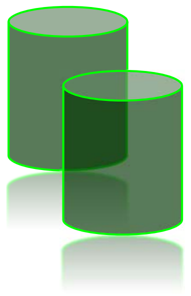
CZ2007
Introduction to Databases

Querying Relational Databases using SQL
Part-1

Arijit Khan

Assistant Professor
School of Computer Science and Engineering
Nanyang Technological University, Singapore

arijit.khan@ntu.edu.sg



Schedule after Recess Week

SQL

8 Lectures
- Week 8 (Oct 07-Oct 11)
- Week 9 (Oct 14-Oct 18)
- Week 10 (Oct 21-Oct 25)
- Week 11 (Oct 28-Nov 01)

Semi-Structured Data, Quiz-2

2 Lectures
- Week 12 (Nov 02-Nov 08)
- Quiz during Tutorial session
- Quiz syllabus: everything on SQL (Week 8, 9, 10 11)

Summary

- Week 13 (Nov 11-Nov 15)

Why Should You Study Databases?

- **Make more $$$:**
  - Startups need DB talent right away
  - Massive industry…

- **Intellectual (Research)**:
  - Science: data poor to data rich
    - No idea how to handle the data!
  - Fundamental ideas to/from all of CS:
    - Systems, theory, AI, logic, stats, analysis….

Many great computer systems ideas started in DB.

About Me …

- **Instructor (me):** Arijit Khan (http://www.ntu.edu.sg/home/arijit.khan/)
- **Faculty** (Assistant Professor), School of Computer Science and Engineering, NTU Singapore
- **Research:**
  - Graph data querying, mining, and systems
  - Big-Data management and analytics
  - Machine learning
  - Uncertain and probabilistic data
  - Dynamic and stream data
  - Crowdsourcing
- **Office:** N4-02C-94
  [Appointment by email: arijit.khan@ntu.edu.sg]

