Introduction to Data Structure (Data Management) Lecture 6

Felipe P. Vista IV

INTRO TO DATA STRUCTURE

NESTED QUERIES

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies that make **some** products with price < 200

Existential Quantifiers

Using **EXISTS**

```
SELECT DISTINCT C.cname
FROM Company C
WHERE EXISTS (SELECT *
              FROM Product P
              WHERE C.cid = P.cid AND P.price < 200)
```

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001
eBike	210.00	c003

cid	cname	city
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

*SQL EXISTS operator - test for existence of any tuple in a subquery
- returns TRUE if subquery returns one or more tuples

Subqueries in WHERE

product(pname,price,cid)

company(cid,cname,city)

Find all companies that make some products with price < 200

Existential Quantifiers

Using IN

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
                FROM Product P
                WHERE P.price < 200)
```

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001
eBike	210.00	c003

cid	cname	city
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

*SQL IN operator – allow to specify multiple values in WHERE clause
- shorthand for multiple OR conditions

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies that make **some** products with price < 200

Existential Quantifiers

Using ANY

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 200 > ANY (SELECT P.price
                 FROM Product P
                 WHERE P.cid = C.cid)
```

But "ANY" is not supported in sqlite ☹.

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001
eBike	210.00	c003

cid	cname	city
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

*SQL ANY operator – returns TRUE if any of the subquery values satisfy the condition

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies that make **some** products with price < 200

Existential Quantifiers

Unnesting…

```
SELECT DISTINCT C.cname
FROM Company C, Product P
WHERE C.cid = P.cid AND P.price < 200)
```

Existential quantifiers are easy!☺

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001
eBike	210.00	c003

cid	cname	city
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Universal Quantifiers

Find all companies where all their products has price < 200

=

Find all companies that make only products with price < 200

```
Universal quantifiers are challenging!☹
```

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - University ID Num Name (no “(   )“)
  - Ex: 202054321 Juan Dela Cruz
  -
  - Not changing your name to this format
    - you might be marked Absent

    * → absent?

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - University ID Num Name (no “(   )“)
  - Ex: 202054321 Juan Dela Cruz
  - Not changing your name to this format
    - you **will** be marked Absent

    * → Some students still do not follow instructions

    * Sirojbek, SeoMinYeong, Ravshan, Farrukhbek

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Universal Quantifiers

Find **all** companies where **all** their products has price < 200

1. Find *the other* companies with some products having price ≥ 200

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid IN (SELECT P.cid
                FROM Product P
                WHERE P.price ≥ 200)
```

pname	price	cid
bike	119.95	c003
scooter	255.00	c004
genesis	450.99	c001
eBike	210.00	c003

cid	cname	City
c003	Alton	Nabas
c001	Hyundai	Jeonju
c002	BMW	Chennai
c004	Vespa	Pontera

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies where all their products has price < 200

Universal Quantifiers

2. Find *all* companies wherein all their products are priced < 200

```
SELECT DISTINCT C.cname
FROM Company C
WHERE C.cid NOT IN (SELECT P.cid
                    FROM Product P
                    WHERE P.price ≥ 200)
```

pname	price	cid
bike	119.95	c003
scooter	255.00	c004
genesis	450.99	c001
eBike	210.00	c003

cid	cname	City
c003	Alton	Nabas
c001	Hyundai	Jeonju
c002	BMW	Chennai
c004	Vespa	Pontera

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies where all their products have a price < 200

Universal Quantifiers

Using EXISTS

```
SELECT DISTINCT C.cname
FROM Company C
WHERE NOT EXISTS (SELECT *
                  FROM Product P
                  WHERE C.cid = P.cid AND P.price < 200)
```

S
F
W (T/F) ?

F
T ×

pname	price	cid
bike	119.95	c003
scooter	255.00	c004
genesis	450.99	c001
eBike	210.00	c003

cid	cname	City
c003	Alton	Nabas
c001	Hyundai	Jeonju
c002	BMW	Chennai
c004	Vespa	Pontera

erator - test for existence of any tuple in a subquery
- returns TRUE if subquery returns one or more tuples

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies where all their products have a price < 200

Universal Quantifiers

Using ALL

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 200 ≥ ALL (SELECT P.price
                 FROM Product P
                 WHERE P.cid = C.cid)
```

But “ALL” is also not supported in sqlite ☹.

