Updated 1/21/2022

Topic 1: Data models and query languages
Unit 1: SQL
Lecture 1

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

1/18/2022

Topic 1: Data Models and Query Languages

PRELIMINARY

- **Lecture 1 (Tue 1/18):** Introduction, SQL, PostgreSQL setup, SQL Activities
- **Lecture 2 (Fri 1/21):** SQL
- **Lecture 3 (Tue 1/25):** SQL, Logic & Relational Calculus
- **Lecture 4 (Fri 1/28):** Logic & Relational Calculus
- **Lecture 5 (Tue 2/1):** Relational Algebra & Codd's Theorem
- **Lecture 6 (Fri 2/4):** Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 2/8):** Datalog & Recursion,
- **Lecture 8 (Fri 2/11):** Datalog & Recursion,
- **Lecture 9 (Tue 2/15):** Alternative Data Models

6240

Pointers to relevant concepts & supplementary material:

- **Unit 1. SQL**: [SAMS'12], [CS3200'18] [Cow'03] Ch3 & Ch5, [Complete'08] Ch6, [Silberschatz+'20] Ch3.8
- **Unit 2. Logic & Relational Calculus**: First-Order Logic (FOL), relational calculus (RC): [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'20] Ch27.1 & Ch27.2, [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4, [Barker-Plummer+'11] Ch11
- **Unit 3. Relational Algebra & Codd's Theorem**: Relational Algebra (RA), Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'20] Ch2.6, [Alice'95] Ch4.4 & Ch5.4
- **Unit 4. Datalog & Recursion**: Datalog, recursion, Stratified Datalog with negation, Datalog evaluation strategies, stable model semantics: [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [G., Suciu'10]
- **Unit 5. Alternative Data Models**: NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Outline: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k

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 ("separation of concerns")

Simple SQL Query

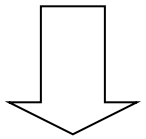
Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

302

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
FROM    Product
WHERE   price > 100
```



?

Simple SQL Query

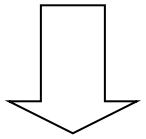
Our friend here shows that you can follow along in Postgres.
Just install the database from the text file "302 - ..."
available in our sql folder from our course web page

302

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
FROM   Product
WHERE  price > 100
```



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

Selection
& Projection

Selection vs. Projection

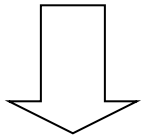
Product

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

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

Where does the selection happen?

?



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

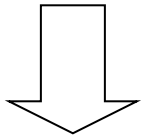
Selection & Projection

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

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



PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

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

Selection vs. Projection

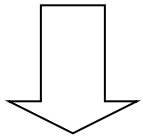
Product

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

```
SELECT category
FROM   Product
```

⇨ ?

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

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

Set vs. Bag semantics

Think of a dictionary: keys mapping to # of occurences

Gadgets : 2
Photography : 1
Houshold : 1

?

Category
Gadgets
Photography
Household

underlying set also called the "support" of the bag

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

```
SELECT category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

Set vs. Bag semantics

Think of a dictionary: keys mapping to # of occurences

Gadgets : 2
Photography : 1
Houshold : 1

