SQL

CASA0025: Building Spatial Applications with Big Data

Ollie Ballinger

Outline

1. What is a Database?
2. What is SQL?
3. Single-table queries
4. Multi-table queries

First, an exercise

- You are the head of the UK's National Health Service (NHS).
- There are 1000 COVID testing centers, each keeping track of positive and negative results.
- You need to collect data from these centers, and share them with other government departments

1. What program do you use?
2. What file format do you use?
3. How do you collect data from test centers, and how do you share them to other parts of the government?

Search US edition
The Guardian
For 200 years

Covid: how Excel may have caused loss of 16,000 test results in England

A million-row limit on Microsoft's Excel spreadsheet software may have led to Public Health England misplacing nearly 16,000 Covid test results, it is understood.

The data error, which led to 15,841 positive tests being left off the official daily figures, means than 50,000 potentially infectious people may have been missed by contact tracers and not told to self-isolate.

In this case, the Guardian understands, one lab had sent its daily test report to PHE in the form of a CSV file - the simplest possible database format, just a list of values separated by commas. That report was then loaded into Microsoft Excel, and the new tests at the bottom were added to the main database.

1. What is a Database

What is a database?

1. A SQL database or relational database is a collection of highly structured tables, wherein each row reflects a data entity, and every column defines a specific information field.

2. Relational databases are built using the structured query language (SQL) to create, store, update, and retrieve data.

3. Therefore, SQL is the underlying programming language for all relational database management systems (RDBMS) such as MySQL, Oracle, and Sybase, among others.

Three key advantages of Databases

1. **Access**
   - Many people can connect to the same database
     - You can set privileges so that some people are read only, and others can edit.
   - Since everyone's working on the same version, your data is always up to date
2. **Structure**
   - You can store data in a relational way, avoiding unnecessary repetition
   - You can store lots of data
3. **Efficient Querying**
   - Syntax is very simple and human-readable
   - SQL is a 4th generation programming language
     - Rather than telling it what to *do*, you tell it what you *want.* It then figures out the best way of acquiring that information.

When do you need a Database?

- Multiple simultaneous changes to data (concurrency)
- Data changes on a regular basis
- Large data sets where you only need some observations/variables
- Share huge data set among many people
- Rapid queries with no analysis
- **Web interfaces to data, especially dynamic data**

Different Database Management Systems

- MySQL
  - The most common database system on the web
- MSSQL / Oracle
  - Most common enterprise DB system
- NoSQL Severs: Big Query, MongoDB
  - In Memory Databases that respond quicker than traditional databases
- **Postgres**
  - With PostGIS Extensions for spatial data
- **DuckDB**
  - Serverless OLAP (on-line analytical processing)

Imagine a library

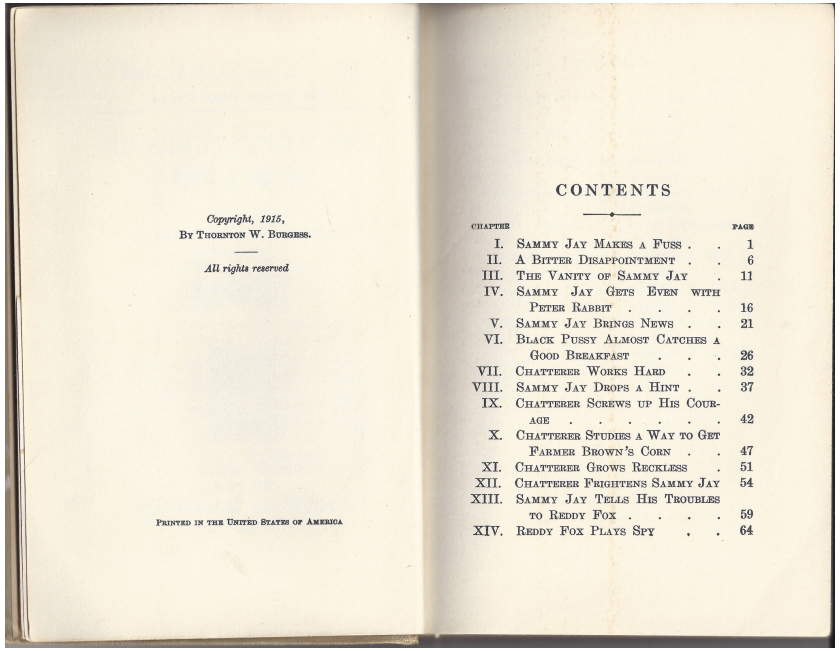
Librarian

DBMS

Shelves

Databases

Books



Copyright, 1915,
By Thornton W. Burgess.

