CS 5530

Database Systems
Spring 2020

Finish SQL Intro

Relational Algebra

Multiple Join

- Join operates on any two relations

```
Patrons JOIN CheckedOut
```

- Result of a join is itself a table that can be joined

```
(Patrons JOIN CheckedOut) JOIN Inventory
```

- Parentheses not needed

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
4	999-9999

Query to get **titles and authors** of books checked out by Joe?

Titles

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

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
4	999-9999

Query to get **phone number** of person holding Harry Potter?

Titles

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

Modifying Tables

```
INSERT INTO <table name>
  VALUES( value1, value2, … );
```

- Inserts in to every column, in order

Modifying Tables

```
INSERT INTO <table name>
  VALUES( value1, value2, … );
```

- Inserts in to every column, in order

```
INSERT INTO Titles VALUES
  ("978-0441172719", "Dune", "Herbert");
```

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0441172719	Dune	Herbert

Modifying Tables

```
INSERT INTO <table name> (col1, col5)
  VALUES( value1, value5 );
```

- Insert only into `col1` and `col5`
  - Use this if `NULL/DEFAULT` column, or `AUTO_INCREMENT`

Modifying Tables

```
DELETE FROM <table name>
  WHERE ...
```

- Delete a row from a table, e.g.

```
DELETE FROM Titles
  WHERE Author = ‘Herbert’;
```

Modifying Tables

```
DELETE FROM <table name>
  WHERE ...
```

- Delete a row from a table, e.g.

```
DELETE FROM Titles
  WHERE Author = 'Herbert';
```

- Leave off the 'where' clause for fun times!

Modifying Tables

```
UPDATE <table name>
   SET col1 = val1, col2 = val2
   WHERE ...
```

•Modify a row, e.g.

```
UPDATE Titles
   SET Title=‘Dune'
   WHERE ISBN = '978-0441172719';
```

Modifying Tables

- Add a column
  - `ALTER TABLE <table> ADD <column>`
  - e.g.

```
ALTER TABLE Titles
ADD PubDate DATE NOT NULL;
```

Modifying Tables

- Change a column
  - ```
    ALTER TABLE <table>
    MODIFY <column> <new type>;
    ```
- e.g.
  - ```
    ALTER TABLE Inventory
    MODIFY Serial INT UNSIGNED;
    ```
- Converts to new type if possible, otherwise picks a default

Modifying Tables

- Add a foreign key
  - ```
    ALTER TABLE <table> ADD
    FOREIGN KEY(<column>)
    REFERENCES <table>(column)
    ```
  - e.g.

```
ALTER TABLE Phones
ADD FOREIGN KEY(CardNum)
REFERENCES Patrons(CardNum);
```

- Not allowed if FK constraint fails
```

Integrity Constraints

- All of this stuff is about integrity constraints
  - primary key
  - foreign keys
  - not null
  - unique

- Maintain the integrity and meaning of the data

Integrity Constraints

- SQL is pretty good about enforcing them
  - i.e. matching the relational model

Integrity Constraints

- SQL is pretty good about enforcing them
  - i.e. matching the relational model
- Unless you don’t want it to…
  - It can be costly in terms of performance
  - You could easily just not specify a FK

Query Languages

- A query language is *not* a programming language
  - May not even be Turing-complete

Programming Languages

- Any PL can implement any computational problem

- But we start with an abstract algorithm description

Programming Languages

- Any PL can implement any computational problem
- But we start with an abstract algorithm description

```
function merge(left, right)
  var result := empty list
  while left not empty and right not empty
    if first(left) ≤ first(right) then
    ...
```

C
Java
Python
...

