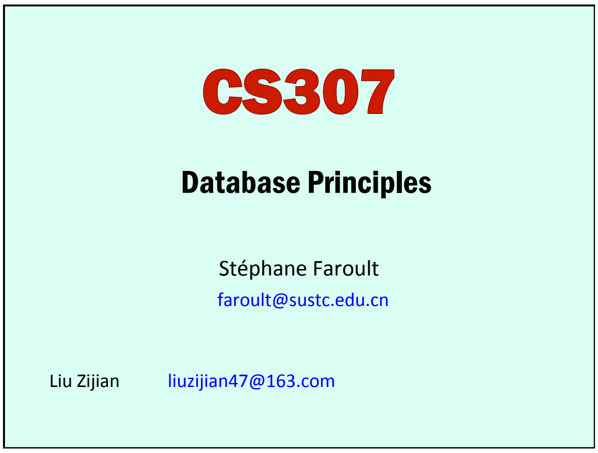
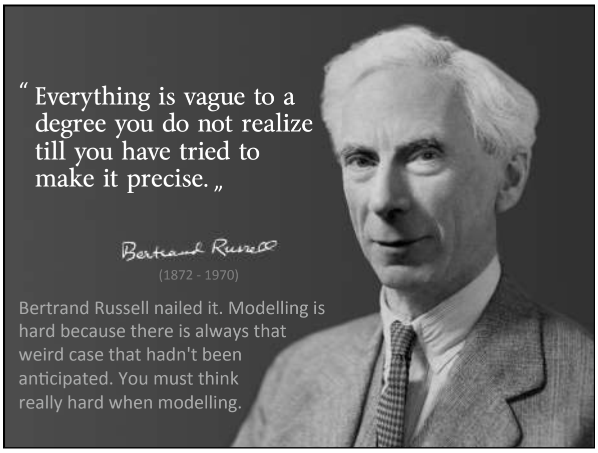


CS307

Database Principles

Stéphane Faroult
faroult@sustc.edu.cn

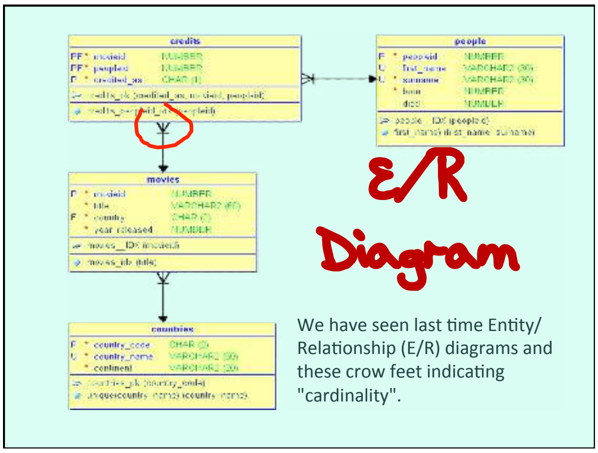
Liu Zijian liuzijian47@163.com



> “ Everything is vague to a degree you do not realize till you have tried to make it precise. „

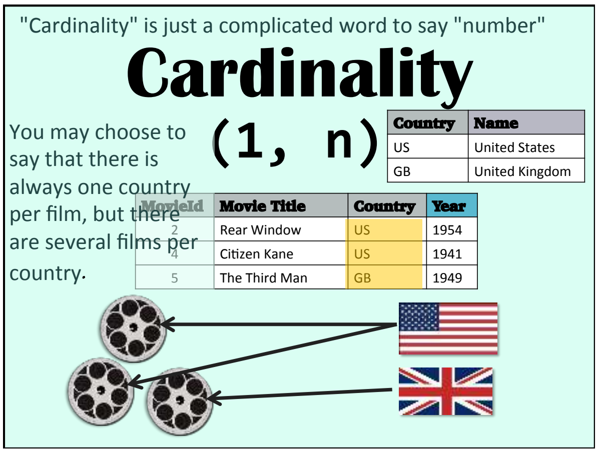
Bertrand Russell
(1872 - 1970)

Bertrand Russell nailed it. Modelling is hard because there is always that weird case that hadn't been anticipated. You must think really hard when modelling.


credits
FF * movieid NUMBER
FF * peopleid NUMBER
F * credited_as CHAR (1)
credits_pk (credited_as, movieid, peopleid)
credits_peopleid_idx (peopleid)
people
P * peopleid NUMBER
U first_name VARCHAR2 (30)
U * surname VARCHAR2 (30)
* born NUMBER
died NUMBER
people_IDX (peopleid)
first_name) (first_name, surname)
movies
P * movieid NUMBER
* title VARCHAR2 (60)
F * country CHAR (2)
* year_released NUMBER
movies_IDX (movieid)
movies_idx (title)
countries
P * country_code CHAR (2)
U * country_name VARCHAR2 (30)
* continent VARCHAR2 (20)
countries_pk (country_code)
unique(country_name) (country_name)

E/R Diagram

We have seen last time Entity/ Relationship (E/R) diagrams and these crow feet indicating "cardinality".



"Cardinality" is just a complicated word to say "number"

Cardinality

You may choose to say that there is always one country per film, but there are several films per country.

(1, n)

Country	Name
US	United States
GB	United Kingdom

MovieId	Movie Title	Country	Year
2	Rear Window	US	1954
4	Citizen Kane	US	1941
5	The Third Man	GB	1949

Cardinality

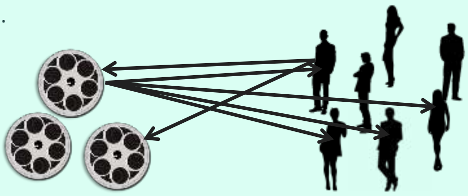
(0, n)

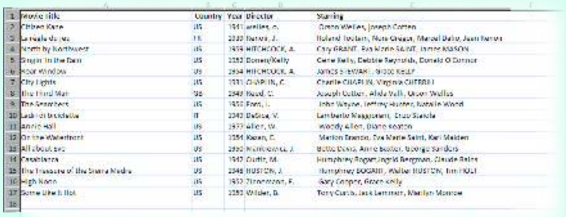
In some cases, some attributes haven't always a value; "main language" for instance, because you have silent films. In that case one speaks about a (0, n) cardinality.

Cardinality

(m, n)

(m,n) cardinality qualifies a "many-to-many" relationship; several actors usually appear in a film, and an actor usually plays in several films. (m,n) cardinality implies a relationship table.





1	Movie title	Country	Year	Director	Starring
2	Citizen Kane	US	1941	Welles, O.	Orson Welles, Joseph Cotten
3	La règle du jeu	FR	1939	Renoir, J.	Roland Toutain, Nora Gregor, Marcel Dalio, Jean Renoir
4	North by Northwest	US	1959	HITCHCOCK, A.	Cary GRANT, Eva Marie SAINT, James MASON
5	Singin' in the Rain	US	1952	Donen/Kelly	Gene Kelly, Debbie Reynolds, Donald O'Connor
6	Rear Window	US	1954	HITCHCOCK, A.	James STEWART, Grace KELLY
7	City Lights	US	1931	CHAPLIN, C.	Charlie CHAPLIN, Virginia CHERRILL
8	The Third Man	GB	1949	Reed, C.	Joseph Cotten, Alida Valli, Orson Welles
9	The Searchers	US	1956	Ford, J.	John Wayne, Jeffrey Hunter, Natalie Wood
10	Ladri di biciclette	IT	1948	DeSica, V.	Lamberto Maggiorani, Enzo Staiola
11	Annie Hall	US	1977	Allen, W.	Woody Allen, Diane Keaton
12	On the Waterfront	US	1954	Kazan, E.	Marlon Brando, Eva Marie Saint, Karl Malden
13	All about eve	US	1950	Mankiewicz, J.	Bette Davis, Anne Baxter, George Sanders
14	Casablanca	US	1942	Curtiz, M.	Humphrey Bogart, Ingrid Bergman, Claude Rains
15	The Treasure of the Sierra Madre	US	1948	HUSTON, J.	Humphrey BOGART, Walter HUSTON, Tim HOLT
16	High Noon	US	1952	Zinnemann, F.	Gary Cooper, Grace Kelly
17	Some Like It Hot	US	1959	Wilder, B.	Tony Curtis, Jack Lemmon, Marilyn Monroe
18					

So you have seen how we went from a raw spreadsheet...

Movies

MovieId	Movie Title	Country	Year
2	Rear Window	US	1954
4	Citizen Kane	US	1941

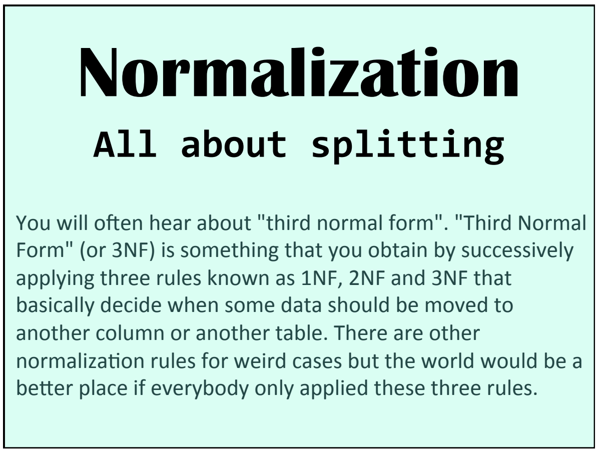
Credits

MovieId	PersonId	Credited
2	1	D
2	3	A
2	4	A
4	2	D
4	2	A
4	5	A

People

Id	Firstname	Surname	Born	Died
1	Alfred	Hitchcock	1899	1980
2	Orson	Welles	1915	1985
3	James	Stewart	1908	1997
4	Grace	Kelly	1929	1982
5	Joseph	Cotten	1905	1994

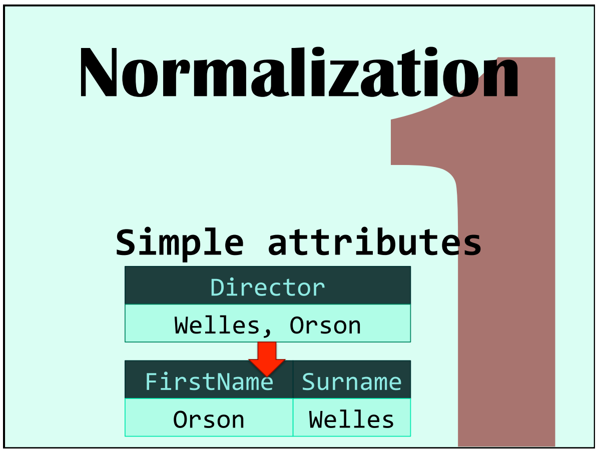
... to a relatively clean database model allowing us to handle weird cases. This process is known as normalization.



Normalization

All about splitting

You will often hear about "third normal form". "Third Normal Form" (or 3NF) is something that you obtain by successively applying three rules known as 1NF, 2NF and 3NF that basically decide when some data should be moved to another column or another table. There are other normalization rules for weird cases but the world would be a better place if everybody only applied these three rules.

