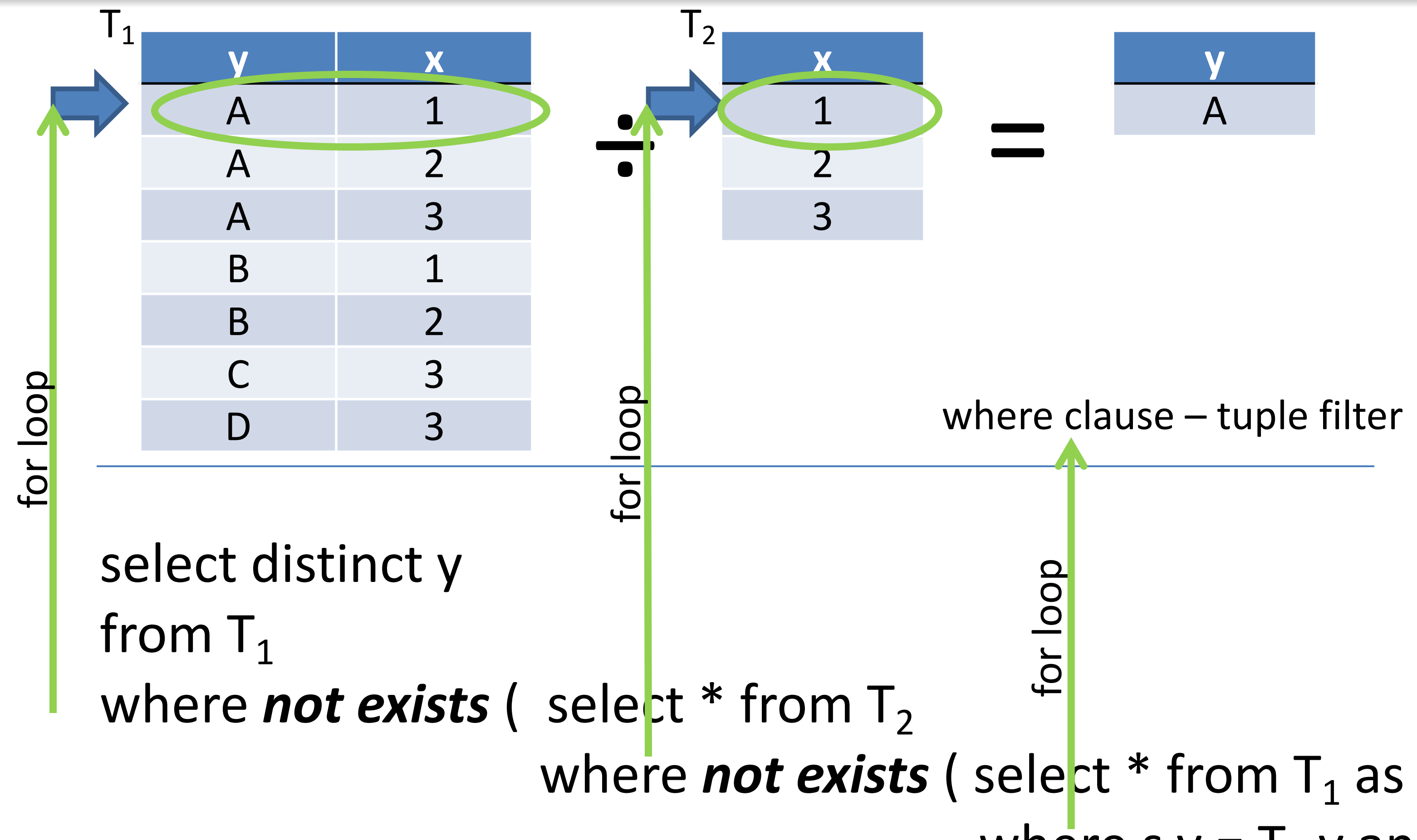
x
1
2
3

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists ( select * from T1 as s
                                          where s.y = T1.y and
                                                s.x = T2.x
                                     )
                 );
```

Division - analysis



T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2

x
1
2
3

$=$

y
A

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists ( select * from T1 as s
                                       where s.y = T1.y and
                                             s.x = T2.x
                                     )
                 );
```

