

Relational Algebra (and SQL)

Announcements

Homework 1: Due beginning of class Thursday

Project 1 Part 1: End of class Thursday

Project 1 Part 2:

Available Thursday; due in 2 weeks

Comment: lecture timing is bad

Yes, sorry. Will attempt to avoid.

This week: Relational Algebra. Next week: SQL.

-> 1 week to do the project part 2

Comment: Live SQL example from ER

Goal: full example next week, with queries

Q: Still confused about aggregates

Example in the middle of class

Reading

Ramakrishnan Sections 4.1 and 4.2

Helpful Reference

https://en.wikipedia.org/wiki/Relational_algebra

Queries: ask your data a question

Declarative language: ask *what* you want, now *how*

Meaning: Queries are high level, “readable”

Not Turing complete (can’t execute any algorithm)

Supports easy, efficient access to large databases

Domain specific language for data access

SQL: Data Definition Language (DDL)

Data Manipulation Language (DML)

Basic Single Table SELECT

SELECT (*output*) FROM (*input*)
WHERE (*condition*)

SELECT * FROM Students

SELECT name FROM Students

SELECT name FROM Students WHERE age < 21

SELECT name, login FROM Students WHERE gpa >= 3

| sid | name | login | age | gpa |
|-----|--------|---------|-----|-----|
| 1 | eugene | ewu | 20 | 2.5 |
| 2 | luis | gravano | 25 | 3.5 |
| 3 | martha | martha | 32 | 3.9 |

Single Table Semantics

conceptual evaluation method:

1. FROM clause: retrieve Students relation
2. WHERE clause: Check conditions, discard tuples that fail
3. SELECT clause: Delete unwanted fields

Real evaluation is *much* more efficient, but must produce the same answers.

Ambiguous names

E.g. Students: (sid, name, ...)

Enrolled: (sid, cid, grade)

Qualified names: Use table name: Students.age

Rename: AS (optional): shortcuts, ambiguity, clarity

```
SELECT S.sid, S.name FROM Students AS S
```

```
SELECT S.sid, S.name FROM Students S
```

Related data: Multiple tables

What does this return?

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid = E.sid AND  
       E.grade = 'A'
```

Enrolled

| sid | cid | grade |
|-----|-----|-------|
| 1 | 2 | A |
| 1 | 3 | B |
| 2 | 2 | A+ |

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid = E.sid AND  
       E.grade = 'A'
```

Students

| sid | name |
|-----|--------|
| 1 | eugene |
| 2 | luis |

Result

| name | cid |
|--------|-----|
| eugene | 2 |

Multi-Table Semantics

- Modify the FROM clause evaluation
 - 1. FROM clause: compute *cross-product* of Students and

Enrolled

| sid | cid | grade |
|-----|-----|-------|
| 1 | 2 | A |
| 1 | 3 | B |
| 2 | 2 | A+ |

Students

| sid | name |
|-----|--------|
| 1 | eugene |
| 2 | luis |

Cross-product

| sid | cid | grade | sid | name |
|-----|-----|-------|-----|--------|
| 1 | 2 | A | 1 | eugene |
| 1 | 3 | B | 1 | eugene |
| 2 | 2 | A+ | 1 | eugene |
| 1 | 2 | A | 2 | luis |
| 1 | 3 | B | 2 | luis |
| 2 | 2 | A+ | 2 | luis |

Multi-Table Semantics

Modify the FROM clause evaluation

1. FROM clause: compute *cross-product* of Students, Enrolled
2. WHERE clause: Check conditions, discard tuples that fail
3. SELECT clause: Delete unwanted fields

Formal Relational Query Languages

Formal basis for real languages e.g., SQL

Relational Algebra

Function of operations applied to relations

Operational: step-by-step execution plans

Relational Calculus

Logical, describes what data users want

(not operational, fully declarative)

Definitions

Relation (for this lecture)

Instance is a set of tuples

Schema defines field names and types (domains)

Students(sid int, name text, major text, gpa int)

Fields: Reference by name: (e.g. major)

Reference by position starting at 1: (e.g. 3)

Names are for humans; positions for “intermediate” results

Definitions

Query is a function over **relation instances**

$$Q(R_1, \dots, R_n) = R_{\text{result}}$$

Schemas of output relations are well defined by query Q.

