Before we start…

This is the **Introduction to Databases Design and Query** workshop

- Download material: dartgo.org/db
- More info: rc.dartmouth.edu

DARTMOUTH
Information, Technology and Consulting

Introduction to Database Design and Query

Christian Darabos, Ph.D.
christian.Darabos@dartmouth.edu

Slides download: dartgo.org/db

Overview

- DB design and UML and case study
  Source: http://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html

- DB query and SQL
  Source: https://www.mysqltutorial.org/basic-mysql-tutorial.aspx

Part 1

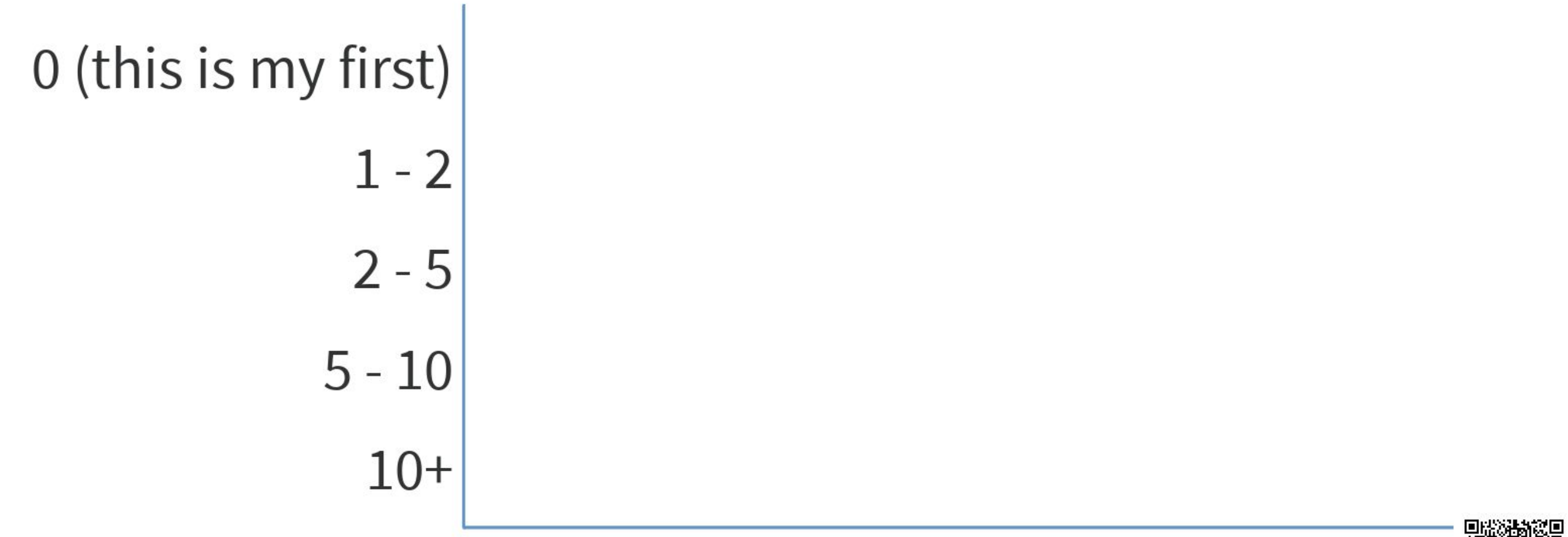
DB design and UML and case study

Source:
http://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html

Right-click > Open link in new window
To keep open slides and poll

dartgo.org/poll

How many RC/RTL workshops have you attended? (excluding this one)



Right-click > Open link in new window
To keep open slides and poll

dartgo.org/poll

In one word, what is a database to you? (word cloud)

Definition of a Relational Database (SQL)

- a database type structured to recognize relations among stored items of information

- designed to store text, dates/times, integers, floating-point number

- implemented as a series of tables

Mental Model

- Think of a database as a set of spreadsheets
- Each spreadsheet (or table) represents a type of entity (person, object, concept, etc.)
- Better than Excel because it also models the relationship between the entities

Why use a Relational Database

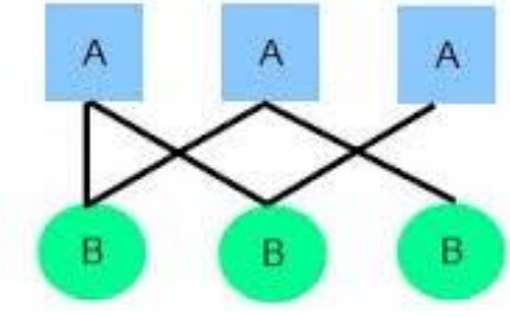
- concurrent (simultaneous) read and **write**
- **powerful selecting, filtering and sorting** cross-referencing tables
- large quantity of structured storage and standardized distribution
- minimize post-processing (simple analytics tools pre-implemented)
- automate using any scripting and programming languages (R, Matlab, Python, C++, Java, PHP)
- web-proof

SQL vs. NoSQL

SQL	NoSQL
Relational Databases	distributed database
table based	document based, key-value pairs, graph databases or wide-column stores
predefined schema	dynamic schema for unstructured data
vertically scalable (more powerful hardware)	horizontally scalable (more hardware)
SQL (structured query language)	proprietary language
MySql, Oracle, Sqlite, Postgres, MariaDB, …	MongoDB, BigTable, Redis, RavenDb, Cassandra, …

Methodologies

- **(Enhanced) Entity-Relationship — (E)ER —model**: a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.



- **Unified Modeling Language (UML)**: a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.

Database Design Environment

- pen and paper

Case Study

- A retail back-office manager wants to track the sales at multiple retail locations (shops):
  - customers purchase products in a shop
  - purchased products are bundled in a sale
  - each sale is attributed to a vendor

Enhanced Entity-Relationship modeling steps

1. Identify Entities and their Attributes
2. Identify Relationships
3. Keys Assignment
4. Attributes Data Types
5. Normalization

Identify Entities (I)

- entities are the individuals and objects (concrete or abstract) of which you need to represent the interactions or relationships

- **entities types** are stored as tables (i.e. customers, stores, products, etc.)

- **entities** are stored as entries (line items) in the tables (ie. John, JCrew 747, cashmere sweater, etc.)

Go to poll window now :)

dartgo.org/poll

What are the possible entities in the case study presented?

Top

Identify Entities (II)

Entities: types of information.

Identify Attributes (I)

- attributes are the relevant characteristics of the entities
- attribute types are stored as columns in entity type tables (i.e. customers number, stores' street address, products unit price, etc.)
- attributes of each entity are stored as elements (column items) in the tables (ie. 12345, Main St, $200, etc.)

What are relevant attribute types of 'Customers'?

Top

What are relevant attribute types of 'Sale'?

Top

Identify Attributes (II)

Entities: Entities with attributes.

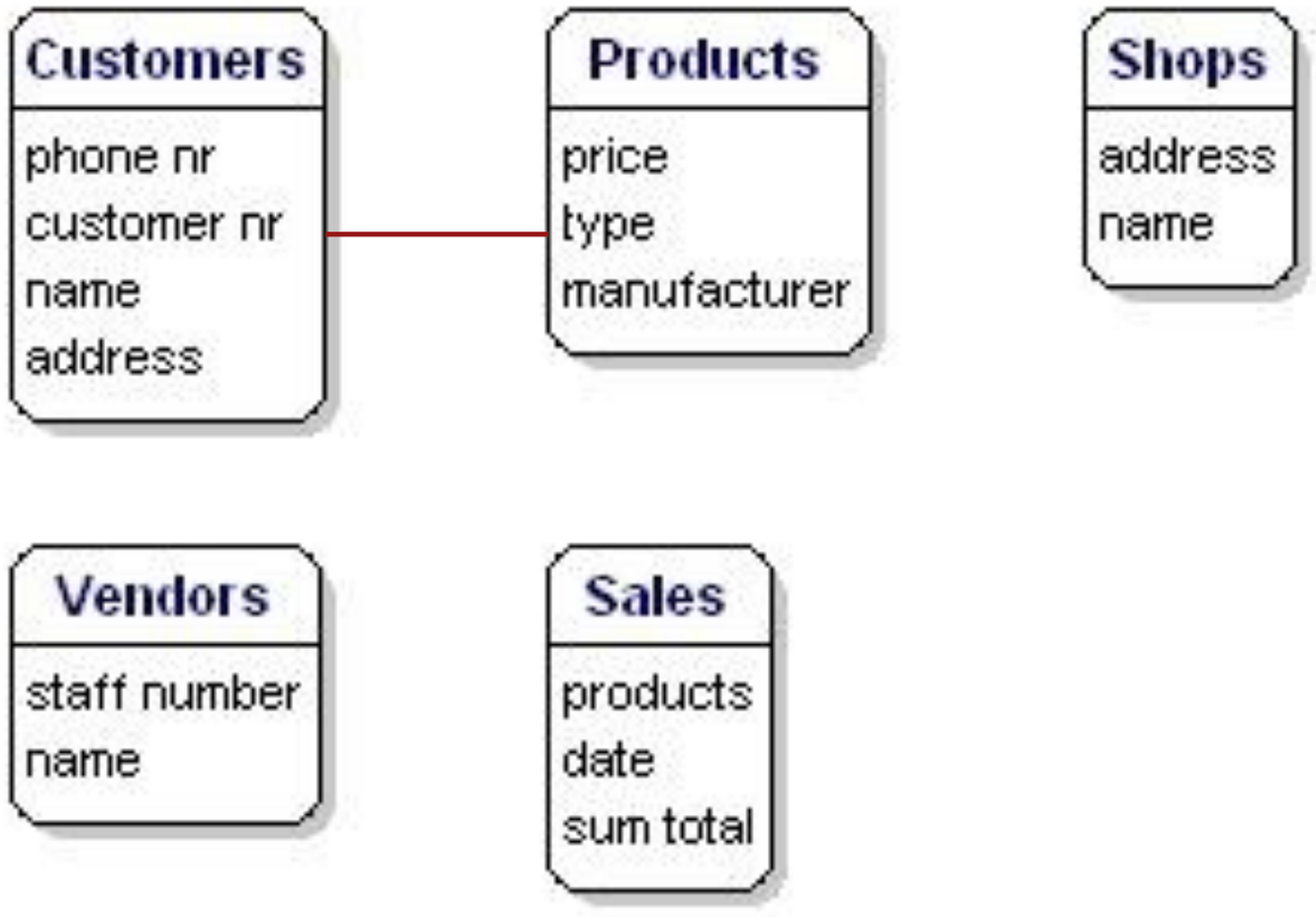
Identify Relationships (I)

English grammar structure	EER structure	Example
Common noun	Entity type	*customer*
Proper noun	Entity	*Jane Doe*
Transitive verb	Relationship type	*Jane* **buys** *a pen*
Intransitive verb	Attribute type	*Jane* **moved**
Adjective	Attribute for entity	*Jane is* **blond**
Adverb	Attribute for relationship	*Jane buys a pen* **impulsively**

Identify Relationships (II)

Entities: Entities with attributes.

Identify Relationships (II)



Entities: Entities with attributes.

Identify Relationships (II)

Entities: Entities with attributes.

Identify Relationships (II)

Entities: Entities with attributes.

Identify Relationships (II)

Entities: Entities with attributes.

