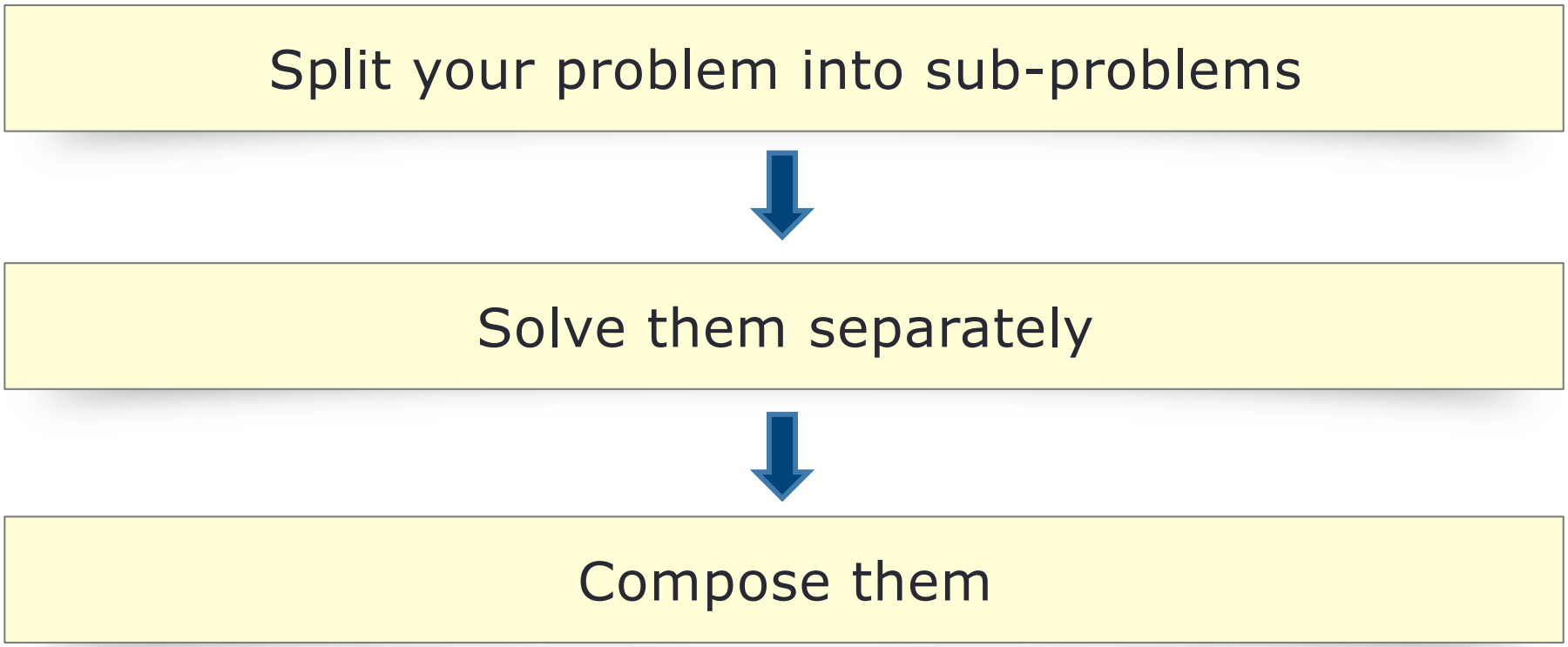
SQL – Subqueries

CS 4750
Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.5.3]

Subqueries: Core Idea



The smaller the problem, the simpler to solve, the easier to debug

Subqueries

- Subquery = a query that is part of another query
- A subquery can have subqueries

Usage:

- Return a single constant that can be used to compute an associated value in a `SELECT` clause
- Return a single constant that can be compared to another value in a `WHERE` clause
- Return a relation that can be compared or evaluated in a `WHERE` clause
- Return a relation that can be used as input for another query, in a `FORM` clause

Equivalent Query Example

Find the average salary for each job

practice_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT job, AVG(sal)
FROM practice_emp
GROUP BY job
```

job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea (self join):

1. Self-join `practice_emp`
2. Use one copy to aggregate, group by job
3. Use one copy to keep the original job

```
SELECT E1.job, AVG(E2.sal) AS AvgSal
FROM practice_emp E1, practice_emp E2
WHERE E1.job = E2.job
GROUP BY E1.job
```

(Equivalent) **Subquery (SELECT)**

Find the average salary for each job

practice_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

"Correlated" query
Recomputed for each tuple
(can't be run independently
of the outer query)

```
SELECT E1.job,
       (SELECT AVG(E2.sal)
        FROM practice_emp AS E2
        WHERE E1.job = E2.job) AS AvgSal
FROM practice_emp E1
GROUP BY E1.job
```

job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea:

1. Group by job
2. For each tuple, compute aggregate

A subquery in `SELECT` returns a single value – used to compute an associated value

(Equivalent) **Subquery (FROM)**

Find the average salary for each job

practice_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

"Uncorrelated" query
Independent of outer query

