Updated 1/11/2020

T1: Data models and query languages
L1: SQL (a refresher)

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

https://northeastern-datalab.github.io/cs7240/sp20/

1/7/2020

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Structured Query Language: SQL

- Influenced by relational calculus (= First Order Logic)
- SQL is a declarative query language
  - We say what we want to get
  - We don’t say how we should get it

Simple SQL Query

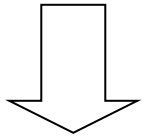
302

Our friend here shows that you can follow along in SQLite. Just install the database from the text file "300 - ..." available in our sql folder

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  pName, price, manufacturer
FROM    Product
WHERE   price > 100
```



PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Selection
& Projection

Selection vs. Projection

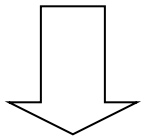
Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

One **projects** onto some attributes (columns)
-> happens in the **SELECT** clause

One **selects** certain entires=tuples (rows)
-> happens in the **WHERE** clause
-> acts like a **filter**

```
SELECT  pName, price
FROM    Product
WHERE   price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Eliminating Duplicates

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
PowerGizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Set vs. Bag semantics

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

```
SELECT DISTINCT category
FROM   Product
```

Category
Gadgets
Photography
Household

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Keys and Foreign Keys

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is a foreign key vs. a key here?

Keys and Foreign Keys

Key

Foreign key

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Key

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is a foreign key vs. a key here?

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

Foreign key: must match field in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

SuperTouch	$249.99	Computer	NewCom

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

violate Foreign Key constraint

SuperTouch	$249.99	Computer	NewCom

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

violate Foreign Key constraint

SuperTouch	$249.99	Computer	NewCom

```
Delete from Company
where CName = 'Canon';
```

Referential Integrity

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key constraint: minimal subset of the fields of a relation is a unique identifier for a tuple.

```
Insert into Product values ('Gizmo', 14.99, 'Gadgets', 'Hitachi');
```

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: must match field in a relational table that matches a candidate key of another table

```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

violate Foreign Key constraint

SuperTouch	$249.99	Computer	NewCom

```
Delete from Company
where CName = 'Canon';
```

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Joins

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

Joins

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Q: Find all products under $200 manufactured in Japan; return their names and prices!