Query Languages

- Much like an algorithm, it is powerful to specify a query in simpler language

$$\pi_{\texttt{Title, Author}} (\texttt{Titles} \times \texttt{Inventory})$$

Formal Query Languages

- Relational Algebra
  - Operational – describe plan of execution

- Relational Calculus
  - Declarative – describe what you want in the end

Formal Query Languages

- Relational Algebra
    - Operational – describe plan of execution

- Relational Calculus
    - Declarative – describe what you want in the end

- These form the basis of practical languages like SQL

Relational Algebra

- Algebra that operates on relations

- The $\boxed{\text{domain}}$ is relational instances
  - Operators consume and produce relational instances

$1 + 2 = 3$

Patrons x Phones → R3

Relational Algebra

- Five basic operators
  - π Projection — select
  - σ Selection — where
  - $\times$ Cross Product — Join
  - $\cup$ Set Union
  - $-$ Set Difference

Relational Algebra

- Five basic operators

  - π Projection
  - σ Selection
  - × Cross Product
  - ∪ Set Union
  - — Set Difference

$1 + 5 \rightarrow$

$7 / 5 \rightarrow$

$\pi((R1 \times R2) - R3)$

- Operations are $\boxed{\textit{closed}}$: input/output is always a relation

  - Thus, we can compose operations

Projection

- Get certain columns from relation
- $\pi_{\text{column list}}$ (relation)

Projection

- Get certain columns from relation
- $\pi_{\text{column list}}$ (relation)
- Example:
- $\pi_{\text{Title, Author}}$ (Titles)

Projection

- Get certain columns from relation

- $\pi_{\text{column list}}$ (relation)

- Example:

- $\pi_{\text{Title, Author}}$ (Titles)

- $\pi_{\text{Title, Serial}}$ (Titles $\times$ Inventory)

Projection

- SQL vs. Relational Algebra

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

Projection

- SQL vs. Relational Algebra

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

- `SELECT Author FROM Titles;`

```
Herbert
Herbert
```

$\pi_{Author}(Titles)$

$\rightarrow$

Author
Herbert
~~Herbert~~

Projection

- SQL vs. Relational Algebra

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

- $\pi_{\texttt{Author}}$ `(Titles)`

Projection

- SQL vs. Relational Algebra

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

- $\pi_{\texttt{Author}}$ `(Titles)`

```
Herbert
```

- Output of π must be a relation (no duplicates)

Projection

- Why does SQL not eliminate duplicates by default?

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

- `SELECT Author FROM Titles;`

```
Herbert
Herbert
```

- If we want the count, we can `SELECT COUNT` …

Projection

- Why does SQL not eliminate duplicates by default?
  - Because it's computationally expensive

Titles

ISBN	Title	Author
978-0441172720	Children of Dune	Herbert
978-0441172719	Dune	Herbert

Selection

- Filter rows on condition

- $\sigma_{\text{condition}}$ (relation)

Selection

- Filter rows on condition
- $\sigma_{\text{condition}}$ (relation)
- Example:
- $\sigma_{\text{CardNum} > 3}$ (Patrons)

Selection

- Filter rows on condition
- $\sigma_{\text{condition}}$ (relation)

- Example:
- $\sigma_{\text{CardNum} > 3}$ (Patrons)
- $\pi_{\text{Phone}}(\sigma_{\text{Patrons.CardNum} > 3}$ (Patrons $\times$ Phones))

Selection + Projection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	~~4~~
Dan	~~5~~

$$\pi_{Name}(\sigma_{CN > 3}(Patrons)) \rightarrow \overline{Dan}$$

Selection vs. Projection

- Selection (σ)
  - Eliminates rows

- Projection (π)
  - Eliminates columns

Selection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	4
Dan	5

- $\sigma_{\text{CardNum} > 3}$ (Patrons)

Selection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	4
Dan	5

- $\sigma_{\text{CardNum} > 3}$ (Patrons)
- What about duplicate elimination?