```
SELECT E1.job, AvgSal
FROM practice_emp E1,
     (SELECT job, AVG(sal) AS AvgSal
      FROM practice_emp
      GROUP BY job) AS E2
WHERE E1.job = E2.job
GROUP BY E1.job
```

job	AvgSal
Analyst	3500.0000
Clerk	1437.5000
Manager	3158.3333
President	6500.0000
Salesman	1800.0000

Idea:

1. Compute aggregate for each job
2. Join the original `practice_emp`

A subquery in `FROM` returns a relation – used as input for another query

Subqueries in WHERE

Find employee name (or names) who earns the highest salary for each job

practice_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

```
SELECT E1.ename
FROM practice_emp E1
WHERE E1.sal =
      (SELECT MAX(E2.sal)
       FROM practice_emp AS E2
       WHERE E1.job = E2.job)
```

"Correlated"

ename
Allen
Jones
Scott
King
Ford
Miller

[more subqueries in WHERE later]

A subquery in `WHERE` returns a single value – to be compared to another value in a `WHERE` clause

Subqueries in WITH

Find employee name (or names) who earns the highest salary for each job

practice_emp

empno	ename	job	sal
7369	Smith	Clerk	1200
7499	Allen	Salesman	2000
7521	Ward	Salesman	1650
7566	Jones	Manager	3375
7654	Martin	Salesman	1650
7698	Blake	Manager	3250
7782	Clark	Manager	2850
7788	Scott	Analyst	3500
7839	King	President	6500
7844	Turner	Salesman	1900
7876	Adams	Clerk	1500
7900	James	Clerk	1350
7902	Ford	Analyst	3500
7934	Miller	Clerk	1700

"Uncorrelated"

```
WITH temp AS
     (SELECT job, MAX(sal) AS maxSal
      FROM practice_emp
      GROUP BY job)
SELECT E1.ename
FROM   practice_emp AS E1, temp AS T
WHERE  E1.sal = T.maxSal AND
       E1.job = T.job
```

ename
Allen
Jones
Scott
King
Ford
Miller

A subquery in `WITH` clause returns a temporary relation that can be used by an associated query

[WITH -- not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and XAMPP MariaDB]

Subqueries and Set Operations

UNION

```
(sub-result1)
UNION
(sub-result2)
```

INTERSECT
[not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and local XAMPP 10.4.11-MariaDB]

```
(sub-result1)
INTERSECT
(sub-result2)
```

EXCEPT
[not supported by MySQL 5.6, 5.7; work on MySQL 8.0 (GCP and CS server) and local XAMPP 10.4.11-MariaDB]

```
(sub-result1)
EXCEPT
(sub-result2)
```

Requirements:

- Same number of columns
- Same order of columns
- Same column data types

We talked about `UNION` and `INTERSECT`. Let's consider `EXCEPT`

Example: Let's Solve A Problem

Use the following schema. Find IDs and names of all customers who have purchased products sold by company 7777 only. Do not list customers who have purchased from any other companies.

```
Product(pid, name, cid)
    -- cid is foreign key to Company.cid
Company(cid, cname, city)
Customer(custId, name, city)
Purchase(purchase_date, pid, custId, quantity, price)
    -- pid is foreign key to Product.pid,
    -- custId is foreign key to Customer.custId
```

Assume each customer may purchase the same product multiple times

How should we solve this problem?

Example: Let's Solve A Problem

How should we solve this problem?

1. (Find all customers who have purchased)

   EXCEPT

2. (Find all customers who have purchased from other companies, not 7777)

Use EXCEPT to Solve the Problem

1. (Find all customers who have purchased)

   EXCEPT

2. (Find all customers who have purchased from other companies, not 7777)

(sub-result1)

custId	name
991	Humpty
~~992~~	~~Dumpty~~
~~992~~	~~Dumpty~~
~~994~~	~~Minnie~~
~~995~~	~~Duh~~
~~995~~	~~Duh~~
997	Duh
~~998~~	~~Duh~~

–

(sub-result2)

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

(difference)

custId	name
991	Humpty
997	Duh

Use EXCEPT to Solve the Problem (2)

1. (Find all customers who have purchased)

Find which companies the customers have purchased.
Then, find the names of the customers

```
SELECT T1.custId, T2.cid
FROM Purchase T1 NATURAL JOIN Product T2
GROUP BY T1.custId, T2.cid
```

custId	cid
991	7777
992	7777
992	7778
994	7778
995	7777
995	7779
997	7777
998	7779

Got all customers who have purchased.

Still need to find the names of the customers

Use EXCEPT to Solve the Problem (3)

1. (Find all customers who have purchased)

Find which companies the customers have purchased
Then, find the names of the customers