pname	price	cid
bike	119.95	c003
scooter	255.00	c004
genesis	450.99	c001
eBike	210.00	c003

cid	cname	City
c003	Alton	Nabas
c001	Hyundai	Jeonju
c002	BMW	Chennai
c004	Vespa	Pontera

tor – returns TRUE if all of the subquery values satisfy the condition

Subqueries in WHERE

```
product(pname,price,cid)
company(cid,cname,city)
```

Find all companies where **all** their products have a price < 200

Universal Quantifiers

Using **ALL**

```
SELECT DISTINCT C.cname
FROM Company C
WHERE 200 ≥ ALL (SELECT P.price
                 FROM Product P
                 WHERE P.cid = C.cid)
```

WHERE 200 ≥ (SELECT * from C

But "ALL" is also not supported in sqlite ☹.

Can we unnest a universal quantifier query?

--- **NO**

Monotone Queries

product(pname,price,cid)

company(cid,cname,city)

A query is **monotone** if :

- When a tuple(record) is added to one or more tables, the result of the query will not lose any of previous tuple results

Product

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001

Company

cid	cname	City
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

P.cID = c.cID

?

pname	city
?????	?????

Monotone Queries

product(pname,price,cid)

company(cid,cname,city)

A query is **monotone** if :

- When a tuple(record) is added to one or more tables, the result of the query will not lose any of previous tuple results

Product

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001

Company

cid	cname	City
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

pname	city
bike	Nabas
genesis	Jeonju

Monotone Queries

product(pname,price,cid)

company(cid,cname,city)

A query is monotone if :

- When a tuple(record) is added to one or more tables, the result of the query will not lose any of previous tuple results

Product

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001

Company

cid	cname	City
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

pname	city
bike	Nabas
genesis	Jeonju

Product

pname	price	cid
bike	119.95	C003 ✗
scooter	255.00	c004
genesis	450.99	C001 \
→ eBike	210.00	c003 ✗ ✗

Company

cid	cname	City
C003 ✗	Alton	Nabas
c001 \	Hyundai	Jeonju
C002	BMW	Chennai

pname	city
?????	?????

Monotone Queries

```
product(pname,price,cid)

company(cid,cname,city)
```

A query is monotone if :

- When a tuple(record) is added to one or more tables, the result of the query will not lose any of previous tuple results

Product

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001

Company

cid	cname	City
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

→

pname	city
bike	Nabas
genesis	Jeonju

Product

pname	price	cid
bike	119.95	C003
scooter	255.00	c004
genesis	450.99	C001
eBike	210.00	c003

Company

cid	cname	City
C003	Alton	Nabas
c001	Hyundai	Jeonju
C002	BMW	Chennai

→

pname	city
~~bike~~	~~Nabas~~
genesis	Jeonju
~~ebike~~	~~Nabas~~

Monotone Queries

Theorem:

If Q is a SELECT-FROM-WHERE query that do not have subqueries, and no aggregates, then it is MONOTONE

sum, count, Ave

Monotone Queries

Theorem:

If Q is a SELECT-FROM-WHERE query that do not have subqueries, and no aggregates, then it is MONOTONE

Proof:

We use the nested loop semantics: If we insert a tuple in a relation R_i, this will not remove any tuples from the answer

Monotone Queries

Theorem:

If Q is a SELECT-FROM-WHERE query that do not have subqueries, and no aggregates, then it is MONOTONE

Proof:

We use the nested loop semantics: If we insert a tuple in a relation R_i, this will not remove any tuples from the answer

```
SELECT a1,a2,..,ak
FROM R1 AS x1,…,Rn AS xn
WHERE conditions
```

```
for x1 in R1: do
   for x2 in R2: do
      …
         for xn in Rn: do
            if conditions
              output(a1,a2,…,ak)
```

Monotone Queries

`product(pname,price,cid)`

`company(cid,cname,city)`

* The following query:

Find all companies where all their products have a price < 200

- is not a monotone

Product

pname	price	cid
bike	119.95	C003

Company

cid	cname	City
C003	Alton	Nabas

cname
?????