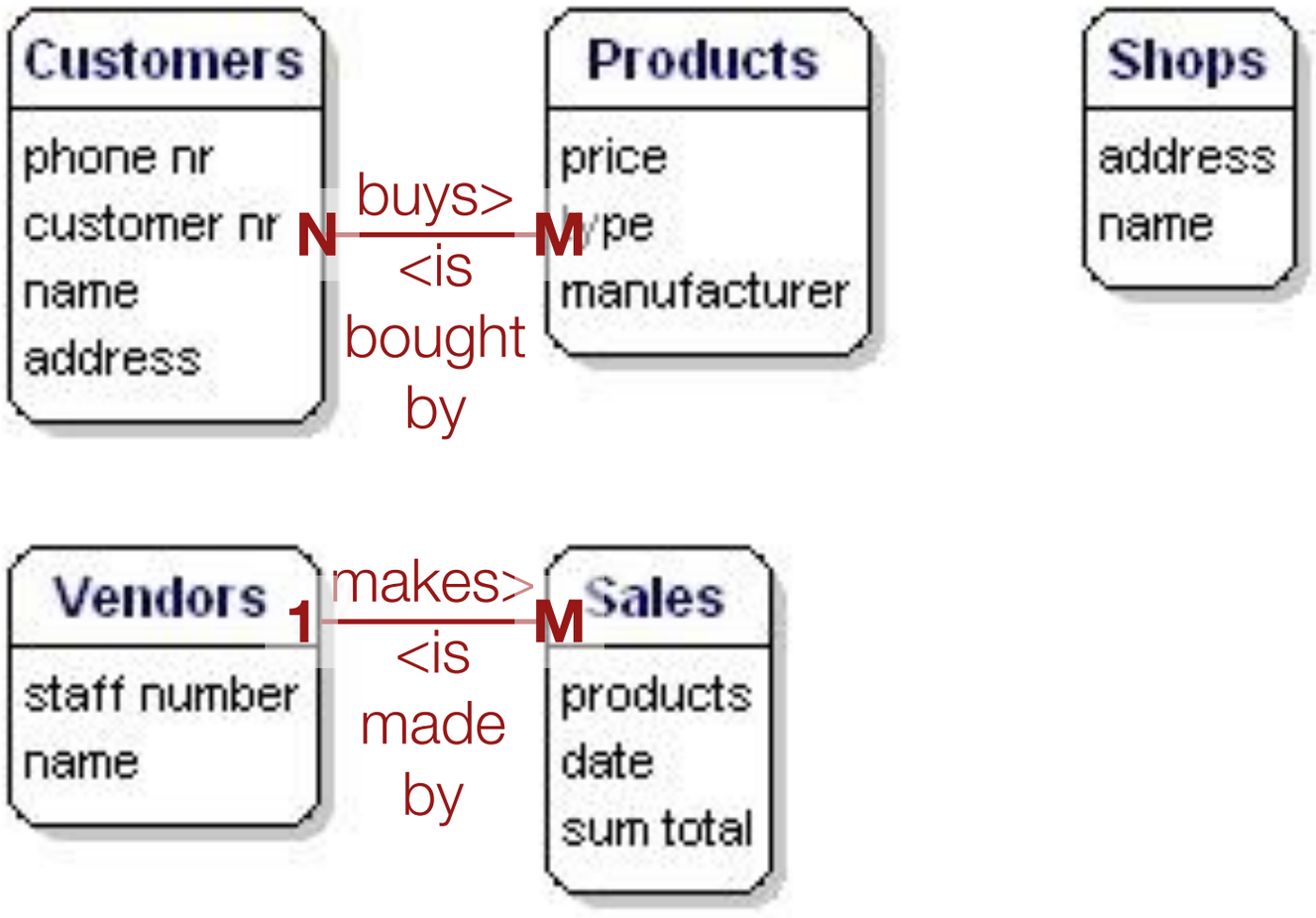
Identify Relationships (II)

Entities: Entities with attributes.

What's a possible relationship between "Sales" and "Vendors"?

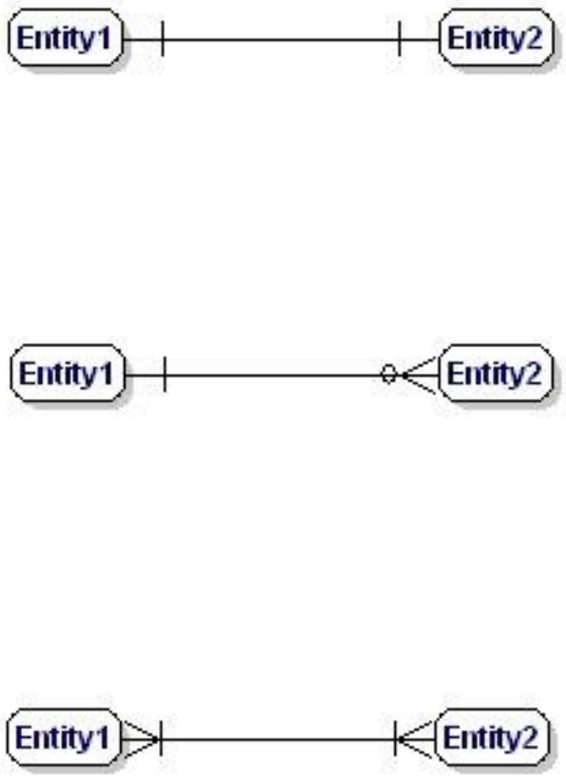
Top

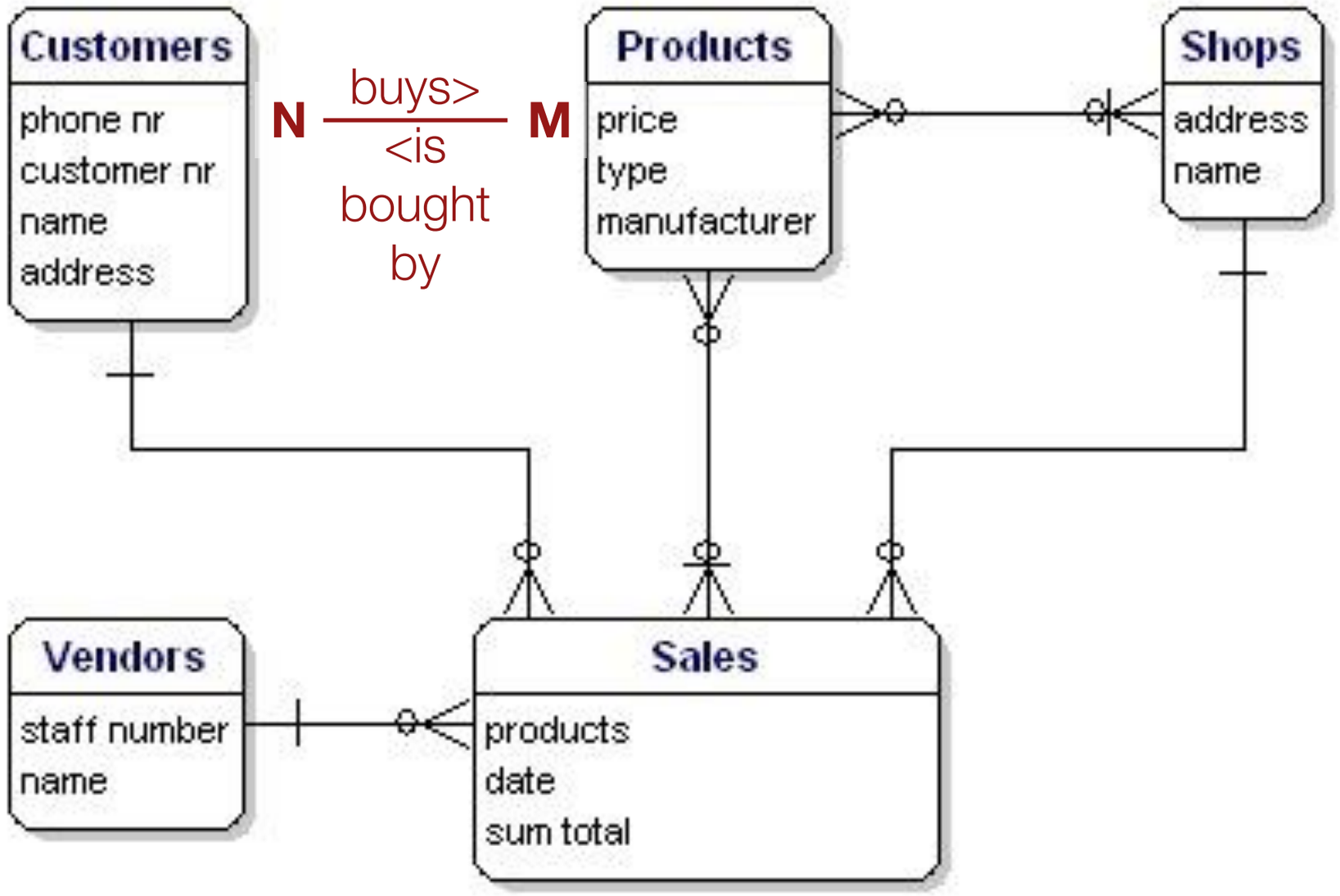
Identify Relationships (II)



Entities: Entities with attributes.

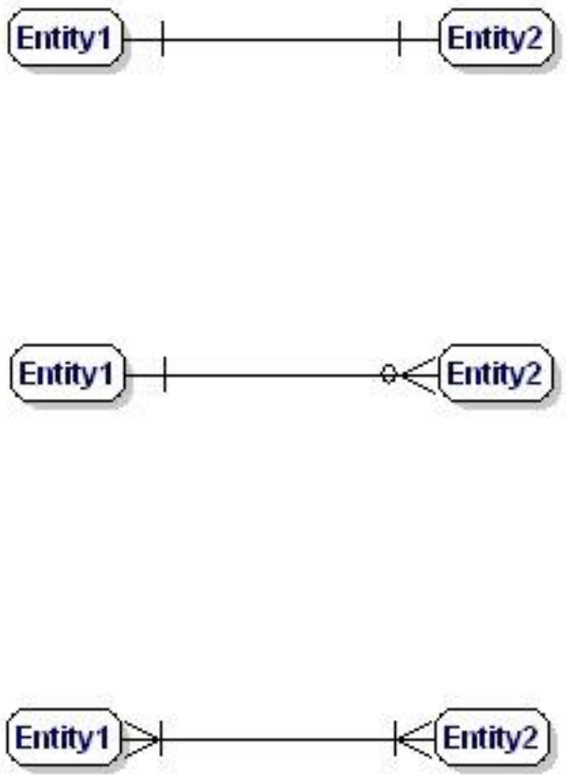
Identify Relationships (III)

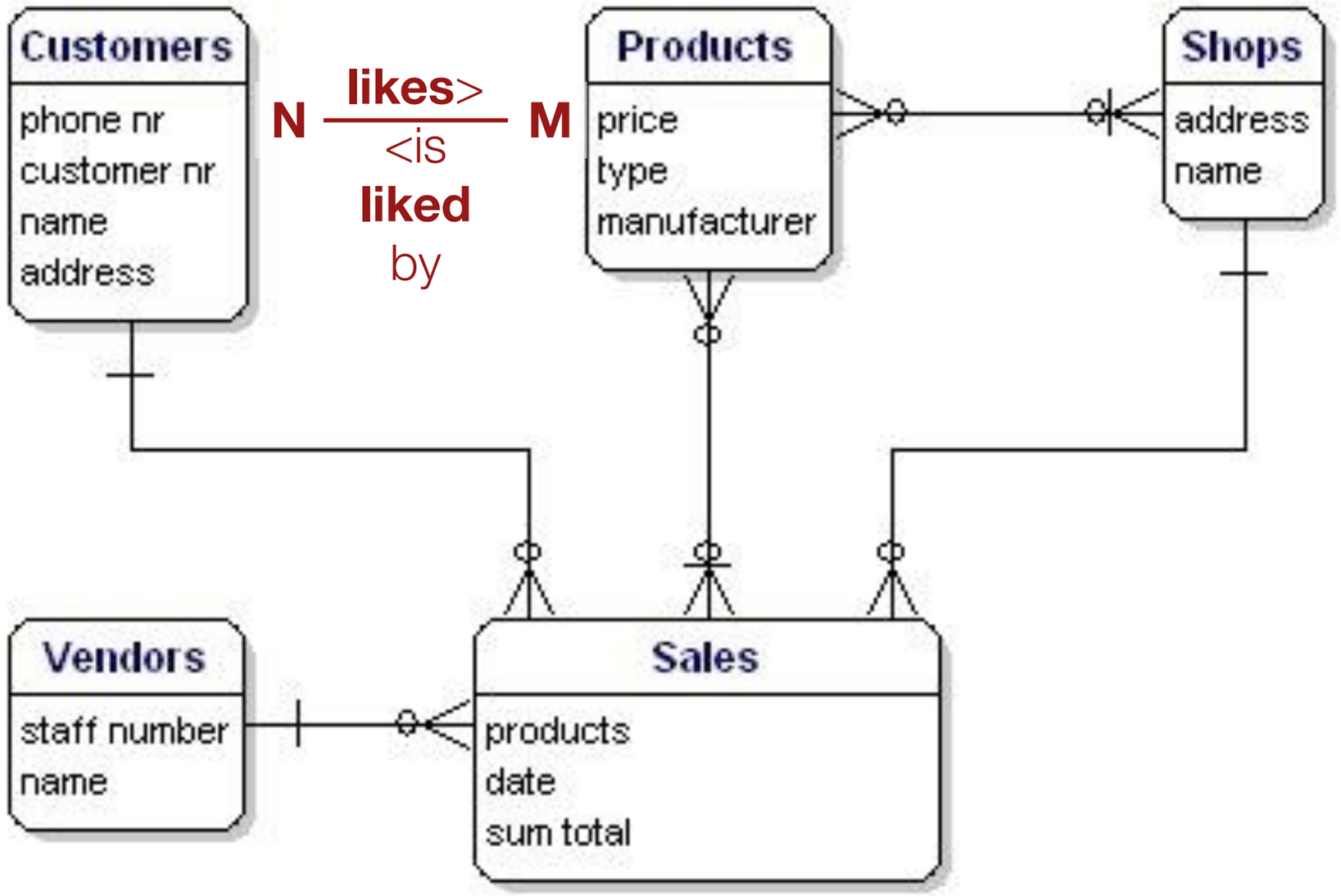




Relationships between the entities.

