UNIVERSITÀ DEGLI STUDI
DI MILANO

DIPARTIMENTO DI INFORMATICA

La gestione dei dati da un punto di vista informatico

Dario Malchiodi –
`malchiodi.di.unimi.it`

Iniziamo con una domanda:

che cos'è l'informatica?

E se vi avessi chiesto:

che cos'è l'astrofisica?

Che cos'è l'astrofisica

- Direste che il lavoro di un astronomo consiste nell'utilizzare un telescopio?

Che cos'è l'informatica

We need to do away with the myth that computer science is about computers. Computer science is no more about computers than astronomy is about telescopes, biology is about microscopes or chemistry is about beakers and test tubes. Science is not about tools, it is about how we use them and what we find out when we do.

[Fellows, Parberry]

Informatica

Che cos'è?

- **Elaborazione** (trasformazione)
- **automatica** (eseguibile in modo meccanico)
- dell'**informazione** (al fine di ridurre l'incertezza)

Le **basi di dati** rappresentano uno strumento che riguarda tutti e tre questi aspetti.

Riferimenti bibliografici

- R. Elmasri, S.B. Navathe, Sistemi di basi di dati - Fondamenti, Pearson-Addison Wesley, 2011 (*warning*: gli argomenti che vedremo noi corrispondono ai capitoli 5, 6 e 8, ma tenete conto che si tratta di un libro di testo che tratta gli argomenti in modo abbastanza approfondito).

Database (o base di dati, o DB)

Insieme di dati

- tra loro coerentemente collegati,
- il cui significato può essere memorizzato (*persistenza*),
- relativi a un *universo del discorso* (o *minimondo*) che rappresenta una proiezione del mondo reale,
- progettato con uno scopo specifico.

Data Base Management System (DBMS)

- Software che permette di
  - creare,
  - manutenere,
  - interrogare
  - e modificare

  un database.

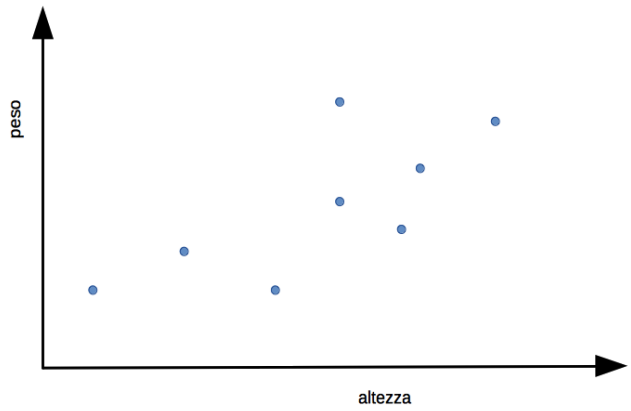
Vantaggi nell'uso di un DB

- Separazione tra dati e programmi
- *Viste* multiple
- Rappresentazione di relazioni complesse tra i dati
- Condivisione sicura dei dati (transazioni)
- Protezione dei dati
- Limitazione della ridondanza
- Vincoli di integrità
- Accesso efficiente
- Backup / recovery

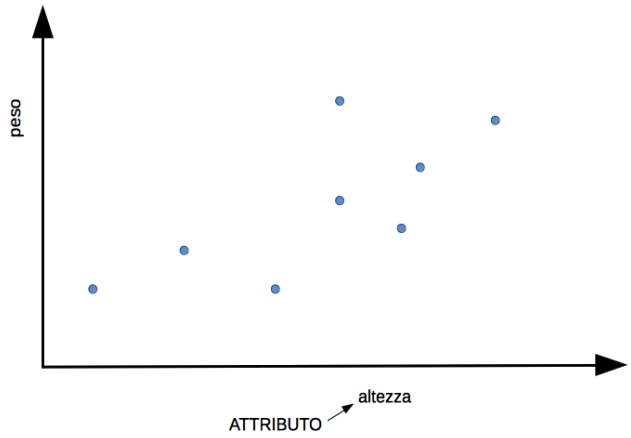
Basi di dati relazionali

- DB descritto come un insieme di *relazioni*
- relazione: un insieme di *entità* (e.g., i pazienti di uno studio medico)
- entità: oggetto o concetto dell'universo del discorso (e.g., la signora Maria Rossi, oppure una patologia, o ancora un medicinale)
- *attributo*: proprietà che descrive le entità (e.g., il peso di un paziente, o la sua altezza)
- *associazione*: rapporto tra entità (e.g., tra pazienti e medicinali)
- *schema*: descrizione di come è strutturato il DB