```
SELECT   pName, price
FROM     Product, Company
WHERE    manufacturer=cName
  and    country='Japan'
  and    price <= 200
```

Join b/w Product and Company

PName	Price
SingleTouch	$149.99

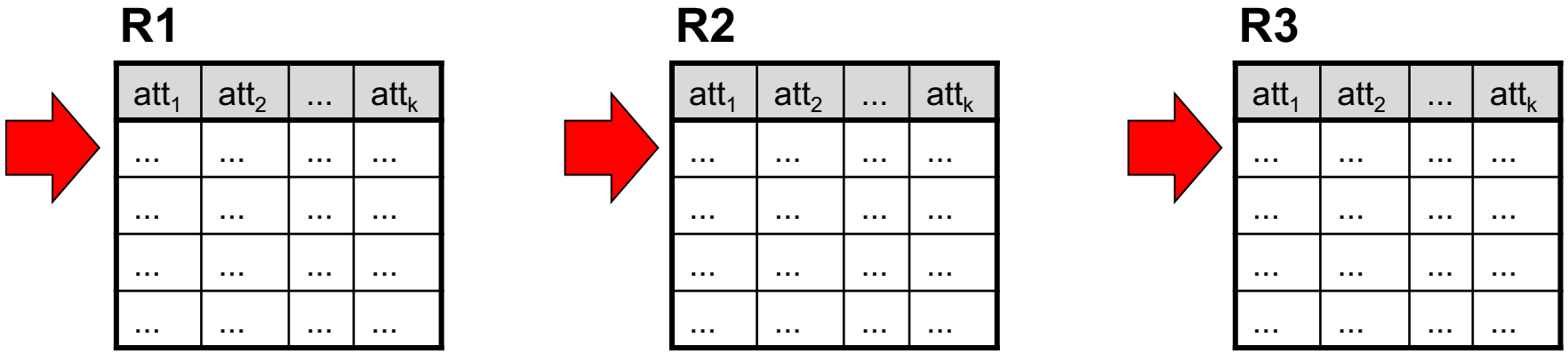
Meaning (Semantics) of conjunctive SQL Queries

SELECT $a_1, a_2, \ldots, a_k$
FROM R_1 as x_1, R_2 as x_2, …, R_n as x_n
WHERE Conditions

Conceptual evaluation strategy (nested for loops):

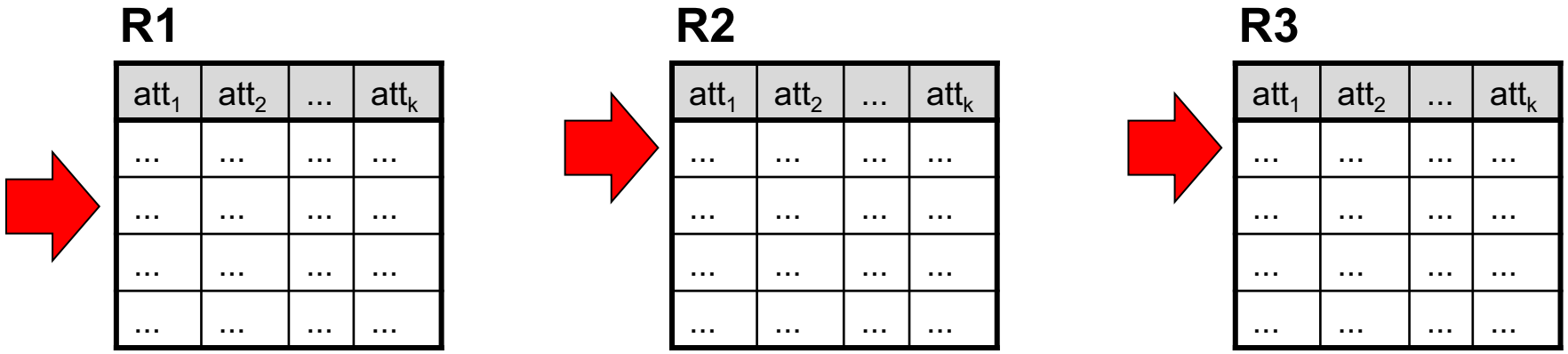
```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        …..
            for xn in Rn do
                if Conditions
                    then Answer = Answer ∪ {(a1,…,ak)}
return Answer
```

Meaning (Semantics) of conjunctive SQL Queries



Answer = {}
for x_1 **in** R_1 **do**
  for x_2 **in** R_2 **do**
    …..
      for x_n **in** R_n **do**
        if Conditions
          then Answer = Answer $\cup$ $\{(a_1,\ldots,a_k)\}$
return Answer

Meaning (Semantics) of conjunctive SQL Queries



Answer = {}
for x_1 **in** R_1 **do**
  for x_2 **in** R_2 **do**
    …..
      for x_n **in** R_n **do**
        if Conditions
          then Answer = Answer $\cup$ $\{(a_1,\ldots,a_k)\}$
return Answer

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - FROM: Compute the cross-product of relation-list.
  - WHERE: Discard resulting tuples if they fail qualifications.
  - SELECT: Delete attributes that are not in target-list.
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute the same answers.

Note: we say “semantics” not “execution order”

- The preceding slides show what a join means
- Not actually how the DBMS executes it under the covers

Data independence

- Logical data independence:
  - specify a set of attributes, not the logical navigation path to compute the connection among them
- Physical data independence:
  - specify a query, not the physical access paths to compute it

Quiz

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM
WHERE
```