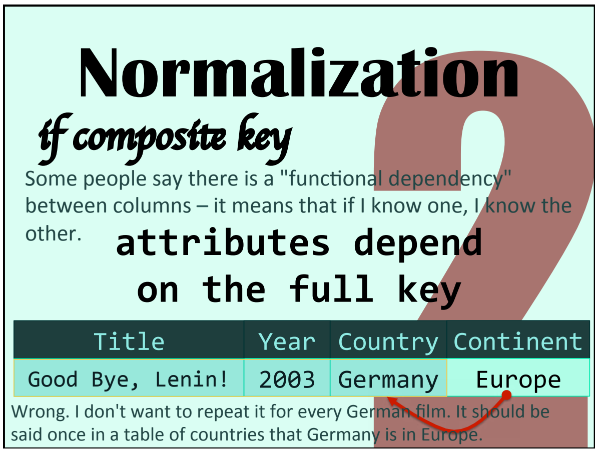


Normalization

Simple attributes

Director
Welles, Orson

FirstName	Surname
Orson	Welles



Normalization

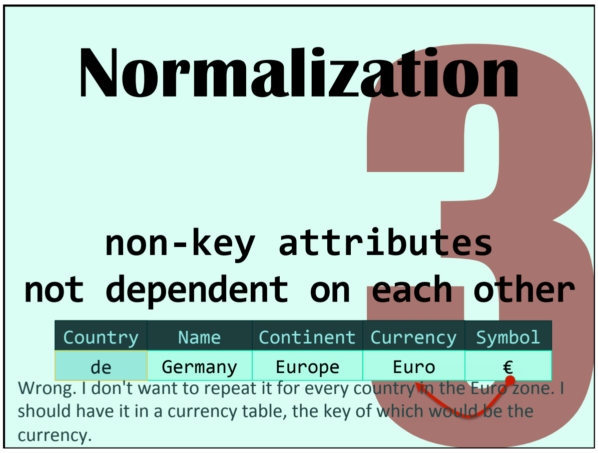
if composite key

Some people say there is a "functional dependency" between columns – it means that if I know one, I know the other.

attributes depend on the full key

Title	Year	Country	Continent
Good Bye, Lenin!	2003	Germany	Europe

Wrong. I don't want to repeat it for every German film. It should be said once in a table of countries that Germany is in Europe.

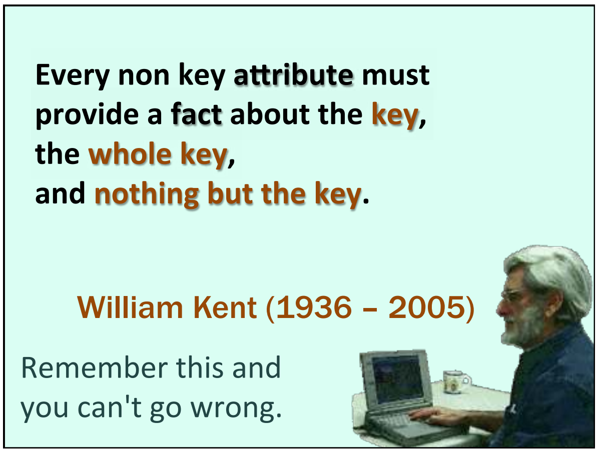


Normalization

non-key attributes not dependent on each other

Country	Name	Continent	Currency	Symbol
de	Germany	Europe	Euro	€

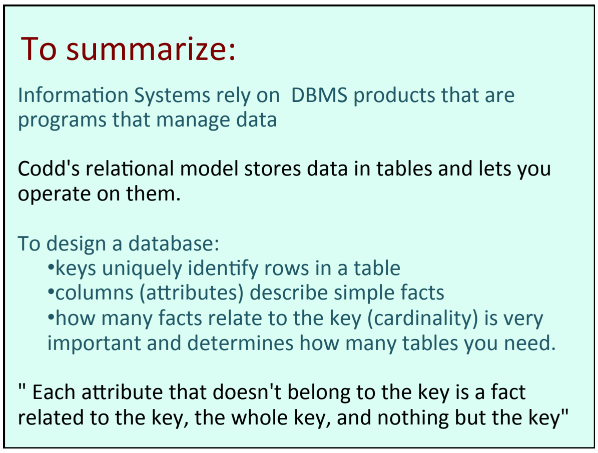
Wrong. I don't want to repeat it for every country in the Euro zone. I should have it in a currency table, the key of which would be the currency.



Every non key attribute must provide a fact about the key, the whole key, and nothing but the key.

William Kent (1936 – 2005)

Remember this and you can't go wrong.



To summarize:

Information Systems rely on DBMS products that are programs that manage data

Codd's relational model stores data in tables and lets you operate on them.

To design a database:

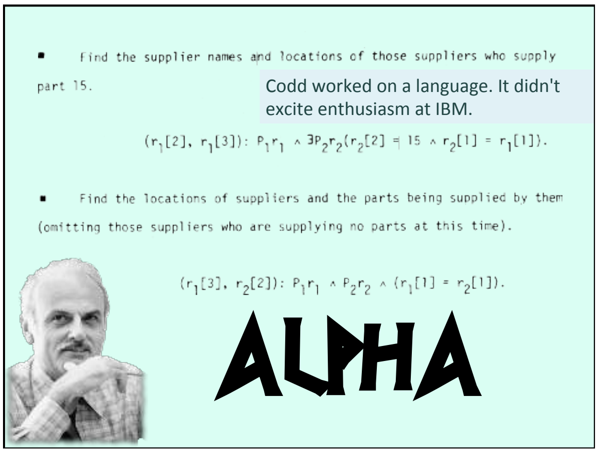
- •keys uniquely identify rows in a table
- •columns (attributes) describe simple facts
- •how many facts relate to the key (cardinality) is very important and determines how many tables you need.

" Each attribute that doesn't belong to the key is a fact related to the key, the whole key, and nothing but the key"

Now how can we pratically manage data in a database?

Special language needed!

We must be able to use a language to query a database, either interactively or from within a program: a query language.



- Find the supplier names and locations of those suppliers who supply part 15.

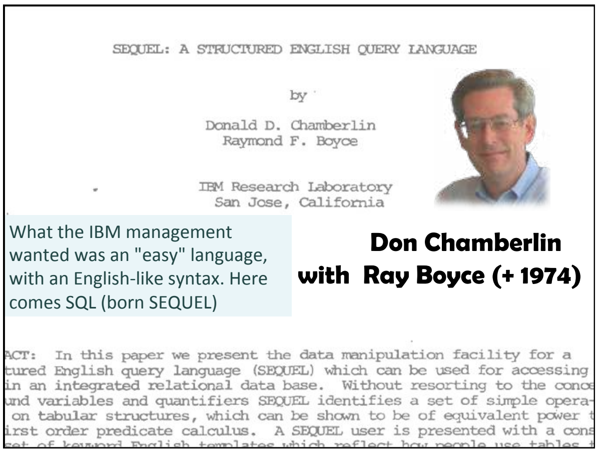
Codd worked on a language. It didn't excite enthusiasm at IBM.

$$(r_1[2], r_1[3]) : P_1 r_1 \wedge \exists P_2 r_2 (r_2[2] = 15 \wedge r_2[1] = r_1[1]).$$

- Find the locations of suppliers and the parts being supplied by them (omitting those suppliers who are supplying no parts at this time).

$$(r_1[3], r_2[2]) : P_1 r_1 \wedge P_2 r_2 \wedge (r_1[1] = r_2[1]).$$

ALPHA



SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

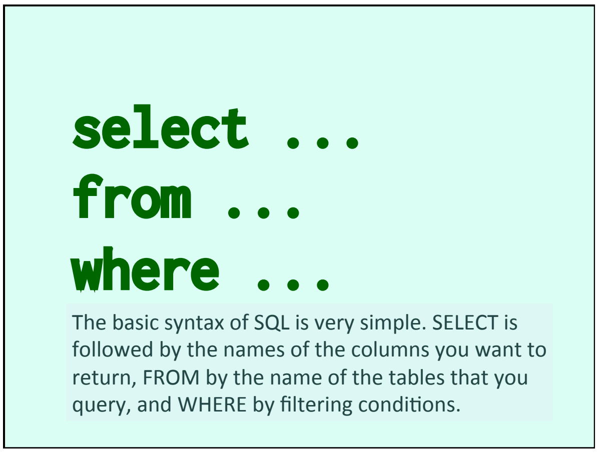
Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

What the IBM management wanted was an "easy" language, with an English-like syntax. Here comes SQL (born SEQUEL)

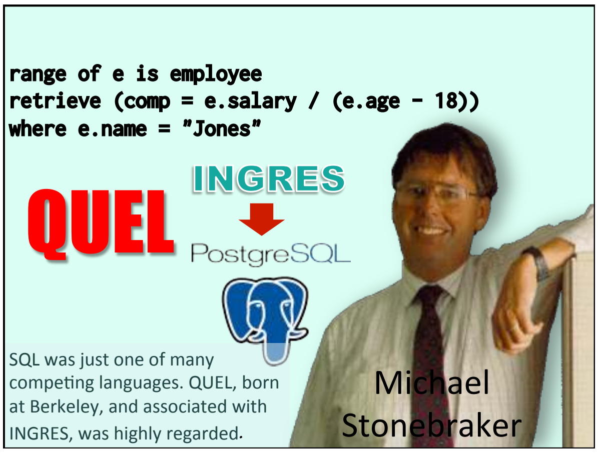
Don Chamberlin with Ray Boyce (+ 1974)

ACT: In this paper we present the data manipulation facility for a
tured English query language (SEQUEL) which can be used for accessing
in an integrated relational data base. Without resorting to the conc
und variables and quantifiers SEQUEL identifies a set of simple opera-
on tabular structures, which can be shown to be of equivalent power t
irst order predicate calculus. A SEQUEL user is presented with a cons



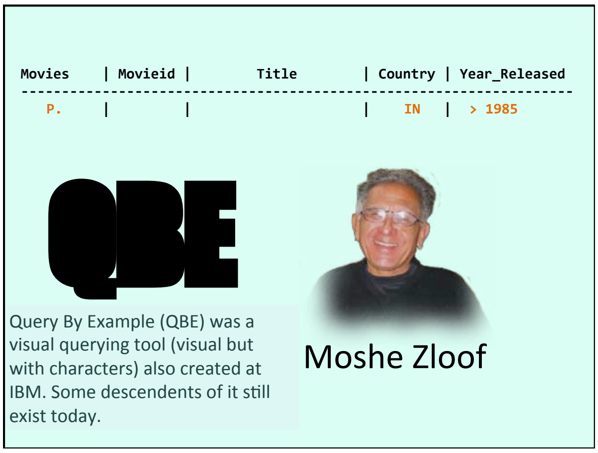
```
select ...
from ...
where ...
```

The basic syntax of SQL is very simple. SELECT is followed by the names of the columns you want to return, FROM by the name of the tables that you query, and WHERE by filtering conditions.



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

SQL was just one of many competing languages. QUEL, born at Berkeley, and associated with INGRES, was highly regarded.



```
Movies    | Movieid |        Title        | Country | Year_Released
---------------------------------------------------------------------
   P.     |         |                     |    IN   |  > 1985
```

Query By Example (QBE) was a visual querying tool (visual but with characters) also created at IBM. Some descendents of it still exist today.

