L02-L06: SQL

CS3200 Database design (sp18 s2)

1/11/2018

L01: SQL introduction

SQL overview

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!

SQL stands for
Structured Query Language

- Many standards out there:
  - ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….
  - Vendors support various subsets

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

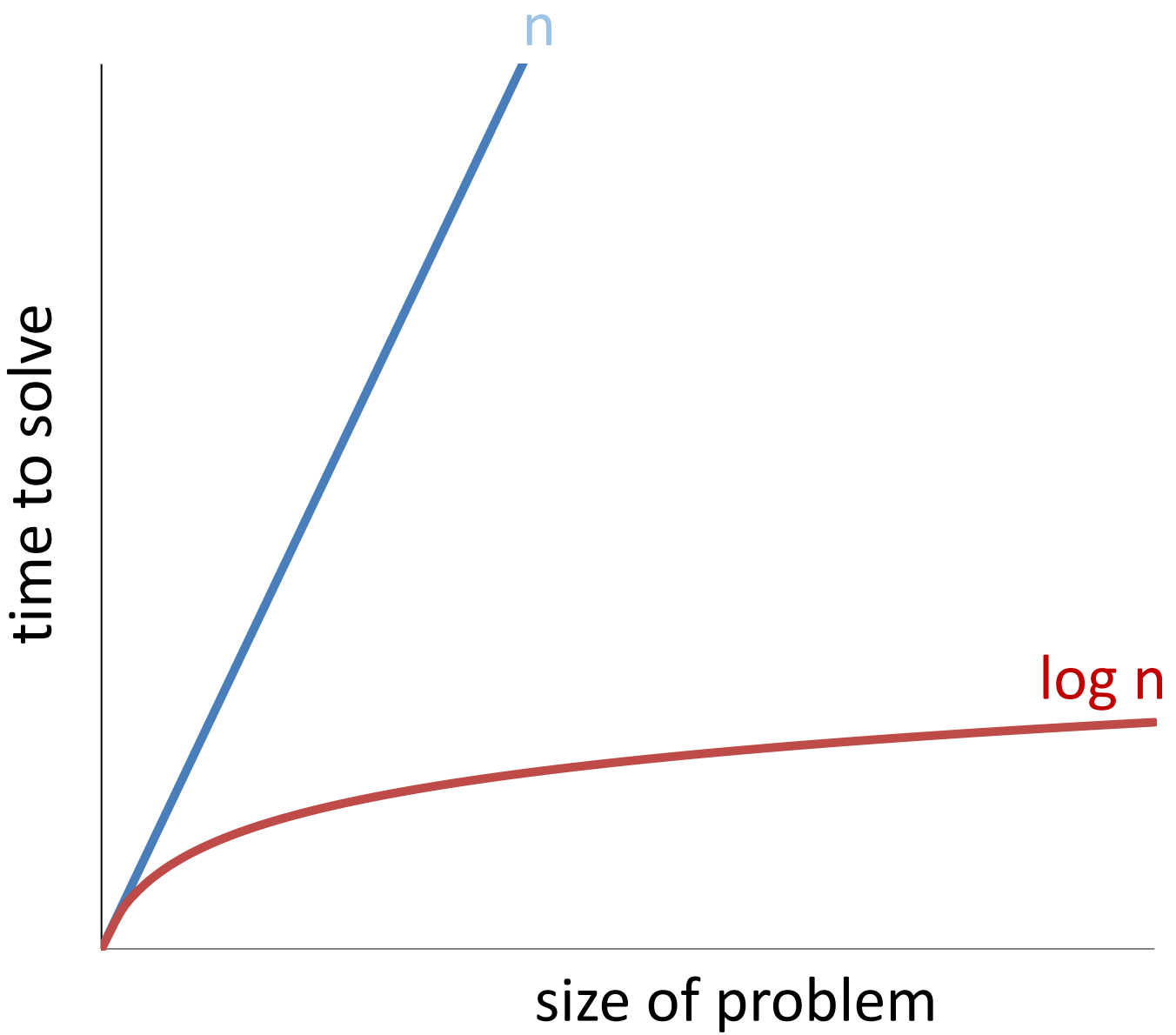
SQL Has Three Major Sub-Languages

- Data Definition Language (DDL)
  - Define a relational schema (create, alter, and drop tables; establish constraints
  - Create/alter/drop tables and their attributes
- Data Manipulation Language (DML)
  - Insert/delete/modify tuples in tables
  - Commands that maintain and query a database (our main focus!)
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data

An Algorithm

- Stand up and think of the number 1
- Pair off with someone standing, add your numbers together, and adopt the sum as your new number
- One of you should sit down; the other should go back to step 2

Scalability



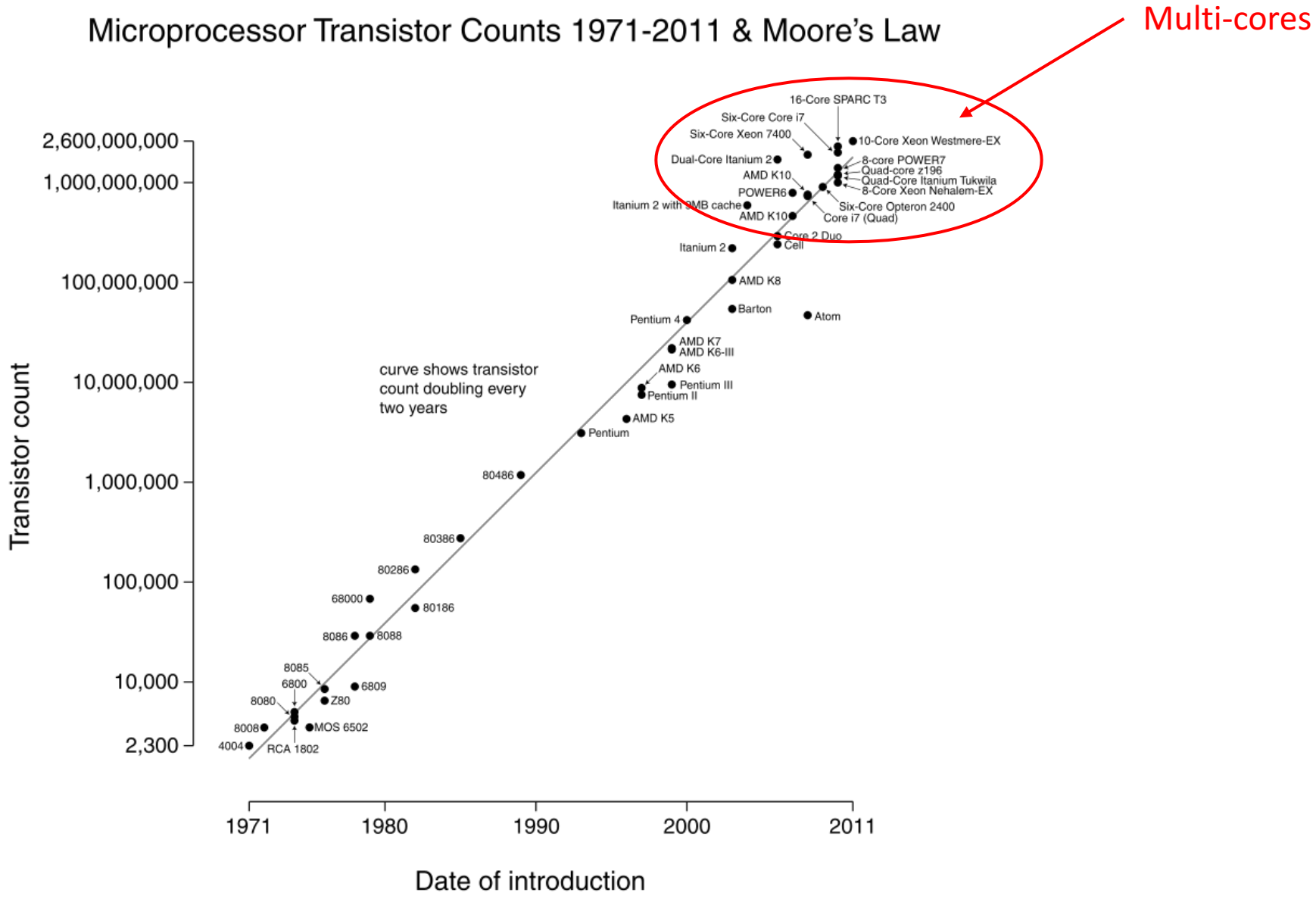
Most spectacular these days: theoretic potential for perfect scaling!