Use positional or named field notation

Names in results inherited from input (unless renamed)

Relational Algebra Overview

Core 5 operations

PROJECT (π)

SELECT (σ)

UNION (\cup)

SET DIFFERENCE ($-$)

CROSSPRODUCT (\times)

Additional operations

RENAME (ρ)

INTERSECT (\cap)

JOIN (\bowtie)

DIVIDE ($/$)

Instances Used Today: Library

Students, Reservations

RI

| sid | rid | day |
|-----|-----|-------|
| 1 | 101 | 10/10 |
| 2 | 102 | 11/11 |

SI

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

Project

$$\pi_{\langle \text{attr1}, \dots \rangle}(A) = R_{\text{result}}$$

Extract desired fields (subset of columns)

Schema is subset of input schema in the projection list

$\pi_{\langle a, b, c \rangle}(A)$ has output schema (a, b, c) w/ types carried over

Project

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

$\pi_{\text{name,age}}(S2) =$

| name | age |
|-------|-----|
| aziz | 21 |
| barak | 21 |
| trump | 88 |
| rusty | 21 |

Project

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

$$\pi_{\text{age}}(S2) =$$

| age |
|-----|
| 21 |
| 88 |

Where did all the rows go?

Real systems typically don't remove duplicates. Why?

Select

$$\sigma_{\langle p \rangle}(A) = R_{\text{result}}$$

Select subset of rows that satisfy condition p

Won't have duplicates in result. Why?

Result schema same as input

Select

S1

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

$$\sigma_{\text{age} < 30} (S1) =$$

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |

$$\pi_{\text{name}}(\sigma_{\text{age} < 30} (S1)) =$$

| name |
|--------|
| eugene |
| barak |

Commutativity

$$A + B = B + A$$

$$A * B = B * A$$

$$A + (B * C) = (B * C) + A$$

Associativity:

$$A + (B + C) = (A + B) + C$$

$$A + (B * C) = (A + B) * C$$

Commutativity

$$A + B = B + A$$

$$A * B = B * A$$

$$A + (B * C) = (B * C) + A$$

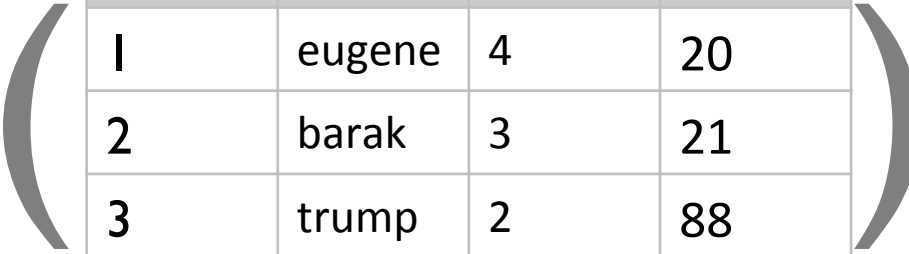
Associativity:

$$A + (B + C) = (A + B) + C$$


~~$$A + (B * C) = (A + B) * C$$~~

Commutativity

$$\pi_{\text{age}}(\sigma_{\text{age} < 30} (SI))$$

$\sigma_{\text{age} < 30}$ 

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |



| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |

Commutativity

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI))$$

$\sigma_{\text{age} < 30}$ $\left(\begin{array}{|c|c|c|c|} \hline \text{sid} & \text{name} & \text{gpa} & \text{age} \\ \hline 1 & \text{eugene} & 4 & 20 \\ \hline 2 & \text{barak} & 3 & 21 \\ \hline 3 & \text{trump} & 2 & 88 \\ \hline \end{array} \right)$

$\underbrace{\hspace{15em}}$

$\pi_{\text{age}} \left(\begin{array}{|c|c|c|c|} \hline \text{sid} & \text{name} & \text{gpa} & \text{age} \\ \hline 1 & \text{eugene} & 4 & 20 \\ \hline 2 & \text{barak} & 3 & 21 \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \text{age} \\ \hline 20 \\ \hline 21 \\ \hline \end{array}$