But those three guys (the founders of Oracle) took SQL and created an SQL-compatible system on other systems than the IBM systems, thus helping turn SQL into a standard.

TWO main components

Codd had required in his seminal paper that a good database language should allow to deal as easily with contents (data) as containers (tables), something that SQL does reasonably well.

CREATE

ALTER

DROP

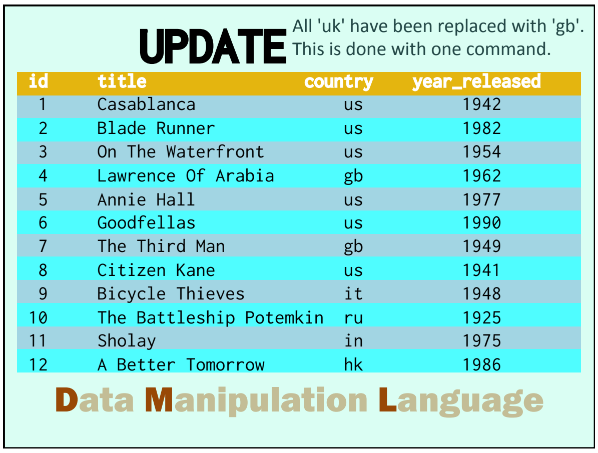
The data definition language (usually called DDL) deals with tables (as well as other database "objects" that we'll see later). Three commands are enough for creating a new table, changing its structure (for instance adding a new column) or deleting it.

Data Definition Language

INSERT The data manipulation language (DML) deals with data

id	title	country	year_released
1	Casablanca	us	1942
2	Blade Runner	us	1982
3	On The Waterfront	us	1954
4	Lawrence Of Arabia	uk	1962
5	Annie Hall	us	1977
6	Goodfellas	us	1990
7	The Third Man	uk	1949
8	Citizen Kane	us	1941
9	Bicycle Thieves	it	1948
10	The Battleship Potemkin	ru	1925
11	Sholay	in	1975
12	A Better Tomorrow	hk	1986

Data Manipulation Language

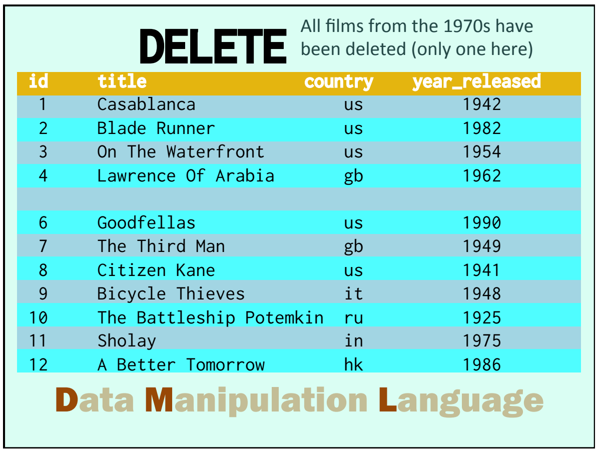


UPDATE

All 'uk' have been replaced with 'gb'. This is done with one command.

id	title	country	year_released
1	Casablanca	us	1942
2	Blade Runner	us	1982
3	On The Waterfront	us	1954
4	Lawrence Of Arabia	gb	1962
5	Annie Hall	us	1977
6	Goodfellas	us	1990
7	The Third Man	gb	1949
8	Citizen Kane	us	1941
9	Bicycle Thieves	it	1948
10	The Battleship Potemkin	ru	1925
11	Sholay	in	1975
12	A Better Tomorrow	hk	1986

Data Manipulation Language

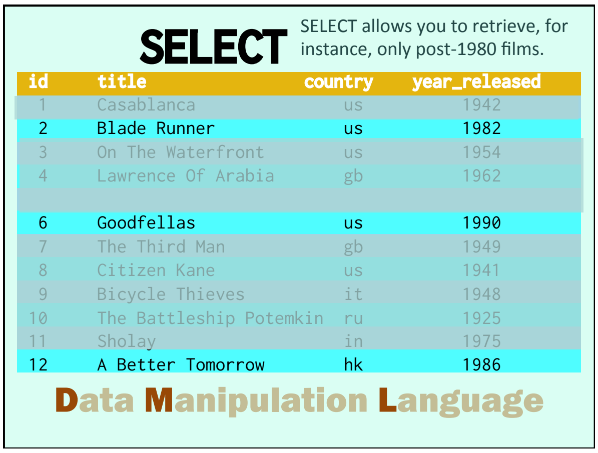


DELETE

All films from the 1970s have been deleted (only one here)

id	title	country	year_released
1	Casablanca	us	1942
2	Blade Runner	us	1982
3	On The Waterfront	us	1954
4	Lawrence Of Arabia	gb	1962
6	Goodfellas	us	1990
7	The Third Man	gb	1949
8	Citizen Kane	us	1941
9	Bicycle Thieves	it	1948
10	The Battleship Potemkin	ru	1925
11	Sholay	in	1975
12	A Better Tomorrow	hk	1986

Data Manipulation Language

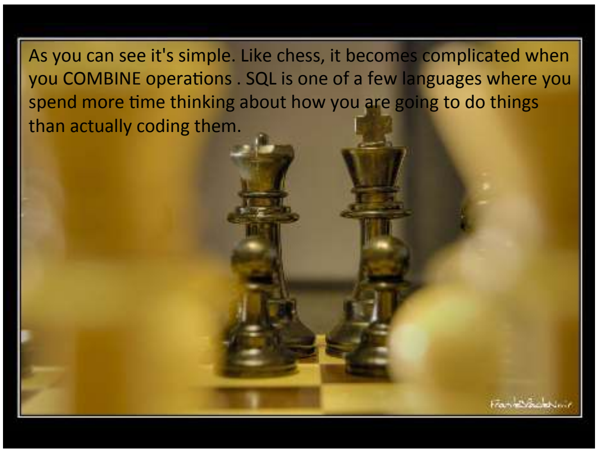


SELECT

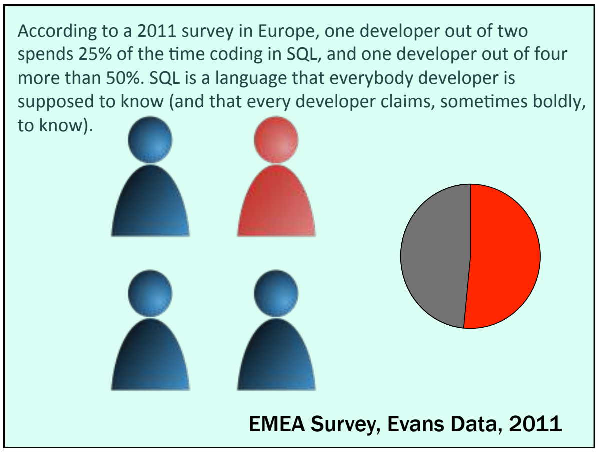
SELECT allows you to retrieve, for instance, only post-1980 films.

id	title	country	year_released
1	Casablanca	us	1942
2	Blade Runner	us	1982
3	On The Waterfront	us	1954
4	Lawrence Of Arabia	gb	1962
6	Goodfellas	us	1990
7	The Third Man	gb	1949
8	Citizen Kane	us	1941
9	Bicycle Thieves	it	1948
10	The Battleship Potemkin	ru	1925
11	Sholay	in	1975
12	A Better Tomorrow	hk	1986

Data Manipulation Language

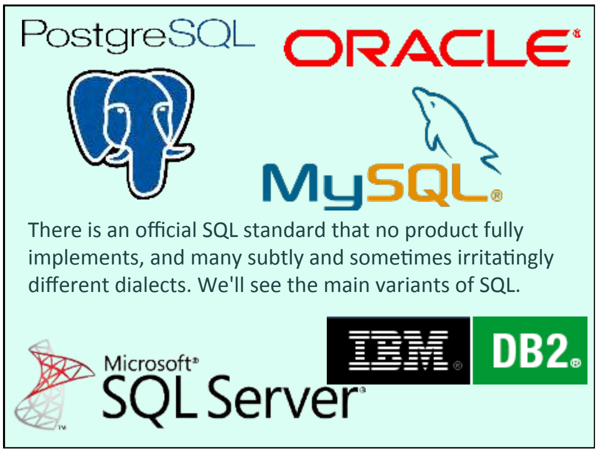


As you can see it's simple. Like chess, it becomes complicated when you COMBINE operations . SQL is one of a few languages where you spend more time thinking about how you are going to do things than actually coding them.



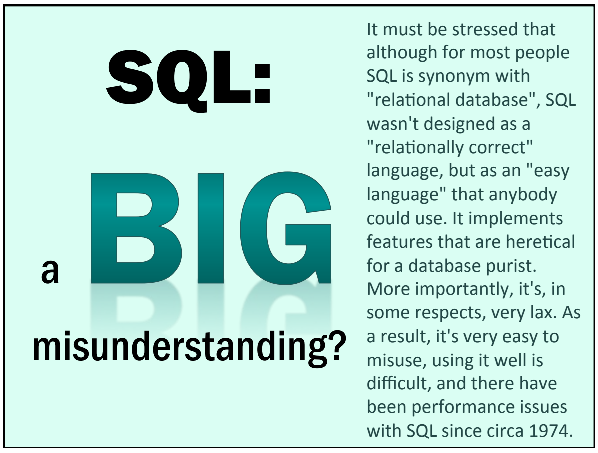
According to a 2011 survey in Europe, one developer out of two spends 25% of the time coding in SQL, and one developer out of four more than 50%. SQL is a language that everybody developer is supposed to know (and that every developer claims, sometimes boldly, to know).

EMEA Survey, Evans Data, 2011



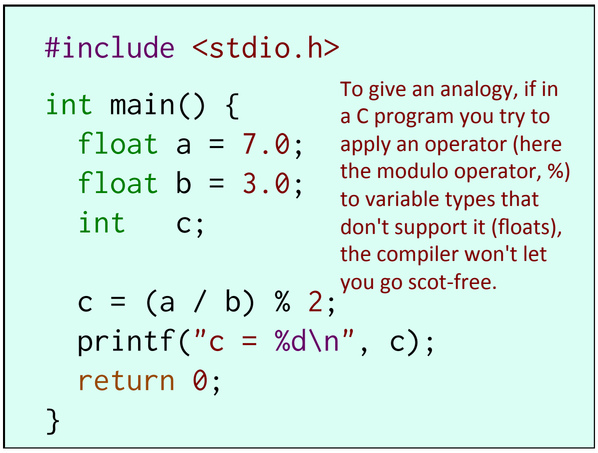
PostgreSQL
ORACLE
MySQL
Microsoft SQL Server
IBM DB2

There is an official SQL standard that no product fully implements, and many subtly and sometimes irritatingly different dialects. We'll see the main variants of SQL.