- perfect scaling
  - given sufficient resources, performance does not degrade as the database becomes larger
- key: parallel processing

- cost: number of processors polynomial in the size of the DB
  - remember our in-class counting exercise

- all (most) relational operators highly parallelisable

Moore's law



Source: http://en.wikipedia.org/wiki/Moore's_law

What is SQL?

The Positives

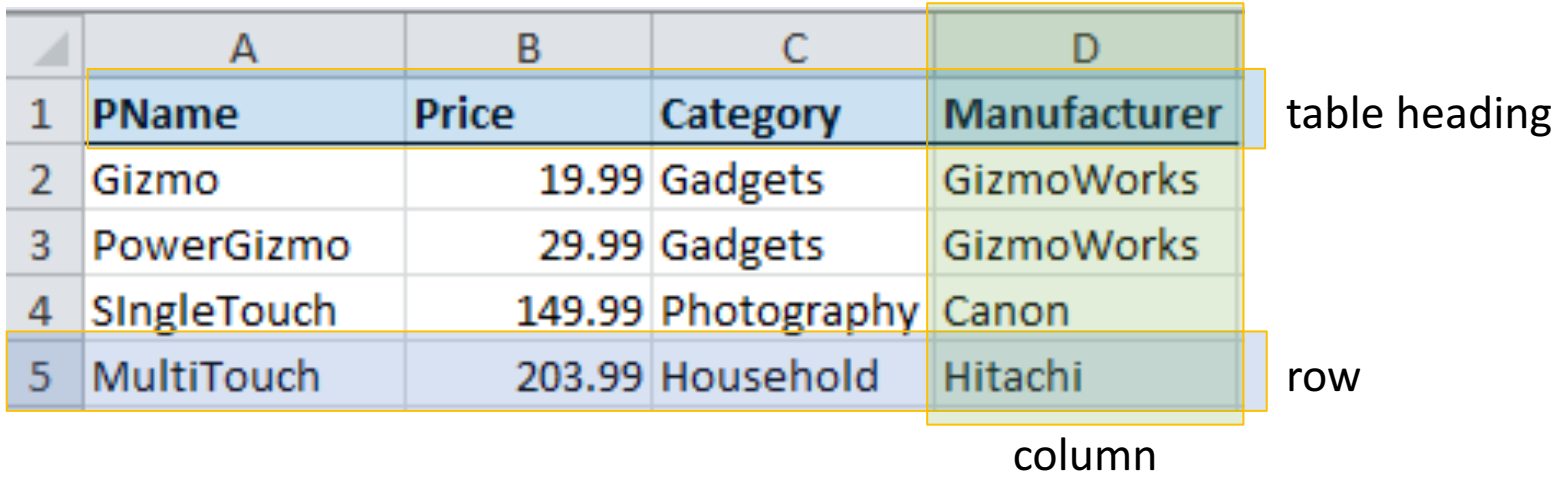
- It's a language (like English, Spanish, German, …)
- There are only a few key words that you have to learn – it's fairly simple
- It's major purpose is to communicate with a database and ask a database for data
- It's a declarative language (you define what to do)

The Challenges

- Simplicity has it's cost – it gets complex quickly
  - Imagine only having 2 verbs (go, put, wait) to express all you do in a lifetime
  - It's either infeasible or you have to combine a lot basic actions to construct a more complex action
    (e.g. skydiving = put parachute into backpack, put the backpack on your back, go airplane, wait until airplane is at 14k feet, go to open door, go outside airplane, …)
- Declarative programming is perceived as non-intuitive (well, decide for yourself ☺)