Quiz

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

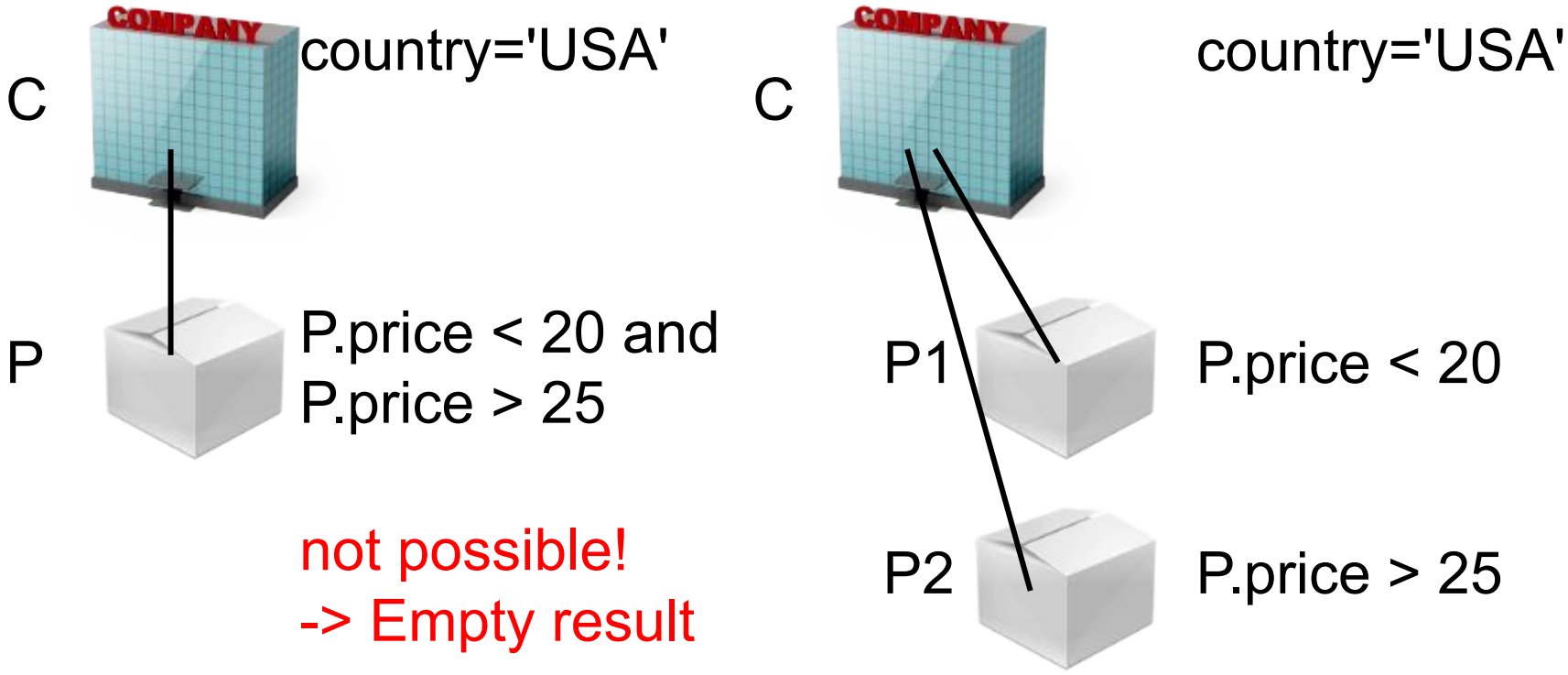
```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and P.price < 20
    and P.price > 25
    and P.manufacturer = cName
```

Wrong! Gives empty result: There is no product with price <20 and >25

Quiz

Product (<u>pName</u>, price, category, manufacturer)
Company (<u>cName</u>, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.