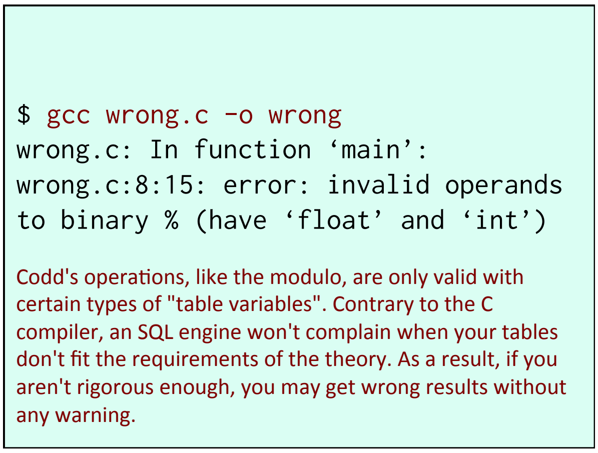
SQL: a BIG misunderstanding?

It must be stressed that although for most people SQL is synonym with "relational database", SQL wasn't designed as a "relationally correct" language, but as an "easy language" that anybody could use. It implements features that are heretical for a database purist. More importantly, it's, in some respects, very lax. As a result, it's very easy to misuse, using it well is difficult, and there have been performance issues with SQL since circa 1974.



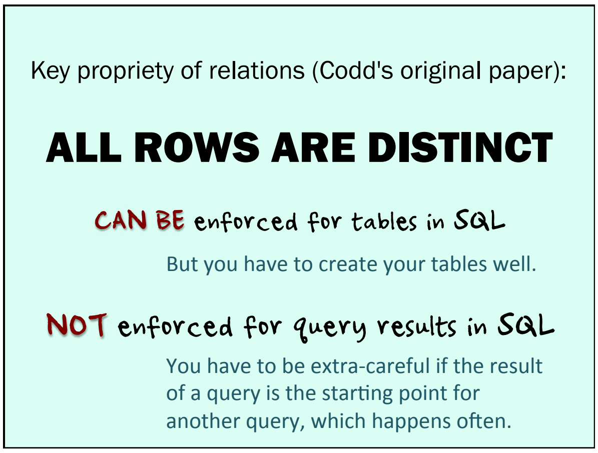
```
#include <stdio.h>

int main() {
  float a = 7.0;
  float b = 3.0;
  int   c;

  c = (a / b) % 2;
  printf("c = %d\n", c);
  return 0;
}
```

To give an analogy, if in a C program you try to apply an operator (here the modulo operator, %) to variable types that don't support it (floats), the compiler won't let you go scot-free.



```
$ gcc wrong.c -o wrong
wrong.c: In function ‘main’:
wrong.c:8:15: error: invalid operands
to binary % (have ‘float’ and ‘int’)
```

Codd's operations, like the modulo, are only valid with certain types of "table variables". Contrary to the C compiler, an SQL engine won't complain when your tables don't fit the requirements of the theory. As a result, if you aren't rigorous enough, you may get wrong results without any warning.



Key propriety of relations (Codd's original paper):

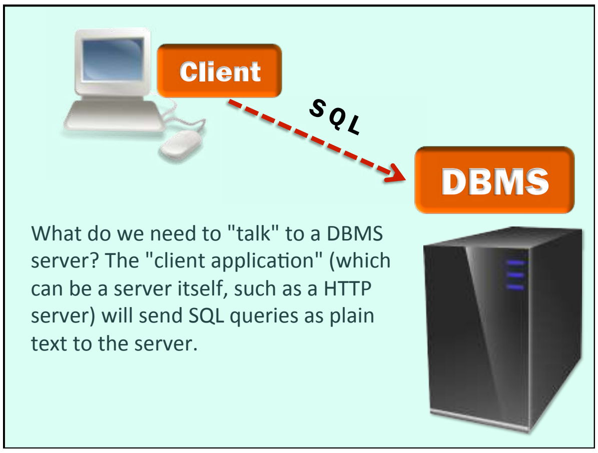
ALL ROWS ARE DISTINCT

CAN BE enforced for tables in SQL

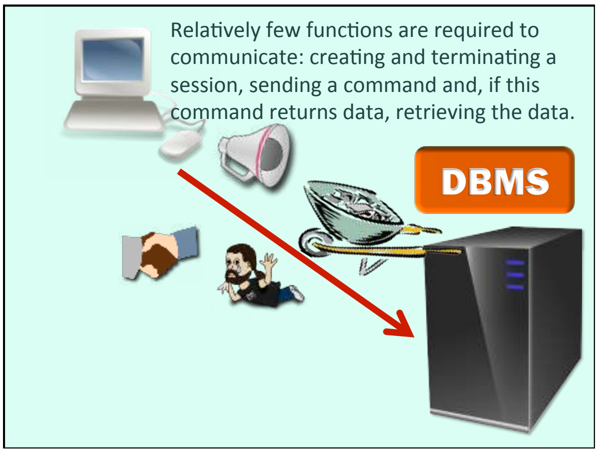
But you have to create your tables well.

NOT enforced for query results in SQL

You have to be extra-careful if the result of a query is the starting point for another query, which happens often.

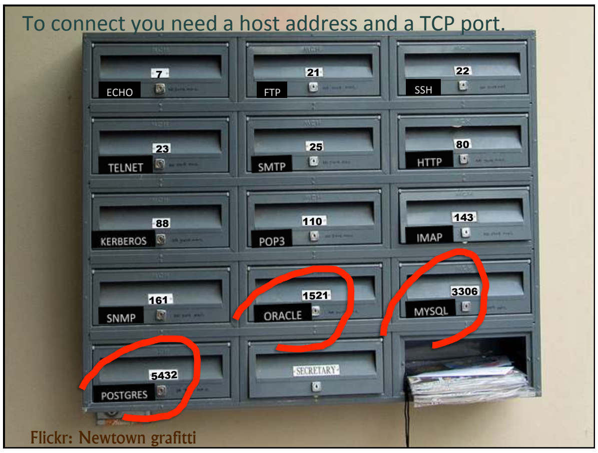


What do we need to "talk" to a DBMS server? The "client application" (which can be a server itself, such as a HTTP server) will send SQL queries as plain text to the server.



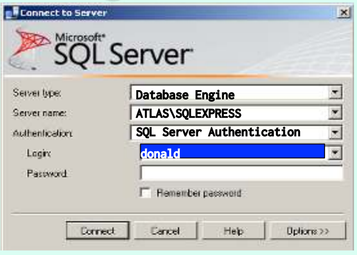
Relatively few functions are required to communicate: creating and terminating a session, sending a command and, if this command returns data, retrieving the data.

To connect you need a host address and a TCP port.



Flickr: Newtown grafitti

```
$ mysql --user=donald --host=atlas --port=3306
$ psql --username donald --host atlas --port 5432
$ sqlplus donald@//atlas:1521/ORCL
$ clpplus donald@//atlas:50000/DB2
```



Connect to Server

Server type: Database Engine

Server name: ATLAS\SQLEXPRESS

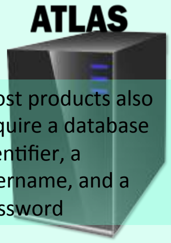
Authentication: SQL Server Authentication

Login: donald

Password:

□ Remember password

ATLAS



Most products also require a database identifier, a username, and a password

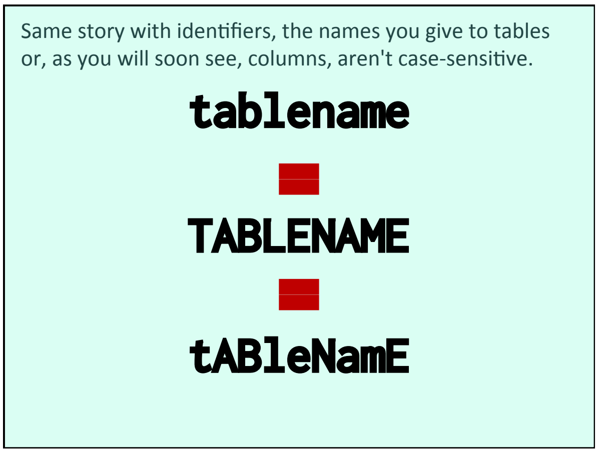
If you want to practice

http://edu.konagora.com/SQLsandbox.php

No registration required!

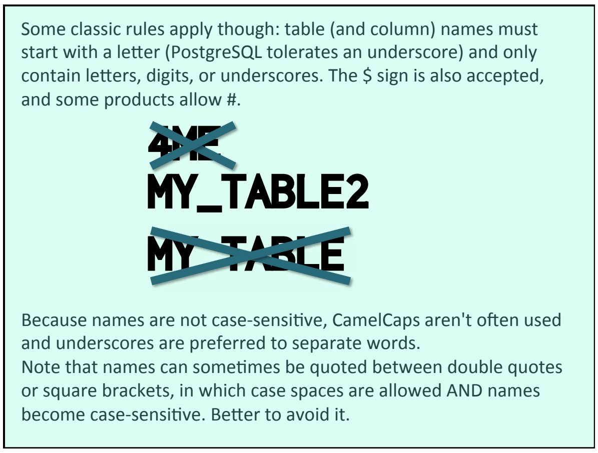
```
cReaTE tABle  table_name
    (                              ,
                                   ,
               ...
    )
```

This is the syntax (simplified, some products can take a lot of additional options) for creating a table. The weird capitalization here is only to explain that SQL keywords (words that have a special meaning in SQL) are NOT case sensitive and can be typed in any case you want. I mostly use lowercase but some people have different habits.



Same story with identifiers, the names you give to tables or, as you will soon see, columns, aren't case-sensitive.

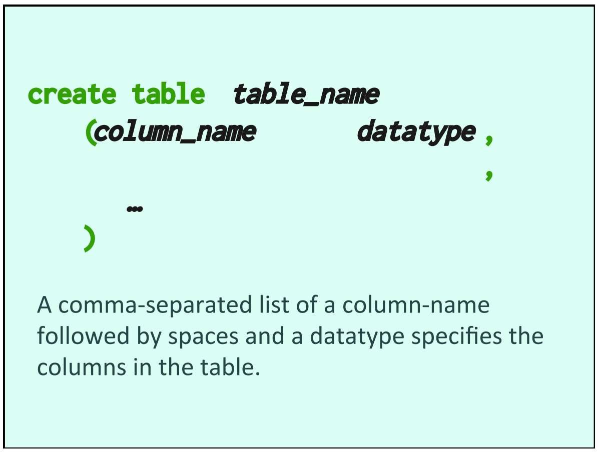
```
tablename
=
TABLENAME
=
tABleNamE
```



Some classic rules apply though: table (and column) names must start with a letter (PostgreSQL tolerates an underscore) and only contain letters, digits, or underscores. The $ sign is also accepted, and some products allow #.

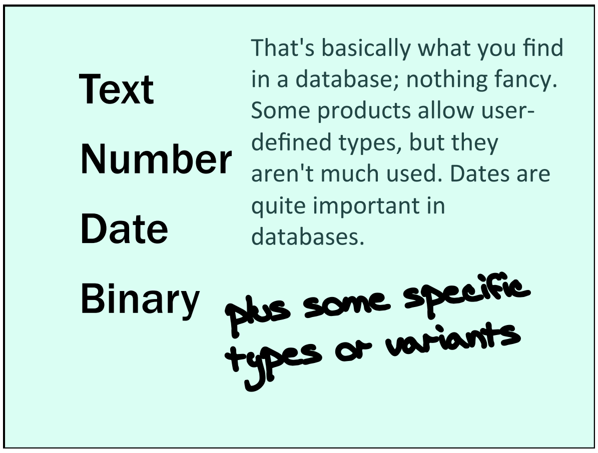
```
~~4ME~~
MY_TABLE2
~~MY TABLE~~
```

Because names are not case-sensitive, CamelCaps aren't often used and underscores are preferred to separate words.
Note that names can sometimes be quoted between double quotes or square brackets, in which case spaces are allowed AND names become case-sensitive. Better to avoid it.



```
create table  table_name
    (column_name        datatype ,
                                 ,
       ...
    )
```

A comma-separated list of a column-name followed by spaces and a datatype specifies the columns in the table.