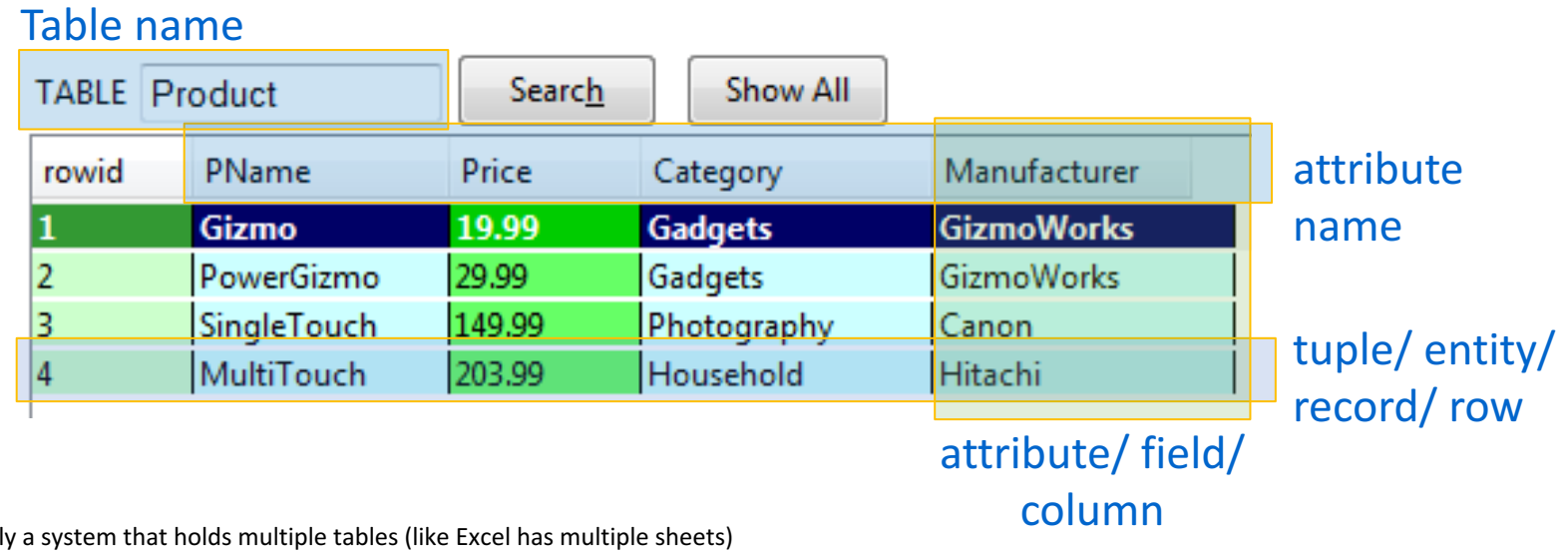
Different symantics between Excel and Database tables

Excel



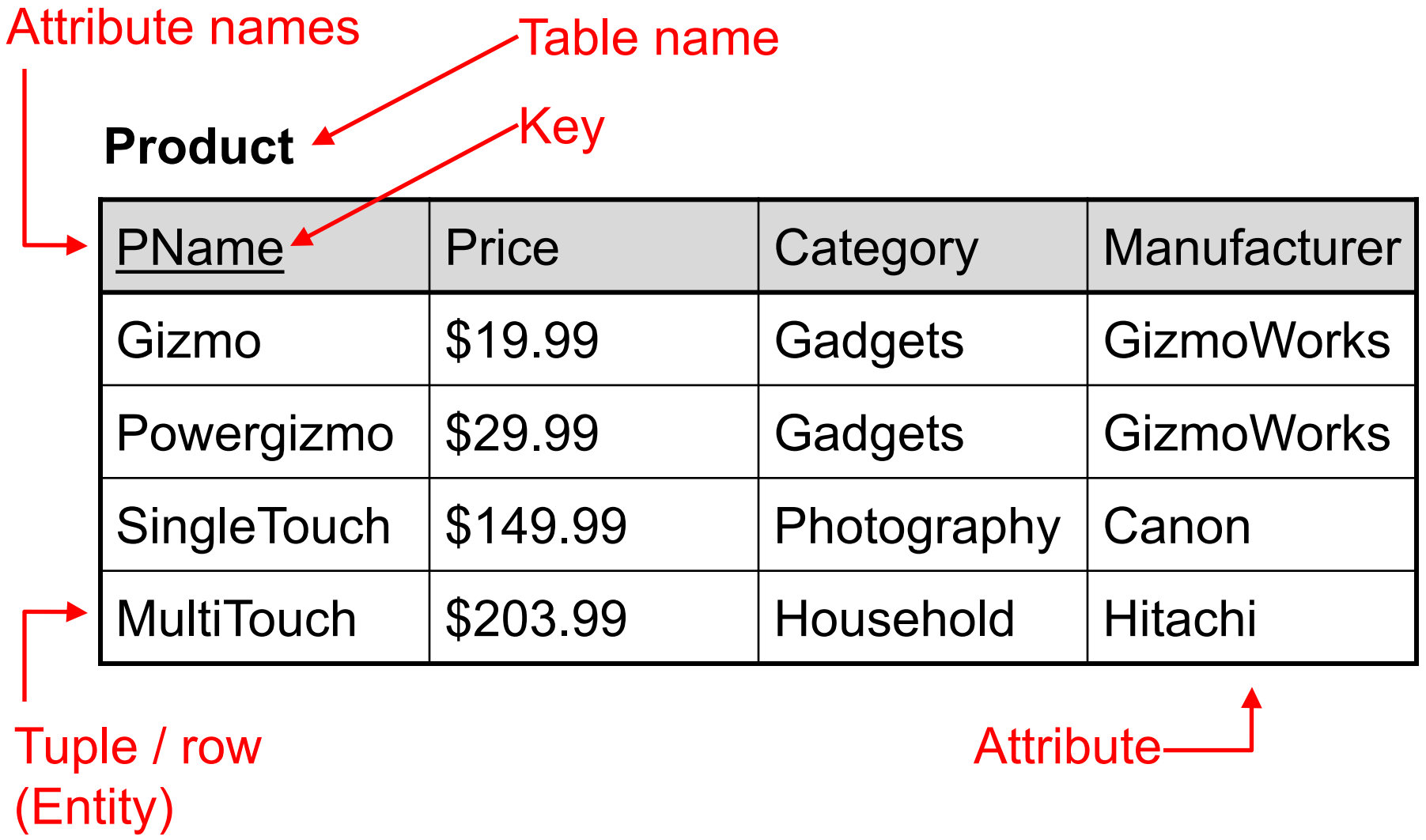
	A	B	C	D
1	**PName**	**Price**	**Category**	**Manufacturer**
2	Gizmo	19.99	Gadgets	GizmoWorks
3	PowerGizmo	29.99	Gadgets	GizmoWorks
4	SIngleTouch	149.99	Photography	Canon
5	MultiTouch	203.99	Household	Hitachi

Database[1]



[1] A Database (DB) is simply a system that holds multiple tables (like Excel has multiple sheets)