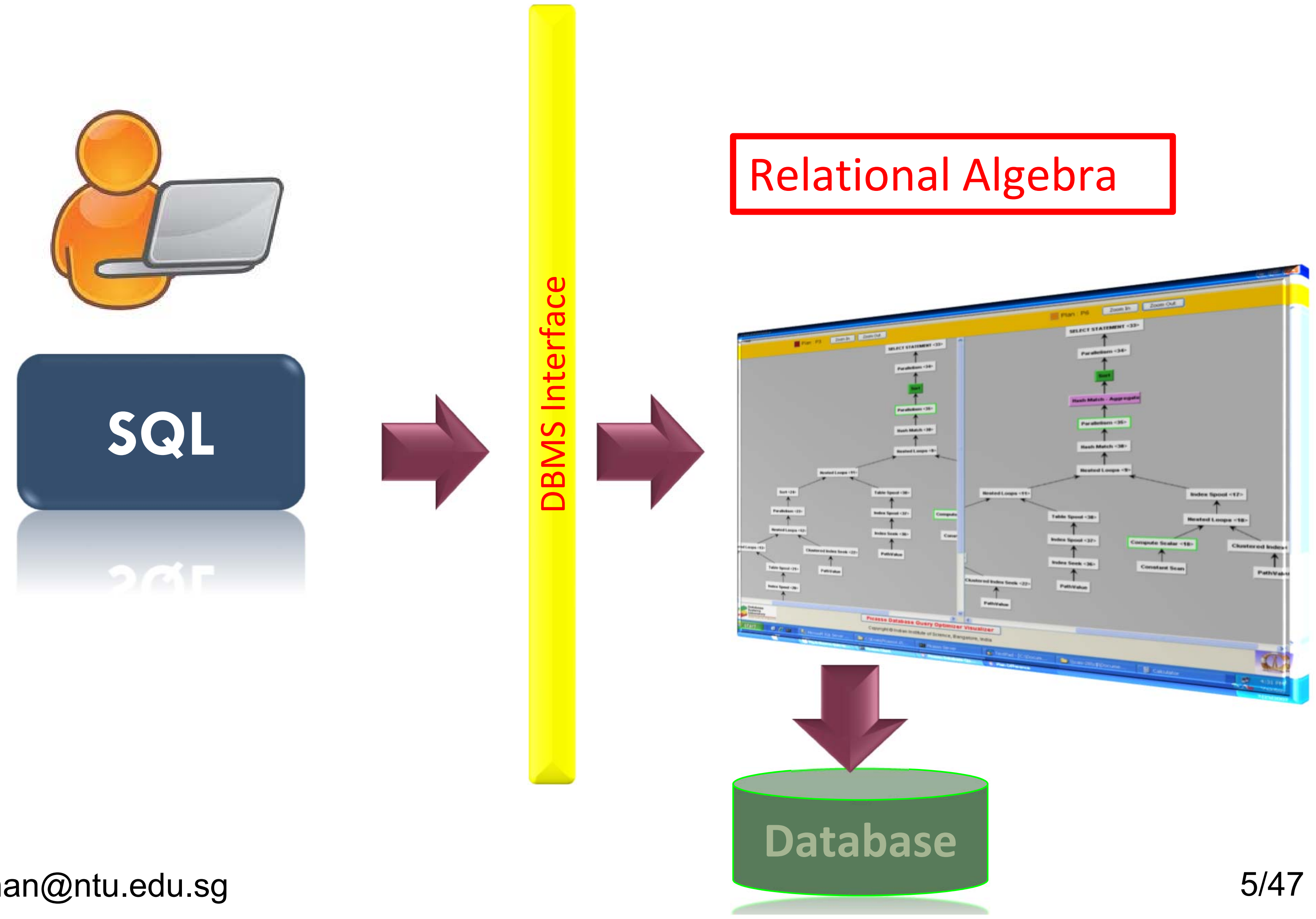
- **University of California, Santa Barbara (UCSB)**
  PhD (2008-2013)
- **IBM TJ Watson, NY**
  Intern, 2010
- **Yahoo! Labs, Barcelona**
  Intern, 2010
- **ETH Zurich, Switzerland**
  Post-doc (2014-2015)
- **NTU, Singapore**
  Assistant Professor
  (2016-now)

Ask Questions!

The important thing is not to stop questioning.

Albert Einstein

Querying RDBMS
(Relational Database Management System)