Text

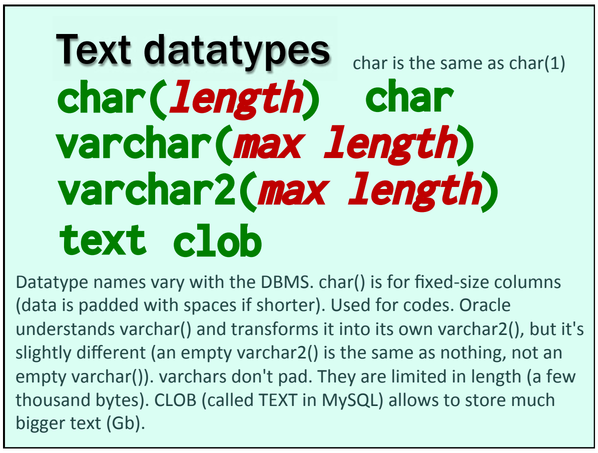
Number

Date

Binary

That's basically what you find in a database; nothing fancy. Some products allow user-defined types, but they aren't much used. Dates are quite important in databases.

plus some specific types or variants

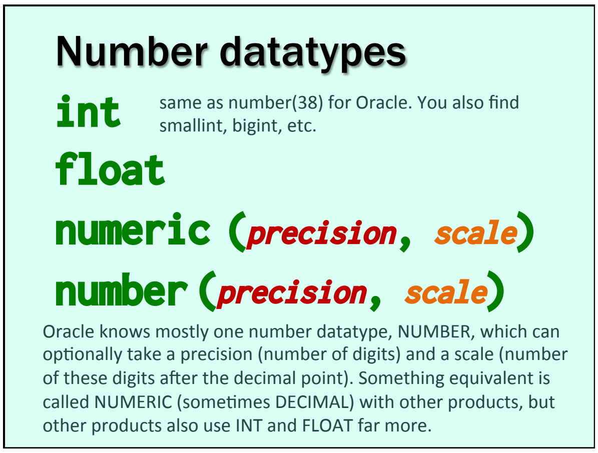


Text datatypes

char is the same as char(1)

char(*length*) **char**
varchar(*max length*)
varchar2(*max length*)
text **clob**

Datatype names vary with the DBMS. char() is for fixed-size columns (data is padded with spaces if shorter). Used for codes. Oracle understands varchar() and transforms it into its own varchar2(), but it's slightly different (an empty varchar2() is the same as nothing, not an empty varchar()). varchars don't pad. They are limited in length (a few thousand bytes). CLOB (called TEXT in MySQL) allows to store much bigger text (Gb).



Number datatypes

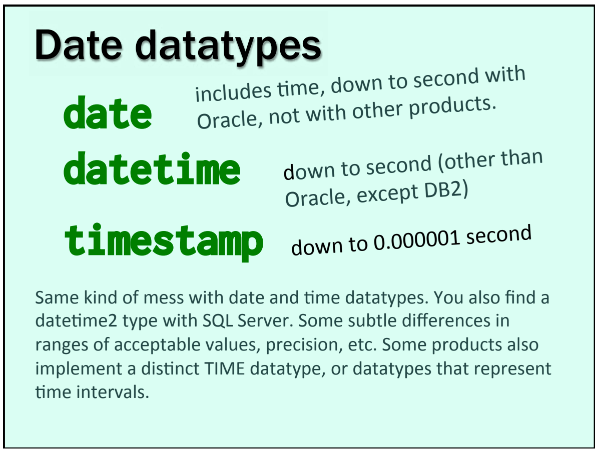
int same as number(38) for Oracle. You also find smallint, bigint, etc.

float

numeric (*precision*, *scale*)

number (*precision*, *scale*)

Oracle knows mostly one number datatype, NUMBER, which can optionally take a precision (number of digits) and a scale (number of these digits after the decimal point). Something equivalent is called NUMERIC (sometimes DECIMAL) with other products, but other products also use INT and FLOAT far more.



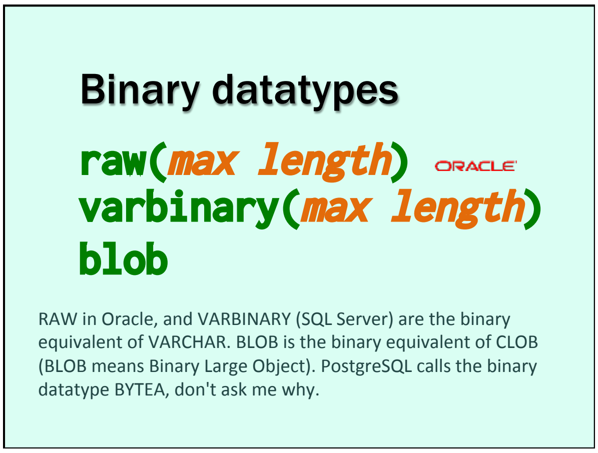
Date datatypes

date includes time, down to second with Oracle, not with other products.

datetime down to second (other than Oracle, except DB2)

timestamp down to 0.000001 second

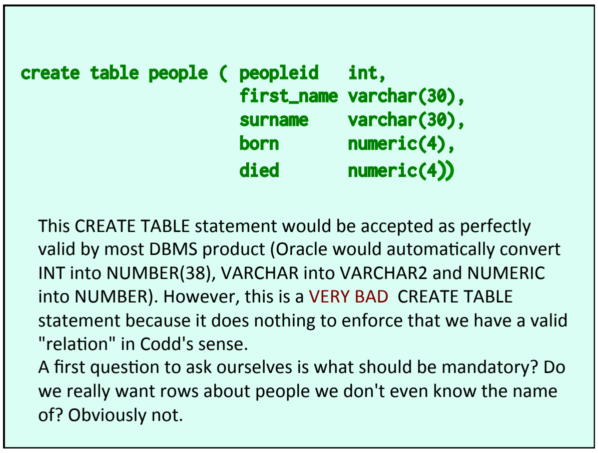
Same kind of mess with date and time datatypes. You also find a datetime2 type with SQL Server. Some subtle differences in ranges of acceptable values, precision, etc. Some products also implement a distinct TIME datatype, or datatypes that represent time intervals.



Binary datatypes

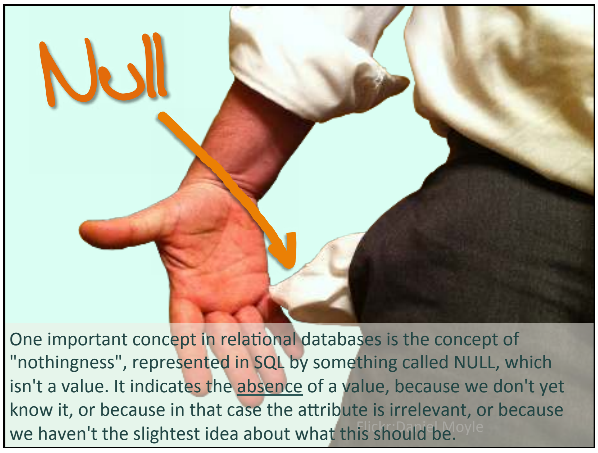
raw(*max length*) ORACLE
varbinary(*max length*)
blob

RAW in Oracle, and VARBINARY (SQL Server) are the binary equivalent of VARCHAR. BLOB is the binary equivalent of CLOB (BLOB means Binary Large Object). PostgreSQL calls the binary datatype BYTEA, don't ask me why.



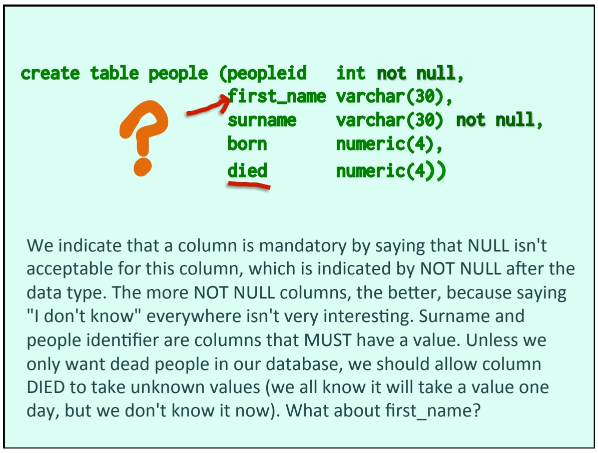
```
create table people ( peopleid   int,
                      first_name varchar(30),
                      surname    varchar(30),
                      born       numeric(4),
                      died       numeric(4))
```

This CREATE TABLE statement would be accepted as perfectly valid by most DBMS product (Oracle would automatically convert INT into NUMBER(38), VARCHAR into VARCHAR2 and NUMERIC into NUMBER). However, this is a VERY BAD CREATE TABLE statement because it does nothing to enforce that we have a valid "relation" in Codd's sense.
A first question to ask ourselves is what should be mandatory? Do we really want rows about people we don't even know the name of? Obviously not.



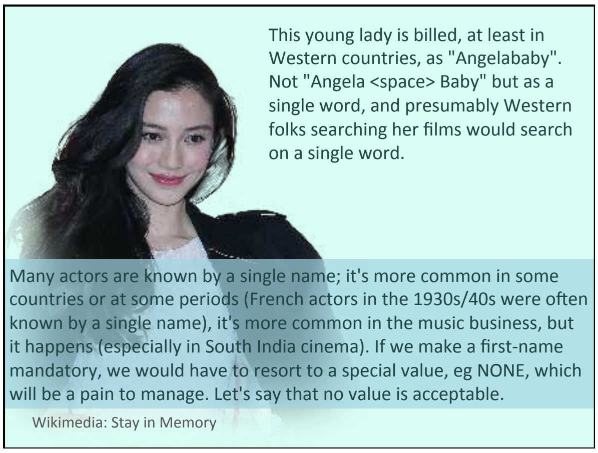
One important concept in relational databases is the concept of "nothingness", represented in SQL by something called NULL, which isn't a value. It indicates the absence of a value, because we don't yet know it, or because in that case the attribute is irrelevant, or because we haven't the slightest idea about what this should be.