Tables in SQL



Data Types in SQL

- Atomic types
  - Character strings: CHAR(20), VARCHAR(50)
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATETIME, …
- Record (aka tuple)
  - Every attribute must have an atomic type
- Table (aka relation)
  - A set of tuples (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)
```

Basic SQL

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

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

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

Simple SQL Query

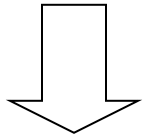
Our friend here shows that you can follow along in SQLite. Just install the database from the text file "300 - ..." available in our sql folder

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



Selection

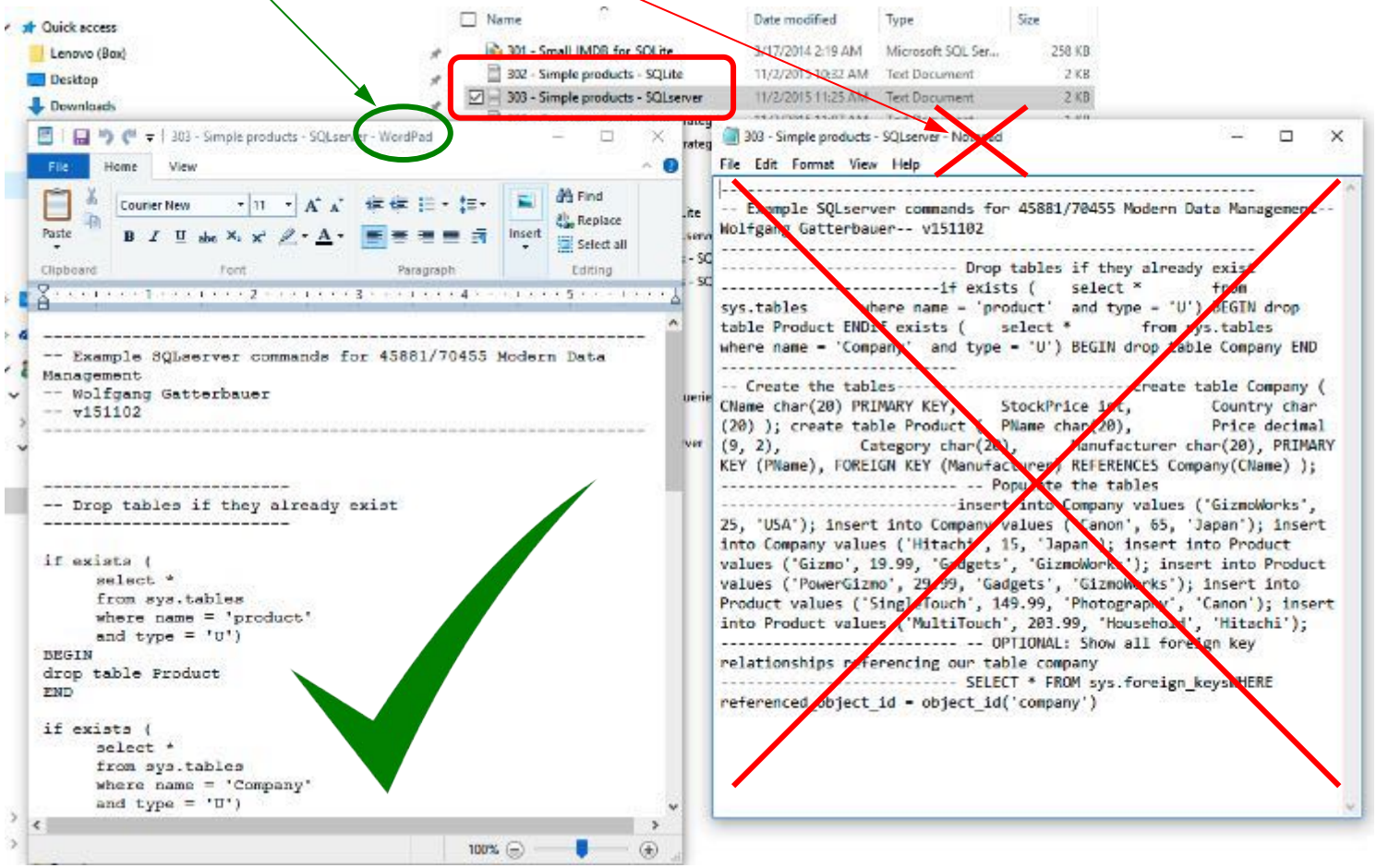
PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

Practice with your own local databases

302

If you are using Windows:

1. *Download the appropriate text files from our repository*
2. *Open them with "**Wordpad**" (not "**Notepad**" which messes up the text!)*
3. *Paste the SQL commands into your SQLite version, and execute*

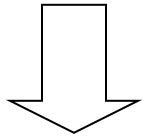


Simple SQL Query

Product

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

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



PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Selection
& Projection

Selection vs. Projection

One **projects** onto some attributes (columns)
-> happens in the **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
FROM    Product
WHERE   price > 100
```

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

PName	Price
SingleTouch	$149.99
MultiTouch	$203.99

A Few Details

- SQL commands are case insensitive:
  - SELECT = Select = select
  - Product = product

- But values are not:
  - Different: 'Boston', 'boston'
  - (Notice: in general, but default settings will vary from DBMS to DBMS. Just to be safe, always assume values to be case sensitive!)

- Use single quotes for constants:
  - 'abc' - yes
  - "abc" - no

Eliminating Duplicates

Product

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

Set vs. Bag semantics

```
SELECT  category
FROM    Product
```

Category
Gadgets
Gadgets
Photography
Household

```
SELECT  DISTINCT category
FROM    Product
```

Category
Gadgets
Photography
Household

Ordering the Results


```
SELECT   pName, price, manufacturer
FROM     Product
WHERE    category='Gadgets'
     and price > 10
ORDER BY price, pName
```

- Ties in attribute *price* broken by attribute *pname*
- Ordering is ascending by default. Descending:

```
... ORDER BY price ASC, pname DESC
```

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  DISTINCT category
FROM    Product
ORDER BY category
```

?

```
SELECT  category
FROM    Product
ORDER BY pName
```

?

```
SELECT  DISTINCT category
FROM    Product
ORDER BY pName
```

?

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  DISTINCT category
FROM    Product
ORDER BY category
```

Category
Gadgets
Household
Photography

```
SELECT  category
FROM    Product
ORDER BY pName
```

Category
Gadgets
Household
Gadgets
Photography

```
SELECT  DISTINCT category
FROM    Product
ORDER BY pName
```

Syntax error on large DBMSs (Oracle, PostgreSQL, SQL server) / unpredictable results on others(MySQL, SQLite)

"ORDER BY items must appear in the select list if SELECT DISTINCT is specified."

L02: SQL Basics

Announcements!

- Microphone
- If you still have SQLite trouble, please ask Disha or Priyal for help during lecture!
- Piazza: please be specific on Piazza with your problem, so we can help you remotely.
  - Compare: "I can't install SQLite. What should I do?" (-> come to office hours) vs. "I get error message XYZ when I do ZYX. Here is a screenshot. What should I do?"
- Piazza: please also post your lessons learned (e.g., John's comment on FF v56)
- Textbooks
- Homework #1 will be released by tonight together with PostgreSQL installation guide (you have 2 weeks)
- Python, Jupyter

Some history

Some "birth-years"

- 2004: Facebook

- 1998: Google
- 1995: Java, Ruby
- 1993: World Wide Web
- 1991: Python

- 1985: Windows

- 1974: SQL

SQL: Declarative Programming

SQL

```
select (e.salary / (e.age - 18)) as comp
from employee as e
where e.name = "Jones"
```

Declarative Language: you say what you want without having to say how to do it.