Commutativity

$$\sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

π_{age} (

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |


)

}


| age |
|-----|
| 20 |
| 21 |
| 88 |


Commutativity

$$\sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$


π_{age} 

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |



$\sigma_{\text{age} < 30}$ 

| age |
|-----|
| 20 |
| 21 |
| 88 |



=

| age |
|-----|
| 20 |
| 21 |

Commutativity

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30} (SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30} (SI))?$$

Commutativity

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) \neq \sigma_{\text{age} < 30}(\pi_{\text{name}}(SI))$$

Commutativity

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) \neq \sigma_{\text{age} < 30}(\pi_{\text{name, age}}(SI))$$

Commutativity

Does Project and Select commute?

$$\pi_{\text{age}}(\sigma_{\text{age} < 30}(SI)) = \sigma_{\text{age} < 30}(\pi_{\text{age}}(SI))$$

What about

$$\pi_{\text{name}}(\sigma_{\text{age} < 30}(SI)) = \pi_{\text{name}}(\sigma_{\text{age} < 30}(\pi_{\text{name, age}}(SI)))$$

OK!

Union, Set-Difference

$$A \text{ op } B = R_{\text{result}}$$

A, B must be *union-compatible*

Same number of fields

Field i in each schema have same type

Result Schema borrowed from first arg (A)

Student(sid int, age int) U Class(cid int, max int) = ?