Flickr:Daniel Moyle



```
create table people (peopleid   int not null,
                     first_name varchar(30),
                     surname    varchar(30) not null,
                     born       numeric(4),
                     died       numeric(4))
```

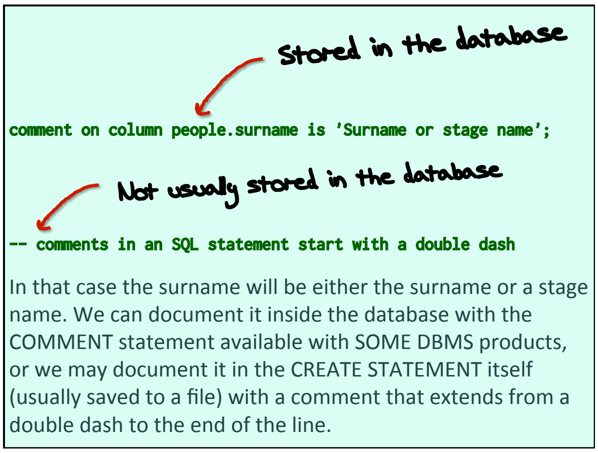
We indicate that a column is mandatory by saying that NULL isn't acceptable for this column, which is indicated by NOT NULL after the data type. The more NOT NULL columns, the better, because saying "I don't know" everywhere isn't very interesting. Surname and people identifier are columns that MUST have a value. Unless we only want dead people in our database, we should allow column DIED to take unknown values (we all know it will take a value one day, but we don't know it now). What about first_name?



This young lady is billed, at least in Western countries, as "Angelababy". Not "Angela <space> Baby" but as a single word, and presumably Western folks searching her films would search on a single word.

Many actors are known by a single name; it's more common in some countries or at some periods (French actors in the 1930s/40s were often known by a single name), it's more common in the music business, but it happens (especially in South India cinema). If we make a first-name mandatory, we would have to resort to a special value, eg NONE, which will be a pain to manage. Let's say that no value is acceptable.

Wikimedia: Stay in Memory



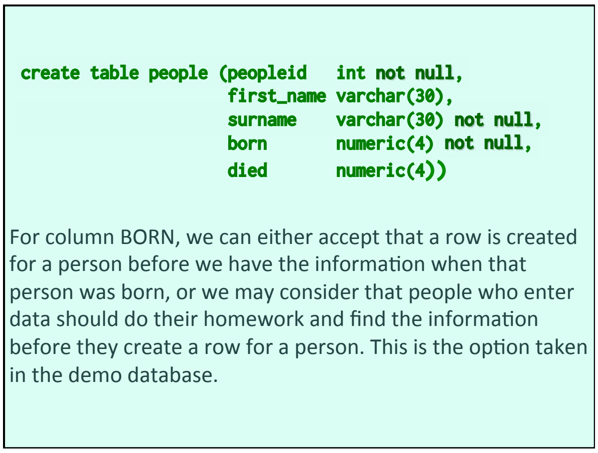
Stored in the database

```
comment on column people.surname is 'Surname or stage name';
```

Not usualy stored in the database

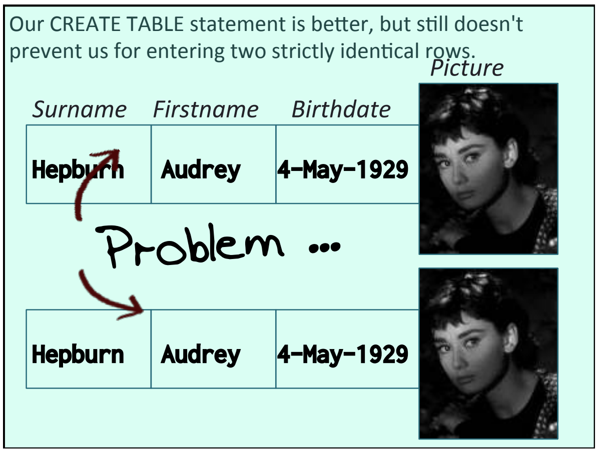
```
-- comments in an SQL statement start with a double dash
```

In that case the surname will be either the surname or a stage name. We can document it inside the database with the COMMENT statement available with SOME DBMS products, or we may document it in the CREATE STATEMENT itself (usually saved to a file) with a comment that extends from a double dash to the end of the line.



```
create table people (peopleid   int not null,
                     first_name varchar(30),
                     surname    varchar(30) not null,
                     born       numeric(4) not null,
                     died       numeric(4))
```

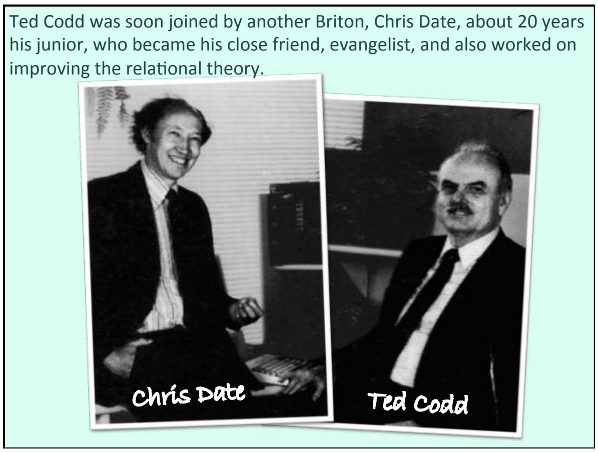
For column BORN, we can either accept that a row is created for a person before we have the information when that person was born, or we may consider that people who enter data should do their homework and find the information before they create a row for a person. This is the option taken in the demo database.



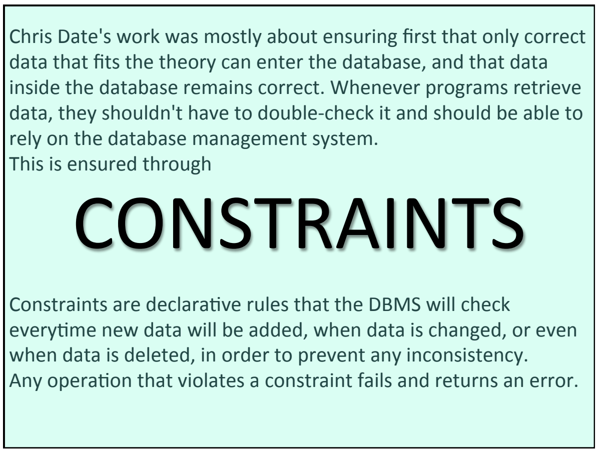
Our CREATE TABLE statement is better, but still doesn't prevent us for entering two strictly identical rows.

Surname	Firstname	Birthdate	Picture
Hepburn	Audrey	4-May-1929	
Hepburn	Audrey	4-May-1929	

Problem ...



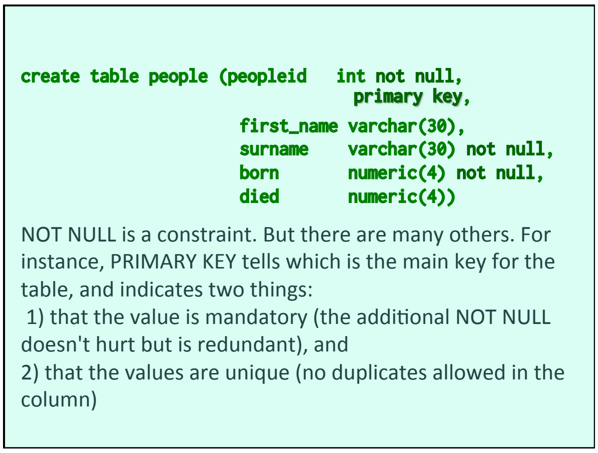
Ted Codd was soon joined by another Briton, Chris Date, about 20 years his junior, who became his close friend, evangelist, and also worked on improving the relational theory.



Chris Date's work was mostly about ensuring first that only correct data that fits the theory can enter the database, and that data inside the database remains correct. Whenever programs retrieve data, they shouldn't have to double-check it and should be able to rely on the database management system.
This is ensured through

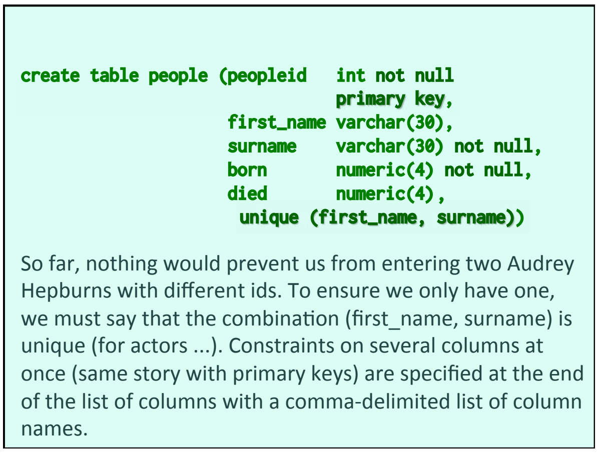
CONSTRAINTS

Constraints are declarative rules that the DBMS will check everytime new data will be added, when data is changed, or even when data is deleted, in order to prevent any inconsistency.
Any operation that violates a constraint fails and returns an error.



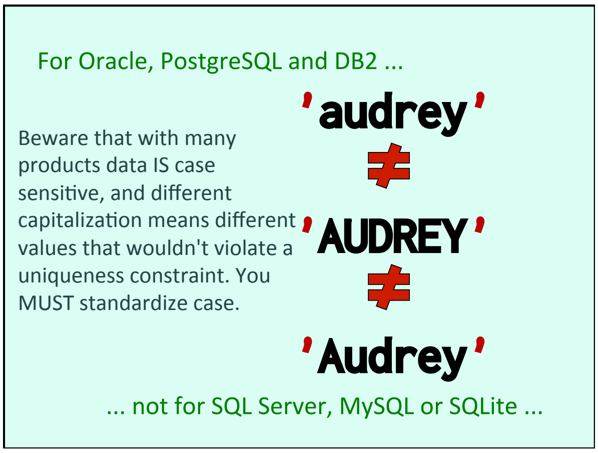
```
create table people (peopleid   int not null,
                                  primary key,
                    first_name varchar(30),
                    surname    varchar(30) not null,
                    born       numeric(4) not null,
                    died       numeric(4))
```

NOT NULL is a constraint. But there are many others. For instance, PRIMARY KEY tells which is the main key for the table, and indicates two things:
1) that the value is mandatory (the additional NOT NULL doesn't hurt but is redundant), and
2) that the values are unique (no duplicates allowed in the column)



```
create table people (peopleid   int not null
                                primary key,
                    first_name varchar(30),
                    surname    varchar(30) not null,
                    born       numeric(4) not null,
                    died       numeric(4),
                     unique (first_name, surname))
```

So far, nothing would prevent us from entering two Audrey Hepburns with different ids. To ensure we only have one, we must say that the combination (first_name, surname) is unique (for actors ...). Constraints on several columns at once (same story with primary keys) are specified at the end of the list of columns with a comma-delimited list of column names.

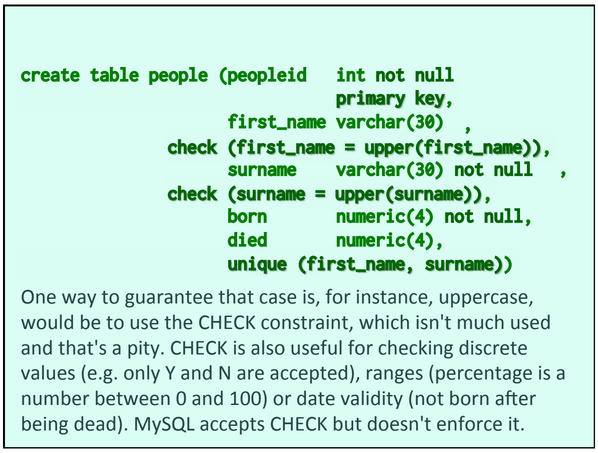


For Oracle, PostgreSQL and DB2 ...