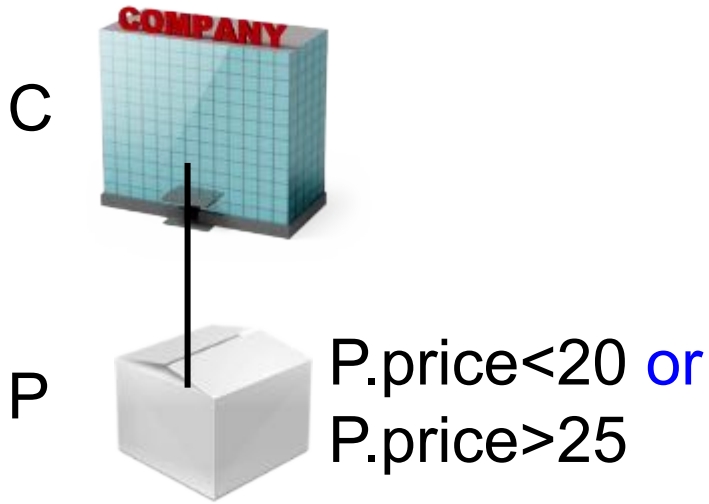
Quiz

Product (pName, price, category, manufacturer)
Company (cName, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P, Company
WHERE   country = 'USA'
    and (P.price < 20
    or   P.price > 25)
    and  P.manufacturer = cName
```

Returns companies with single product w/price (<20 or >25)



Quiz

Product (<u>pName</u>, price, category, manufacturer)
Company (<u>cName</u>, stockPrice, country)

Q: Find all US companies that manufacture both a product below $20 and a product above $25.

```
SELECT  DISTINCT cName
FROM    Product as P1, Product as P2, Company
WHERE   country = 'USA'
    and P1.price < 20
    and P2.price > 25
    and P1.manufacturer = cName
    and P2.manufacturer = cName
```

Quiz

P1

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
...	...	...	...

P2

PName	Price	Category	Manufacturer
...	...	...	...
Powergizmo	$29.99	Gadgets	GizmoWorks

Company

CName	StockPrice	Country
GizmoWorks	25	USA
...	...	...

```
SELECT  DISTINCT cName
FROM    Product as P1, Product as P2, Company
WHERE   country = 'USA'
    and P1.price < 20
    and P2.price > 25
    and P1.manufacturer = cName
    and P2.manufacturer = cName
```

Cname
GizmoWorks

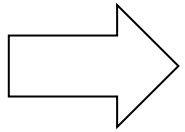
Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Grouping and Aggregation

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
~~Banana~~	~~1~~	~~50~~
Banana	2	10
Banana	4	10



Product	TotalSales
Bagel	40
Banana	20

Notice: we use "sales" for total number of products sold

Find total quantities for all purchases with price over $1 grouped by product.

From → Where → Group By → Select

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
~~Banana~~	~~1~~	~~50~~
Banana	2	10
Banana	4	10

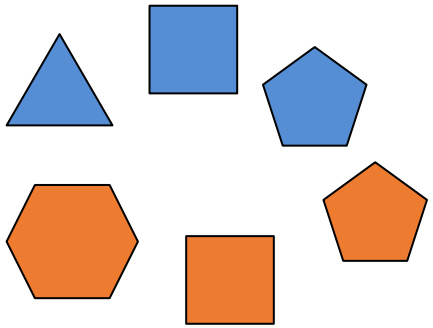
Product	TotalSales
Bagel	40
Banana	20

Select contains
- grouped attributes
- and aggregates

```
4  SELECT    product, sum(quantity) as TotalSales
1  FROM      Purchase
2  WHERE     price > 1
3  GROUP BY  product
```

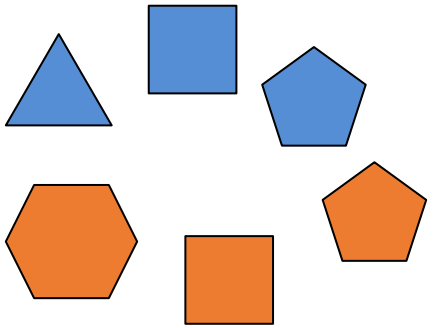
Groupings illustrated with colored shapes

group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

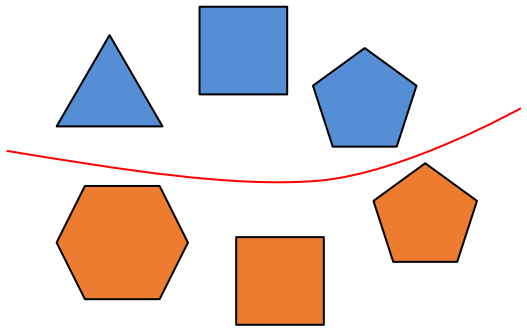
group by numc (# of corners)