Procedural Language: you have to specify exact steps to get the result.

SQL: was not the only Attempt

SQL

```
select (e.salary / (e.age - 18)) as comp
from employee as e
where e.name = "Jones"
```

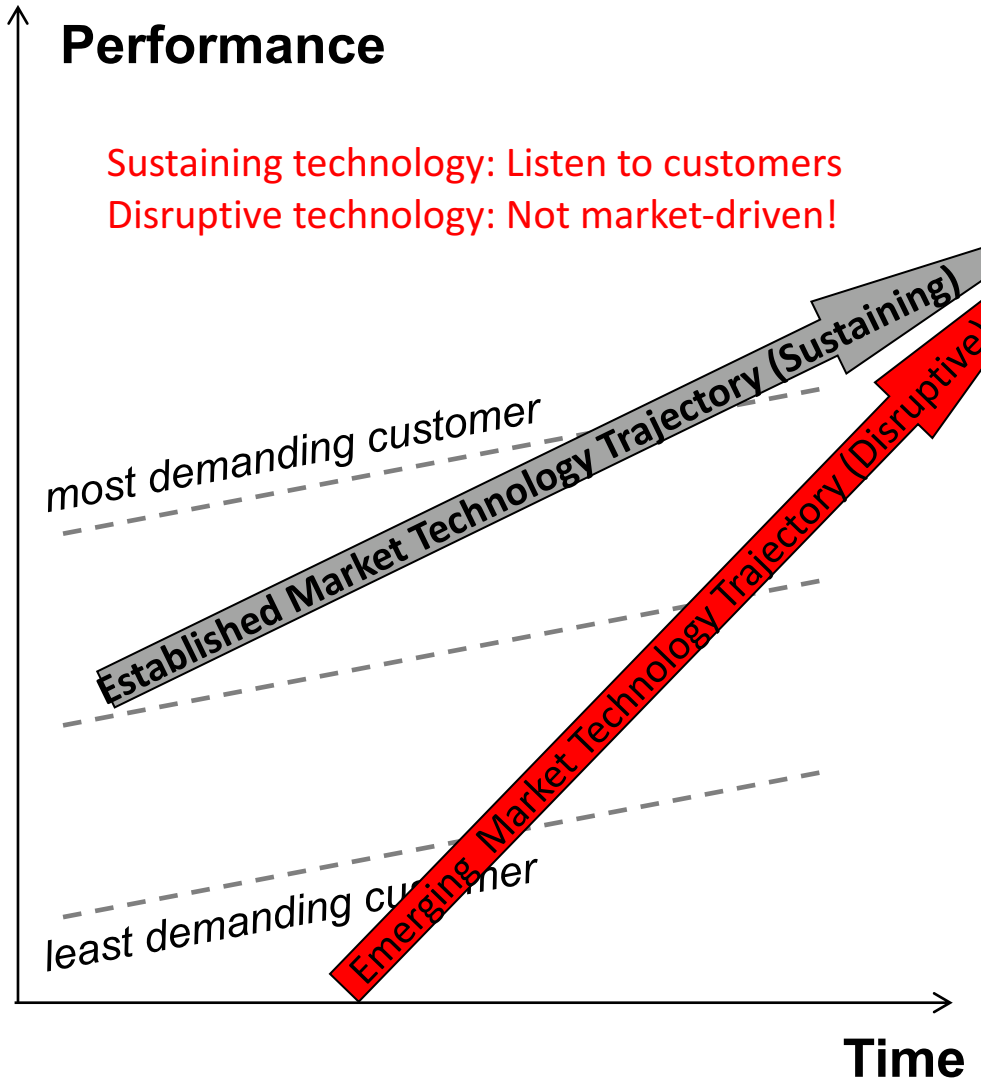
QUEL

```
range of e is employee
retrieve (comp = e.salary / (e.age - 18))
where e.name = "Jones"
```

Commercially not used anymore since ~1980

Source: http://en.wikipedia.org/wiki/QUEL_query_languages

Disruptive Innovation



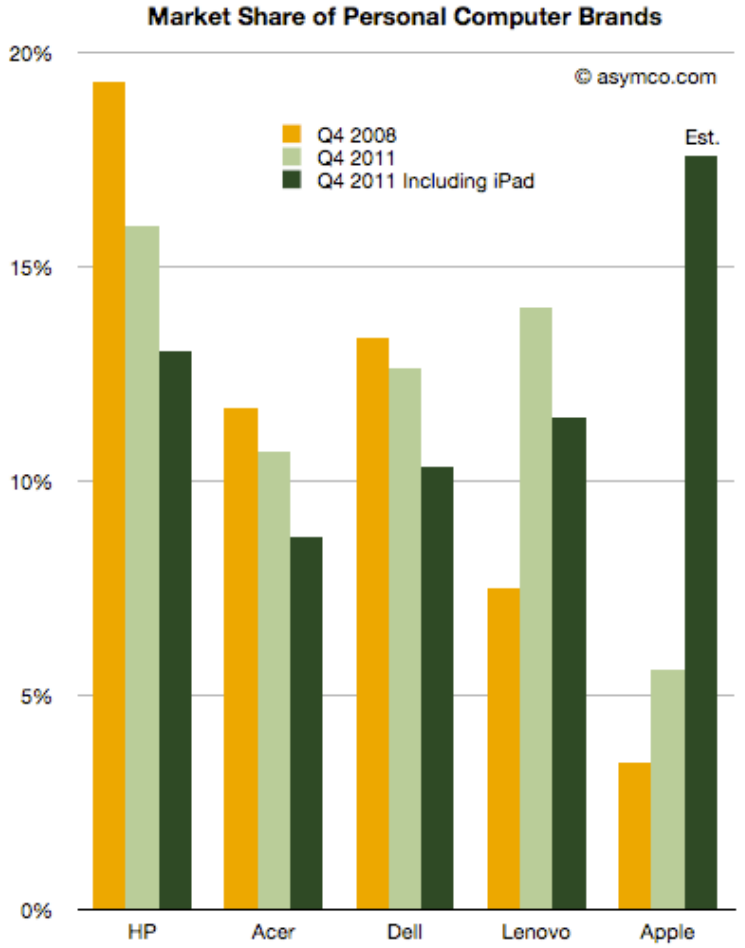
- Disruptive innovations are generally not acceptable for the mass market when they are introduced. Only the fringes of the market pick up the innovation in the first iteration
- It performs worse in one or more areas, but is typically simpler, more reliable, or more convenient than existing technologies.
- It is less profitable than existing technologies. Leading firms' most profitable customers generally can't use it and don't want it.
- As the innovator continues to refine their product the utility value to the market increases
- Its performance trajectory is steeper than that of existing technologies.
- Large organizations are fundamentally incapable of successfully bringing it to market.