Union, Set-Difference

$$A \text{ op } B = R_{\text{result}}$$

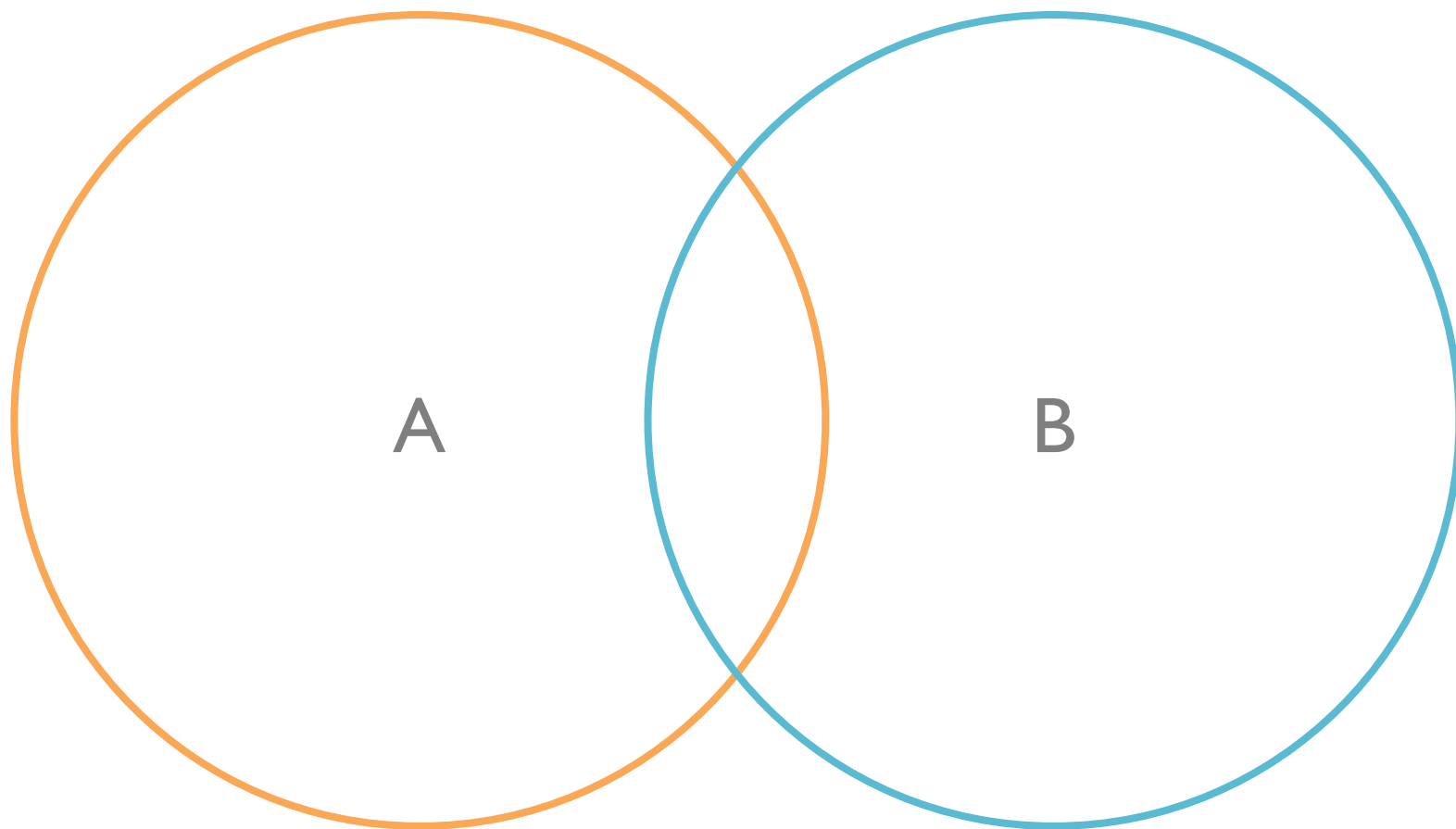
A, B must be *union-compatible*

Same number of fields

Field i in each schema have same type

Result Schema borrowed from first arg (A)

Student(sid int, age int) U Class(cid int, max int) =
 $R_{\text{result}}(\text{sid int, age int})$



Union, Intersect, Set-Difference

S1

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

$S1 \cup S2 =$

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 4 | aziz | 3.2 | 21 |
| 5 | rusty | 3.5 | 21 |
| 3 | trump | 2 | 88 |
| 2 | barak | 3 | 21 |

Union, Intersect, Set-Difference

S1

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

$S1 - S2 =$

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |

Note on Set Difference & Performance

Most operators are monotonic
increasing size of inputs \rightarrow outputs grow
can compute *incrementally*

Set Difference is *not monotonic*:

compute $A - B$

add data in X to B: $B2 = B \cup X$

compute $A - B2$

could be smaller than $A - B$

Set difference is *blocking*:

For $T - S$, must wait for all S tuples before any results

Intersect

$$A \cap B = R_{\text{result}}$$

A, B must be *union-compatible*

Intersect

S1

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

S2

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 4 | aziz | 3.2 | 21 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |
| 5 | rusty | 3.5 | 21 |

$S1 \cap S2 =$

| sid | name | gpa | age |
|-----|-------|-----|-----|
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

Intersect

$$A \cap B = R_{\text{result}}$$

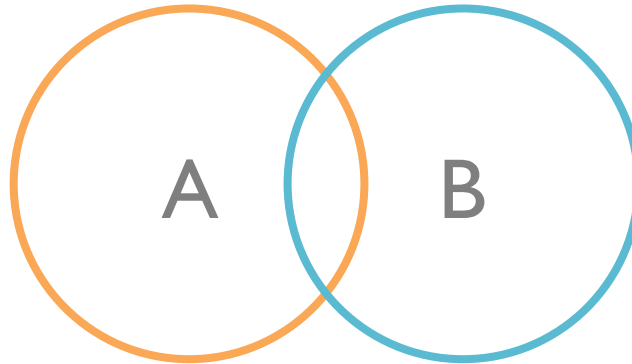
A, B must be *union-compatible*

Can we express using core operators?

$$A \cap B = ?$$

Intersect

$$A \cap B = R_{\text{result}}$$

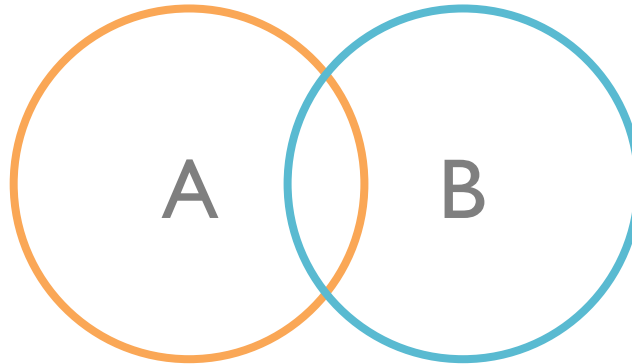


Can we express using core operators?