Selection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	4
Dan	5

- $\sigma_{\text{CardNum} > 3}$ (Patrons)

- What about duplicate elimination? No need
  - Any relation is already a set
  - A subset of a set is also a set

Selection + Projection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	4
Dan	5

- $\pi_{CardNum}(\sigma_{CardNum > 3}(Patrons))$ → {4, 5}

Selection + Projection

Patrons

Name	CardNum
~~Joe~~	~~1~~
~~Ann~~	~~2~~
~~Ben~~	~~3~~
Dan	4
Dan	5

- $\pi_{Name}(\sigma_{CardNum > 3}\ (Patrons)\ ) \rightarrow \{\text{“Dan”}\}$

removes duplicates

Cross Product

- Binary operator (takes two operands)
- relation1 × relation2 → relation3

Cross Product

- Binary operator (takes two operands)
- relation1 × relation2 → relation3
- Example:
- R1 × R2

R1

Col1
a
b

×

R2

Col1
x
y
z

=

R3

R1.Col1	R2.Col1
a	x
a	y
a	z
b	x
b	y
b	z

Cross Product

- Binary operator (takes two operands)
- relation1 × relation2 → relation3
- Example:
- R1 × R2

R1

Col1
a
b

×

R2

Col1
x
y
z

=

R3

R1.Col1	R2.Col1
a	x
a	y
a	z
b	x
b	y
b	z

- Column names must be disambiguated

Quiz

- Does cross product need to eliminate duplicates?

Quiz

- Does cross product need to eliminate duplicates?
  - No
  - All R1 rows are unique and all R2 rows are unique

Cross Product

- Is cross product commutative?
  - $R1 \times R2 == R2 \times R1$?

Cross Product

- Is cross product commutative?
  - $R1 \times R2 == R2 \times R1$?
- Yes
  - SQL analog (JOIN) may give columns in a different order, but it doesn't matter

Set Union

- relation1 ∪ relation2 → relation3

- Think of it like "add"

Col1
a
b

∪

Col1
x
y
z

=

Col1
a
b
x
y
z

Set Union

- relation1 ∪ relation2 → relation3

- Relations must be union compatible (same schema)
  - i.e. same number of columns, same column *types*

float	int
…	…

∪

float	int
…	…

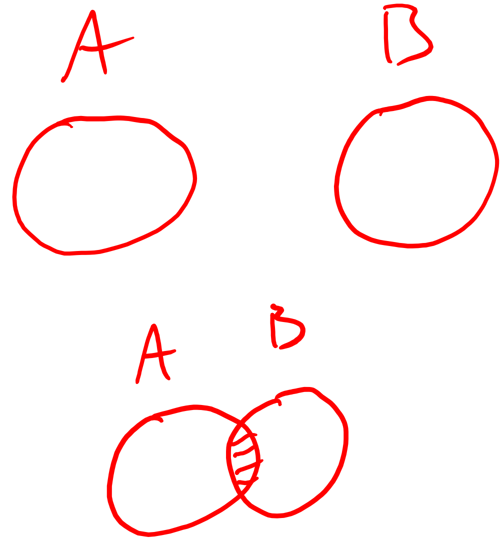
∪

float
…

✓ ✗

Set Union

- What about duplicates?



Set Union

- What about duplicates?
  - They must be removed

Set Union

- Is union commutative?
  - $R1 \cup R2 == R2 \cup R1$?

Set Union

- Is union commutative?
  - R1 ∪ R2 == R2 ∪ R1?

- Yes
  - Again, SQL may reorder tuples, but doesn't matter

Col1
a
b

∪

Col1
x
y
z

=

Col1
a
b
x
y
z

=

Col1
x
y
z
a
b

Set Difference

- relation1 — relation2 → relation3
- Relations must be union compatible

Set Difference

- relation1 — relation2 → relation3
- Think of it as “subtract”
- Remove tuples from relation1 that are also in relation2

A
x
y
z

B
a
b
x

A - B	B - A
y	a
z	b

Set Difference

- Is set difference commutative?
  - $R1 - R2 == R2 - R1$?