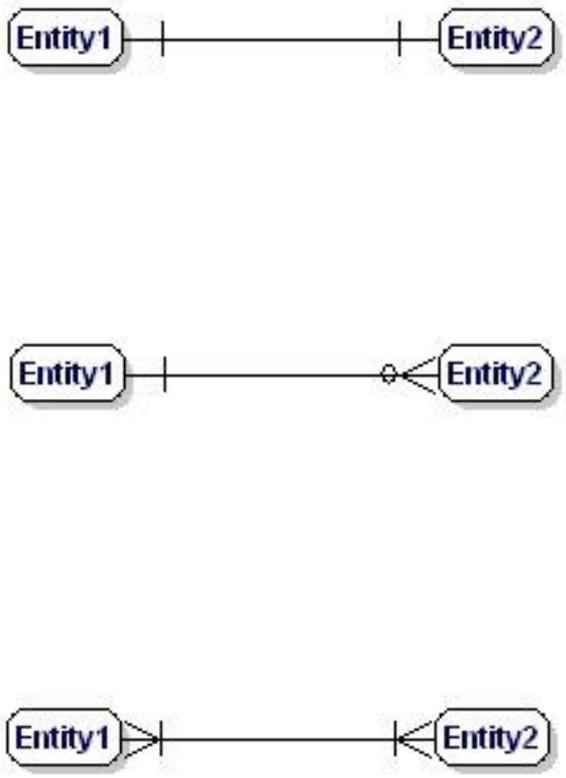
Identify Relationships (III)

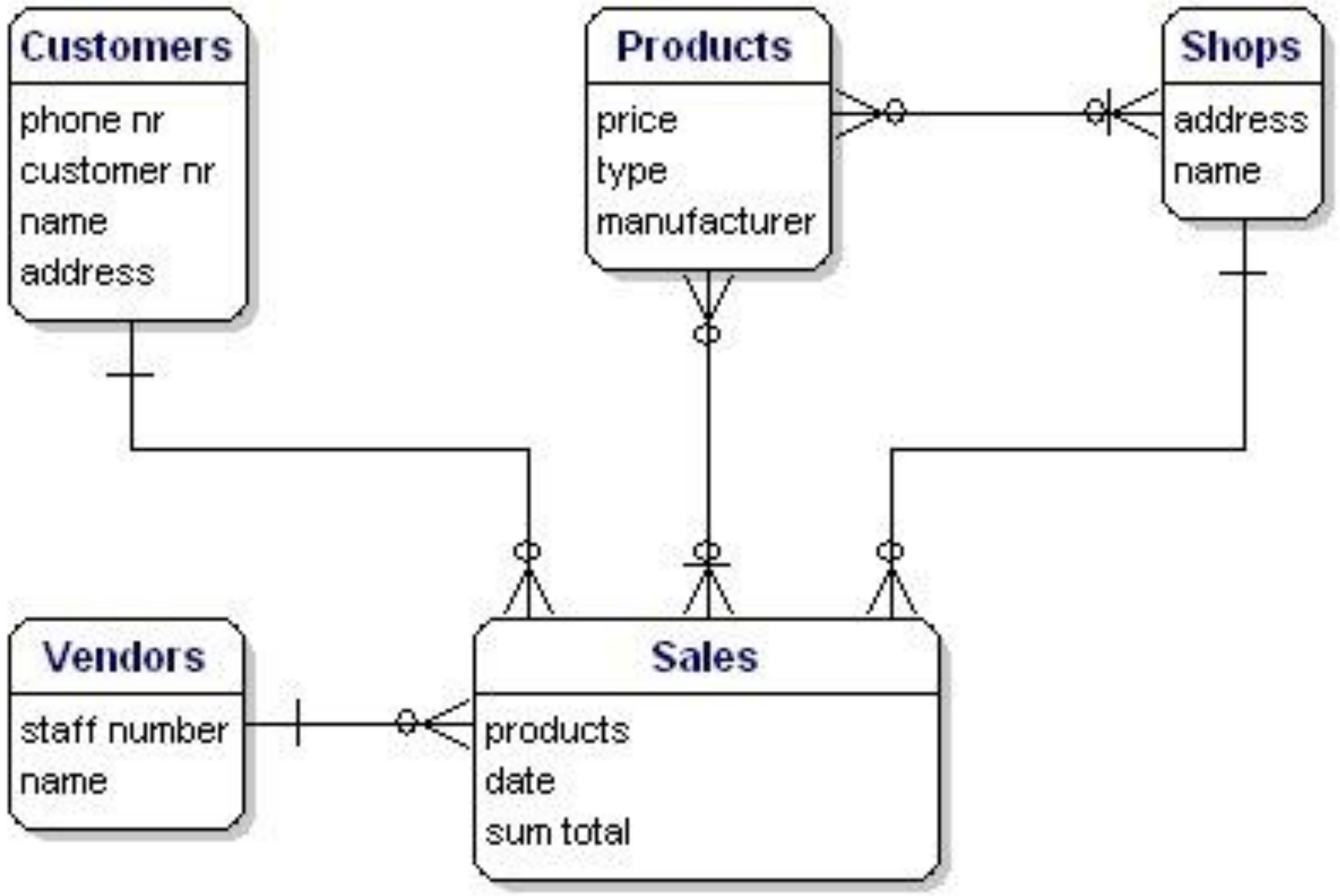




Relationships between the entities.

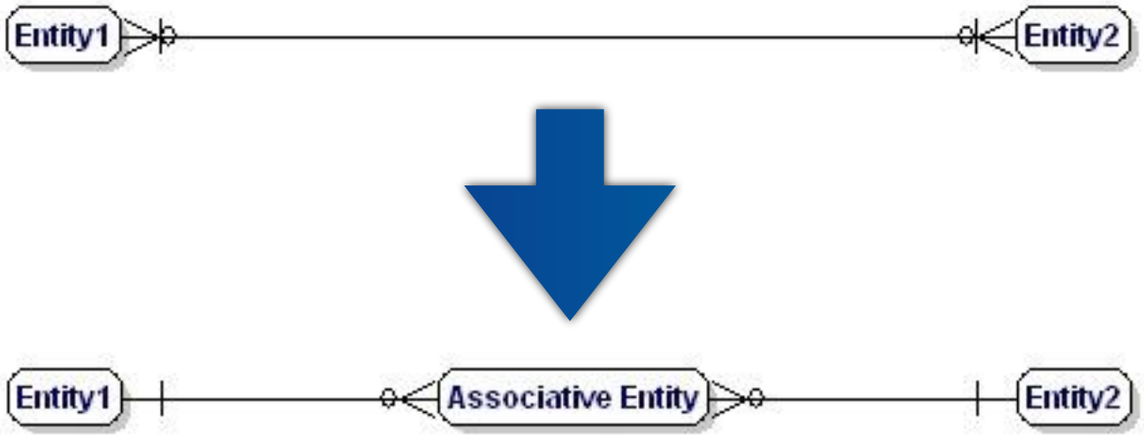
Identify Relationships (III)

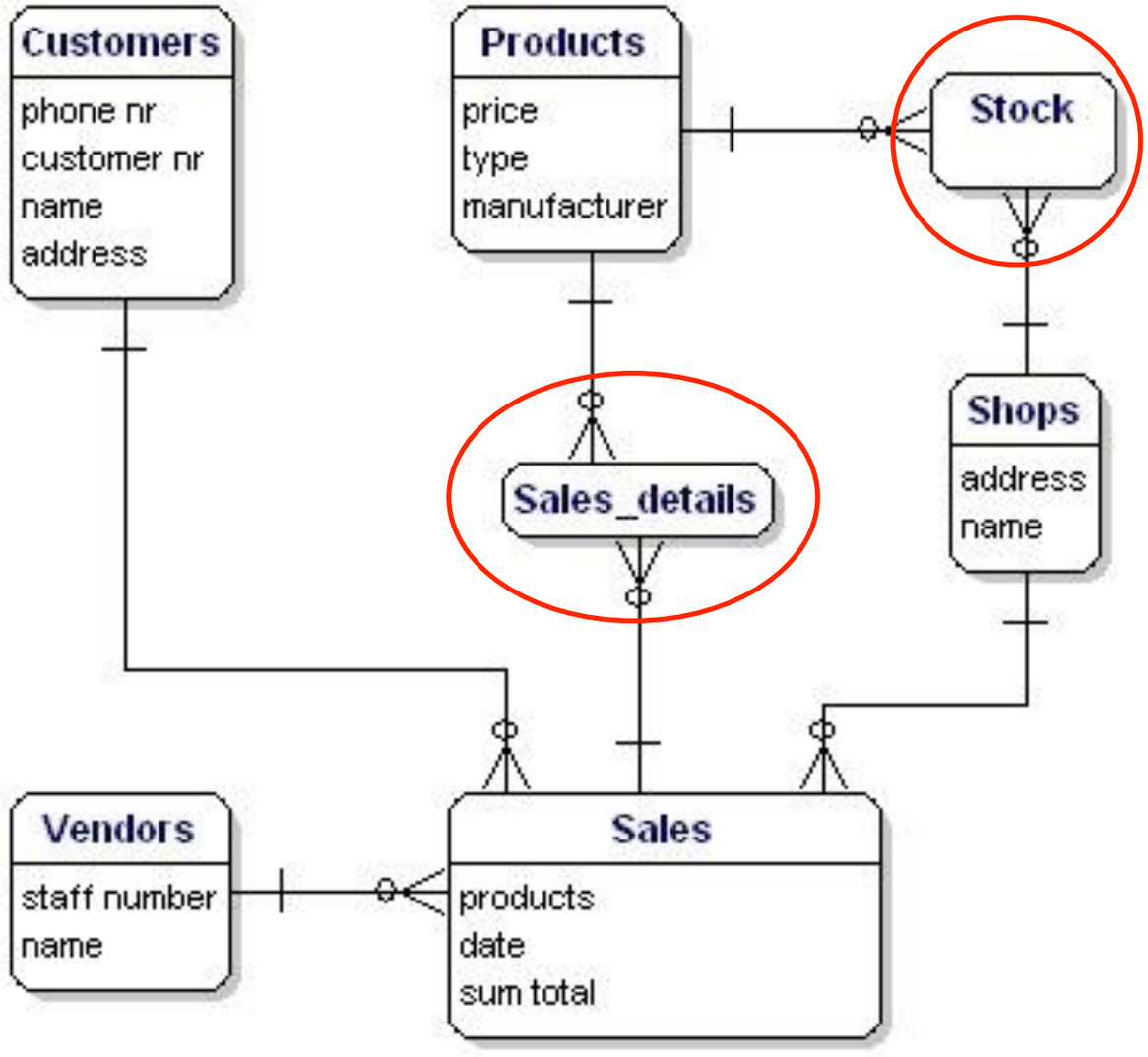




Relationships between the entities.

Identify Relationships (III)

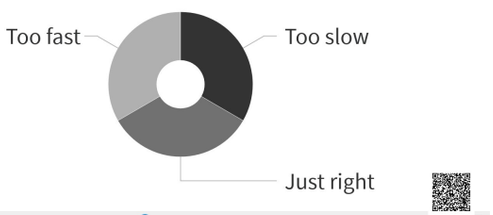




Relationships between the entities.

How's the pace of this workshop?

Too slow **A** Just right **B** Too fast **C**



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Keys Assignment (I)

- **primary key** (PK) is a set of one or more data attributes that uniquely identify an entity

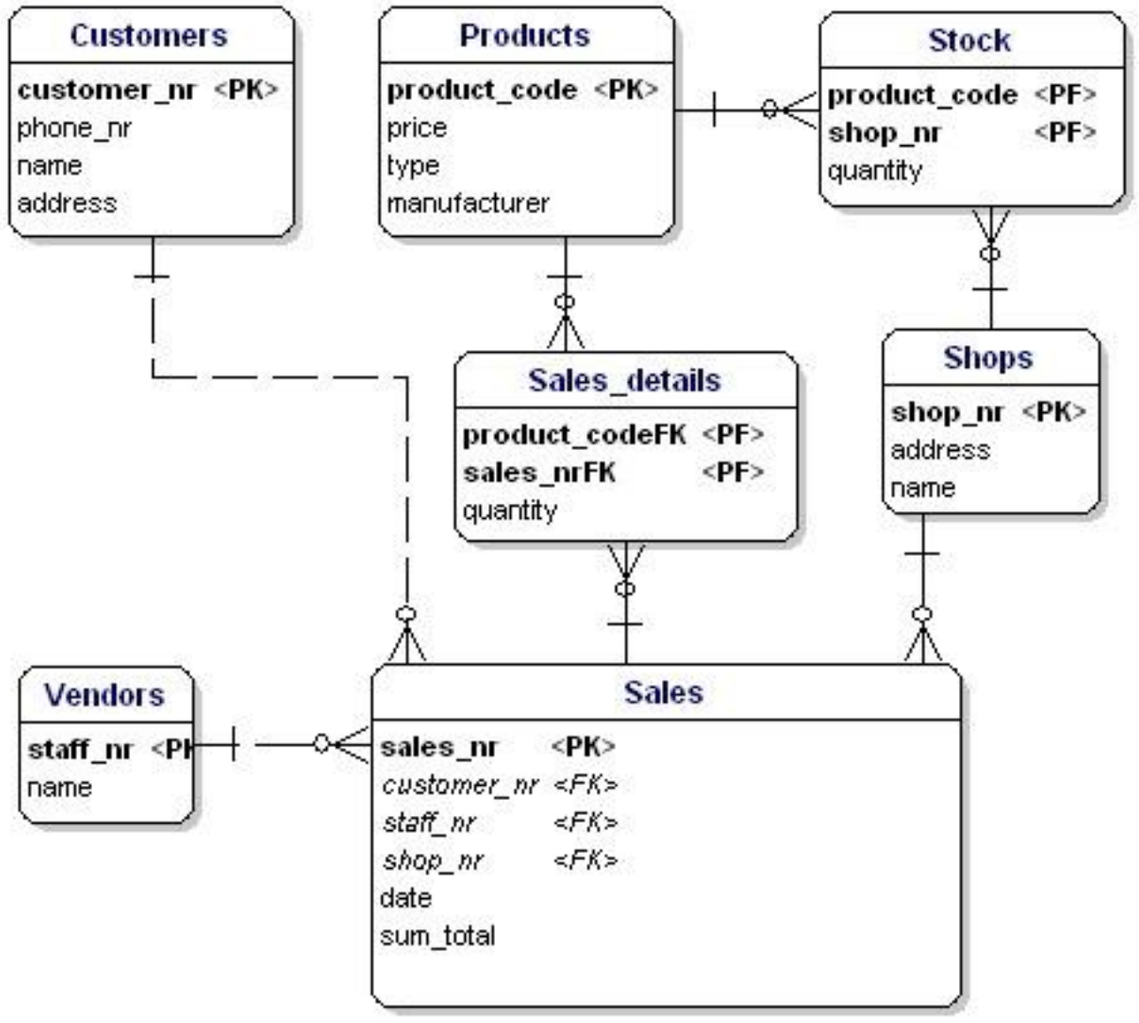
- **foreign key** (FK) in an entity is the reference to the primary key of another entity