$A \cap B = A - ?$ (think venn diagram)

Intersect

$$A \cap B = R_{\text{result}}$$



Can we express using core operators?

$$A \cap B = A - (A - B)$$

Cross-Product

$$A(a_1, \dots, a_n) \times B(a_{n+1}, \dots, a_m) = R_{\text{result}}(a_1, \dots, a_m)$$

Each row of A paired with each row of B

Result schema: Combine A and B's fields, inherit if possible

Conflict: students and reservations have *sid* field: need positions

Cross-Product

SI

| sid | name | gpa | age |
|-----|--------|-----|-----|
| 1 | eugene | 4 | 20 |
| 2 | barak | 3 | 21 |
| 3 | trump | 2 | 88 |

RI

| sid | rid | day |
|-----|-----|-------|
| 1 | 101 | 10/10 |
| 2 | 102 | 11/11 |

SI x RI =

| (sid) | name | gpa | age | (sid) | rid | day |
|-------|--------|-----|-----|-------|-----|-------|
| 1 | eugene | 4 | 20 | 1 | 101 | 10/10 |
| 2 | barak | 3 | 21 | 1 | 101 | 10/10 |
| 3 | trump | 2 | 88 | 1 | 101 | 10/10 |
| 1 | eugene | 4 | 20 | 2 | 102 | 11/11 |
| 2 | barak | 3 | 21 | 2 | 102 | 11/11 |
| 3 | trump | 2 | 88 | 2 | 102 | 11/11 |

Rename

$\rho(<\text{new_name}>(<\text{mappings}>), Q)$

Explicitly defines/changes field names of schema

$\rho(C(1 \rightarrow \text{sid1}, 5 \rightarrow \text{sid2}), S1 \times R1)$

C =

| sid1 | name | gpa | age | sid2 | rid | day |
|------|--------|-----|-----|------|-----|-------|
| 1 | eugene | 4 | 20 | 1 | 101 | 10/10 |
| 2 | barak | 3 | 21 | 1 | 101 | 10/10 |
| 3 | trump | 2 | 88 | 1 | 101 | 10/10 |
| 1 | eugene | 4 | 20 | 2 | 102 | 11/11 |
| 2 | barak | 3 | 21 | 2 | 102 | 11/11 |
| 3 | trump | 2 | 88 | 2 | 102 | 11/11 |