Set Difference

- Is set difference commutative?
  - R1 — R2 == R2 — R1?
  - No

R1

Col1
a
b
x

R2

Col1
x
y
z

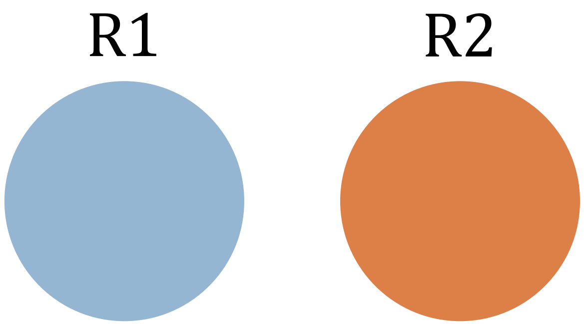
R1—R2

Col1
a
b

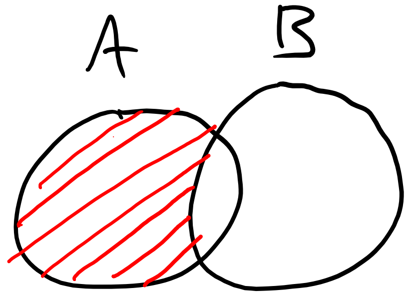
R2—R1

Col1
y
z

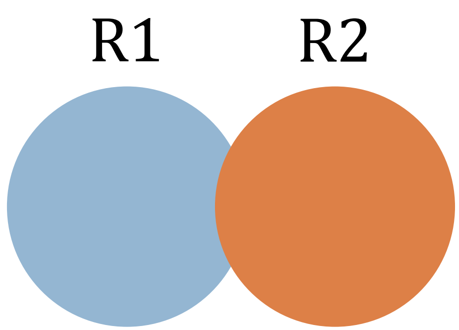
Set Difference



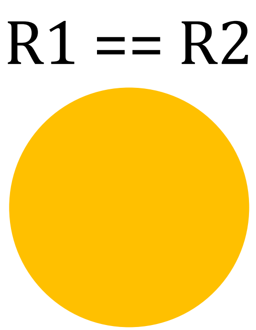
R1 — R2 == R1
R2 — R1 == R2



A − B



R1 — R2 == subset(R1)
R2 — R1 == subset(R2)



R1 — R2 == R2 — R1 == {}

Relational Algebra

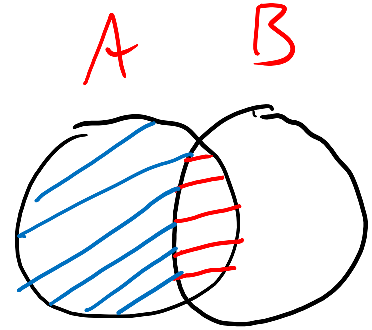
- Five basic operators
  - π Projection
  - σ Selection
  - $\times$ Cross Product
  - — Set Difference
  - $\cup$ Set Union

$A \cap B$

- What about set intersection, or other common operations?

Set Intersection

- Can we formulate set intersection ($\cap$) using only these?
  - $\times$
  - $-$
  - $\cup$



$A \cap B$

$A - B$

$B - (B - A)$

$\boxed{A - (A - B)}$

Set Intersection

- Example R1 ∩ R2

$$= R1 - \boxed{(R1 - R2)}$$

R1

Col1
a
b
x

R2

Col1
x
y
z

R1 $\begin{pmatrix} a \\ b \\ x \end{pmatrix}$ R2 $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow$ R1 - R2: a, b

$$\begin{pmatrix} a \\ b \\ x \end{pmatrix} - \begin{pmatrix} a \\ b \end{pmatrix} - \begin{pmatrix} x \end{pmatrix}$$

Exercise

1. Locations that are retail only
2. Locations that are corporate only
3. Locations that are both retail and corporate
4. Managers who are friends (manage at same address)

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.

- π Projection
- σ Selection
- $\times$ Cross Product
- $\cup$ Set Union
- $-$ Set Difference
- $\cap$ Set Intersection

Exercise

- Using relation algebra, generate queries to find:

1. Locations that are retail only

$$\pi_{Addr}(Retail) - \pi_{Addr}(Corp)$$

2. Locations that are corporate only

3. Locations that have both retail and corporate

$$\pi_{Addr}(Retail) \cap \pi_{Addr}(Corp)$$

4. Managers who are friends (manage at same address)

$$\pi_{C.M,R.M}(\sigma_{C.A=R.A}(Corp \times Retail))$$