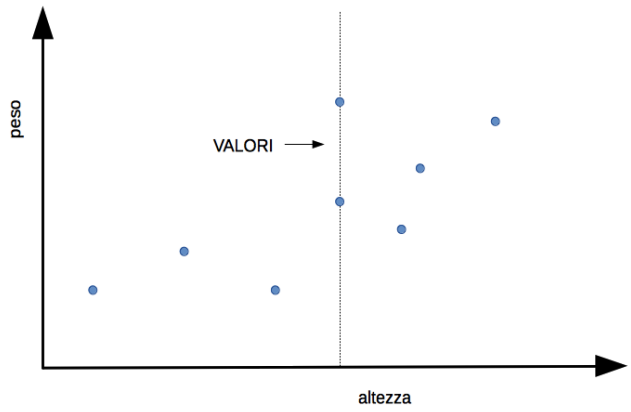
Un esempio



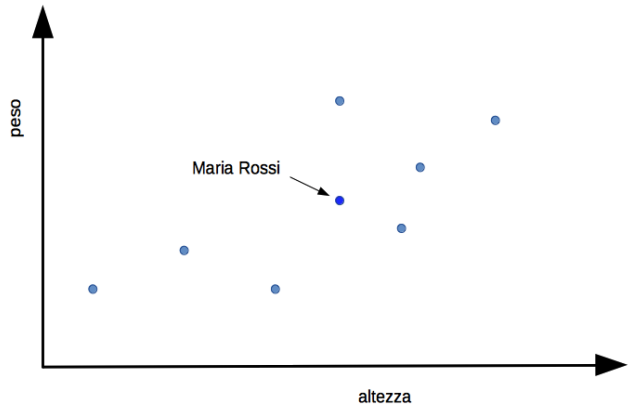
Un esempio



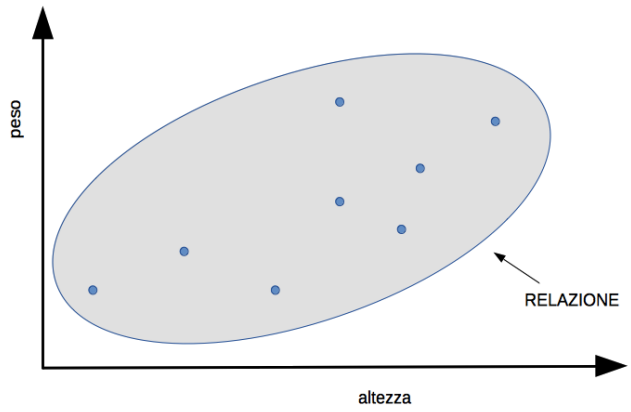
Un esempio



Un esempio



Un esempio



Le relazioni che considereremo (1)

city

ID	Name	CountryCode	District	Population

Le relazioni che considereremo (2)

Out[1]:

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238

Le relazioni che considereremo (3)

country

Code	Name	Continent	Region
SurfaceArea	IndepYear	Population	LifeExpectancy
GNP	GNPOld	LocalName	GovernmentForm
HeadOfState	Capital	Code2	

Le relazioni che considereremo (4)

Out[2]:

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy
ABW	Aruba	North America	Caribbean	193.0	NaN	103000	78.4
AFG	Afghanistan	Asia	Southern and Central Asia	652090.0	1919.0	22720000	45.9
AGO	Angola	Africa	Central Africa	1246700.0	1975.0	12878000	38.3
AIA	Anguilla	North America	Caribbean	96.0	NaN	8000	76.1
ALB	Albania	Europe	Southern Europe	28748.0	1912.0	3401200	71.6
AND	Andorra	Europe	Southern Europe	468.0	1278.0	78000	83.5
ANT	Netherlands Antilles	North America	Caribbean	800.0	NaN	217000	74.7
ARE	United Arab Emirates	Asia	Middle East	83600.0	1971.0	2441000	74.1
ARG	Argentina	South America	South America	2780400.0	1816.0	37032000	75.1
ARM	Armenia	Asia	Middle East	29800.0	1991.0	3520000	66.4

Le relazioni che considereremo (5)

countrylanguage

CountryCode	Language	IsOfficial	Percentage

Le relazioni che considereremo (6)

Out[3]:

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	1.9
AFG	Uzbek	F	8.8
AGO	Ambo	F	2.4

SQL

- SQL (Structured Query Language) è un linguaggio che permette di
  - definire una base di dati
  - consultare una base di dati
  - aggiornare una base di dati
- Le *frasi* scritte in questo linguaggio prendono il nome di *query*

mySQL

- mySQL è un DBMS relazionale open source che useremo come riferimento
  - ma la maggior parte delle query che vedremo funzionerebbe anche con altri DBMS

Terminologia SQL

A	*B*	*C*	…
			…
a	*b*	*c*	…
			…
			…

- una relazione viene implementata costruendo una *tabella*

Terminologia SQL

A	*B*	*C*	…
			…
a	*b*	*c*	…
			…
			…

- le *righe* (o *tuple*) di questa tabella rappresentano le entità

Terminologia SQL

A	B	C	…
			…
a	b	c	…
			…
			…

- le sue *colonne* rappresentano invece gli attributi

Notazione

A	B	C	…
			…
a	b	c	…
			…
			…

- $R(A, B, C, \ldots)$ indica una rlazione R che lega gli attributi A, B, C e così via