All rights reserved

Printed in the United States of America

CONTENTS

Tables

Finding the Library

- Addressing System
  - IP or Domain Name -> Address to the Library
- Port Numbers
  - Large Apartment Block => Got the find the right door
- A database is software running on a computer or server

- CASA has a database that lives at 128.40.150.34:3306
  - bart150-34.bartlett.ucl.ac.uk:3306
  - dev.spatialdatacapture.org:3306

2. What is SQL

SQL Motivation

- But why use SQL?
  - The relational model of data is the most widely used model today
    - Main Concept: the *relation*- essentially, a table

Remember: The reason for using the relational model is data independence!

Logical data independence:
protection from changes in the *logical structure of the data*

SQL is a logical, declarative query language. We use SQL because we happen to use the relational model.

SQL Introduction

- SQL is a standard language for querying and manipulating data
- SQL is a **very high-level** programming language
  - This works because it is optimized well!
- Many standards out there:
  - ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….
  - Vendors support various subsets

SQL stands for
Structured Query Language

Probably the world's most successful **parallel** programming language (multicore?)

SQL is a

- Data Definition Language (DDL)
  - Define relational *schemata*
  - Create/alter/delete tables and their attributes

- Data Manipulation Language (DML)
  - Insert/delete/modify tuples in tables
  - Query one or more tables – discussed next!

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema

Let's break this definition down

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A multiset is an unordered list (or: a set with multiple duplicate instances allowed)

List: [1, 1, 2, 3]
Set: {1, 2, 3}
Multiset: {1, 1, 2, 3}

i.e. no *next()*, etc. methods!

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An attribute (or column) is a typed data entry present in each tuple in the relation

*Attributes must have an **atomic** type in standard SQL, i.e. not a list, set, etc.*

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

*Also referred to sometimes as a **record***

A tuple or row is a single entry in the table having the attributes specified by the schema

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

Data Types in SQL

- Atomic types:
  - Characters: CHAR(20), VARCHAR(50)
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATETIME, …

- Every attribute must have an atomic type
  - Hence tables are flat

Table Schemas

- The **schema** of a table is the table name, its attributes, and their types:

```
Product(Pname: string, Price: float, Category:
string, Manufacturer: string)
```

- A **key** is an attribute whose values are unique; we underline a key

```
Product(Pname: string, Price: float, Category:
string, Manufacturer: string)
```

Key constraints

A key is a minimal subset of attributes that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
  - i.e. if two tuples agree on the values of the key, then they must be the same tuple!

```
Students(sid:string, name:string, gpa: float)
```

1. Which would you select as a key?
2. Is a key always guaranteed to exist?
3. Can we have more than one key?

NULL and NOT NULL

- To say "don't know the value" we use NULL
  - NULL has (sometimes painful) semantics, more details later

```
Students(sid:string, name:string, gpa: float)
```

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be NOT NULL, e.g., "name" in this table

Summary of Schema Information

- Schema and Constraints are how databases understand the semantics (meaning) of data
- They are also useful for optimization
- SQL supports general constraints:
  - Keys and foreign keys are most important
  - We'll give you a chance to write the others

2. Single-table queries

What you will learn about in this section

1. The SFW query
2. Other useful operators: LIKE, DISTINCT, ORDER BY

AND OR ALTER TABLE AS BETWEEN CREATE DATABASE CREATE TABLE CREATE INDEX CREATE VIEW DELETE DROP DATABASE DROP INDEX DROP TABLE EXISTS GROUP BY HAVING IN INSERT INTO INNER JOIN LEFT JOIN RIGHT JOIN FULL JOIN LIKE ORDER BY SELECT SELECT * SELECT DISTINCT SELECT INTO SELECT TOP TRUNCATE TABLE UNION UNION ALL UPDATE WHERE

Finding your way around

- Since a single server can support many databases, each containing many tables, with each table having a variety of columns, it's easy to get lost when you're working with databases.
- These commands will help figure out what's available:
  - SHOW DATABASES;
  - SHOW TABLES IN database;
  - SHOW COLUMNS IN table;
  - DESCRIBE table;
    - shows the columns and their types

SQL Query

- Basic form (there are many many more bells and whistles)

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

Call this a SFW query.