Beware that with many products data IS case sensitive, and different capitalization means different values that wouldn't violate a uniqueness constraint. You MUST standardize case.

'audrey'
≠
'AUDREY'
≠
'Audrey'

... not for SQL Server, MySQL or SQLite ...



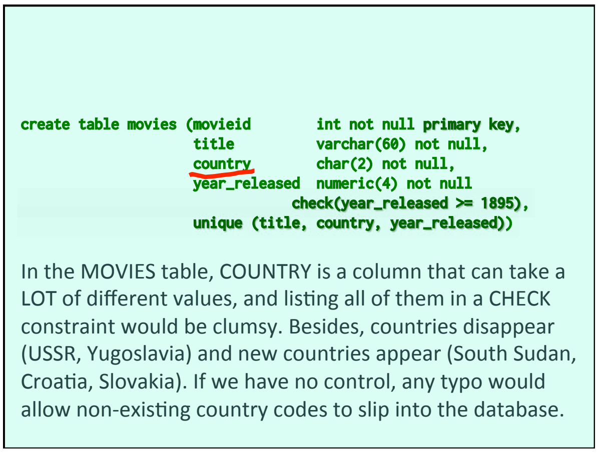
```
create table people (peopleid   int not null
                                primary key,
                     first_name varchar(30)  ,
               check (first_name = upper(first_name)),
                     surname    varchar(30) not null   ,
               check (surname = upper(surname)),
                     born       numeric(4) not null,
                     died       numeric(4),
                     unique (first_name, surname))
```

One way to guarantee that case is, for instance, uppercase, would be to use the CHECK constraint, which isn't much used and that's a pity. CHECK is also useful for checking discrete values (e.g. only Y and N are accepted), ranges (percentage is a number between 0 and 100) or date validity (not born after being dead). MySQL accepts CHECK but doesn't enforce it.

Picture: Douglas Robertson

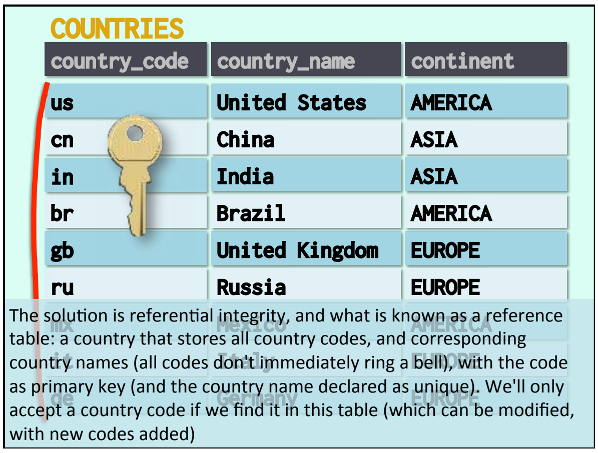
Referential Integrity

But there is far better than CHECK, which is a fairly static control: Referential Integrity



```
create table movies (movieid        int not null primary key,
                     title          varchar(60) not null,
                     country        char(2) not null,
                     year_released  numeric(4) not null
                                check(year_released >= 1895),
                     unique (title, country, year_released))
```

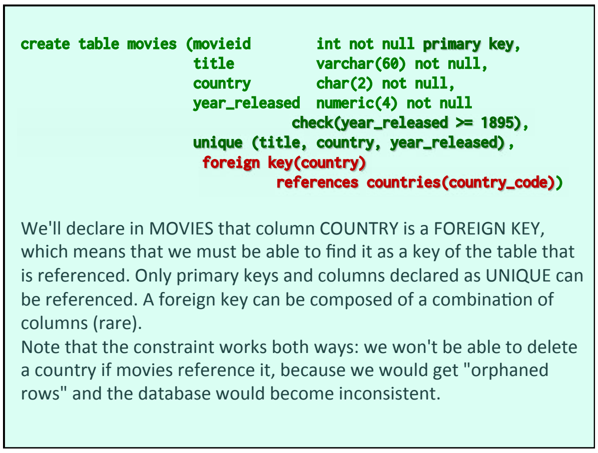
In the MOVIES table, COUNTRY is a column that can take a LOT of different values, and listing all of them in a CHECK constraint would be clumsy. Besides, countries disappear (USSR, Yugoslavia) and new countries appear (South Sudan, Croatia, Slovakia). If we have no control, any typo would allow non-existing country codes to slip into the database.



COUNTRIES

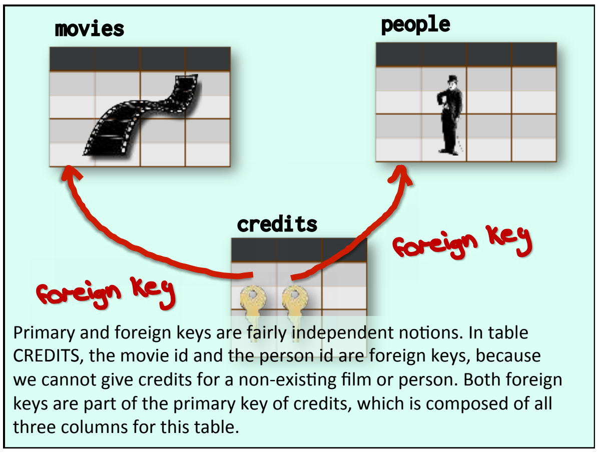
country_code	country_name	continent
us	United States	AMERICA
cn	China	ASIA
in	India	ASIA
br	Brazil	AMERICA
gb	United Kingdom	EUROPE
ru	Russia	EUROPE
mx	Mexico	AMERICA
it	Italy	EUROPE
de	Germany	EUROPE

The solution is referential integrity, and what is known as a reference table: a country that stores all country codes, and corresponding country names (all codes don't immediately ring a bell), with the code as primary key (and the country name declared as unique). We'll only accept a country code if we find it in this table (which can be modified, with new codes added)

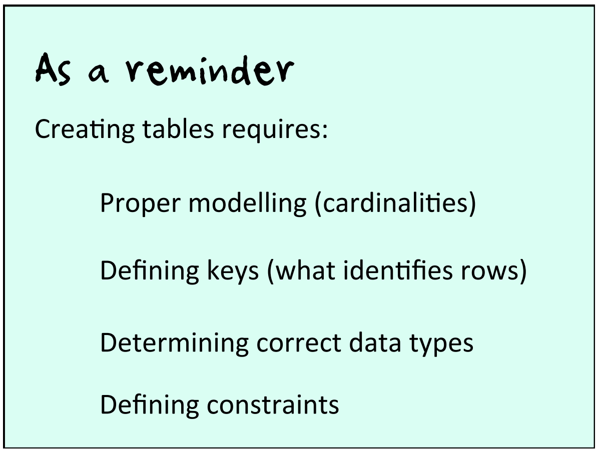


```
create table movies (movieid        int not null primary key,
                     title          varchar(60) not null,
                     country        char(2) not null,
                     year_released  numeric(4) not null
                                check(year_released >= 1895),
                     unique (title, country, year_released),
                      foreign key(country)
                               references countries(country_code))
```

We'll declare in MOVIES that column COUNTRY is a FOREIGN KEY, which means that we must be able to find it as a key of the table that is referenced. Only primary keys and columns declared as UNIQUE can be referenced. A foreign key can be composed of a combination of columns (rare).
Note that the constraint works both ways: we won't be able to delete a country if movies reference it, because we would get "orphaned rows" and the database would become inconsistent.



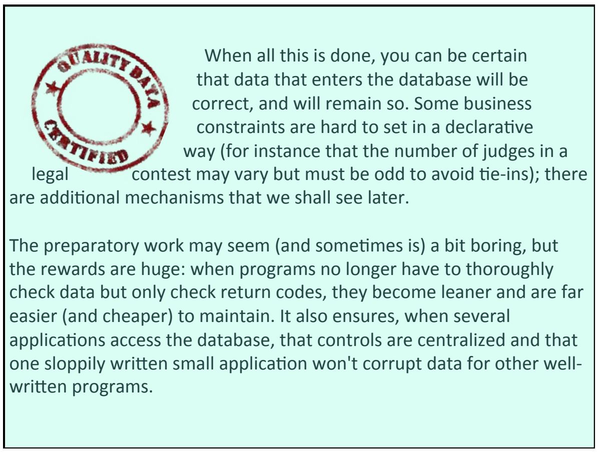
Primary and foreign keys are fairly independent notions. In table CREDITS, the movie id and the person id are foreign keys, because we cannot give credits for a non-existing film or person. Both foreign keys are part of the primary key of credits, which is composed of all three columns for this table.



As a reminder

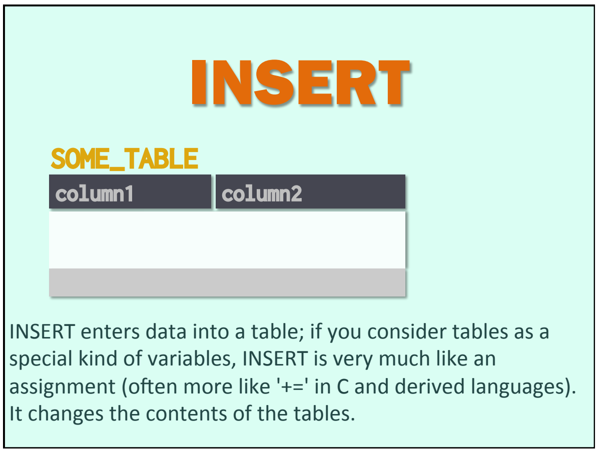
Creating tables requires:

- Proper modelling (cardinalities)
- Defining keys (what identifies rows)
- Determining correct data types
- Defining constraints


QUALITY DATA CERTIFIED

When all this is done, you can be certain that data that enters the database will be correct, and will remain so. Some business constraints are hard to set in a declarative way (for instance that the number of judges in a legal contest may vary but must be odd to avoid tie-ins); there are additional mechanisms that we shall see later.

The preparatory work may seem (and sometimes is) a bit boring, but the rewards are huge: when programs no longer have to thoroughly check data but only check return codes, they become leaner and are far easier (and cheaper) to maintain. It also ensures, when several applications access the database, that controls are centralized and that one sloppily written small application won't corrupt data for other well-written programs.