- **indexed** fields are “indexed” in a separate manner to increase make their referencing faster at the expense of space

What are each entities' PRIMARY KEY? (response format: table - attribute type)

Top

Keys Assignment (II)



Primary keys and foreign keys.

Keys Assignment (III)

MyISAM versus Innodb

The following table provides a brief comparison of the engine types. The abbreviation ACID stands for Atomicity, Consistency, Isolation, Durability.

MyISAM	Innodb
Not ACID-compliant and non-transactional	ACID-compliant and hence fully transactional with ROLLBACK and COMMIT and support for Foreign Keys
MySQL 5.0 default engine	Rackspace Cloud default engine
Offers compression	Offers compression
Requires full repair and rebuild of indexes and tables	Provides automatic recovery from crashes via the replay of logs
Changed database pages written to disk instantly	Dirty pages converted from random to sequential before commit and flush to disk
No ordering in storage of data	Row data stored in pages in PK order
Table-level locking	Row-level locking

MyISAM vs. Innodb

Attributes Data Types (I)

Text:

- CHAR(length) - includes text (characters, numbers, punctuations...). CHAR saves a fixed amount of positions.
- VARCHAR(length) - includes text (characters, numbers, punctuation...). VARCHAR is the same as CHAR, the difference is that VARCHAR only takes as much space as necessary.
- TEXT - can contain large amounts of text. Depending on the type of database this can add up to gigabytes.

Numbers:

- INT - contains a positive or negative whole number. Variations: TINYINT, SMALLINT, MEDIUMINT, BIGINT…
  - INT is 4 bytes : -2147483647 to +2147483646, **UNSIGNED** from 0 to 4294967296.
  - INT8, or BIGINT, 0 to 18446744073709551616, but takes up to 8 bytes of diskspace
- FLOAT, DOUBLE - MySQL calculating with floating point numbers is not perfect, (1/3)*3 will result in 0.9999999, not 1.

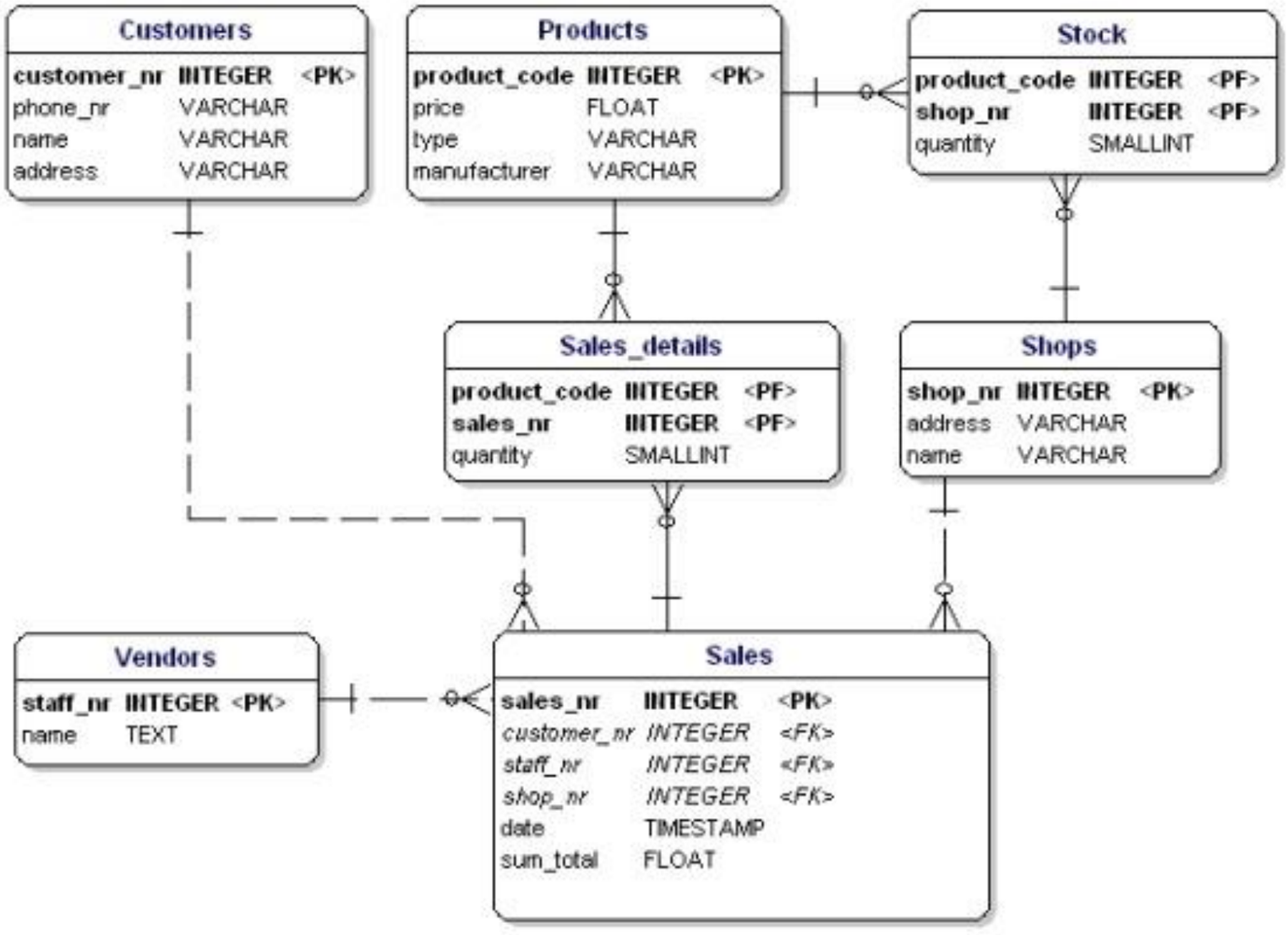
Other types:

- BLOB - for binary data such as (serialized) files.
- INET - for IP addresses. Also useable for netmasks.

For each PRIMARY KEY, what data type would you assign? (response format: entity-attribute type-data type)

Top

Attributes Data Types (II)

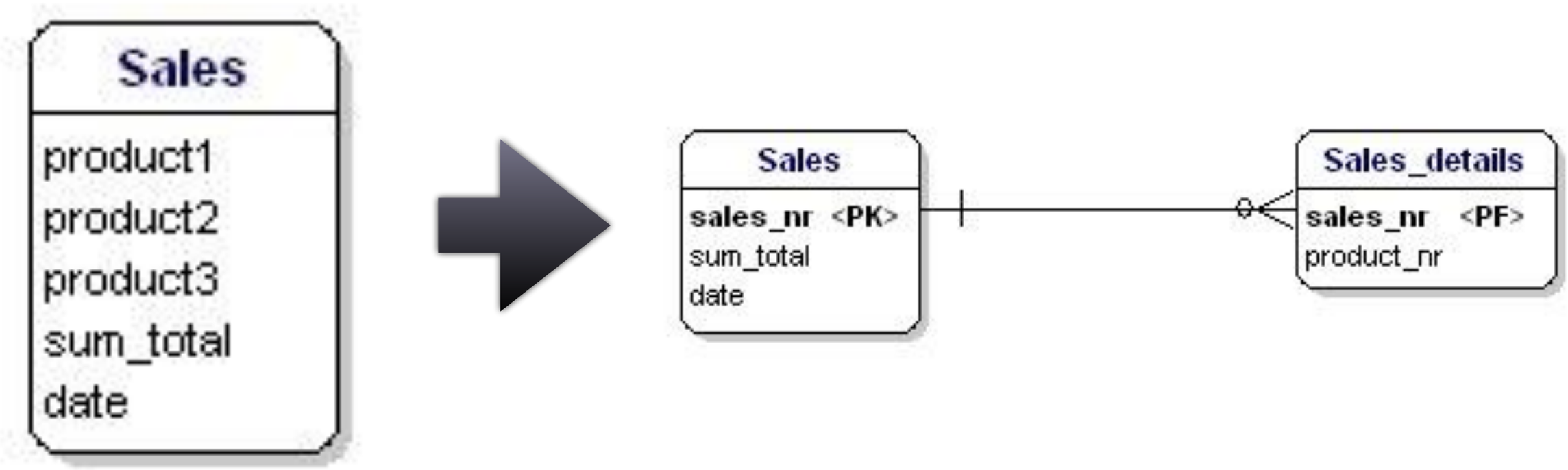


Data model displaying data types.

Normalization (I)

- **Normalization makes your data model flexible and reliable. It does generate some overhead because you usually get more tables, but it enables you to do many things with your data model without having to adjust it.**

Normalization (I)



1st normal form:

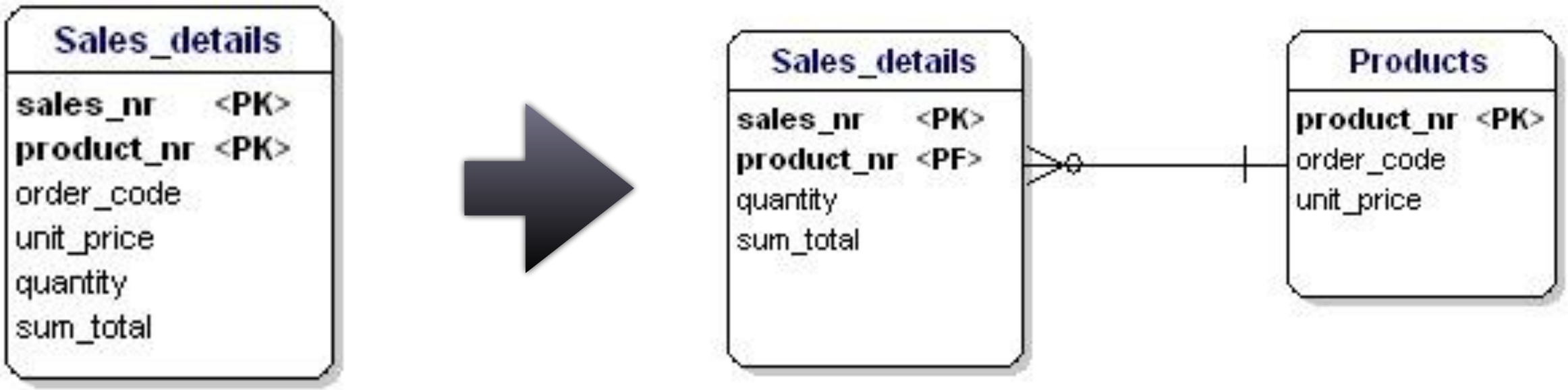
~~no repeating groups of columns in an entity~~

Normalization (II)

2nd normal form:

all attributes of an entity should be fully dependent on the whole primary key.