```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

Groupings illustrated with colored shapes

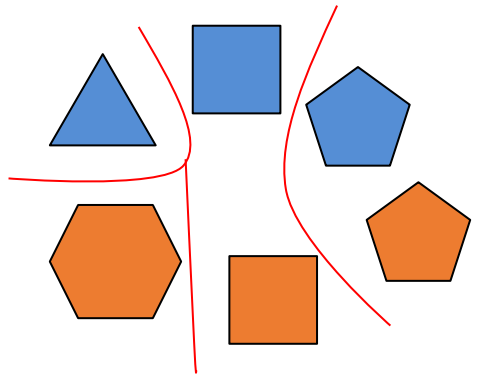
group by color



```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

color	anc
blue	4
orange	5

group by numc (# of corners)

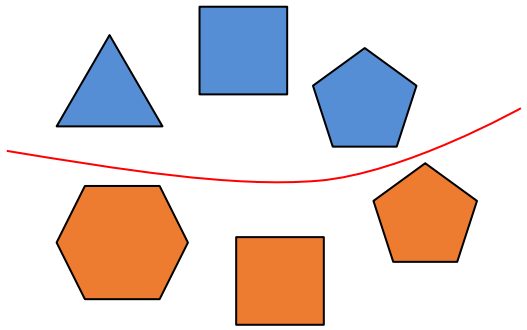


```
SELECT  numc
FROM    Shapes
GROUP BY numc
```

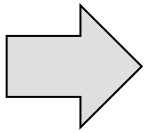
numc
3
4
5
6

Groupings illustrated with colored shapes

group by color

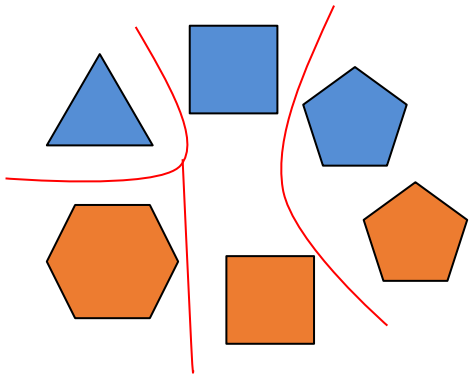


```
SELECT  color,
        avg(numc) anc
FROM    Shapes
GROUP BY color
```

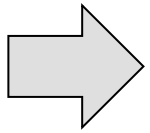


color	anc
blue	4
orange	5

group by numc (# of corners)