Simple SQL Query: Selection

Selection is the operation of filtering a relation's tuples on some condition

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 *
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

Simple SQL Query: Projection

Projection is the operation of producing an output table with tuples that have a subset of their prior attributes

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  Category = 'Gadgets'
```

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks

Notation

Input schema

Product(PName, Price, Category, Manfacturer)

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets'
```

Output schema

Answer(PName, Price, Manfacturer)

A Few Details

- SQL **commands** are case insensitive:
  - Same: SELECT, Select, select
  - Same: Product, product
- **Values** are **not:**
  - Different: 'Seattle', 'seattle'
- Use single quotes for constants:
  - 'abc' - yes
  - "abc" - no

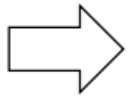
LIKE: Simple String Pattern Matching

```
SELECT *
FROM   Products
WHERE  PName LIKE '%gizmo%'
```

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = any sequence of characters
  - _ = any single character

DISTINCT: Eliminating Duplicates

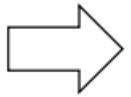
```
SELECT DISTINCT Category
FROM    Product
```



Category
Gadgets
Photography
Household

Versus

```
SELECT Category
FROM   Product
```



Category
Gadgets
Gadgets
Photography
Household

ORDER BY: Sorting the Results

```
SELECT    PName, Price, Manufacturer
FROM      Product
WHERE     Category='gizmo' AND Price > 50
ORDER BY  Price, PName
```

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

SQL Operators and Functions

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Equals
!= or <>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Operator	Description
AND	Combines multiple conditions
OR	Combines multiple conditions
BETWEEN	Only values between
EXISTS	Searches for presence of row meeting criteria
IN	Finds values within provided list
LIKE	Finds values similar to provided value
NOT	Negates other operators (e.g. NOT BETWEEN, NOT IN)
IS NULL	Returns if value is null

Function	Description
AVG(*col*)	Returns average of all values in column
COUNT(*col*)	Returns count of all rows in column
FIRST(*col*) *and* LAST(*col*)	Returns first or last value from all values in column
MIN(*col*) *and* MAX(*col*)	Returns maximum or minimum value from all values in column
SUM(*col*)	Returns sum of all values in range
LEN(*col*)	Returns length of a text value
ROUND(*col*, *val*)	Rounds numeric value to given number of decimals (*val*)
UCASE(*col*)	Coverts text field to upper case
LCASE(*col*)	Coverts text field to lower case

3. Multi-table queries

Foreign Key constraints

- Suppose we have the following schema:

```
Students(sid: string, name: string, gpa: float)
Enrolled(student_id: string, cid: string, grade: string)
```

- And we want to impose the following constraint:
  - 'Only bona fide students may enroll in courses' i.e. a student must appear in the Students table to enroll in a class

Students

sid	name	gpa
101	Bob	3.2
123	Mary	3.8

Enrolled

student_id	cid	grade
123	564	A
123	537	A+

student_id alone is not a key- what is?

We say that student_id is a foreign key that refers to Students

Declaring Foreign Keys

```
Students(sid: string, name: string, gpa: float)
Enrolled(student_id: string, cid: string, grade: string)

CREATE TABLE Enrolled(
      student_id CHAR(20),
      cid        CHAR(20),
      grade      CHAR(10),
      PRIMARY KEY (student_id, cid),
      FOREIGN KEY (student_id) REFERENCES Students(sid)
)
```

Keys and Foreign Keys

Company

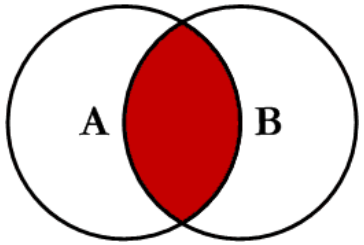
CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is a foreign key vs. a key here?