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

=

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                   where not exists ( select * from T1 as s
                                               where s.y = T1.y and
                                                         s.x = T2.x
                                    )
                 );
```

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2

x
1
2
3

$=$

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (select * from T_1 as s
where s.y = 'A' and
s.x = 1
)
);

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

✗

=

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (

y	x
A	1

)
);

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

$\div$

T_2

x
1
2
3

✗ =

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists ( select * from T1 as s
                                       where s.y = 'A' and
                                             s.x = 2
                                     )
                 );
```

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

✗ =

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (

y	x
A	2

)
);

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1
2
3

✗
✗

=

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists (



                    )
);
```

y	x
A	2

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x	
1	✗
2	✗
3	✗

=

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists (
```

y	x
A	3

```
                    )
);
```

Division - analysis

T_1

y	x
A	1 ✓
A	2
A	3
B	1
B	2
C	3
D	3

÷

T_2

x
1 ✗
2 ✗
3 ✗

=

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (

x
empty

);

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

✓

÷

T_2

x
1
2
3

✗

=

y
A

where clause – tuple filter

select distinct y

from T_1

where ***not exists*** (select * from T_2

where ***not exists*** (se

y	x
A	1

s

where s.y = T_1.y and

s.x = T_2.x

)

);

Division - analysis

T_1

y	x	
A	1	✓
A	2	
A	3	
B	1	
B	2	
C	3	
D	3	

÷

T_2

x	
1	✗
2	✗
3	

=

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (se

y	x
A	2

s
where s.y = T_1.y and
s.x = T_2.x
)
);

Division - analysis

T_1

y	x
A	1
A	2
A	3
B	1
B	2
C	3
D	3

✓

÷

T_2

x
1
2
3

✗
✗
✗

=

y
A

where clause – tuple filter

select distinct y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (se

y	x
A	3

s
where s.y = T_1.y and
s.x = T_2.x
)
);

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	
B	1	
B	2	
C	3	
D	3	

÷

T_2

x	
1	✗
2	✗
3	✗

=

y
A

where clause – tuple filter

select ***distinct*** y
from T_1
where ***not exists*** (

x
empty

);

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	✓
B	1	
B	2	
C	3	
D	3	

$\div$

T_2

x	
1	✗
2	
3	

$=$

y
A

where clause – tuple filter

select ***distinct*** y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (se … s
where s.y = T_1.y and
s.x = T_2.x
)
);

y	x
B	1

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	✓
B	1	
B	2	
C	3	
D	3	

÷

T_2

x	
1	✗
2	✗
3	

=

y
A

where clause – tuple filter

select ***distinct*** y
from T_1
where ***not exists*** (select * from T_2
where ***not exists*** (se

y	x
B	2

s
where s.y = T_1.y and
s.x = T_2.x
)
);

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	✓
B	1	
B	2	
C	3	
D	3	

÷

T_2

x	
1	✗
2	✗
3	✓

=

y
A

where clause – tuple filter

```
select distinct y
from T1
where not exists (  select * from T2
                    where not exists (se                    s
                                   where s.y = T1.y and
                                         s.x = T2.x
                                   )
                 );
```

y	x
B	3

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	✓
B	1	✗
B	2	
C	3	
D	3	

÷

T_2

x	
1	✗
2	✗
3	✓

=

y
A

where clause – tuple filter

select ***distinct*** y
from T_1
where ***not exists*** (

x
3

);

Division - analysis

T_1

y	x	
A	1	✓
A	2	✓
A	3	✓
B	1	✗
B	2	✗
C	3	✗
D	3	✗

÷

T_2

x
1
2
3

=

y
A

where clause – tuple filter

select ***distinct*** y

from T_1

where ***not exists*** (select * from T_2

where ***not exists*** (select * from T_1 as s

where s.y = T_1.y and

s.x = T_2.x

)

);

MySQL and Set

- There are at least three other techniques to rewrite Division SQL queries in MySQL.

- http://users.abo.fi/soini/divisionEnglish.pdf
- See Canvas for a cache.
- A clear pictorial explanation is also provided.