```
SELECT  numc
FROM    Shapes
GROUP BY numc
```



numc
3
4
5
6

Same as:

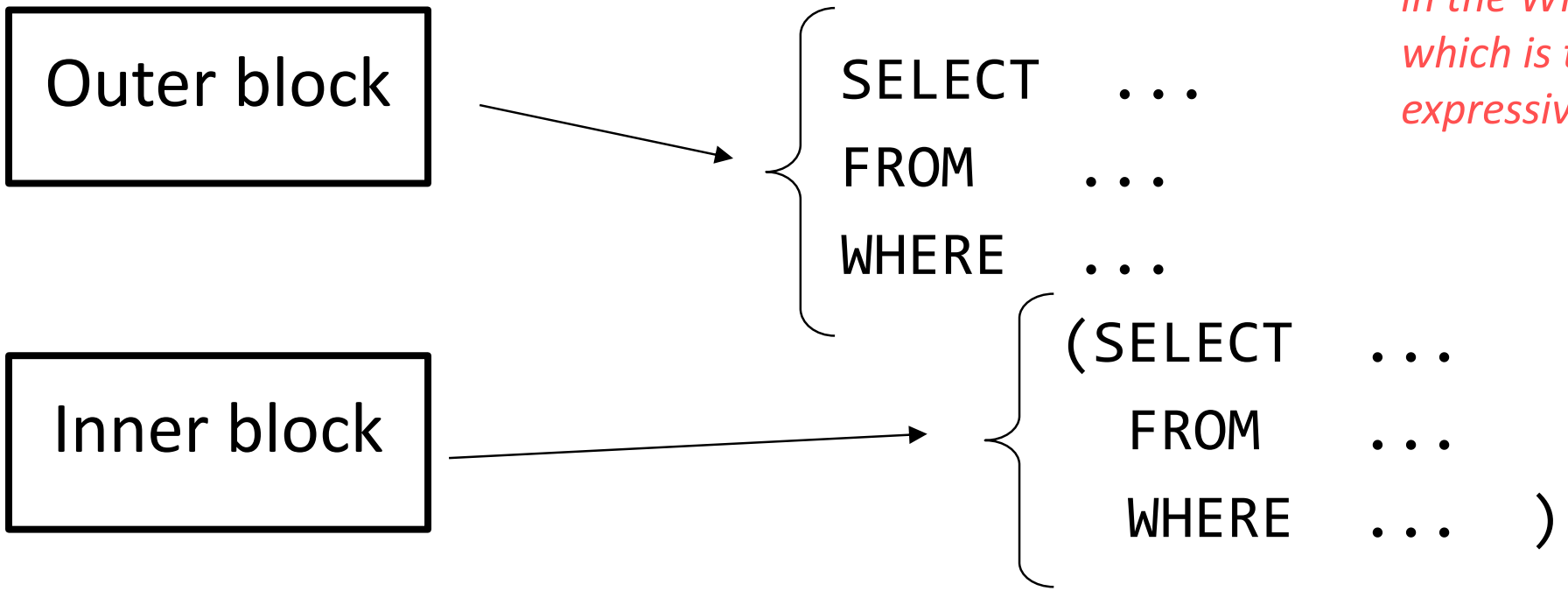
```
SELECT  DISTINCT numc
FROM    Shapes
```

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Subqueries = Nested queries

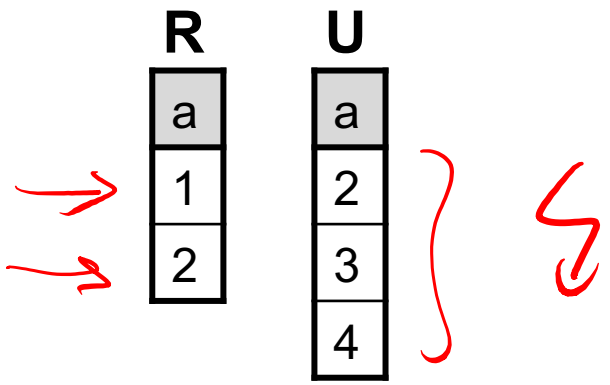
We only focus on nestings in the WHERE clause, which is the most expressive type of nesting



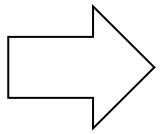
- We can nest queries because SQL is compositional:
  - Everything (inputs / outputs) is represented as multisets
  - the output of one query can thus be used as the input to another (nesting)
  - Subqueries return relations
- This is extremely powerful!

Subqueries in WHERE

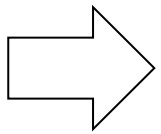
What do these queries compute?



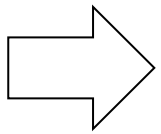

```
SELECT  a
FROM    R
WHERE   a IN
        (SELECT * from U)
```

 ?

```
SELECT  a
FROM    R
WHERE   a < ANY
        (SELECT * from U)
```

 ?

```
SELECT  a
FROM    R
WHERE   a < ALL
        (SELECT * from U)
```

 ?

Subqueries in WHERE

What do these queries compute?

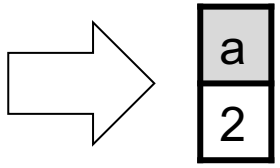
R

a
1
2

U

a
2
3
4

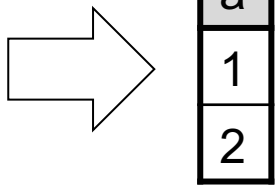
```
SELECT  a
FROM    R
WHERE   a IN
        (SELECT * from U)
```



a
2

Since 2 is in the set (bag) (2, 3, 4)

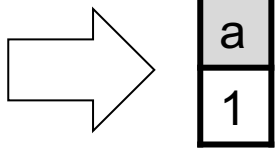
```
SELECT  a
FROM    R
WHERE   a < ANY
        (SELECT a from U)
```



a
1
2

Since 1 and 2 are < than at least one ("any") of 2, 3 or 4

```
SELECT  a
FROM    R
WHERE   a < ALL
        (SELECT * from U)
```



a
1

Since 1 is < than each ("all") of 2, 3, and 4