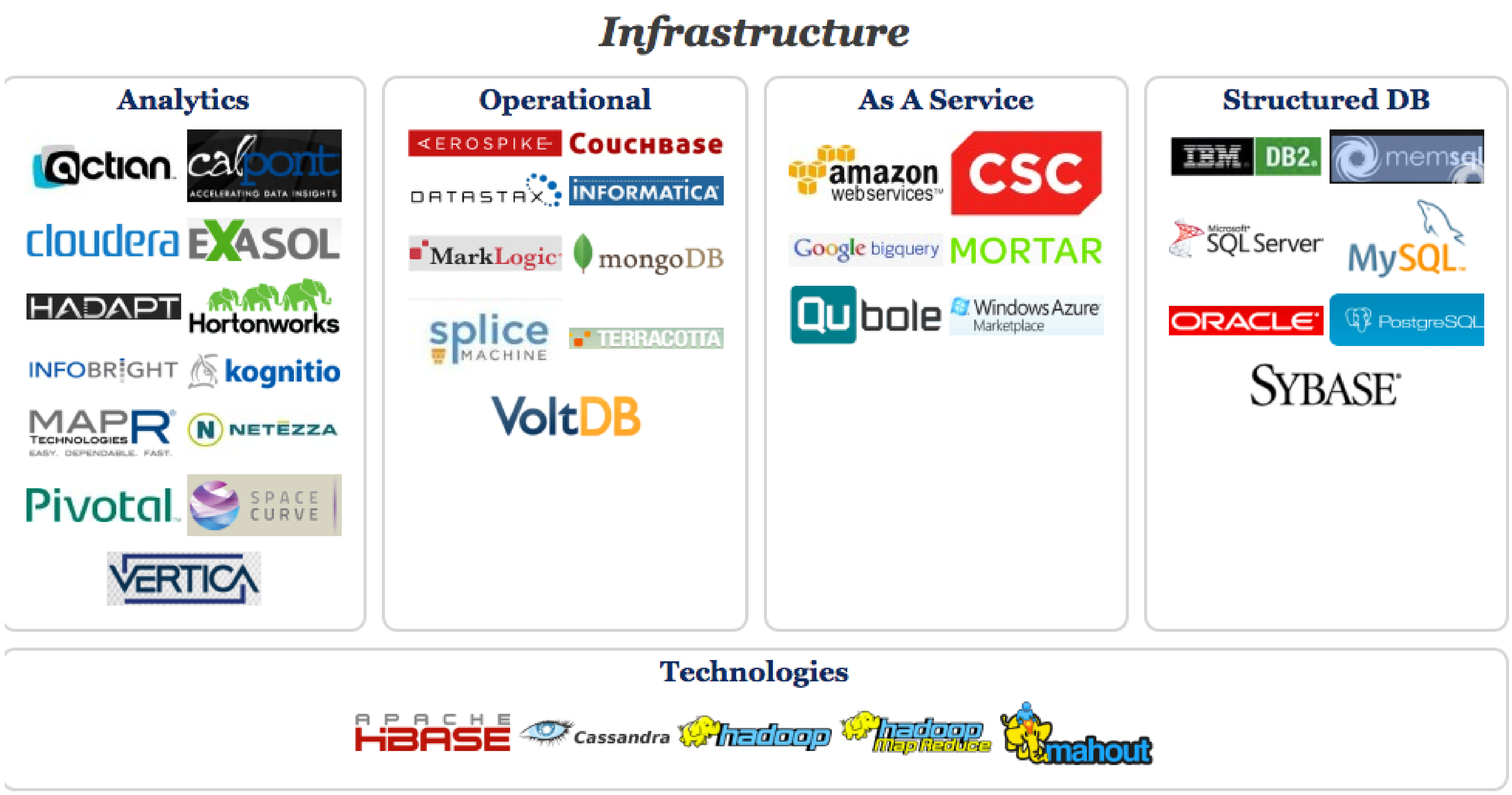


What is SQL?

- **Structured Query Language** (SQL) – standard query language for relational databases. Pronounced "**S-Q-L**" or "**sequel**"

- **A brief history:**
  - First proposal of SEQUEL (IBM Research, System R, 1974)
  - First implementation in SQL/DS (IBM) and Oracle (1981)
  - Around 1983 there is a "de facto standard"
  - Became official standard in 1986 – defined by the American National Standards Institute (ANSI), and in 1987 – by the International Organization for Standardization (ISO)
  - ANSI SQL89
  - ANSI SQL92 (SQL2)
  - ANSI SQL99 (SQL3)
  - ANSI SQL 2003 (added OLAP, XML, etc.)
  - ANSI SQL 2006 (added more XML)
  - … …. …
  - ANSI SQL 2016 (added pattern matching, JSON, etc.)

Present Days: Big Data



***New* technology. *Same* SQL Principles**

What SQL we shall study?

- All major database vendors (Oracle, IBM, Microsoft, Sybase) conform to SQL standard

- Although database companies have added “proprietary” extensions (different dialects)

- Commercial systems offer features that are not part of the standard
  – Incompatibilities between systems
  – Incompatibilities with newer standards (e.g. triggers in SQL:1999)

– We concentrate more on the principles

– (mostly) We will study SQL92 - a basic subset

Best Practice (as we learn SQL)

- Run your query in the Lab (they usually provide MySQL?)

- (It may not compile, but might still be correct!)
  Always check in **Google**

- Consult with the **Book** and course material

  - Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom
  - (Book available online)

Best Practice (as we learn SQL)

- Run your query in the Lab (they usually provide MySQL?)

