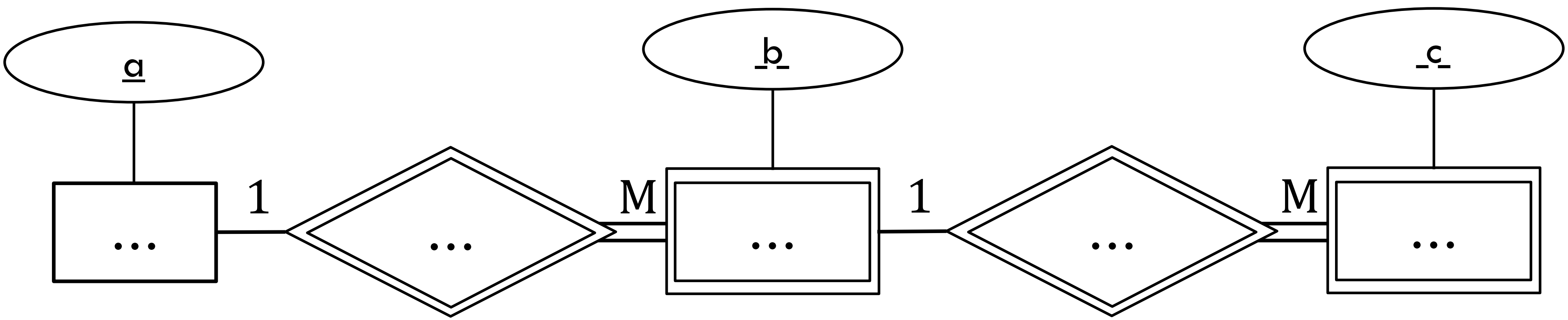
CS 5530

Database Systems
Spring 2020

Adv. Queries I

Weak Entity Chain

- Keys get progressively more complex down the chain

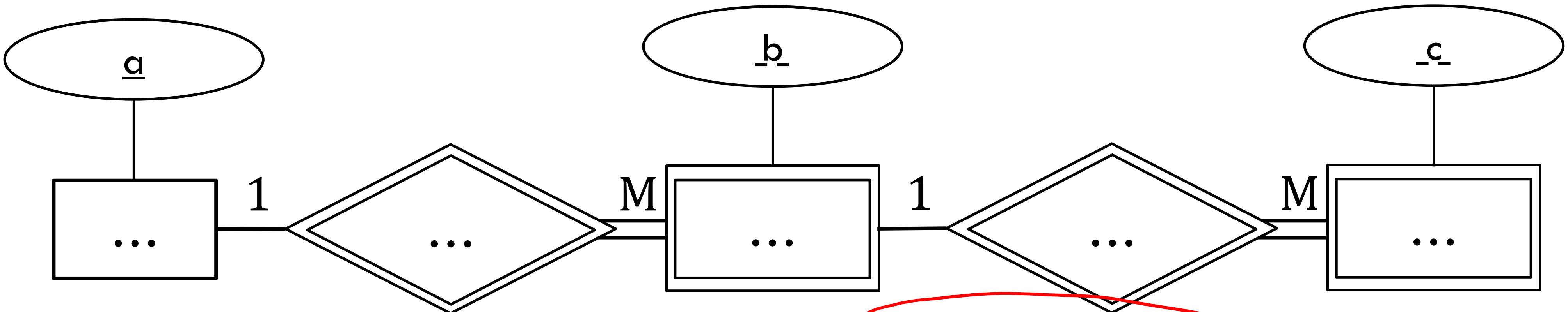


{a} {b, a} {c, b, a}

- This only becomes a concern in actual database
  - In ER it doesn't matter

Translating Weak Entity Chain

- Primary keys should be simple



a
...

PK

b	a
...	...