```
SELECT DISTINCT category
FROM   Product
```

Category
Gadgets
Photography
Household

underlying set also called the "support" of the bag

Outline: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k

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 here a key vs. a foreign key?

Keys and Foreign Keys

Key

Product

Foreign key

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

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 attributes of a relation is a unique identifier for a tuple.

Foreign key: attribute 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 attributes 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: attribute 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 attributes of a relation is a unique identifier for a tuple.

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

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute 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 attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

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

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute 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 attributes of a relation is a unique identifier for a tuple.

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

tuple violates key constraint

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

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

tuple violates 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 attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

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

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

tuple violates foreign key constraint

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

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 attributes of a relation is a unique identifier for a tuple.

tuple violates key constraint

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

Gizmo	$14.99	Gadgets	Hitachi

Foreign key: attribute in a relational table that matches a candidate key of another table

tuple violates foreign key constraint

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

SuperTouch	$249.99	Computer	NewCom

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

Outline: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k

Joins

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
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

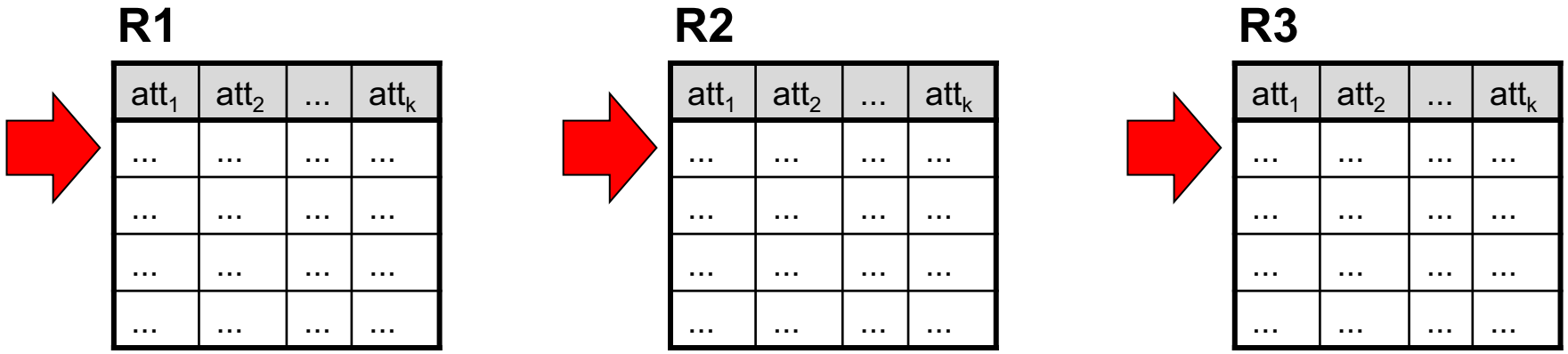
Meaning (Semantics) of conjunctive SQL Queries

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):

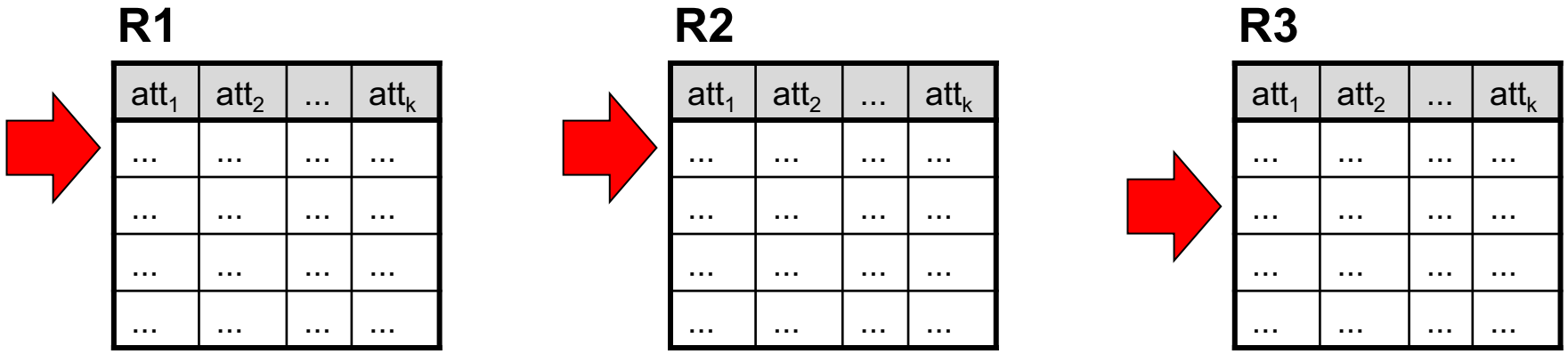
```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        .....
            for xn in Rn do
                if Conditions
                    then Answer = Answer ∪ {(a1,…,ak)}
return Answer
```

Meaning (Semantics) of conjunctive SQL Queries




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

- Quiz: we say “semantics” not “execution order”. Why?

?

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" the rest)
  - 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.

- Quiz: we say “semantics” not “execution order”. Why?
  - The preceding slides show what a join means (semantics = meaning): "the logic"
  - Not actually how the DBMS actually executes it (separation of concerns): algebra