Product

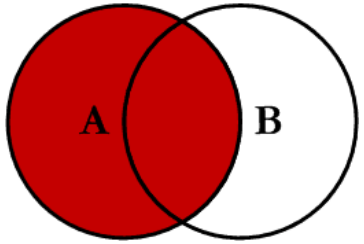
PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Joins



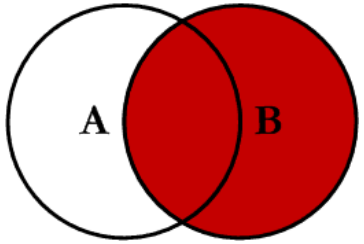
Inner JOIN

```
SELECT <select_list>
FROM Table_A A
INNER JOIN Table_B B
ON A.Key = B.Key
```



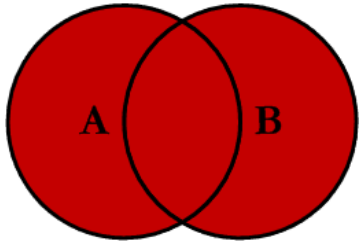
Left JOIN

```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
```



Right JOIN

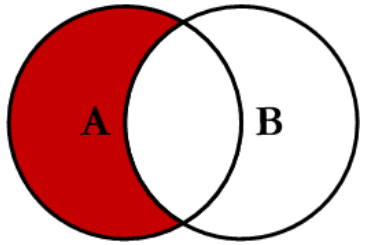
```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
```



Outer JOIN

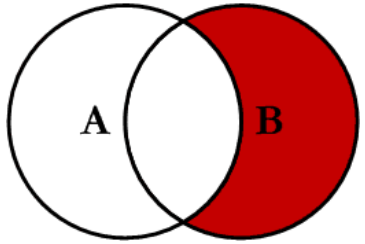
Not available in MySQL

```
SELECT <select_list>
FROM Table_A A
FULL OUTER JOIN Table_B B
ON A.Key = B.Key
```



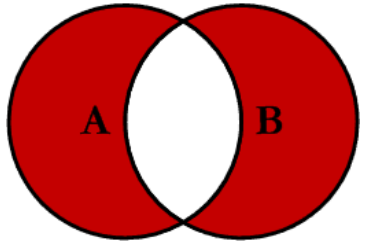
Left excluding JOIN

```
SELECT <select_list>
FROM Table_A A
LEFT JOIN Table_B B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



Right excluding JOIN

```
SELECT <select_list>
FROM Table_A A
RIGHT JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



Outer excluding JOIN

Not available in MySQL

```
SELECT <select_list>
FROM Table_A A
FULL OUTER JOIN Table_B B
ON A.Key = B.Key
WHERE A.Key IS NULL OR B.Key IS NULL
```

Joins

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Ex: 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
```

Note: we will often omit attribute types in schema definitions for brevity, but assume attributes are always atomic types

Joins

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Ex: 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
```

A join between tables returns all unique combinations of their tuples which meet some specified join condition

Joins

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Ex: 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
```

A join between tables returns all unique combinations of their tuples which meet some specified join condition

Joins

```
Product(PName, Price, Category, Manufacturer)
Company(CName, StockPrice, Country)
```

Several equivalent ways to write a basic join in SQL:

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
       AND Country='Japan'
       AND Price <= 200
```

```
SELECT PName, Price
FROM   Product
JOIN   Company ON Manufacturer = Cname
                  AND Country='Japan'
WHERE  Price <= 200
```

A few more later on...

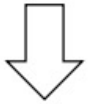
Joins

Product

PName	Price	Category	Manuf
Gizmo	$19	Gadgets	GWorks
Powergizmo	$29	Gadgets	GWorks
SingleTouch	$149	Photography	Canon
MultiTouch	$203	Household	Hitachi

Company

Cname	Stock	Country
GWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan



```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
       AND Country='Japan'
       AND Price <= 200
```

PName	Price
SingleTouch	$149.99

Tuple Variable Ambiguity in Multi-Table

```
Person(name, address, worksfor)
Company(name, address)
```

```
SELECT DISTINCT name, address
FROM            Person, Company
WHERE           worksfor = name
```

Which "address" does this refer to?

Which "name"s??