Source: After “The Innovator’s Dilemma” by Clayton Christensen, 1997.

iPhone: Disruptive Innovation or not?

1: "Business Phones"
Microsoft in 2007

2: Laptops



Source: http://www.youtube.com/watch?v=eywi0h_Y5_U , http://www.asymco.com/2012/01/16/apple-is-the-top-personal-computer-vendor/

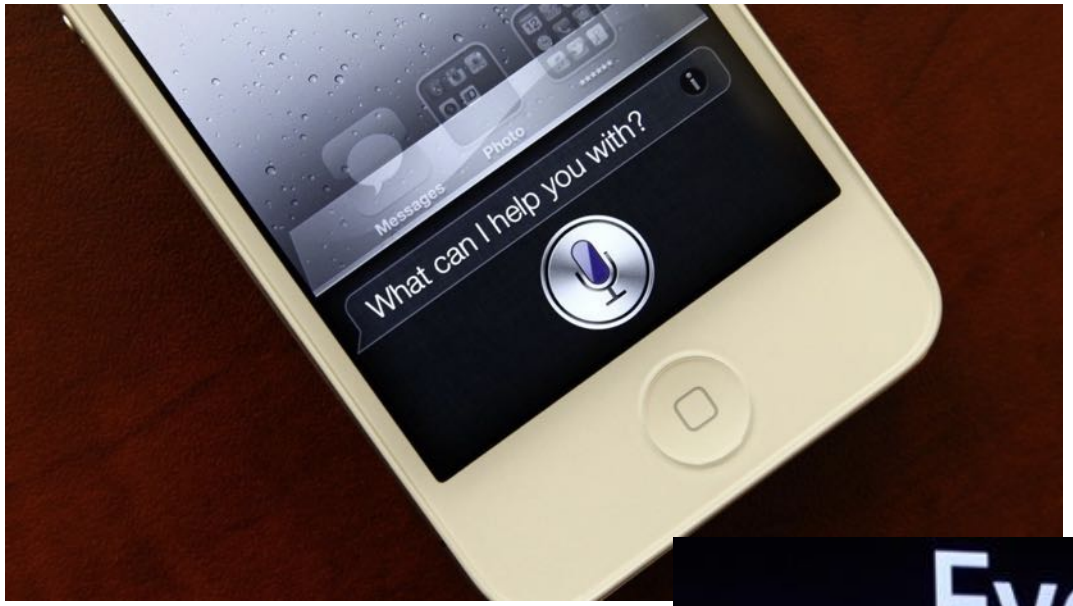
What keyboard without keys can do...

In Feb 2016, SwiftKey was purchased by Microsoft, for 250 M$

Sources: https://en.wikipedia.org/wiki/Swype, https://en.wikipedia.org/wiki/SwiftKey

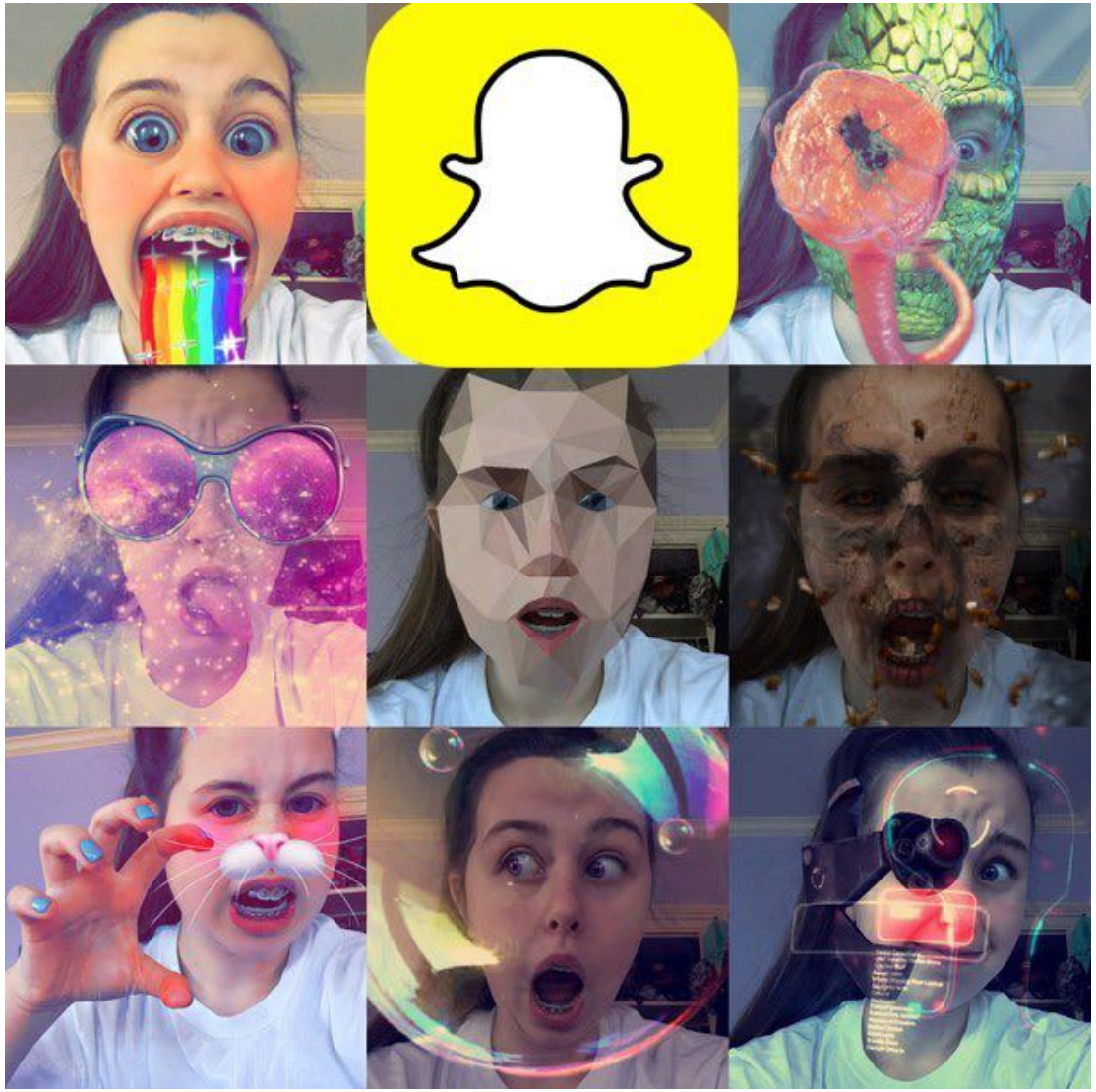
The keyboard of the future?