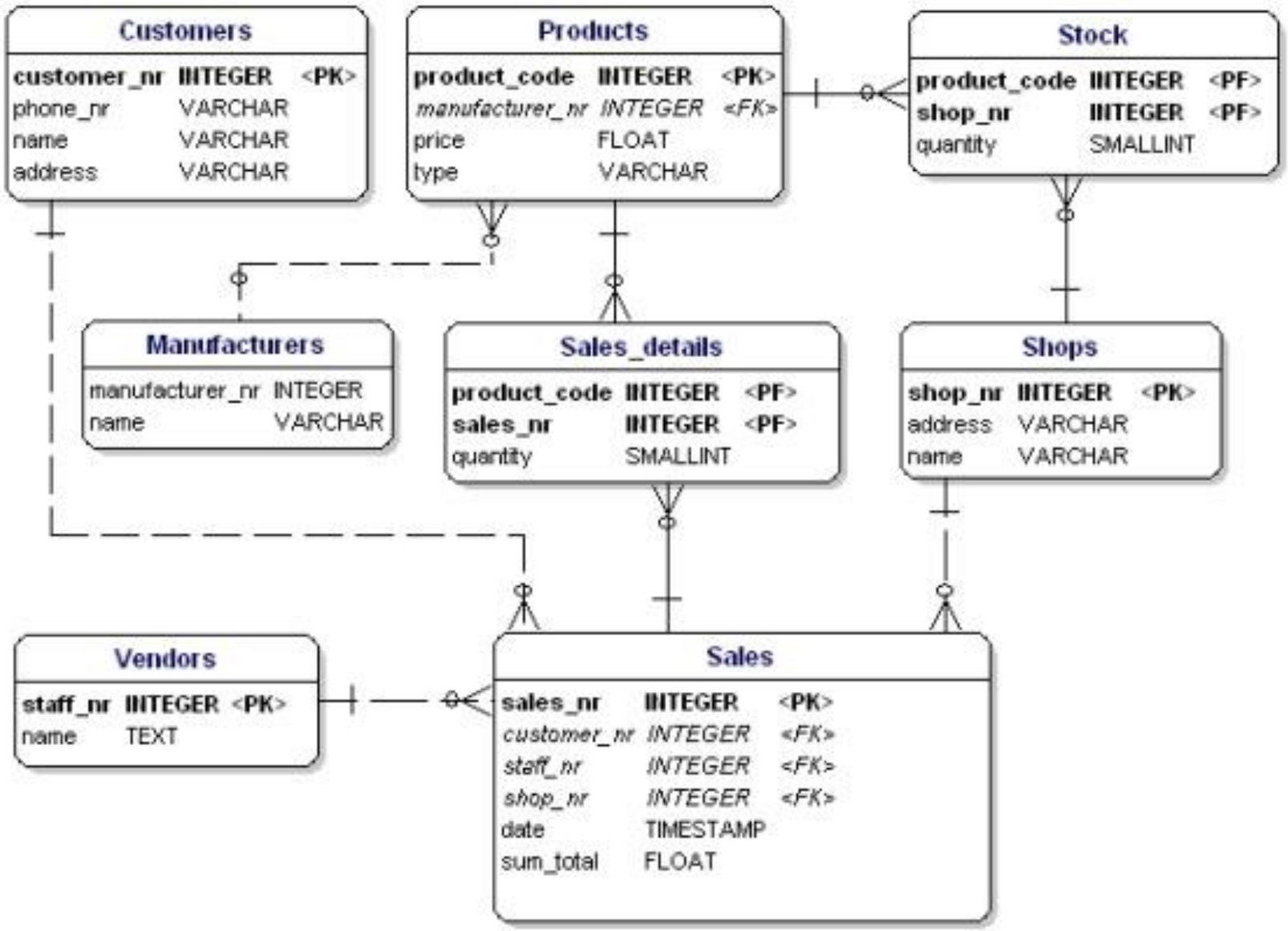
Normalization (III)



3rd normal form:

all attributes need to be directly dependent on the primary key, and not on other attributes.

Normalization (IV)



Data model in accordance with 1st, 2nd and 3d normal form.

Part 2

DB query and SQL

Source:
https://www.mysqltutorial.org/basic-mysql-tutorial.aspx

Quick Poll

dartgo.org/poll

When poll is active, respond at **PollEv.com/dartrc**
Text **DARTRC** to **37607** once to join

In a word or two, to you, what is a Database QUERY?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

What is a QUERY?

- A query is a question:
  - How many clients are named Paul?
  - What is the sales peoples' average sales sum in December?
- The answer is given in the form of a table

Accessing the DB

First, navigate to:

http://dartgo.org/pma

- Log in via DUO/SSO

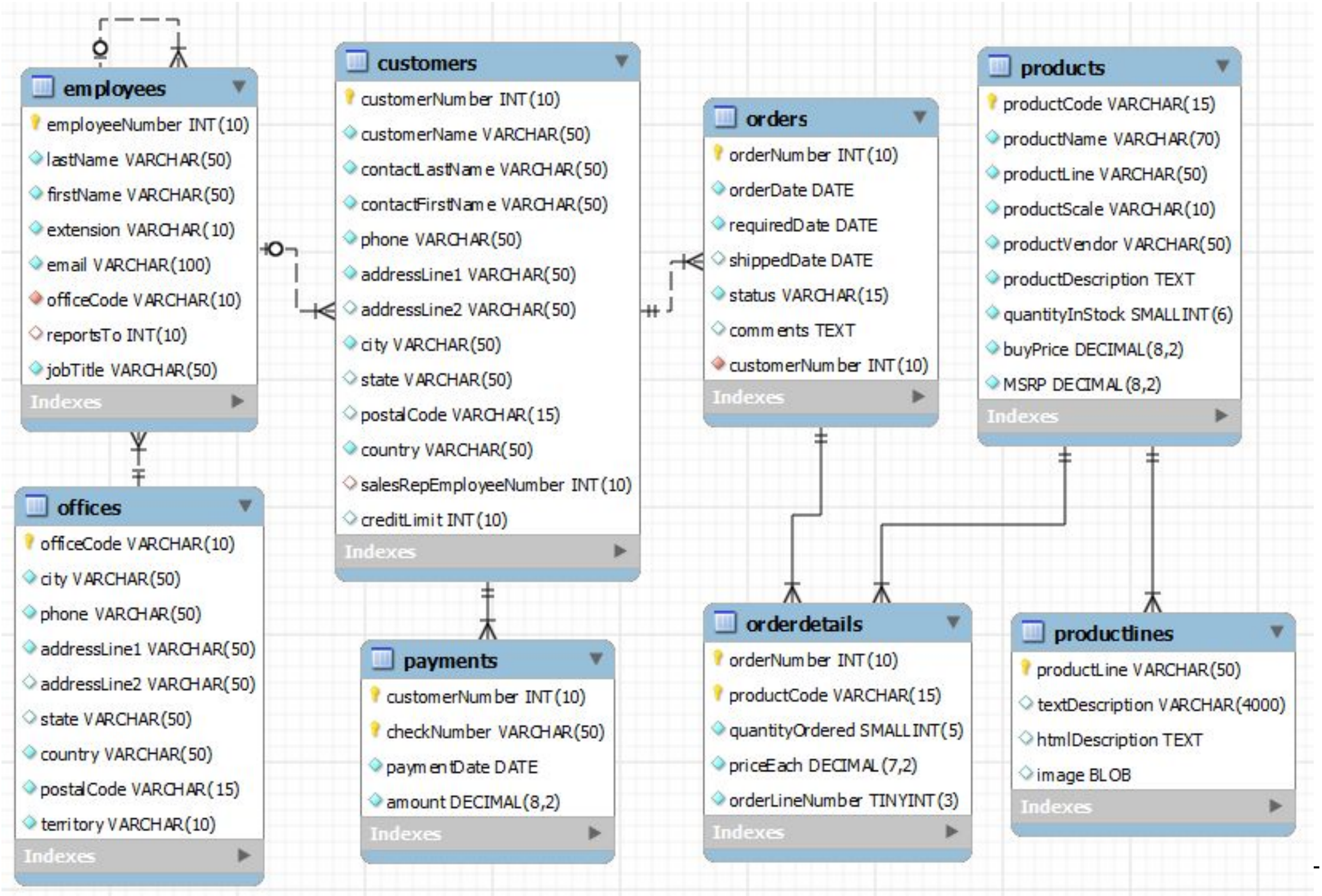
Then:

- username: **workshop**
  password: **learndb**

Case Study

- A toy classic car company keeps track of:
  - Employees, Offices and Customers
  - Orders and Payment methods
  - Products and Product Lines

Case Study DB



Case Study DB

Server: localhost:8889 » Database: retail_example

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers | Designer

	Table	Action						Rows	Type	Collation	Size	Overhead
☐	customers	☆	Browse	Structure	Search	Insert	Empty	Drop	122	InnoDB	latin1_swedish_ci	32 KiB	-
☐	employees	☆	Browse	Structure	Search	Insert	Empty	Drop	23	InnoDB	latin1_swedish_ci	48 KiB	-
☐	offices	☆	Browse	Structure	Search	Insert	Empty	Drop	7	InnoDB	latin1_swedish_ci	16 KiB	-
☐	orderdetails	☆	Browse	Structure	Search	Insert	Empty	Drop	2,996	InnoDB	latin1_swedish_ci	240 KiB	-
☐	orders	☆	Browse	Structure	Search	Insert	Empty	Drop	326	InnoDB	latin1_swedish_ci	64 KiB	-
☐	payments	☆	Browse	Structure	Search	Insert	Empty	Drop	273	InnoDB	latin1_swedish_ci	16 KiB	-
☐	productlines	☆	Browse	Structure	Search	Insert	Empty	Drop	7	InnoDB	latin1_swedish_ci	16 KiB	-
☐	products	☆	Browse	Structure	Search	Insert	Empty	Drop	110	InnoDB	latin1_swedish_ci	80 KiB	-
	8 tables	Sum							3,864	InnoDB	latin1_swedish_ci	512 KiB	0 B

Check All With selected:

- browse content, edit structure, search, insert, empty or drop
- feel free to click around and explore the UI

Browsing the Content

Using the UI:

- select the “classic_model_cars” database
- select the “employee” table
- click on the “Browse” tab

```
SELECT * FROM `employees`
```

UI Search Form

Using the UI's "Search" tab, search for:

- all employees with last name "Firrelli"
- all employees whose first name is NOT "Leslie"
- all employees whose job title contains "sale"

UI Search Form

Using the UI's "Search" tab, search for:

- all employees with last name "Firrelli"

```
SELECT * FROM `employees` WHERE `lastName` LIKE 'firrelli'
```

- all employees whose first name is NOT "Leslie"

```
SELECT * FROM `employees` WHERE `firstName` NOT LIKE 'leslie'
```

- all employees whose job title contains "sale"

```
SELECT * FROM `employees` WHERE `jobTitle` LIKE '%sale%'
```

UI Limitations

The search form in the UI is limited:

- one table at a time
- one value at a time
- no arithmetic
- no grouping

SQL Query in UI

- at the database level
  - select the database
  - SQL (or Query tab for more advanced users)
- at the table level
  - select the database
  - select the table
  - SQL tab
  - hit “SELECT *” button, if table name is missing

SELECT statement

- The SELECT statement controls which columns and rows that you want to see of the tables specified in the FROM section of the statement.

- The result(s) of a SELECT statement is always a table

- SELECT * shows ALL the columns

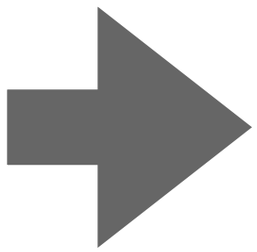
```
SELECT
    lastname, firstname, jobtitle
FROM
    employees;
```

```
SELECT * FROM employees;
```

Eliminate Duplicates

In order to remove these duplicate rows, you use the DISTINCT clause in the SELECT statement.

```
1 SELECT
2     lastname
3 FROM
4     employees
5
```



```
1 SELECT DISTINCT
2     lastname
3 FROM
4     employees
5
```

You can use the DISTINCT clause with more than one column.

```
1 SELECT DISTINCT
2     state, city
3 FROM
4     customers
```

Quick Poll

dartgo.org/poll

What cities do we have offices in?

Top

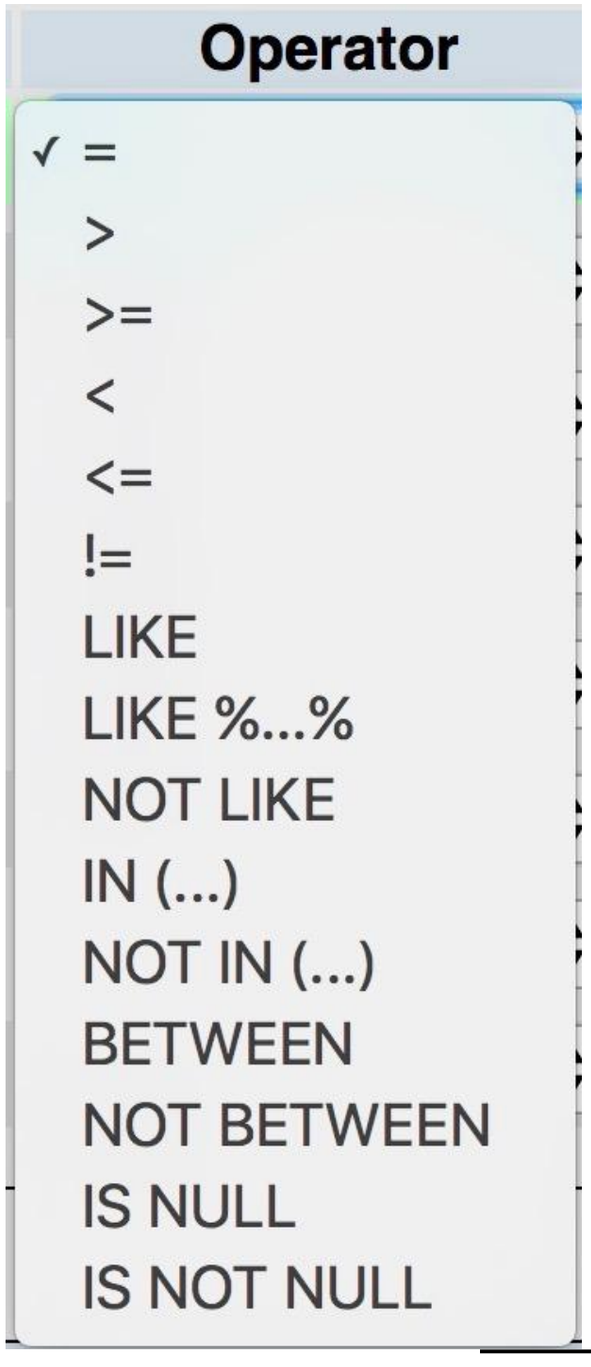
What query did you run to answer the question: What cities do we have offices in?