- (It may not compile, but might still be correct!)
  Always check in **Google**

- Consult with the **Book** and course material
  - Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom
  - (Book available online)

Other sources:

1. Database System Concepts – book by Avi Silberschatz, Henry F. Korth, and S. Sudarshan

2. CMU database group course lecture videos in Youtube – by Andy Pavlo (https://www.youtube.com/channel/UCHnBsf2rH-K7pn09rb3qvkA)

3. Comparison of different SQL implementations – by Troels Arvin (http://troels.arvin.dk/db/rdbms/)

What we want to do with SQL?

Today's lecture: Chapter 6.1 of the Book "Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom

- Manage and query the database (a set of relations / tables)

What we want to do on the relations?
• Retrieve • Insert • Delete • Update

More about SQL

Declarative Language

•SQL is a ***declarative language*** (non-procedural).

•A SQL query specifies *what* to retrieve but not *how* to retrieve it.

What is a procedural language ??

More about SQL

Declarative Language

•SQL is a ***declarative language*** (non-procedural).

•A SQL query specifies *what* to retrieve but not *how* to retrieve it.

What is a procedural language ??

- Procedure/ Functions – Imperative Languages
- Write instructions on *how* to do it
- C, C++, Java

More about SQL

Declarative Language

•SQL is a ***declarative language*** (non-procedural).

•A SQL query specifies *what* to retrieve but not *how* to retrieve it.

What is a procedural language ??

- Procedure/ Functions – Imperative Languages
- Write instructions on *how* to do it
- C, C++, Java

SQL is Not a complete programming language

- It does not have control or iteration commands.

Stuffs supported by SQL

Data Manipulation Language (DML)

- Perform queries
- Perform updates (add/ delete/ modify)

✓

Data Definition Language (DDL)

✓

Embedded SQL

We shall not study this !

Stuffs supported by SQL

Data Manipulation Language (DML)

- Perform queries
- Perform updates (add/ delete/ modify)

✓

Data Definition Language (DDL)

- Creates databases, tables, indices
- Create views
- Specify authorization
- Specify integrity constraints

✓

Embedded SQL

We shall not study this !

Stuffs supported by SQL

Data Manipulation Language (DML)

- Perform queries
- Perform updates (add/ delete/ modify)

✓

Data Definition Language (DDL)

- Creates databases, tables, indices
- Create views
- Specify authorization
- Specify integrity constraints

✓

Embedded SQL

Wrap a high-level programming language around DML to do more sophisticated queries/updates

We shall not study this !

Roadmap (SQL)

Today's lecture: Chapter 6.1 of the Book "Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom

- Introduction to SQL ✓
- **Querying single relation**

Study-at-Home slides at the end of every lecture

- They will be in the syllabus of Quiz-2 and Final Exam
- More examples
- Study them at home, will be discussed at the beginning of next lecture
- If any questions, ask me !!

Roadmap (SQL)

- Introduction to SQL ✓
- **Querying single relation**

Lecture-1

- Ordering Tuples
- Multi-relation queries
- Subqueries

Lecture-2

- Set operations
- Bag semantics
- Join expressions
- Aggregation

Lectures-3 & 4

Recap: Roadmap (SQL)

- Groupings
- Creation of tables
- Database modifications
- Constraints
- Views

Lecture-5 & 6

- Triggers
- Indexes

Lecture-7 & 8

That would be all about Quiz-2!!

Questions?

Tables in SQL

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

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!