Monotone Queries

product(pname,price,cid)

company(cid,cname,city)

* The following query:

Find all companies where all their products have a price < 200

- is not a monotone

Product

pname	price	cid
bike	119.95	C003

Company

cid	cname	City
C003	Alton	Nabas

cname
Alton

Monotone Queries

```
product(pname,price,cid)
company(cid,cname,city)
```

* The following query:

Find all companies where **all** their products have a price < 200

- is not a monotone

Product

pname	price	cid
bike	119.95	C003

Company

cid	cname	City
C003	Alton	Nabas

cname
Alton

Product

pname	price	cid
bike	119.95	C003
eBike	210.00	c003

Company

cid	cname	City
C003	Alton	Nabas

cname
?????

Monotone Queries

```
product(pname,price,cid)
company(cid,cname,city)
```

* The following query:

Find all companies where **all** their products have a price < 200

- is not a monotone

Product

pname	price	cid
bike	119.95	C003

Company

cid	cname	City
C003	Alton	Nabas

cname
Alton

Product

pname	price	cid
bike	119.95	C003
eBike	210.00	c003

Company

cid	cname	City
C003	Alton	Nabas

cname

Monotone Queries

- SFW Q × sQ
× Agg

\ × sQ
× Agg

```
product(pname,price,cid)
company(cid,cname,city)
```

* The following query:

Find all companies where all their products have a price < 200

- is not a monotone

Product

pname	price	cid
bike	119.95	C003

Company

cid	cname	City
C003	Alton	Nabas

→

cname
Alton

Product

pname	price	cid
bike	119.95	C003
eBike	210.00	c003

Company

cid	cname	City
C003	Alton	Nabas

→

cname

->
—

* **Effect**: This query cannot be written as SELECT-FROM-WHERE without using nested subqueries

When to Use Nested Queries

- That is, they cannot be SFW queries

When to Use Nested Queries

- That is, they cannot be SFW queries
- Queries with universal quantifiers or negation

Negation of Quantifiers

When to Use Nested Queries

- That is, they cannot be SFW queries
- Queries with universal quantifiers or negation
- Queries that use aggregates in usual ways are not monotone = [illegible]
  - `sum(…)`, etc… are NOT monotone
  - `SELECT count(*) FROM R` is NOT monotone

Negation of Quantifiers

Thank you.

If-Then

Existential Quantification

- Statements that are examples of existential quantification:
  - There are black swans
  - There is a way to get a change of 12 ewans with 4 ewan & 5 ewan coins
  - There exists such integers a, b, c, and d that $a^4 + b^4 + c^4 = d^4$
  - There exists a power of 2 starting with 65

<<BACK

If-Then

Universal Quantification

- Statements that are examples of universal quantification:
  - All swans are white
  - All integers ending with the digit “2” are even
  - For all n, $2 \times n = n + n$

- Fermat’s Last Theorem states that for all $n > 2$, equation $a^n + b^n = c^n$ does not have solutions with positive integers a, b, & c.
  - This is another example universal negation

<<BACK

If-Then

Negation of Quantifications

- Negation of universal quantification is a corresponding existential quantification
- Negation of existential quantification is a corresponding universal quantification

- Example:
  - UQ: "For **all** n, statement A is **true**"
  - Negation of UQ: "There exists **such** n that statement A is **false**"

<<BACK NEXT>>

If-Then

Negation of Quantifications

- Euler's hypothesis is a combination of two universal quantifications:

  ∀ ∀ ∀ ∀

  – For **any** $n > 3$, for **any** positive integer a, it is **impossible to represent** a^n as a sum of $n - 1$ numbers which are the n-th powers of positive integers.

- Negation:

  ∃ ∃

  – There exists **such** $n > 3$ and **such** positive integer a that a^n **can be represented** as a sum of $n - 1$ numbers which are the n-th powers of positive integers.

<<BACK NEXT>>

If-Then

Neg AND → N(x) OR N(x)

Negation of Quantifications

N(x) AND N(y)

- UQ: “**All** positive integers are either even **OR** odd”
- Negation: “There exists **such** positive integer *n* that is not even **AND** not odd”

E

- To negate:
  - We switch universal quantification (UQ) to existential qualification (EQ) and switch OR to AND
  - We switch existential quantification (EQ) to universal qualification (UQ) and switch AND to OR

<<BACK

NEXT>>

If-Then

Negation of Quantifications

- UQ: "All positive integers are either even OR odd"
- Negation: "There exists such positive integer n that is not even AND not odd"

- To negate:
  - We switch universal quantification (UQ) to existential qualification (EQ) and switch OR to AND
  - We switch existential quantification (EQ) to universal qualification (UQ) and switch AND to OR

<<BACK