Sources: http://www.wired.com/2014/06/siri_ai/


Messages
Photo
What can I help you with?


Eyes Free
MODE
engadget

Keyboards and emails?

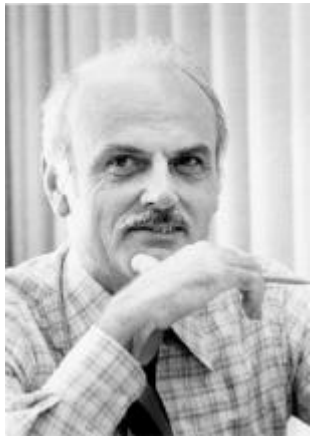


Sources: http://www.buzzfeed.com/benrosen/how-to-snapchat-like-the-teens

What is this? (1975)

Source: http://pluggedin.kodak.com/pluggedin/post/?id=687843

SQL: some history



- Dr. Edgar Codd (IBM)
  - CACM June 1970: "A Relational Model of Data for Large Shared Data Banks" http://seas.upenn.edu/~zives/03f/cis550/codd.pdf
- Standardized
  - 1986 by ANSI: SQL1
  - 1992: Revised: SQL2
    - Approx 580 page document describing syntax and semantics
  - Revised: 1999, 2003, 2008, ...
- Players
  - Microsoft, IBM, Relational Software (Oracle), ….
- Every vendor has a slightly different version of SQL
- But the main commands are standardized

Codd's (disruptive ?) innovation

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

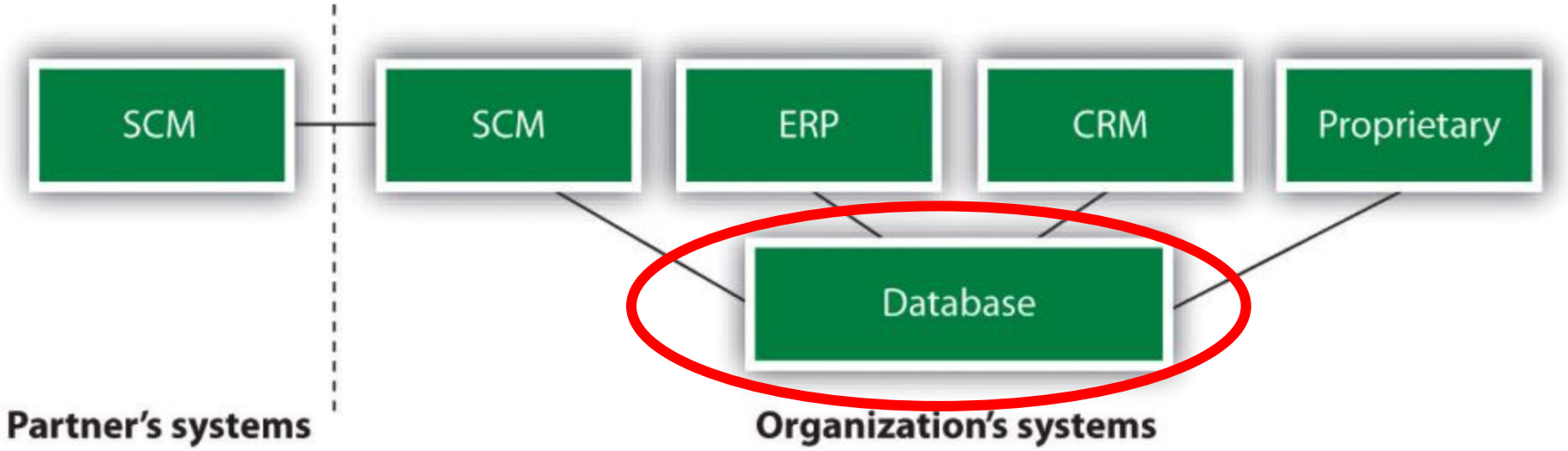
1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Source: http://seas.upenn.edu/~zives/03f/cis550/codd.pdf

SQL and the relational model as standard



Source: "Information Systems: A Manager's Guide to Harnessing Technology (book v1.4)," p.185, Gallaugher, 2012.

Databases we are using

Client/Server Architecture

- There is a single server that stores the database (called DBMS or RDBMS):
  - Usually a beefy system, e.g. IISQLSRV
  - But can be your own desktop…
  - … or a huge cluster running a parallel dbms (later assign.)
- Many clients run apps and connect to DBMS
  - E.g. Microsoft's Management Studio
  - More realistically some Java, Python, or C++ program
- Clients "talk" to server using some protocol

DBMSs we will work with

- SQLlite
  - most widely deployed database engine
  - in particular with embedded systems, browsers, etc., e.g., Microsoft's Windows Phone 8, Apple's iOS, Skype, Firefox
- PostgreSQL
  - popular and powerful open source database (Microsoft)

SQLite vs. PostgreSQL

SQLlite

- open source & cross-platform
- easy to install
- has no server ("embedded")
- ideal for single-user application; has limitations when it comes to concurrency / simultaneous transactions (one writer at a time)
- does not allow partitioning; everything is stored in one single file
- extra functions are written in C/C++

PostgreSQL

- commercial (Microsoft)
- takes a bit longer to install
- uses a server
- ideal for shared repository; allows concurrency (many simultaneous transactions), locking and fine-grained access control
- scales to >GB easily; allows partitioning (distributing) the data across several files / nodes
- supports user-defined functions

SQL overview

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 detail 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

General Constraints

- We can actually specify arbitrary assertions
  - E.g. “There cannot be 25 people in the DB class”

- In practice, we don’t specify many such constraints. Why?
  - Performance!