```
SELECT T3.custId, T3.name
FROM (SELECT T1.custId, T2.cid
      FROM Purchase T1 NATURAL JOIN Product T2
      GROUP BY T1.custId, T2.cid) T
NATURAL JOIN Customer T3
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

Use EXCEPT to Solve the Problem (4)

2. (Find all customers who have purchased from other companies, not 7777)

Find all customers who have purchased from other companies
Then, find the names of the customers

```
SELECT T1.custId
FROM Purchase T1 NATURAL JOIN Product T2
WHERE T2.cid <> 7777
GROUP BY T1.custId
```

custId
992
994
995
998

Got all customers who have not purchased from 7777.
Still need to find the names of the customers

Use EXCEPT to Solve the Problem (5)

2. (Find all customers who have purchased from other companies, not 7777)

Find all customers who have purchased from other companies
Then, find the names of the customers

```
SELECT T3.custId, T3.name
FROM (SELECT T1.custId
      FROM Purchase T1 NATURAL JOIN Product T2
      WHERE T2.cid <> 7777
      GROUP BY T1.custId) T
NATURAL JOIN Customer T3
```

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

Use EXCEPT to Solve the Problem (6)

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId, T2.cid
       FROM Purchase T1 NATURAL JOIN Product T2
       GROUP BY T1.custId, T2.cid) T
 NATURAL JOIN Customer T3)
```

1

EXCEPT

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId
       FROM Purchase T1 NATURAL JOIN Product T2
       WHERE T2.cid <> 7777
       GROUP BY T1.custId) T
 NATURAL JOIN Customer T3)
```

2

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

−

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
997	Duh

Workaround for EXCEPT

```
SELECT result1.custId, result1.name
FROM
   (SELECT T3.custId, T3.name                                    1
    FROM (SELECT T1.custId, T2.cid
          FROM Purchase T1 NATURAL JOIN Product T2
          GROUP BY T1.custId, T2.cid) T
    NATURAL JOIN Customer T3) result1
LEFT OUTER JOIN
   (SELECT T3.custId, T3.name                                    2
    FROM (SELECT T1.custId
          FROM Purchase T1 NATURAL JOIN Product T2
          WHERE T2.cid <> 7777
          GROUP BY T1.custId) T
    NATURAL JOIN Customer T3) result2
ON      result1.custId = result2.custId
WHERE   result2.custId IS NULL
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

−

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
997	Duh

Example: UNION

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId, T2.cid
       FROM Purchase T1 NATURAL JOIN Product T2
       GROUP BY T1.custId, T2.cid) T
 NATURAL JOIN Customer T3)
```

UNION

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId
       FROM Purchase T1 NATURAL JOIN Product T2
       WHERE T2.cid <> 7777
       GROUP BY T1.custId) T
 NATURAL JOIN Customer T3)
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

U

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

Example: INTERSECT

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId, T2.cid
       FROM Purchase T1 NATURAL JOIN Product T2
       GROUP BY T1.custId, T2.cid) T
 NATURAL JOIN Customer T3)
```

INTERSECT

```
(SELECT T3.custId, T3.name
 FROM (SELECT T1.custId
       FROM Purchase T1 NATURAL JOIN Product T2
       WHERE T2.cid <> 7777
       GROUP BY T1.custId) T
 NATURAL JOIN Customer T3)
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

∩

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

Workaround for INTERSECT

```
SELECT DISTINCT result1.custId, result1.name
FROM
   (SELECT T3.custId, T3.name                                    1
    FROM (SELECT T1.custId, T2.cid
          FROM Purchase T1 NATURAL JOIN Product T2
          GROUP BY T1.custId, T2.cid) T
    NATURAL JOIN Customer T3) result1
JOIN    -- inner join
   (SELECT T3.custId, T3.name                                    2
    FROM (SELECT T1.custId
          FROM Purchase T1 NATURAL JOIN Product T2
          WHERE T2.cid <> 7777
          GROUP BY T1.custId) T
    NATURAL JOIN Customer T3) result2
ON      result1.custId = result2.custId
```

custId	name
991	Humpty
992	Dumpty
992	Dumpty
994	Minnie
995	Duh
995	Duh
997	Duh
998	Duh

∩

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

=

custId	name
992	Dumpty
994	Minnie
995	Duh
998	Duh

Wrap-Up

- Subqueries in SELECT, FROM
- Abstract immediate result using WITH
- Equivalent queries
- Intro to subqueries in WHERE
- Subqueries and set operations

Note:

- Avoid nested queries – if aiming for speed
- Be careful of semantics of nested queries
  - Correlated vs. Uncorrelated

What's next?

- Subqueries in WHERE
- Existential and universal quantifiers
- Triggers and constraints