--PK--
FK

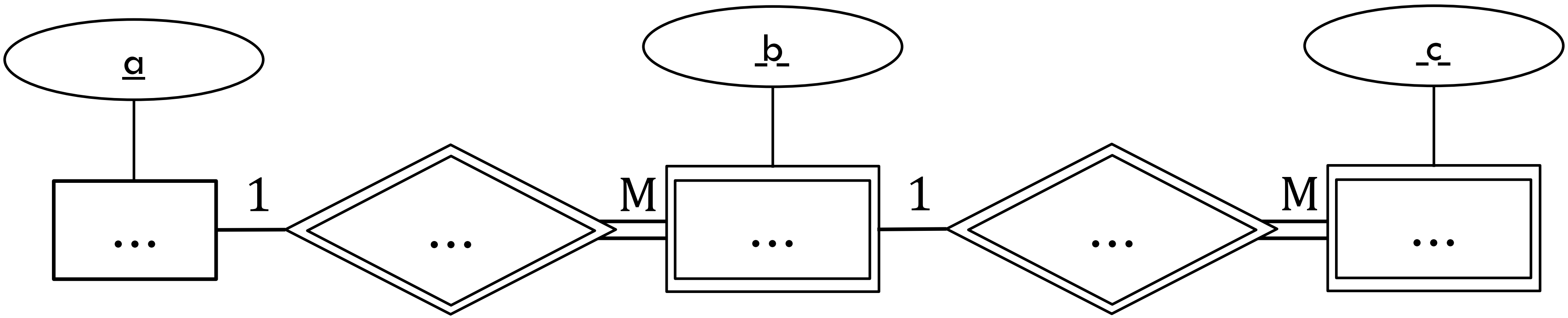
c	b	a
...	...	...

----PK----
--FK--

- Direct translation

Translating Weak Entity Chain

- Primary keys should be simple



a
…

b	a
…	…

c	b	a
…	…	…

PK

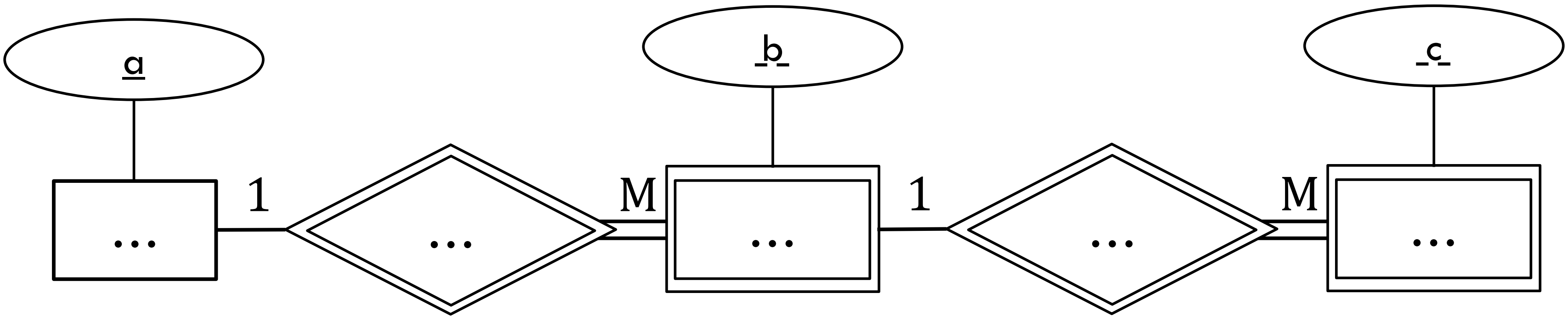
--UQ--

FK

- Achieve same constraint with `UNIQUE` (candidate key)

Translating Weak Entity Chain

- Add a new PK to represent each `{b, a}`



a
...

PK

b	a	bID
...	...	

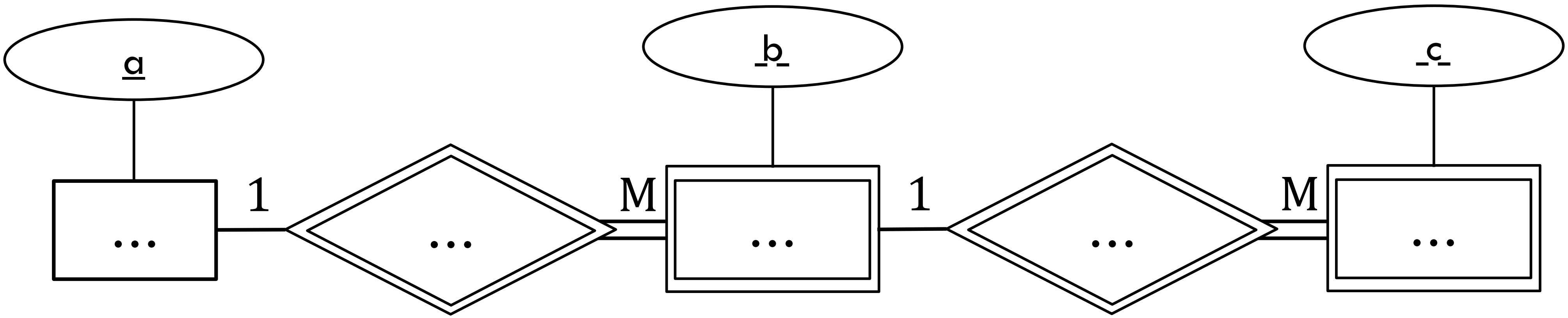
--UQ-- PK

FK

c	b	a
...	...	...