Whenever we do something ugly (or avoid doing something convenient) it’s for the sake of performance

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

Basic SQL

Simple SQL Query

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
FROM    Product
WHERE   manufacturer in ('Canon','Hitachi')
```

Simple SQL Query

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
FROM    Product
WHERE   manufacturer in ('Canon','Hitachi')
```

Selection
& Projection

PName
SingleTouch
MultiTouch

WHERE … IN (…) cp. to Excel

JFYI

	A	B	C	D	E	F
1	*Source: Walkenbach, Excel 2010 Formulas, p396, 2010.*					
2						
3	Is this name contained in the range?					
4	Name:	Barbara	TRUE			
5		Betty	1			
6		Betty	TRUE			
7						
8	**NameRange**					
9						
10	Barbara	Karen	Nancy			
11	Betty	Kimberly	Patricia			
12	Carol	Laura	Ruth			
13	Deborah	Linda	Sandra			
14	Donna	Lisa	Sarah			
15	Dorothy	Margaret	Sharon			
16	Elizabeth	Maria	Susan			
17	Helen	Mary				
18	Jennifer	Michelle				
19						
20						

Assume that there is a range defined for A10:C18 called "NameRange"

WHERE … IN (…) cp. to Excel

JFYI

	A	B	C	D	E	F
1	*Source: Walkenbach, Excel 2010 Formulas, p396, 2010.*					
2						
3	Is this name contained in the range?					
4	Name:	Barbara	TRUE	**{=OR(NameRange=B4)}**		
5		Betty	1	**=COUNTIF(NameRange,B5)**		
6		Betty	TRUE	**=COUNTIF(NameRange,B6)>0**		
7						
8	**NameRange**					
9						
10	Barbara	Karen	Nancy			
11	Betty	Kimberly	Patricia			
12	Carol	Laura	Ruth			
13	Deborah	Linda	Sandra			
14	Donna	Lisa	Sarah			
15	Dorothy	Margaret	Sharon			
16	Elizabeth	Maria	Susan			
17	Helen	Mary				
18	Jennifer	Michelle				
19						
20						

Assume that there is a range defined for A10:C18 called "NameRange"

LIKE

LIKE: Simple String Pattern Matching

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
FROM    Product
WHERE   pname LIKE '%izmo'
```

PName
Gizmo
Powergizmo

% is a wildcard for any sequence of zero or more characters.

More details: http://www.techonthenet.com/sql_server/like.php

LIKE: Simple String Pattern Matching

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
FROM    Product
WHERE   pname LIKE '_izmo'
```

PName
Gizmo

_ is a wildcard for exactly one character.

More details: http://www.techonthenet.com/sql_server/like.php

Table selection using comparison predicates

	Numbers	**Text / Strings**
Simple comparators	= equal to < smaler than <= smaller or equal to > greater than >= greater or equal to <> unequal to	= equal to (exact string)
Complex comparators	BETWEEN value1 AND value2 any values within the range	LIKE equal to (pattern) 'S%' string starting with S '%S' string ending with S '%S%' string containing an S 'S_S' string with S at both ends and any character in the middle
Comparators that **work across types**	IN (value1, value2, …) any values within the given set IS NULL has no value IS NOT NULL has a value	

Note: Combinations of multiple predicates with **AND** & **OR** (use brackets)

Date functions

Arithmetic expressions

```
SELECT  3+2
```

(no column name)
5

```
SELECT  (2*3 + 4*5) as name
```

name
26

Date functions are database-specific

JFYI

Worker

Name	Birthdate
Max	1980-01-01
Fred	1979-02-01
Susan	1990-01-31
Tilda	1988-01-01

We can specify the output column names

```
SELECT  date('now')-date(birthdate) as age
FROM    Worker
```

age
33
34
23
25

This is here SQLite semantics
Date functions are different between different databases.
In real life, you may need to look up how your DB handles date functions:
http://www.sqlite.org/lang_datefunc.html

Joins

Keys and Foreign Keys

Product

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

Company

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

What is a foreign key vs. a key here?

Keys and Foreign Keys

302

Key

Foreign key

Product

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

Key

Company

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

What is a foreign key vs. a key here?

Referential Integrity

Product

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

Company

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

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

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

violates Key constraint

Gizmo	$14.99	Gadgets	Hitachi

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

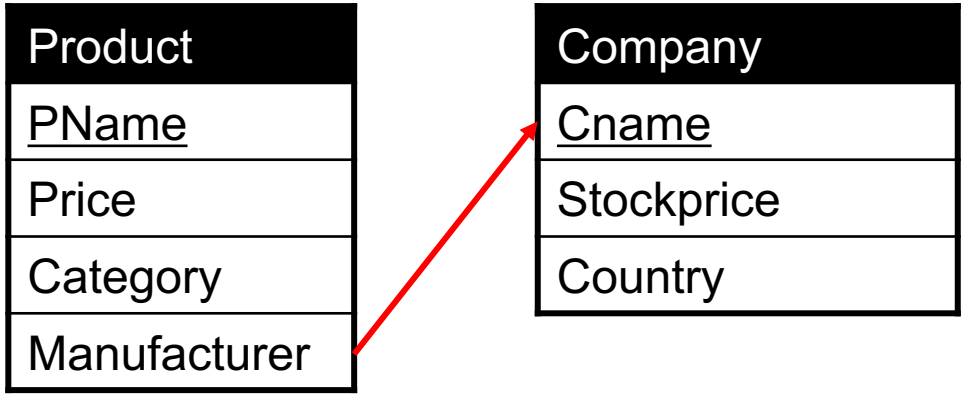
```
Insert into Product values ('SuperTouch', 249.99, 'Computer', 'NewCom');
```

violate Foreign Key constraint

SuperTouch	$249.99	Computer	NewCom

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

(Relational Database) Schema



"Schema": describes the structure of data in terms of the relational data model.

A schema includes tables, columns, PKs, FKs, and other constraints

Product(pname, price, category, manufacturer)
Company(cname, stockprice, country)

Product.manufacturer is FK to Company

Joins

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

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

Company

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

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

Joins

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

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

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

Quiz

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

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

Company

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

What does the query below return?

```
SELECT  pName, StockPrice
FROM    Product, Company
WHERE   manufacturer=cName
    and country = 'USA'
```

Quiz

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

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

Company

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

What does the query below return?

```
SELECT   pName, StockPrice
FROM     Product, Company
WHERE    manufacturer=cName
   and   country = 'USA'
```

PName	StockPrice
Gizmo	25
Powergizmo	25

Table Alias (Tuple Variables)

Person (pName, address, works_for)
University (uName, address)

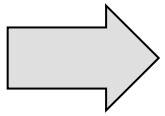
```
SELECT  DISTINCT pName, address
FROM    Person, University
WHERE   works_for = uName
```

Table Alias (Tuple Variables)

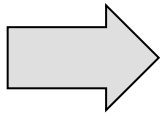
Person (pName, address, works_for)
University (uName, address)

which address?
Error!

```
SELECT  DISTINCT pName, address
FROM    Person, University
WHERE   works_for = uName
```



```
SELECT  DISTINCT Person.pName, University.address
FROM    Person, University
WHERE   Person.works_for = University.uName
```



```
SELECT  DISTINCT X.pName, Y.address
FROM    Person as X, University Y
WHERE   X.works_for = Y.uName
```

Note that "as" is optional !!