Notazione

A	B	C	…
			…
a	b	c	…
			…
			…

- Ogni entità nella relazione verrà indicata con una tupla $(a, b, c, \ldots) \in R(A, B, C, \ldots)$

Per esempio

Peso	Altezza
p	a

- R(Peso, Altezza) indica la relazione dell'esempio precedente
- i suoi attributi sono Peso e Altezza

Per esempio

Peso	Altezza
p	*a*

- (p, a), dove p e a indicano un peso e un'altezza specifici, è la tupla che corrisponde alla sig.ra Maria Rossi

Query SQL (1)

In teoria si può

- definire un intero schema
- creare tabelle
- creare viste

Per esempio

```
CREATE TABLE city (
  ID INT(11) NOT NULL AUTO_INCREMENT,
  Name CHAR(35) NOT NULL DEFAULT "",
  CountryCode CHAR(3) NOT NULL DEFAULT ""
,
  District CHAR(20) NOT NULL DEFAULT "",
  Population INT(11) NOT NULL DEFAULT 0,
  PRIMARY KEY (ID),
  KEY CountryCode (CountryCode),
  CONSTRAINT city_ibfk_1 FOREIGN KEY (Cou
ntryCode)
  REFERENCES country (Code)
);
```

Qualche dettaglio

È possibile indicare per ogni attributo:

- un *tipo* (`INT` , `FLOAT` , `CHAR` , `DATE` , ...)
- dei *vincoli* (`NOT NULL`)
- dei *valori predefiniti* (`DEFAULT`)
- delle *azioni predefinite* (`AUTO_INCREMENT`)
- delle *chiavi* di indicizzazione (`PRIMARY_KEY`)

Tipi di attributo

- Numerici (interi e con virgola)
- Alfanumerici (sequenze di caratteri–alfabetici, numerici o di altro tipo–che prendono il nome di *stringhe*)
  - a lunghezza fissa o variabile
- Logici (o *booleani*, indicano i possibili valori di verità *vero* e *falso*)
- Temporali (data, ora, data e ora)
- Monetari

Tipi di attributo

Perché è necessario specificare il tipo degli attributi?

- nella memoria (o nel disco fisso) di un computer, tutti i dati sono *codificati* in termini numerici
- e già che ci siamo, i numeri ottenuti sono specificati usando una serie di *bit* (cioè espressi usando solo 0 e 1)
- a seconda del *tipo* degli attributi viene utilizzata una codifica diversa

Tipi di dato

La sequenza 01000001, per esempio, può indicare

- il numero intero 65
- la lettera A (in codice ASCII)
- (ma anche altre cose)

È questo il motivo per cui è necessario indicare come *interpretare* le sequenze di bit in modo da trasformarli in modo corretto

Query SQL (2)

In pratica ci occuperemo di

- selezione
- inserimento
- aggiornamento
- cancellazione

Operazione di selezione

- Data una relazione $R(A, B, C, \ldots)$
- e una condizione C che in funzione dei valori di uno o più attributi può essere vera o falsa
- l'operazione di *selezione* produce la relazione $\sigma_C(R(A, B, C, \ldots))$ che si ottiene da R considerando tutte e sole le righe per cui C risulta vera

Query di selezione

Nella sua forma più semplice, una query di selezione permette di leggere **tutte** le righe di una tabella

```
SELECT * FROM <tabella>;
```

- il carattere `*` indica che vogliamo leggere i valori per tutte le colonne
- (dunque in questo caso la condizione C è sempre vera)
- non vedremo esempi di questo tipo (di solito è inutile visualizzare interamente una tabella)

Query di selezione

Una semplice variante delle query di selezione permette di leggere **alcune** righe di una tabella

```
SELECT * FROM <tabella> LIMIT <n>;
```

- il carattere `*` indica che vogliamo leggere i valori per tutte le colonne
- ma **anche** in questo caso la condizione C è sempre vera…
- …perché è la clausola `LIMIT` permette di limitare il numero di righe ottenute
- di norma non possiamo sapere **quali** righe verranno restituite

Per esempio

```
SELECT * FROM city LIMIT 3;
```

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800

Per esempio

```
SELECT * FROM country LIMIT 3;
```

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy
ABW	Aruba	North America	Caribbean	193.0	None	103000	78.4
AFG	Afghanistan	Asia	Southern and Central Asia	652090.0	1919	22720000	45.9
AGO	Angola	Africa	Central Africa	1246700.0	1975	12878000	38.3

Per esempio

```
SELECT * FROM countrylanguage LIMIT 3;
```

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7

Operazione di proiezione

- Data una relazione $R(A, B, C, \ldots)$
- e un sottoinsieme $\mathcal{E} \subseteq \{A, B, C, \ldots\}$ dei suoi attributi
- l'operazione di *proiezione* produce la relazione $\pi_{\mathcal{E}}(R(A, B, C, \ldots))$ che si ottiene da R eliminando tutte e sole le colonne che corrispondono agli attributi in $\mathcal{E}$

Per esempio, la relazione $\pi_{\{A,C\}}(R(A, B, C))$ conterrà una tupla (a, c) in corrispondenza di ogni tupla $(a, b, c) \in R(A, B, C)$.