Translating Weak Entity Chain

- Add a new PK to represent each `{b, a}`



a
…

PK

b	a	bID
…	…	

--UQ-- PK

FK

c	bID
…	…

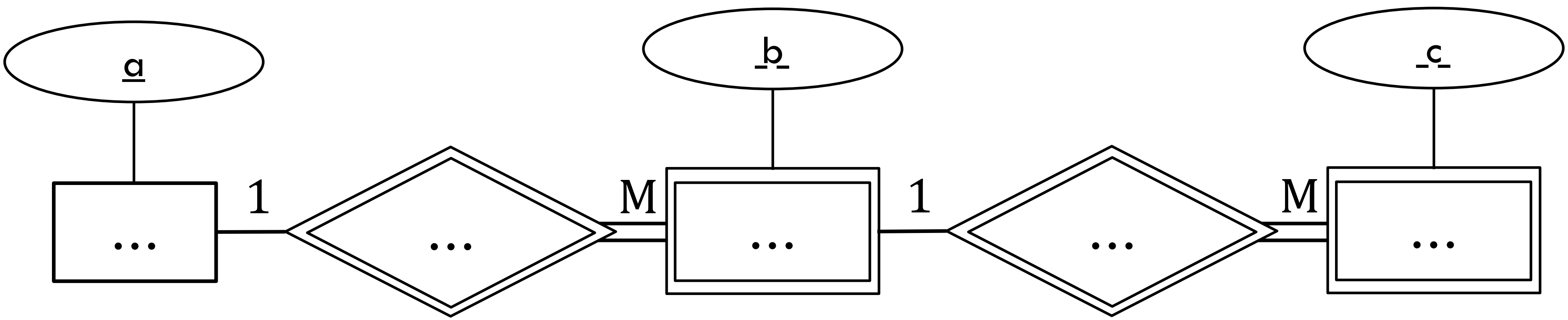
--UQ--

FK

- `bID` now represents complex key in smaller footprint

Translating Weak Entity Chain

- Add a new PK to represent each `{b, a}`



a
...

PK

b	a	bID
...	...	

--UQ-- PK

FK

c	bID	cID
...	...	...

--UQ-- PK

FK

- `bID` now represents complex key in smaller footprint

Example

Courses

Dept	Num
CS	2420
ART	1020
CS	3500

---------PK---------
FK

Classes

Dept	Num	Semester	Location
CS	2420	F18	ASB 220
CS	2420	S20	WEB L104
ART	1020	S20	ART 361

----------------PK--------------
-------FK-------

Example

Courses

Dept	Num
CS	2420
ART	1020
CS	3500

---------PK---------

FK

Classes

Dept	Num	Semester	Location
CS	2420	F18	ASB 220
CS	2420	S20	WEB L104
ART	1020	S20	ART 361

---------------PK--------------

-------FK-------

Example

Courses

Dept	Num	cID
CS	2420	1
ART	1020	2
CS	3500	3

-------UQ------- PK

FK

- Every `{Dept, Num}` matched with a unique `cID`

Example

Courses

Dept	Num	cID
CS	2420	1
ART	1020	2
CS	3500	3

-------UQ------- PK

FK

Classes

cID	Semester	Location
1	F18	ASB 220
1	S20	WEB L104
2	S20	ART 361

----------PK----------

FK

- Now `Classes` can use a smaller FK

Example

Classes

cID	Semester	Location
1	F18	ASB 220
1	S20	WEB L104
2	S20	ART 361

----------PK----------

FK

- Continue down the chain

Example

Classes

cID	Semester	Location	classID
1	F18	ASB 220	x
1	S20	WEB L104	y
2	S20	ART 361	z

----------UQ----------   PK

FK

- Continue down the chain

Example

Classes

cID	Semester	Location	classID
1	F18	ASB 220	x
1	S20	WEB L104	y
2	S20	ART 361	z

----------UQ---------- PK

FK

AsgCats

classID	Name
x	Labs
x	Tests
x	Quiz
y	Labs

-------UQ------- …

FK

•Continue down the chain

Example

Classes

cID	Semester	Location	classID
1	F18	ASB 220	x
1	S20	WEB L104	y
2	S20	ART 361	z

----------UQ---------- PK

FK

AsgCats

classID	Name	acID
x	Labs	1
x	Tests	2
x	Quiz	3
y	Labs	4

-------UQ------- PK

FK

Example

Classes

cID	Semester	Location	classID
1	F18	ASB 220	x
1	S20	WEB L104	y
2	S20	ART 361	z

----------UQ----------   PK

FK

AsgCats

classID	Name	acID
x	Labs	1
x	Tests	2
x	Quiz	3
y	Labs	4

-------UQ-------   PK

FK

Not part of ER model!