Top

Solution

- SELECT DISTINCT city FROM offices

Comparison Operators



= : equals

> : greater than

>= : greater than or equals

< : smaller than

<= : smaller than or equals

!= : not equals

LIKE / NOT LIKE: case insensitive comparison

LIKE %...% : contains

IN / NOT IN (...) : equals one of the values in (...)

BETWEEN / NOT BETWEEN: between 2 values

IS NULL / IS NOT NULL : value is “NULL”

Filterning

The WHERE clause allows you to specify exact rows to select based on a particular filtering expression or condition.

```sql
SELECT
    lastname, firstname, jobtitle
FROM
    employees
WHERE
    jobtitle = 'Sales Rep';
```

Logical Operators

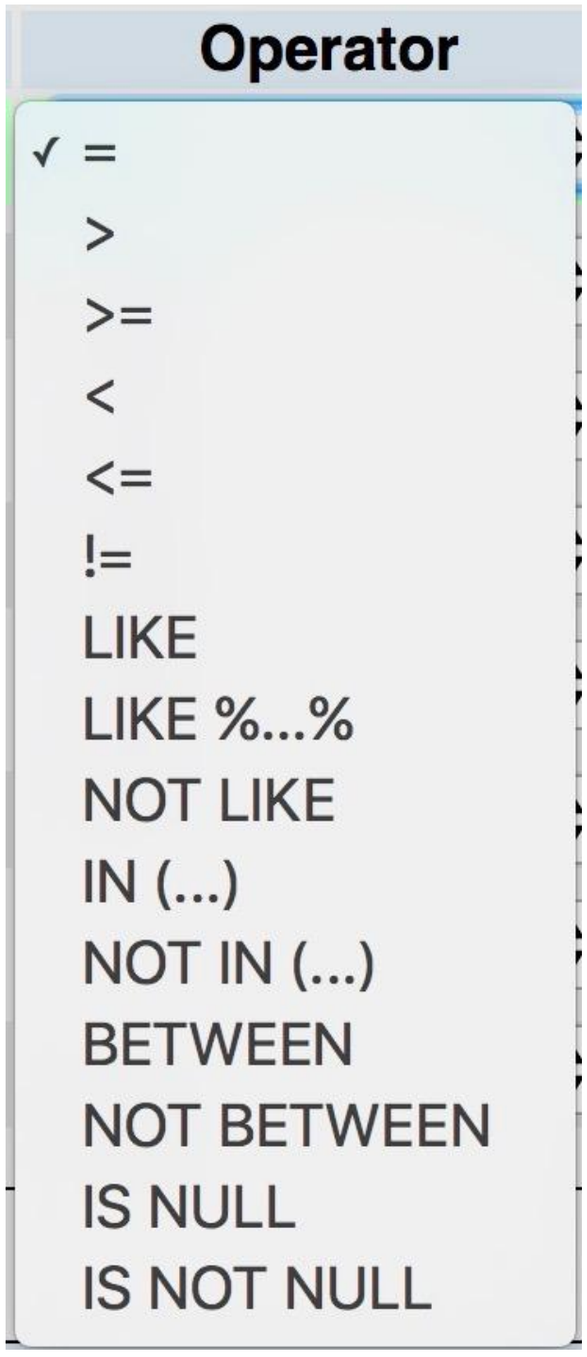
You can have more than one condition by using logical operator like AND and OR.

AND : both conditions have to be satisfied

OR : at least one condition has to be satisfied

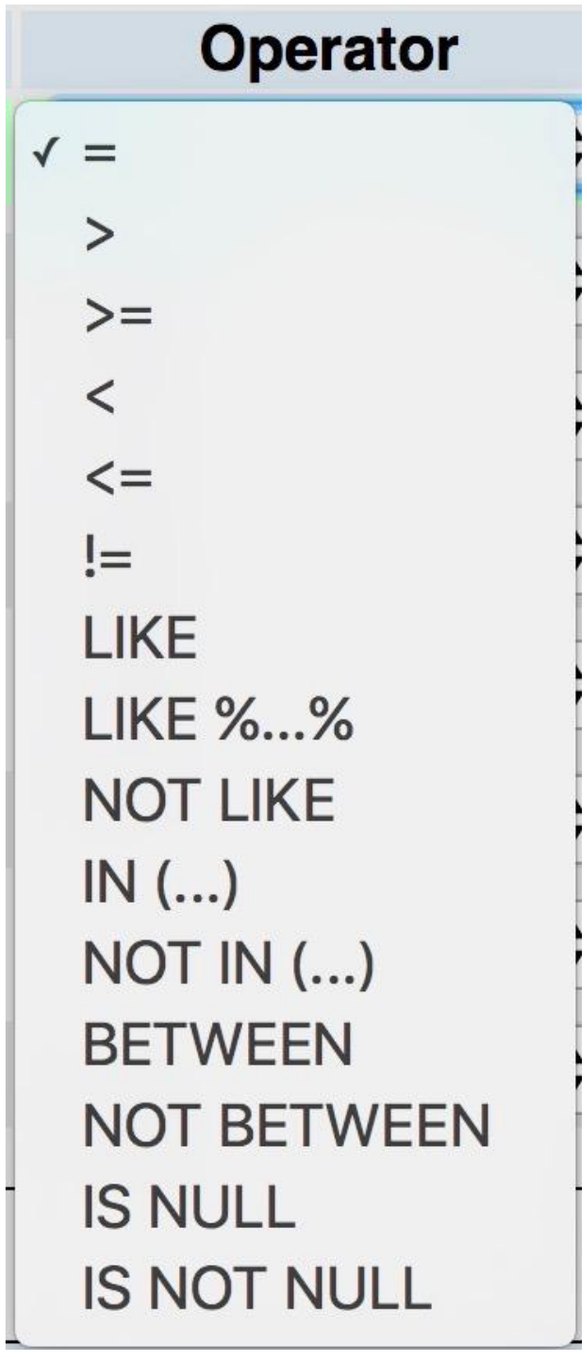
```sql
SELECT
    lastname, firstname, jobtitle
FROM
    employees
WHERE
    jobtitle = 'Sales Rep' AND officeCode = 1;
```

Get Dirty



- list all the unique employee first names
- who reports to employee #1102?
- which sales rep report to #1088?
- whose phone extension starts with a 4?
- whose phone extension contains a 3 or a 5?

Get Dirty



- list all the unique employee first names

```
SELECT DISTINCT firstName FROM `employees`
```

- who reports to employee #1102?

```
SELECT * FROM employees WHERE reportsTo = 1102
```

- which sales rep report to #1088?

```
SELECT * FROM employees WHERE reportsTo = 1088
AND jobTitle LIKE 'sales rep'
```

- whose phone extension starts with a 4?

```
SELECT lastname, extension FROM employees
WHERE extension LIKE 'x4%'
```

- whose phone extension contains a 3 or a 5

```
SELECT lastname, firstname, extension FROM employees
WHERE extension LIKE '%3%' OR extension LIKE '%5%'
```

Sorting

When you use the SELECT statement to query data from a table, the result set is not sorted in any orders. To sort the result set, you use the ORDER BY clause. The ORDER BY clause allows you to:

- Sort a result set by a single column or multiple columns.
- Sort a result set by different columns in ascending (ASC) or descending order (DESC).

Sorting

```sql
SELECT
 contactLastname,
 contactFirstname
FROM
 customers
ORDER BY
 contactLastname;
```

```sql
SELECT
 contactLastname,
 contactFirstname
FROM
 customers
ORDER BY
 contactLastname DESC;
```

```sql
SELECT
 contactLastname,
 contactFirstname
FROM
 customers
ORDER BY
 contactLastname DESC,
 contactFirstname ASC;
```

Aliasing

- To give a column a descriptive name, you use a column alias.

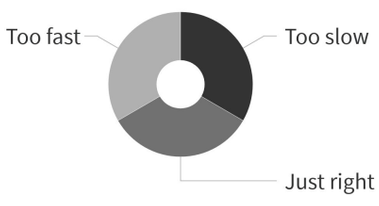
```
SELECT
 CONCAT_WS(', ', lastName, firstname) `Full name`
FROM
 employees
ORDER BY
 `Full name`;
```

Quick Poll

dartgo.org/poll

How's the pace of this workshop?

Too slow **A** Just right **B** Too fast **C**



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Grouping

- The GROUP BY clause, which is an optional part of the SELECT statement, groups a set of rows into a set of summary rows by values of columns or expressions. The GROUP BY clause returns one row for each group.

- We often use the GROUP BY clause with aggregate functions such as SUM, AVG, MAX, MIN, and COUNT. The aggregate function that appears in the SELECT clause provides the information about each group.

Aggregate functions

Name	Description
AVG()	Return the average value of the argument
BIT_AND()	Return bitwise AND
BIT_OR()	Return bitwise OR
BIT_XOR()	Return bitwise XOR
COUNT()	Return a count of the number of rows returned
COUNT(DISTINCT)	Return the count of a number of different values
GROUP_CONCAT()	Return a concatenated string
MAX()	Return the maximum value
MIN()	Return the minimum value
STD()	Return the population standard deviation
STDDEV()	Return the population standard deviation
STDDEV_POP()	Return the population standard deviation
STDDEV_SAMP()	Return the sample standard deviation
SUM()	Return the sum
VAR_POP()	Return the population standard variance
VAR_SAMP()	Return the sample variance
VARIANCE()	Return the population standard variance

GROUP BY

```sql
SELECT
    status, COUNT(*)
FROM
    orders
GROUP BY status;
```

```sql
SELECT
    orderNumber,
    SUM(quantityOrdered * priceEach) AS total
FROM
    orderdetails
GROUP BY orderNumber;
```

Quick Poll

dartgo.org/poll

How many cities do we have offices in?

Top

What query did you run to answer the question: How many cities do we have offices in?

Top

Name
AVG()
BIT_AND()
BIT_OR()
BIT_XOR()
COUNT()
COUNT(DISTINCT)
GROUP_CONCAT()
MAX()
MIN()
STD()
STDDEV()
STDDEV_POP()
STDDEV_SAMP()
SUM()
VAR_POP()
VAR_SAMP()
VARIANCE()

Get Dirty

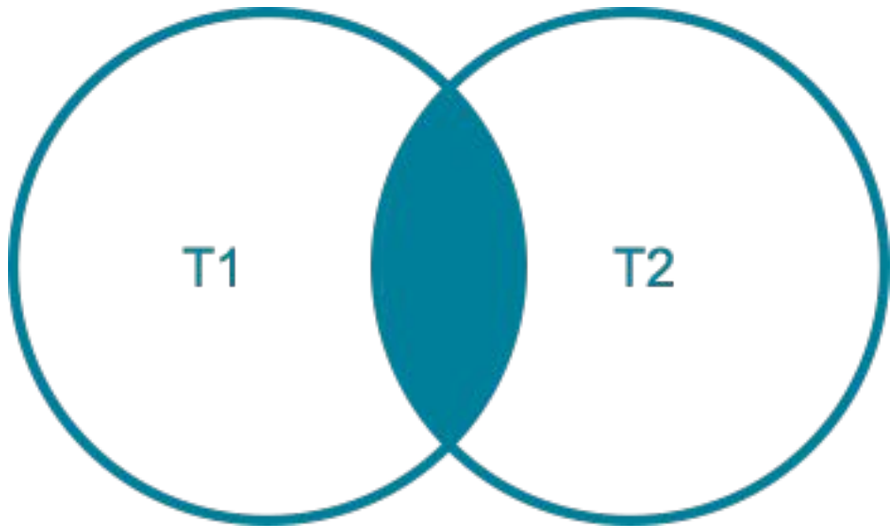
- the total quantity and average item price for each order
- the number, in descending order, of different product for each order
- the cheapest product
- how many products are in the “Vintage Cars” product line