Tuple Variable Ambiguity in Multi-Table

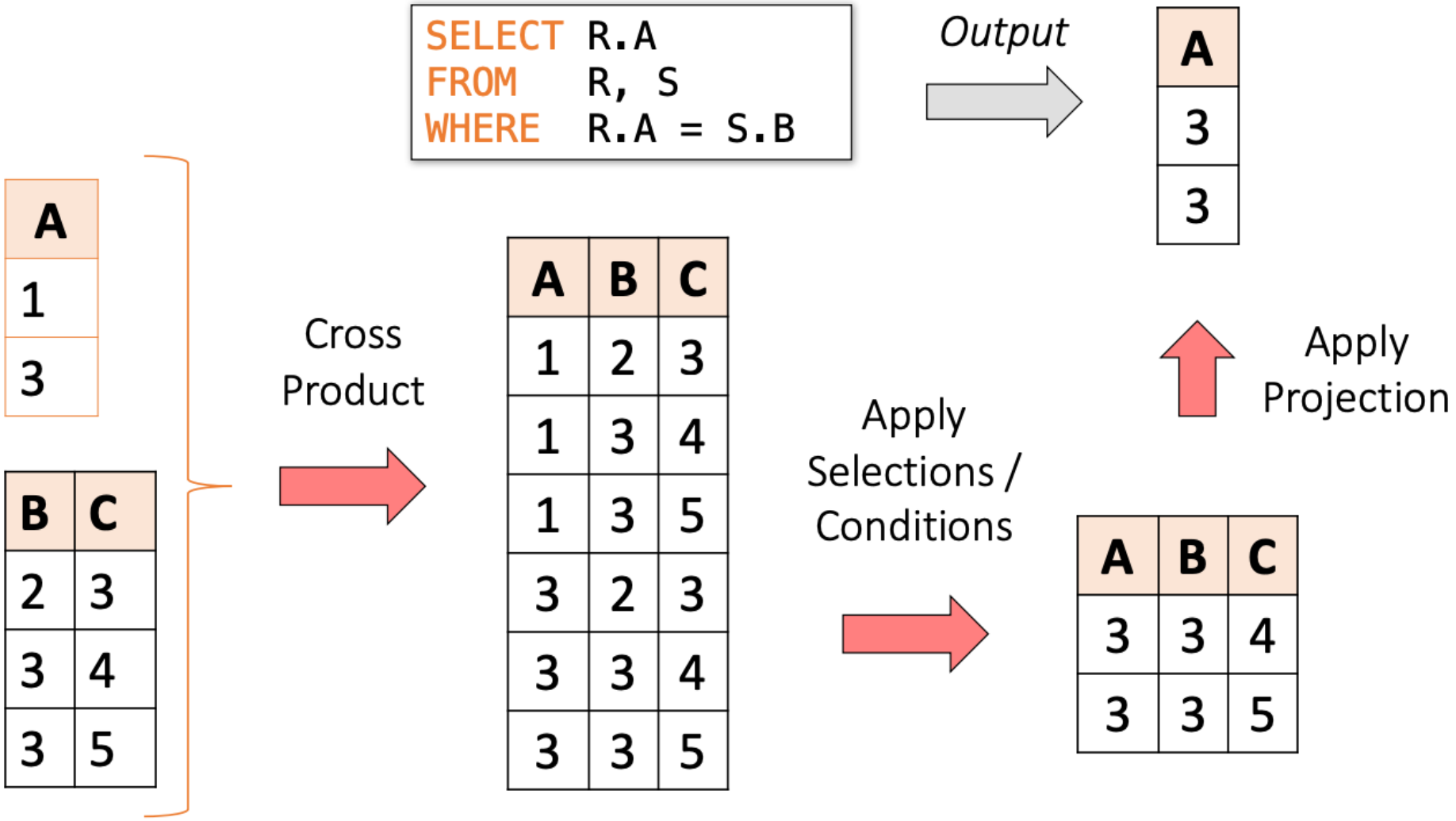
```
Person(name, address, worksfor)
Company(name, address)
```

Both equivalent ways to resolve variable ambiguity

```
SELECT DISTINCT Person.name, Person.address
FROM            Person, Company
WHERE           Person.worksfor = Company.name
```

```
SELECT DISTINCT p.name, p.address
FROM            Person p, Company c
WHERE           p.worksfor = c.name
```

An example of SQL semantics



The semantics of a join

```
SELECT R.A
FROM   R, S
WHERE  R.A = S.B
```

1. Take **cross product**:

$$X = R \times S$$

Recall: Cross product (A X B) is the set of all unique tuples in A,B

Ex: {a,b,c} X {1,2}
= {(a,1), (a,2), (b,1), (b,2), (c,1), (c,2)}

2. Apply **selections / conditions**:

$$Y = \{(r, s) \in X \mid r.A == r.B\}$$

= Filtering!

3. Apply **projections** to get final output:

$$Z = (y.A,) \; for \; y \in Y$$

= Returning only *some* attributes

Remembering this order is critical to understanding the output of certain queries (see later on...)

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

Any Questions?

o.ballinger@ucl.ac.uk

References; Theodoros Rekatsinas, Steven Gray