Example: Aggregate

Three entities: Company, Product, Service Rep

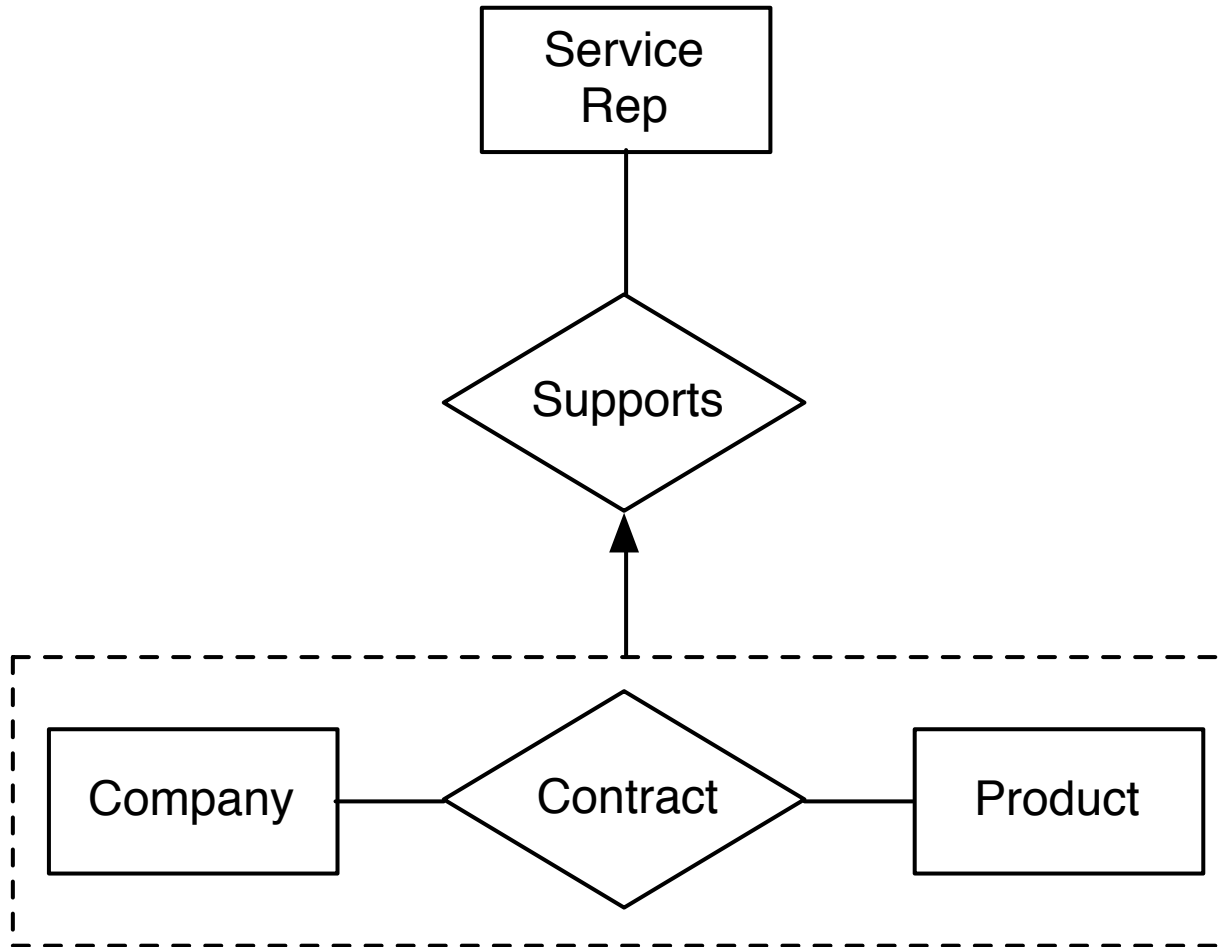
Companies purchase products with contracts

Big contracts have one dedicated service rep

Q1: Draw the E-R diagram (hint: use aggregate)

Q2: Create a schema

Diagram



SQL Schema: Entities

```
CREATE TABLE Company(  
    cid int PRIMARY KEY,  
    email text);  
CREATE TABLE Product(  
    pid int PRIMARY KEY,  
    description text);  
CREATE TABLE ServiceRep(  
    sid int PRIMARY KEY,  
    name text);
```

Relationships (no constraints)

```
CREATE TABLE Contract(  
    cid int REFERENCES Company,  
    pid int REFERENCES Product,  
    PRIMARY KEY (cid, pid));
```

```
CREATE TABLE Supports(  
    cid int,  
    pid int,  
    sid int REFERENCES SupportRep,  
    PRIMARY KEY (cid, pid, sid),  
    FOREIGN KEY (cid, pid) REFERENCES Contract));
```

At most one: Make unique

```
CREATE TABLE Supports(  
  cid int,  
  pid int,  
  sid int REFERENCES SupportRep,  
  PRIMARY KEY (cid, pid, sid),  
  FOREIGN KEY (cid, pid) REFERENCES Contract),  
  UNIQUE (cid, pid));
```

DUPLICATION: UNIQUE + PRIMARY KEY

Reduce duplication

```
CREATE TABLE Supports(  
    cid int,  
    pid int,  
    sid int REFERENCES SupportRep NOT NULL,  
    PRIMARY KEY (cid, pid),  
    FOREIGN KEY (cid, pid) REFERENCES  
Contract));  
DUPLICATION: Same primary key as Contract
```

Combine tables

```
CREATE TABLE Contract(  
    cid int REFERENCES Company,  
    pid int REFERENCES Product,  
    sid int REFERENCES SupportRep,  
    PRIMARY KEY (cid, pid));
```