Name
AVG()
BIT_AND()
BIT_OR()
BIT_XOR()
COUNT()
COUNT(DISTINCT)
GROUP_CONCAT()
MAX()
MIN()
STD()
STDDEV()
STDDEV_POP()
STDDEV_SAMP()
SUM()
VAR_POP()
VAR_SAMP()
VARIANCE()

Get Dirty

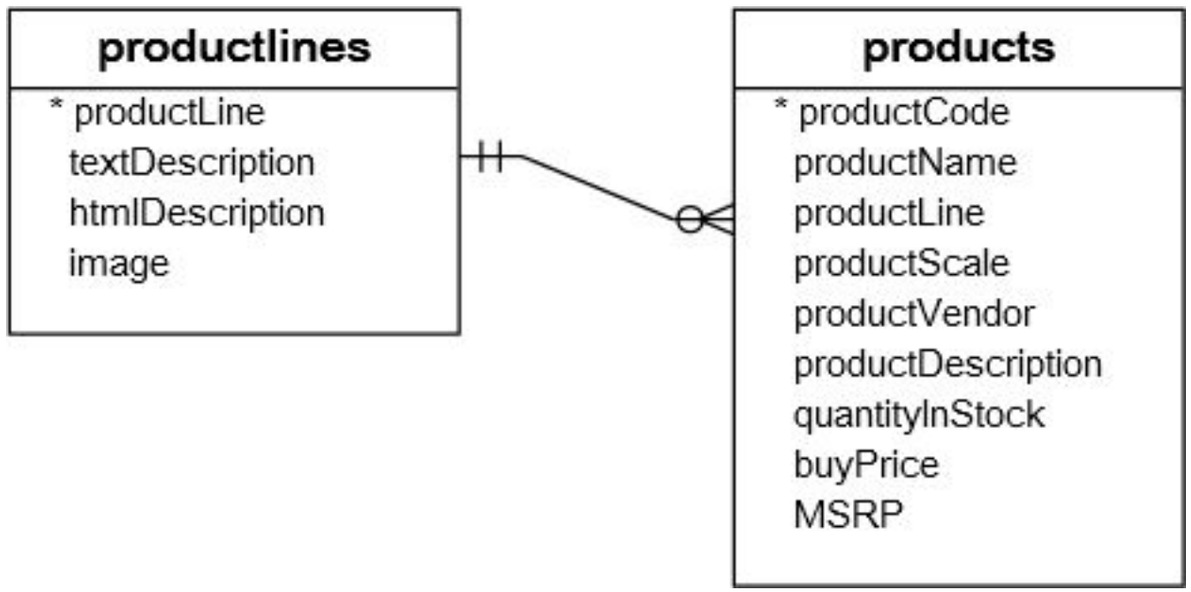
- ```
  SELECT orderNumber, SUM( quantityOrdered ) , AVG( priceEach )
  FROM orderdetails
  GROUP BY orderNumber
  LIMIT 0 , 30
  ```
- ```
  SELECT orderNumber, COUNT( * ) AS nbProducts
  FROM orderdetails
  GROUP BY orderNumber
  ORDER BY nbProducts DESC
  ```
- ```
  SELECT *
  FROM `products`
  ORDER BY MSRP
  LIMIT 1
  ```
  st product
- ```
  SELECT COUNT( * )
  FROM `products`
  WHERE productLine LIKE 'vintage cars'
  GROUP BY productLine
  ```
  the “Vintage

Joining



- The MySQL INNER JOIN clause matches rows in one table with rows in other tables and allows you to query rows that contain columns from both tables.

Join example 1



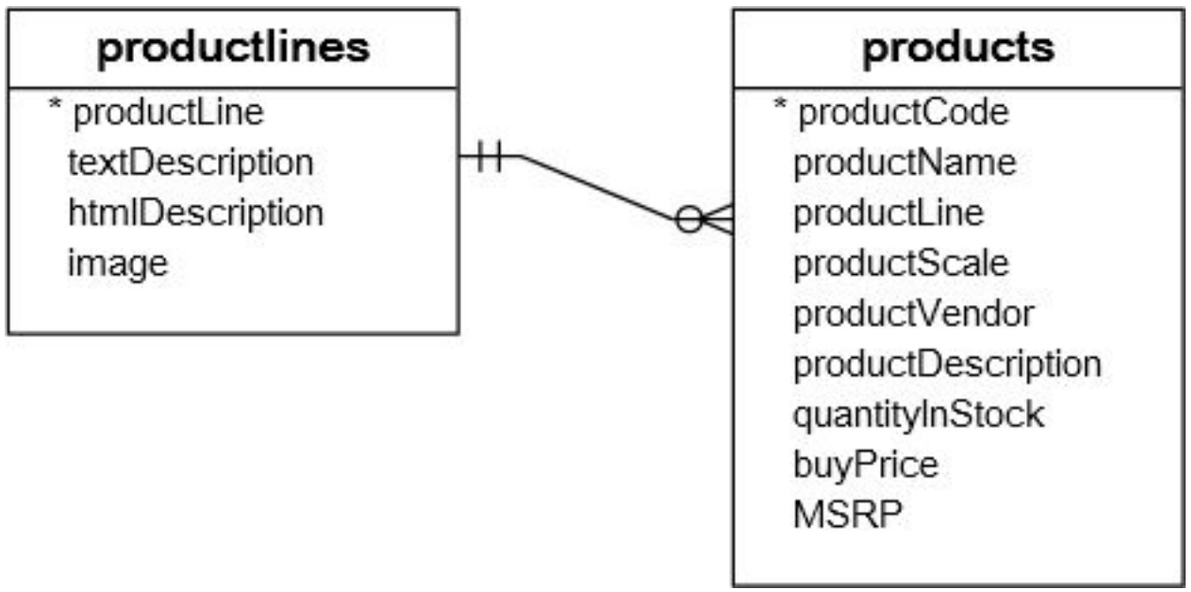
How to get…

- The product code and product name from the products table.

AND

- The text description of product lines from the productlines table.

Join example 1




```sql
SELECT
 productCode,
 productName,
 textDescription
FROM
 products T1
INNER JOIN productlines T2 ON T1.productline = T2.productline;
```

How to get…

- The product code and product name from the products table.

AND

- The text description of product lines from the productlines table.

Join example 1

	productCode	productName	textDescription
	S10_1949	1952 Alpine Renault 1300	Attention car enthusiasts: Make your wildest car ownership dreams come true.
	S10_4757	1972 Alfa Romeo GTA	Attention car enthusiasts: Make your wildest car ownership dreams come true.
	S10_4962	1962 LanciaA Delta 16V	Attention car enthusiasts: Make your wildest car ownership dreams come true.
	S12_1099	1968 Ford Mustang	Attention car enthusiasts: Make your wildest car ownership dreams come true.
	S12_1108	2001 Ferrari Enzo	Attention car enthusiasts: Make your wildest car ownership dreams come true.

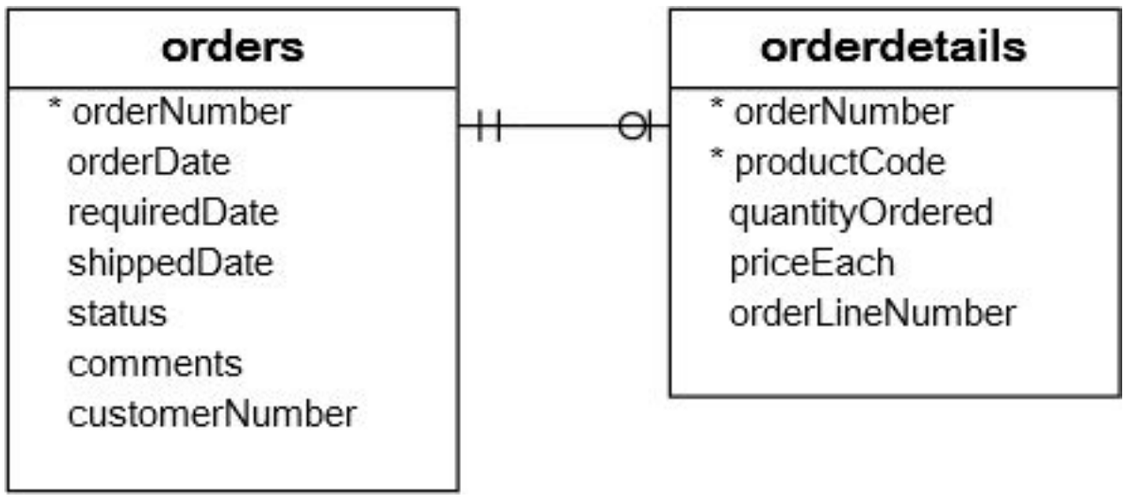
How to get…

- The product code and product name from the products table.

AND

- The text description of product lines from the productlines table.

Join example 2



- We can get the order number, order status and total sales from the *orders* and *orderdetails* tables using the INNER JOIN clause with the GROUP BY clause as follows:

```sql
SELECT
 T1.orderNumber,
 STATUS,
 SUM(quantityOrdered * priceEach) total
FROM
 orders AS T1
INNER JOIN orderdetails AS T2 ON T1.orderNumber = T2.orderNumber
GROUP BY
 orderNumber;
```

	orderNumber	status	total
	10100	Shipped	10223.83
	10101	Shipped	10549.01
	10102	Shipped	5494.78
	10103	Shipped	50218.95
	10104	Shipped	40206.20

Get Dirty

- get a list of employees names and the city of their office. First and last name may remain separate columns.

full name	city
Diane Murphy	San Francisco
Mary Patterson	San Francisco
Jeff Firrelli	San Francisco
Anthony Bow	San Francisco
Leslie Jennings	San Francisco
Leslie Thompson	San Francisco
Julie Firrelli	Boston
Steve Patterson	Boston
Foon Yue Tseng	NYC
George Vanauf	NYC
Gerard Bondur	Paris
Loui Bondur	Paris
Gerard Hernandez	Paris
Pamela Castillo	Paris
Martin Gerard	Paris
Mami Nishi	Tokyo
Yoshimi Kato	Tokyo
William Patterson	Sydney
Andy Fixter	Sydney
Peter Marsh	Sydney
Tom King	Sydney
Larry Bott	London
Barry Jones	London

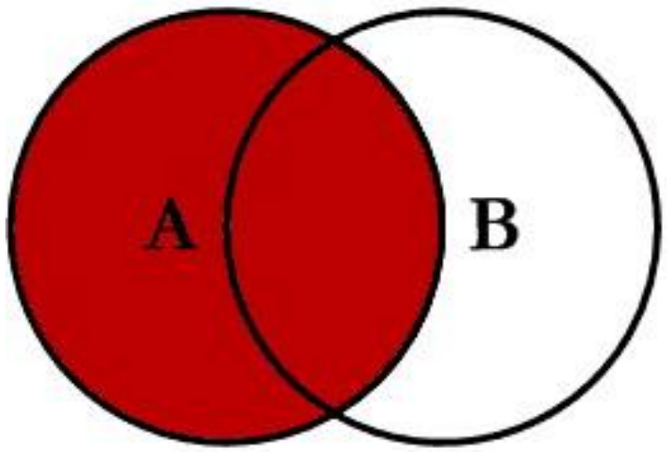
Get Dirty

- get a list of employees names and the city of their office

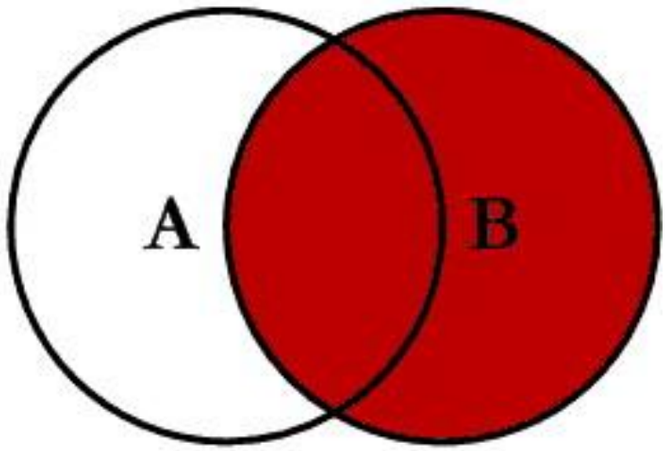
```
SELECT CONCAT_WS( ' ', firstName, lastName ) AS 'full name', city
FROM `employees` T1
INNER JOIN offices T2 ON T1.officeCode = T2.officeCode
```

full name	city
Diane Murphy	San Francisco
Mary Patterson	San Francisco
Jeff Firrelli	San Francisco
Anthony Bow	San Francisco
Leslie Jennings	San Francisco
Leslie Thompson	San Francisco
Julie Firrelli	Boston
Steve Patterson	Boston
Foon Yue Tseng	NYC
George Vanauf	NYC
Gerard Bondur	Paris
Loui Bondur	Paris
Gerard Hernandez	Paris
Pamela Castillo	Paris
Martin Gerard	Paris
Mami Nishi	Tokyo
Yoshimi Kato	Tokyo
William Patterson	Sydney
Andy Fixter	Sydney
Peter Marsh	Sydney
Tom King	Sydney
Larry Bott	London
Barry Jones	London