Attributes (Columns) in a Table

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.*

Tuples (Rows) in a Table

Product

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

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

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

More on Tables

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

Product

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

Schema of a Table

Product

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

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

Product(Pname: *string*, Price: *float*, Manufacturer: *string*)

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

Product(<u>Pname</u>: *string*, Price: *float*, Manufacturer: *string*)

Principle Form of SQL

Basic Structure of SQL

SELECT desired attributes (***A1, A2, … , An)***
FROM one or more tables (***R1, R2, … , Rm)***
WHERE condition about tuples of the tables (***P)***

Mapping to Relational Algebra

$$\Pi_{A1, A2, \ldots, An} (\sigma_P (R1 \times R2 \times \ldots \times Rm))$$

Principle Form of SQL

Basic Structure of SQL

SELECT desired attributes (***A1, A2, … , An)***
FROM one or more tables (***R1, R2, … , Rm)***
WHERE condition about tuples of the tables (***P)***

Today, we talk about "One Table" only ☺

Mapping to Relational Algebra

$$\Pi_{A1,\ A2,\ \ldots,\ An} (\sigma_P (R1 \times R2 \times \ldots \times Rm))$$

Today, we talk about "One Relation" only ☺

SQL Syntax

Reserved words / Keywords

- There is a set of *reserved words* that cannot be used as names for database objects.
- SELECT, FROM, WHERE, etc.

Case-insensitive

- SQL is generally *case-insensitive*.
  Exception: is string constants. 'FRED' not the same as 'fred'.
- Use single quotes for constants:
  ‘abc’ – Okay
  “abc” – Not okay

White-space ignored

- White-space is ignored
- All statements end with a semicolon (;)

Simple SQL Query: Selection

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

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

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

Simple SQL Query: 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

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

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

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	29.99	GizmoWorks

Notation for (SELECT-FROM-WHERE) Query

Input schema

Product(<u>PName</u>, Price, Category, Manfacturer)

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Category = ‘Gadgets’
```

$\Downarrow$

Output schema

Answer(PName, Price, Manfacturer)

DISTINCT: Eliminating Duplicates

```
SELECT DISTINCT Category
FROM   Product
```

→

Category
Gadgets
Photography
Household

Versus

```
SELECT Category
FROM   Product
```

→

Category
Gadgets
Gadgets
Photography
Household

AS: Renaming Attributes

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 AS Product, Price AS Cost, Manufacturer
FROM   Product
WHERE  Category = ‘Gadgets’
```

Product	Cost	Manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	29.99	GizmoWorks

Expressions in SELECT Clause

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*1.4 AS Cost_IN_SGD, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Cost_IN_SGD	Manufacturer
Gizmo	27.99	GizmoWorks
Powergizmo	41.99	GizmoWorks

Questions?

Summary

- Introduction to SQL ✓
- Querying single relation ✓

Study-at-Home

More examples and cases for "Querying Single Relation" (Slides 38-47)
- Complex conditions in WHERE clause
- NULL values and 3-valued logic

Will be in the syllabus of Quiz-2 and Final Exam

Complex Conditions in WHERE Clause: AND

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  Category = 'Gadgets' AND Price < 20
```

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks

Complex Conditions in WHERE Clause: BETWEEN

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  BETWEEN 10 AND 20
```

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks

Complex Conditions in WHERE Clause: IN

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  Manufacturer  IN (‘GizmoWorks’, ‘Samsung’, ‘Hitachi’)
```

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	29.99	GizmoWorks
MultiTouch	203.99	Hitachi

Complex Conditions in WHERE Clause: LIKE (String Pattern Matching)

s **LIKE** p: pattern matching on strings

Patterns are case sensitive

p may contain two special symbols:

% = any sequence of Characters

_ = any single character

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

PName	Price	Category	Manufacturer
Powergizmo	29.99	Gadgets	GizmoWorks

Complex Conditions in WHERE Clause: LIKE (String Pattern Matching)

s **LIKE** p: pattern matching on strings

Patterns are <u>case sensitive</u>

p may contain two special symbols:

% = any sequence of Characters

_ = any single character

More Examples

- 'John%' – Matches any string beginning with "John"
- '%ohn%' – Matches any string containing "ohn" as substring
- '_ _ _' – Matches any string of exactly three characters
- '_ _ _%' – Matches any string of at least three characters
- 'ab\%cd%' – Match all strings beginning with "ab%cd"

NULL Values

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
iPhone 8	NULL	Smartphone	Apple

NULL

Tuples in SQL relations can have NULL as a value for one or more attributes.

Meaning

- *Missing value* : e.g., we know ‘iPhone 8’ has some Price, but we don’t know what it is.
- *Inapplicable* : e.g., ‘iPhone 8’ is not available yet in the market

NULL Values

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
iPhone 8	NULL	Smartphone	Apple

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Price <= 150 OR Price >= 150
```