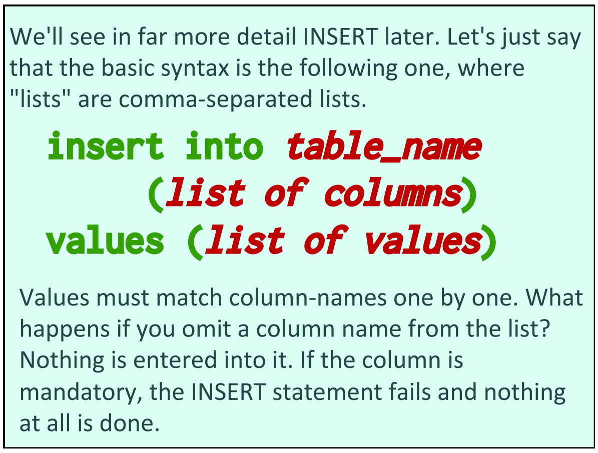


INSERT

SOME_TABLE

column1	column2

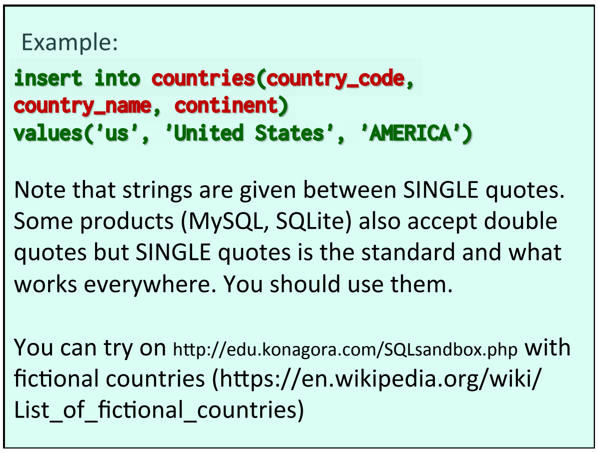
INSERT enters data into a table; if you consider tables as a special kind of variables, INSERT is very much like an assignment (often more like '+=' in C and derived languages). It changes the contents of the tables.



We'll see in far more detail INSERT later. Let's just say that the basic syntax is the following one, where "lists" are comma-separated lists.

```
insert into table_name
       (list of columns)
values (list of values)
```

Values must match column-names one by one. What happens if you omit a column name from the list? Nothing is entered into it. If the column is mandatory, the INSERT statement fails and nothing at all is done.

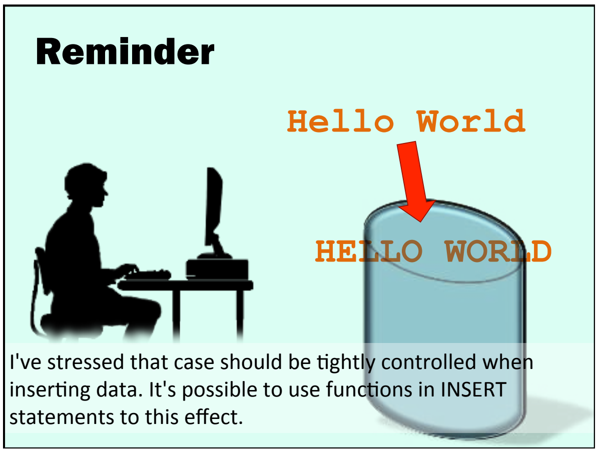


Example:

```
insert into countries(country_code,
country_name, continent)
values('us', 'United States', 'AMERICA')
```

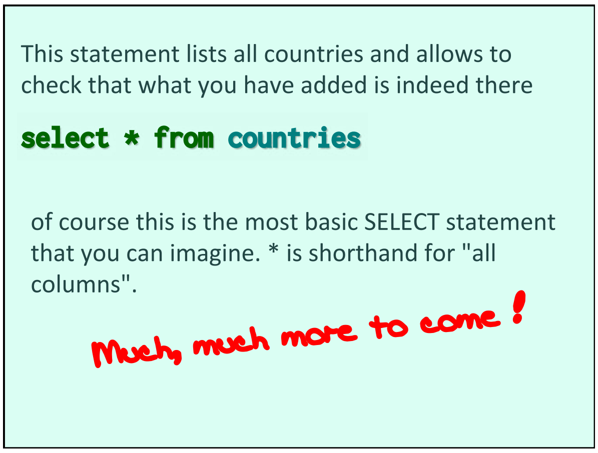
Note that strings are given between SINGLE quotes. Some products (MySQL, SQLite) also accept double quotes but SINGLE quotes is the standard and what works everywhere. You should use them.

You can try on http://edu.konagora.com/SQLsandbox.php with fictional countries (https://en.wikipedia.org/wiki/List_of_fictional_countries)



Reminder

I've stressed that case should be tightly controlled when inserting data. It's possible to use functions in INSERT statements to this effect.

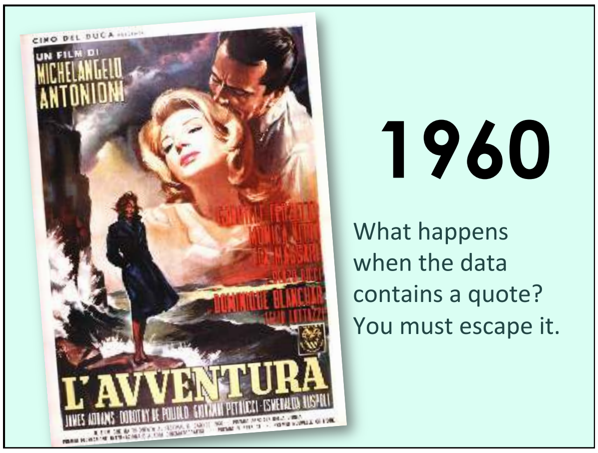


This statement lists all countries and allows to check that what you have added is indeed there

```
select * from countries
```

of course this is the most basic SELECT statement that you can imagine. * is shorthand for "all columns".

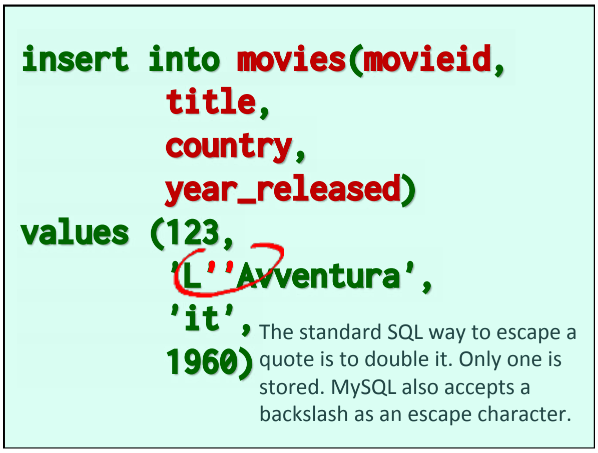
Much, much more to come !



CINO DEL DUCA
UN FILM DI
MICHELANGELO
ANTONIONI
L'AVVENTURA

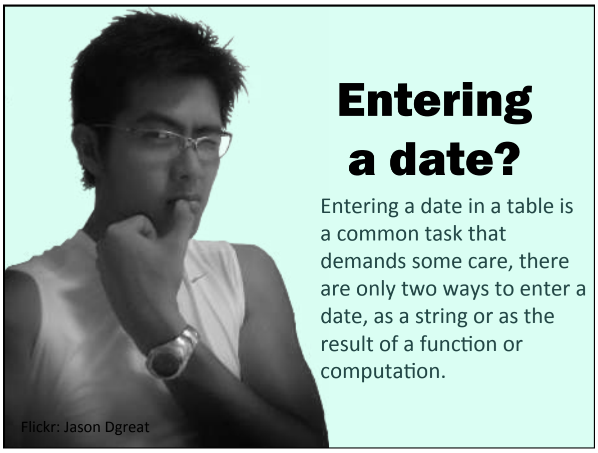
1960

What happens when the data contains a quote? You must escape it.



```
insert into movies(movieid,
       title,
       country,
       year_released)
values (123,
       'L''Avventura',
       'it',
       1960)
```

The standard SQL way to escape a quote is to double it. Only one is stored. MySQL also accepts a backslash as an escape character.



Entering a date?

Entering a date in a table is a common task that demands some care, there are only two ways to enter a date, as a string or as the result of a function or computation.

Flickr: Jason Dgreat

Most products support a default date format, and are able to automatically translate text that is inserted into a date or datetime column if this text matches the default format. You shouldn't rely on this and always specify the format, as here with Oracle:

```
to_date('07/20/1969',
        'MM/DD/YYYY')
```

The reason is that firstly a database administrator CAN change the default format, and secondly that dates can be ambiguous: you may read 11/05 as November 5th, for a European it will be May 11th.

CURRENT_DATE

SYSDATE ORACLE

MySQL CURDATE()

GETDATE() Microsoft SQL Server

The current date is often used. CURRENT_DATE (no time) and CURRENT_TIMESTAMP (time included) are recognized by all products. For historical reasons, all products also have their, still frequently used, own functions.

Don't forget (especially with Oracle) that there may be a time component that the default format for displaying dates may not show.

Flickr:Steve Jurvetson 79

SQL : THE database language.

Connection : server, port, database, username, password.

Create table: data types + constraints.

Keywords and identifiers: not case-sensitive

Data : can be CASE-SENSITIVE

Important things to remember about SQL and databases

Our example database

Film database

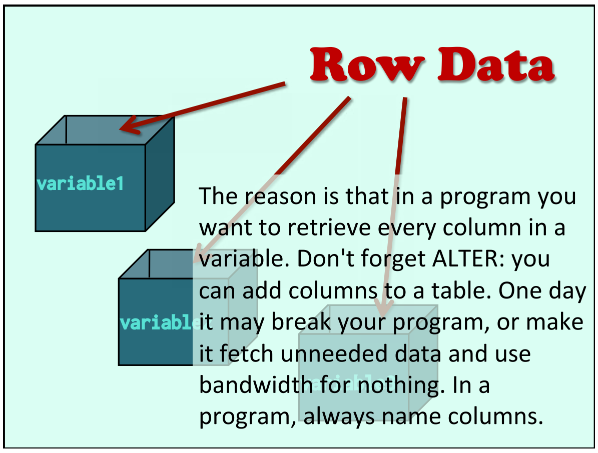
Available online at
http://edu.konagora.com/SQLsandbox.php

Bigger database available as a sqlite file in Sakai
http://sqlite.org/
http://sqlitebrowser.org/
http://www.squirrelsql.org/

select * from movies

To display the full content of a table, you can use select *. * is short-hand for "all columns" and is frequently used in interactive tools (especially when you don't remember column names ...)

You should not use, though, in programs.



Row Data

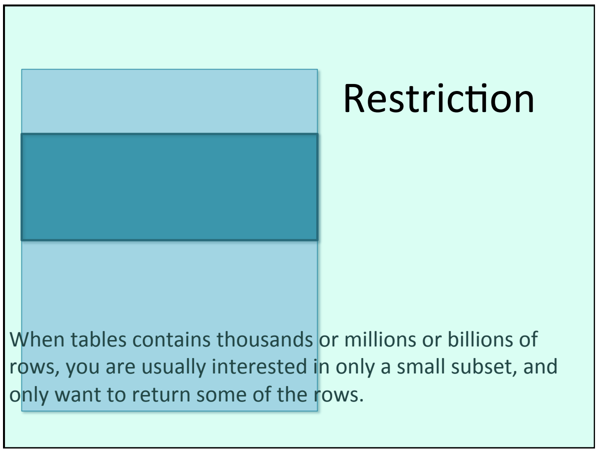
The reason is that in a program you want to retrieve every column in a variable. Don't forget ALTER: you can add columns to a table. One day it may break your program, or make it fetch unneeded data and use bandwidth for nothing. In a program, always name columns.

select * from *table*

≈

print table

Select * displays the full content of the table and is a bit like printing the table variable.



Restriction

When tables contains thousands or millions or billions of rows, you are usually interested in only a small subset, and only want to return some of the rows.