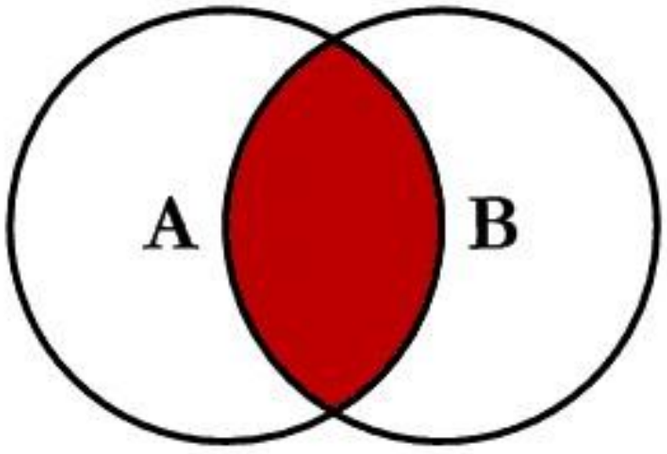
SQL JOINS



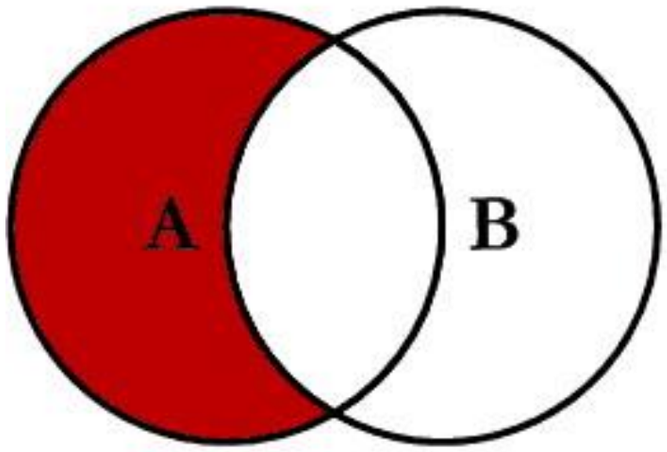

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



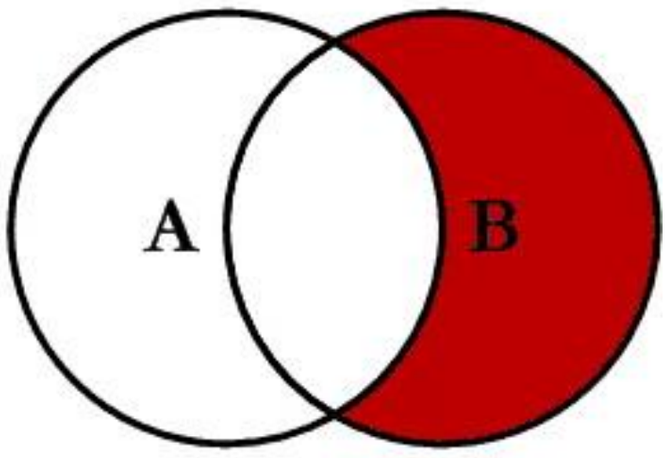

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



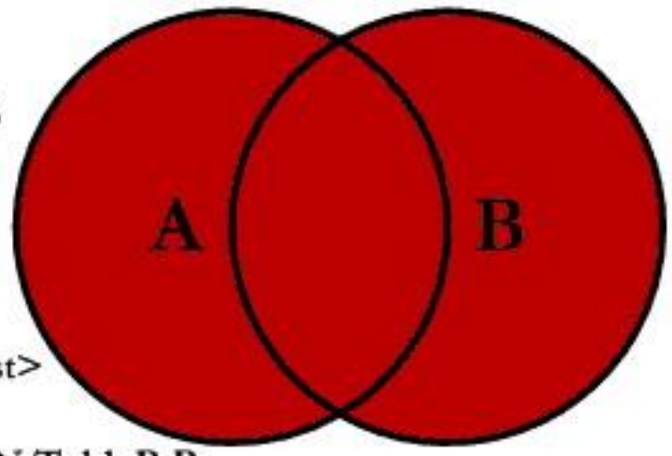

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



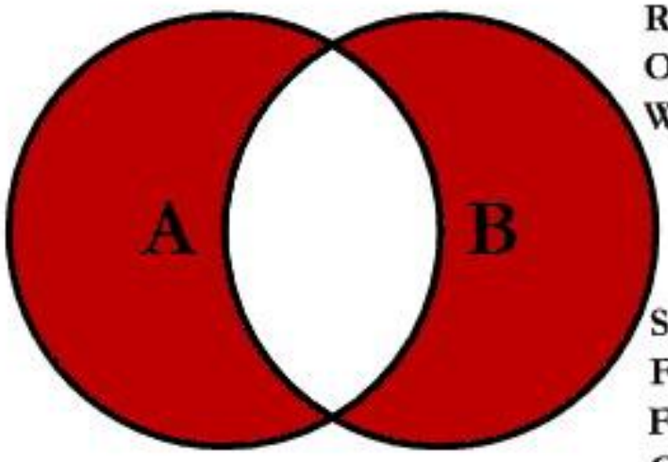

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




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




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




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

String Functions

- https://dev.mysql.com/doc/refman/5.7/en/string-functions.html

Name	Description
ASCII()	Return numeric value of left-most character
BIN()	Return a string containing binary representation of a number
BIT_LENGTH()	Return length of argument in bits
CHAR()	Return the character for each integer passed
CHAR_LENGTH()	Return number of characters in argument
CHARACTER_LENGTH()	Synonym for CHAR_LENGTH()
CONCAT()	Return concatenated string
CONCAT_WS()	Return concatenate with separator
ELT()	Return string at index number
EXPORT_SET()	Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
FIELD()	Return the index (position) of the first argument in the subsequent arguments
FIND_IN_SET()	Return the index position of the first argument within the second argument
FORMAT()	Return a number formatted to specified number of decimal places
FROM_BASE64()	Decode to a base-64 string and return result
HEX()	Return a hexadecimal representation of a decimal or string value
INSERT()	Insert a substring at the specified position up to the specified number of characters
INSTR()	Return the index of the first occurrence of substring
LCASE()	Synonym for LOWER()
LEFT()	Return the leftmost number of characters as specified
LENGTH()	Return the length of a string in bytes
LIKE	Simple pattern matching
LOAD_FILE()	Load the named file
LOCATE()	Return the position of the first occurrence of substring
LOWER()	Return the argument in lowercase
LPAD()	Return the string argument, left-padded with the specified string
LTRIM()	Remove leading spaces
MAKE_SET()	Return a set of comma-separated strings that have the corresponding bit in bits set
MATCH	Perform full-text search
MID()	Return a substring starting from the specified position
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP
OCT()	Return a string containing octal representation of a number
OCTET_LENGTH()	Synonym for LENGTH()
ORD()	Return character code for leftmost character of the argument
POSITION()	Synonym for LOCATE()
QUOTE()	Escape the argument for use in an SQL statement
REGEXP	Pattern matching using regular expressions
REPEAT()	Repeat a string the specified number of times
REPLACE()	Replace occurrences of a specified string
REVERSE()	Reverse the characters in a string
RIGHT()	Return the specified rightmost number of characters
RLIKE	Synonym for REGEXP
RPAD()	Append string the specified number of times
RTRIM()	Remove trailing spaces
SOUNDEX()	Return a soundex string
SOUNDS LIKE	Compare sounds
SPACE()	Return a string of the specified number of spaces
STRCMP()	Compare two strings
SUBSTR()	Return the substring as specified
SUBSTRING()	Return the substring as specified
SUBSTRING_INDEX()	Return a substring from a string before the specified number of occurrences of the delimiter
TO_BASE64()	Return the argument converted to a base-64 string
TRIM()	Remove leading and trailing spaces
UCASE()	Synonym for UPPER()
UNHEX()	Return a string containing hex representation of a number
UPPER()	Convert to uppercase
WEIGHT_STRING()	Return the weight string for a string

Get Dirty

- get a list of employees names and the city of their office
- and the number of customers they work with
- order by the descending nb of customers

full name	city	nb customers
Pamela Castillo	Paris	10
Barry Jones	London	9
Larry Bott	London	8
George Vanauf	NYC	8
Gerard Hernandez	Paris	7
Foon Yue Tseng	NYC	7
Leslie Thompson	San Francisco	6
Loui Bondur	Paris	6
Steve Patterson	Boston	6
Leslie Jennings	San Francisco	6
Martin Gerard	Paris	6
Julie Firrelli	Boston	6
Mami Nishi	Tokyo	5
Peter Marsh	Sydney	5
Andy Fixter	Sydney	5

Get Dirty

- get a list of employees names and the city of their office
- and the number of customers they work with
- order by the descending nb of customers

full name	city	nb customers
Pamela Castillo	Paris	10
Barry Jones	London	9
Larry Bott	London	8
George Vanauf	NYC	8
Gerard Hernandez	Paris	7
Foon Yue Tseng	NYC	7
Leslie Thompson	San Francisco	6
Loui Bondur	Paris	6
Steve Patterson	Boston	6
Leslie Jennings	San Francisco	6
Martin Gerard	Paris	6
Julie Firrelli	Boston	6
Mami Nishi	Tokyo	5
Peter Marsh	Sydney	5
Andy Fixter	Sydney	5

```
SELECT CONCAT_WS( ' ', firstName, lastName ) AS 'full name', T2.city, COUNT( DISTINCT T3.customerNumber ) AS 'nb customers'
FROM `employees` T1
INNER JOIN offices T2 ON T1.officeCode = T2.officeCode
INNER JOIN customers T3 ON T1.employeeNumber = T3.salesRepEmployeeNumber
GROUP BY T1.employeeNumber
ORDER BY `nb customers` DESC
```

Get Dirty

- get a list of employees names and the city of their office, the number of customers they work with, order by the descending nb of customers
- how many sales in 2004
- for how much total money

full name	city	nb customers	nb sales	total $$
Pamela Castillo	Paris	9	14	409910.07
Barry Jones	London	9	14	388872.38
Larry Bott	London	7	9	303470.32
George Vanauf	NYC	7	12	401758.60
Foon Yue Tseng	NYC	6	9	237255.26
Steve Patterson	Boston	6	10	327602.21
Gerard Hernandez	Paris	6	15	418367.27
Martin Gerard	Paris	5	7	207828.89
Leslie Jennings	San Francisco	5	10	291693.96
Julie Firrelli	Boston	5	7	129916.12
Loui Bondur	Paris	5	8	254002.97
Mami Nishi	Tokyo	4	6	151761.45
Peter Marsh	Sydney	4	7	247176.25
Leslie Thompson	San Francisco	4	6	185038.40
Andy Fixter	Sydney	3	5	172377.82

Get Dirty

- get a list of employees names and the city of their office, the number of customers they work with, order by the descending nb of customers

```
SELECT CONCAT_WS(' ', firstName,lastName) AS 'full name', T2.city, COUNT(DISTINCT
T3.customerNumber) AS 'nb customers', count(distinct T4.orderNumber) AS 'nb sales'
FROM `employees` T1
INNER JOIN offices T2 ON T1.officeCode = T2.officeCode
INNER JOIN customers T3 ON T1.employeeNumber = T3.salesRepEmployeeNumber
INNER JOIN orders T4 ON T3.customerNumber = T4.customerNumber
GROUP BY T1.employeeNumber
ORDER BY `nb customers` DESC
```

- and how many sales

```
WHERE T4.orderDate BETWEEN CAST('2004-01-01' AS DATE) AND CAST('2004-12-01' AS DATE)
```

- how many sales in 2004

```
1 SELECT CONCAT_WS(' ', firstName,lastName) AS 'full name', T2.city, COUNT(DISTINCT
  T3.customerNumber) AS 'nb customers', count(distinct T4.orderNumber),
  SUM(T5.priceEach*T5.quantityOrdered) AS 'total revenu'
2 FROM `employees` T1
3 INNER JOIN offices T2 ON T1.officeCode = T2.officeCode
4 INNER JOIN customers T3 ON T1.employeeNumber = T3.salesRepEmployeeNumber
5 INNER JOIN orders T4 ON T3.customerNumber = T4.customerNumber
6 INNER JOIN orderdetails T5 ON T4.ordernumber = T5.ordernumber
7 WHERE T4.orderDate BETWEEN CAST('2004-01-01' AS DATE) AND CAST('2004-12-01' AS DATE)
8 GROUP BY T1.employeeNumber
9 ORDER BY `nb customers` DESC
```

Insert, Update, Replace, Delete and So Much more...

- mysqltutorial.org — “Basic MySQL Tutorial”
- lynda.dartmouth.edu — search “SQL” or “Database”
- dev.mysql.com/doc/

Announcements

More RC workshops:

- https://rc.dartmouth.edu > Training

Support:

- christian.darabos@dartmouth.edu
- research.computing@dartmouth.edu