- Include or not include NULL values?
- Answer in the remaining slides ☺

SQL: 3-Valued Logic

3-value logic

- The logic of conditions in SQL is really 3-valued logic
- TRUE, FALSE, UNKNOWN.

Comparing with NULL

When any value is compared with NULL, the truth value is **UNKNOWN**.

SQL Rules

A query only produces a tuple in the answer if its truth value for the WHERE clause is **TRUE** (not FALSE or UNKNOWN).

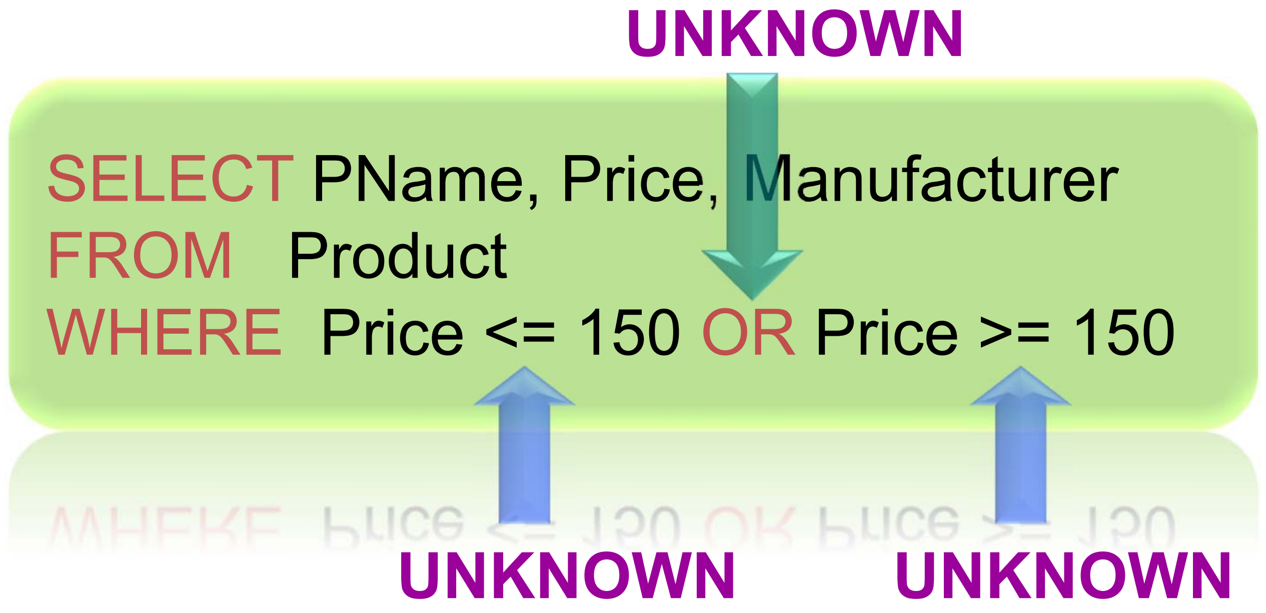
3-Valued Logic

3-Value Logic

To understand how AND, OR, and NOT work in 3-valued logic, think of **TRUE = 1, FALSE = 0,** and **UNKNOWN = ½**

AND, OR, NOT

- AND = MIN
- OR = MAX
- NOT(*x*) = 1-*x*



```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Price <= 150 OR Price >= 150
```

NULL price values will NOT be reported

Questions ??

Thank You !