Simplifying Keys

- Rule of thumb: if your primary key is:
    - Compound
    - Not a small integer (very small strings OK)

1. Convert it to `UNIQUE`
2. Add a new “ID” primary key (integer type)

Simplifying Keys

- Rule of thumb: if your primary key is:
  - Compound
  - Not a small integer (very small strings OK)

1. Convert it to `UNIQUE`
2. Add a new "ID" primary key (integer type)

- This is an optimization – not part of ER model!
  - And there are exceptions… profile it

Query by Instance

Gradescope Page Numbering

- Some are broken into subparts, some are not
  - Please number all parts and subparts

More on Renaming (ρ)

- In the chess database, suppose you have extracted certain white and black IDs

white

wpID
1
2

$\cup$

black

bpID
2
3
4

=

all

?
1
2
3
4

- $\pi_{?}$(all)

More on Renaming (ρ)

- Rename operator works on columns too

$\rho(newRelation_{newCol1\ /\ oldCol1,\ newCol2\ /\ oldCol2}, X)$

- Relation X has “oldCol1” and “oldCol2”
  - No need to specify all columns, only those being renamed

More on Renaming (ρ)

1 → newName

old → new

$\rho(newWhite_{pID\,/\,wpID,}\ white)$

$\rho(newBlack_{pID\,/\,bpID,}\ black)$

$\pi_{pID}(newWhite \cup newBlack)$

white

wpID
1
2

black

bpID
2
3
4

newWhite

pID
1
2

∪

newBlack

pID
2
3
4

Nested Queries

- Give a name to a temp query

- Find Serials of "The Lorax"

```
select Serial from

   (select ISBN from Titles where
     Title = 'The Lorax') as lorax

natural join Inventory;
```

Serial	ISBN
…	…
1004	*978-0394823379*
1005	*978-0394823379*
…	…

ISBN
978-0394823379

Intersect in MySQL

- Lots of ways to formulate it

CorporateLocs

MngrID	Addr
1	455 Pine Rd.
2	123 Fake St.
5	50 S. Campus

RetailLocs

MngrID	Addr
6	400s State St.
3	750 Rose Park
4	455 Pine Rd.

```
SELECT * FROM
(SELECT Addr FROM CorporateLocs) AS corp
NATURAL JOIN
(SELECT Addr FROM RetailLocs) AS retail;
```

- If schemas are the same, natural join = intersect

Intersect in MySQL

- Using `IN`

CorporateLocs

MngrID	Addr
1	455 Pine Rd.
2	123 Fake St.
5	50 S. Campus

RetailLocs

MngrID	Addr
6	400s State St.
3	750 Rose Park
4	455 Pine Rd.

```
SELECT Addr FROM CorporateLocs
WHERE
Addr IN (SELECT Addr FROM RetailLocs);
```

Set Difference → `NOT IN`

- Find all locations that are corporate-only

CorporateLocs

MngrID	Addr
1	455 Pine Rd.
2	123 Fake St.
5	50 S. Campus

RetailLocs

MngrID	Addr
6	400s State St.
3	750 Rose Park
4	455 Pine Rd.

$\pi_{Addr}(\text{CorporateLocs}) - \pi_{Addr}(\text{RetailLocs})$

```
SELECT Addr FROM CorporateLocs
WHERE
Addr NOT IN (SELECT Addr FROM RetailLocs);
```

Exercise

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

Phones

CardNum	Phone
1	555-5555
2	666-6666

1. All Patrons who have not checked out a book
2. All Patrons who have checked out 'The Lorax' AND 'Harry Potter'

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Ponder

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
4	999-9999
5	777-7777

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- First we have to disambiguate

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 × P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

Self-Join

- Then filter by matching phone number

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 × P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

- $\sigma_{P1.Phone=P2.Phone}(P1 \times P2)$

Self-Join

- Then filter by *not* matching card number

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 × P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

- $\sigma_{P1.Phone=P2.Phone\ \&\&\ P1.CardNum\ !=\ P2.CardNum}(P1 \times P2)$

- Demo…

SQL Self-Join

```
select p1.CardNum, p2.CardNum
from Phones p1 join Phones p2
where p1.Phone = p2.Phone
and p1.CardNum != p2.CardNum;
```

- Range variables (renaming) required for self-join

Join

- Default `join` is called an inner join
- `x JOIN y WHERE ...`
  - Gives rows where condition is true
- Equivalent:
  - `x INNER JOIN y`
  - `x, y`

Outer Join

- Two types of outer join: `LEFT` and `RIGHT`

- `ON` clause required!

- `x LEFT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `x`

Outer Join

- Two types of outer join: `LEFT` and `RIGHT`

- `ON` clause required!

- `x LEFT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `x`

- `x RIGHT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `y`

Left Join

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

Joe	1	1001
Joe	1	1004
Dan	4	1005
Ann	2	
Ben	3	

Left Join

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

```
Patrons p LEFT JOIN CheckedOut c
ON p.CardNum = c.CardNum;
```

Name	CardNum	CardNum	Serial
Joe	1	1	1001
Joe	1	1	1004
Ann	2	NULL	NULL
Ben	3	NULL	NULL
Dan	4	4	1005

- Unmatched tuples get `NULL` in right-side columns

Shortcut

```
Patrons NATURAL LEFT JOIN CheckedOut;
```

Name	CardNum	Serial
Joe	1	1001
Joe	1	1004
Ann	2	NULL
Ben	3	NULL
Dan	4	1005

Shortcut

```
Patrons NATURAL LEFT JOIN CheckedOut;
```

Name	CardNum	Serial
Joe	1	1001
Joe	1	1004
Ann	2	NULL
Ben	3	NULL
Dan	4	1005

Only one copy of natural column(s)

Quiz

- Using left join, find CardNums of Patrons who have not checked out a book

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Three-Valued Logic

- `NULL` is not considered a value in SQL
  - (unlike most programming languages)

- Instead, it represents an “unknown”

Example

- `5 == NULL`
- Read this as: “is 5 equal to an unknown value?”
  - The answer is: “unknown”
  - The answer is **not** false

`NULL` in SQL

- When you see NULL in SQL, replace it in your mind with "unknown"

`NULL` in SQL

- `NULL` does not have the reflexive property
  - `NULL` is not equal to `NULL`
  - i.e. “some unknown value is not equal to some unknown value”

`NULL` in SQL

- `WHERE CardNum != NULL        // wrong`
- `WHERE CardNum IS NOT NULL  // right`

NULL

- Boolean operators on `NULL` always return `NULL`
  - `NULL == 5` → `NULL`
  - `NULL != 5` → `NULL`
  - `NULL == NULL` → `NULL`
  - `NULL != NULL` → `NULL`

Nested Query as Condition

- Filter by nested query

x > 5
x = ...

```
SELECT x FROM y WHERE x IN (SELECT …);
```

$\underbrace{\texttt{x IN (SELECT …);}}_{\text{Condition}}$

Nested Query as Condition

- There are several of these operators

- `x` is a value, `A` is a multi-set (e.g. from `SELECT`)

```
x IN A

EXISTS A

x OP ANY A

x OP ALL A
```

Nested Query as Condition

- There are several of these operators
- `x` is a value, `A` is a multi-set (e.g. from `SELECT`)

`x IN A` → true if $x \in A$

`EXISTS A` → true if `A` is not empty

`x OP ANY A` → true if $\exists y \in A$, x OP y = true

`x OP ALL A` → true if $\forall y \in A$, x OP y = true

Example

Find all students younger than *everyone* in Databases

Students

sID	sName	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B
3	3500	C
3	3810	B
4	3500	C

Courses

cID	cName
3500	SW Practice
3810	Architecture
5530	Databases

Example

- Find all students younger than everyone in ‘Databases’

```
select s2.sName from Students s2
where s2.DOB > all
(select DOB from
Students natural join Enrolled
natural join Courses c
where c.cName='Databases');
```

EXISTS

- Filter by complex nested query

```
select x from y where
EXISTS
(select …);
```

- If any rows exist in nested query, `x` is selected

EXISTS

- Filter by complex nested query

```
select x from y where
NOT EXISTS
(select …);
```

- If nested query empty, `x` is selected

Division

- Find students taking all classes

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810