Proiezione

Si può abbinare una query di selezione a un'operazione di *proiezione* quando ci interessa visualizzare un sottoinsieme dei valori per gli attributi

- omettendo il carattere `*`
- specificando i nomi degli attributi separati da virgola

```
SELECT Name, Population FROM city LIMIT 3;
```

Name	Population
Kabul	1780000
Qandahar	237500
Herat	186800

Selezionare da più tabelle

- È possibile indicare più tabelle in una query di selezione
- Se non ci sono problemi di ambiguità, gli attributi si possono indicare come nelle query viste finora
- Il risultato è il *prodotto cartesiano* corrispondente

```
SELECT Continent, Language FROM country, countrylanguage LIMIT 10;
```

Continent	Language
North America	Dutch
Asia	Dutch
Africa	Dutch
North America	Dutch
Europe	Dutch
Europe	Dutch
North America	Dutch
Asia	Dutch
South America	Dutch
Asia	Dutch

Selezionare da più tabelle

Il tutto si vede meglio aggiungendo la parola chiave `DISTINCT` che elimina i doppioni dai risultati della query

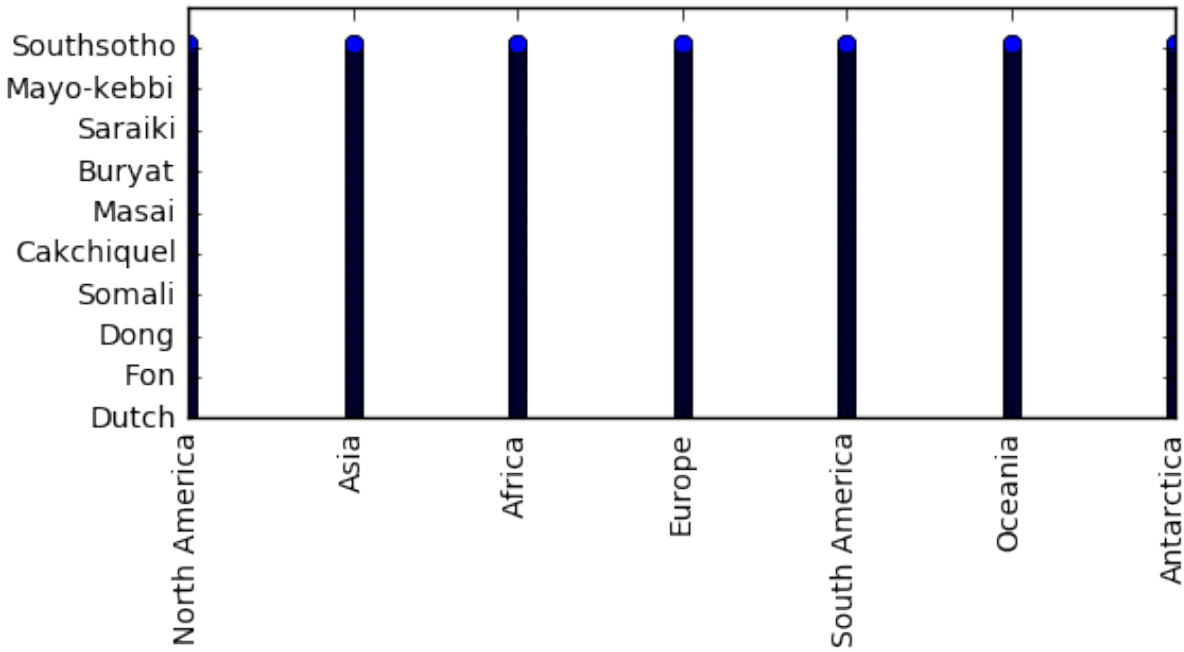
```
SELECT DISTINCT Continent, Language FROM country, countrylanguage LI
MIT 10;
```

Continent	Language
North America	Dutch
Asia	Dutch
Africa	Dutch
Europe	Dutch
South America	Dutch
Oceania	Dutch
Antarctica	Dutch
North America	English
Asia	English
Africa	English

(il che significa che i risultati di una query non si possono descrivere tramite insiemi, ma piuttosto come *multi-insiemi*)

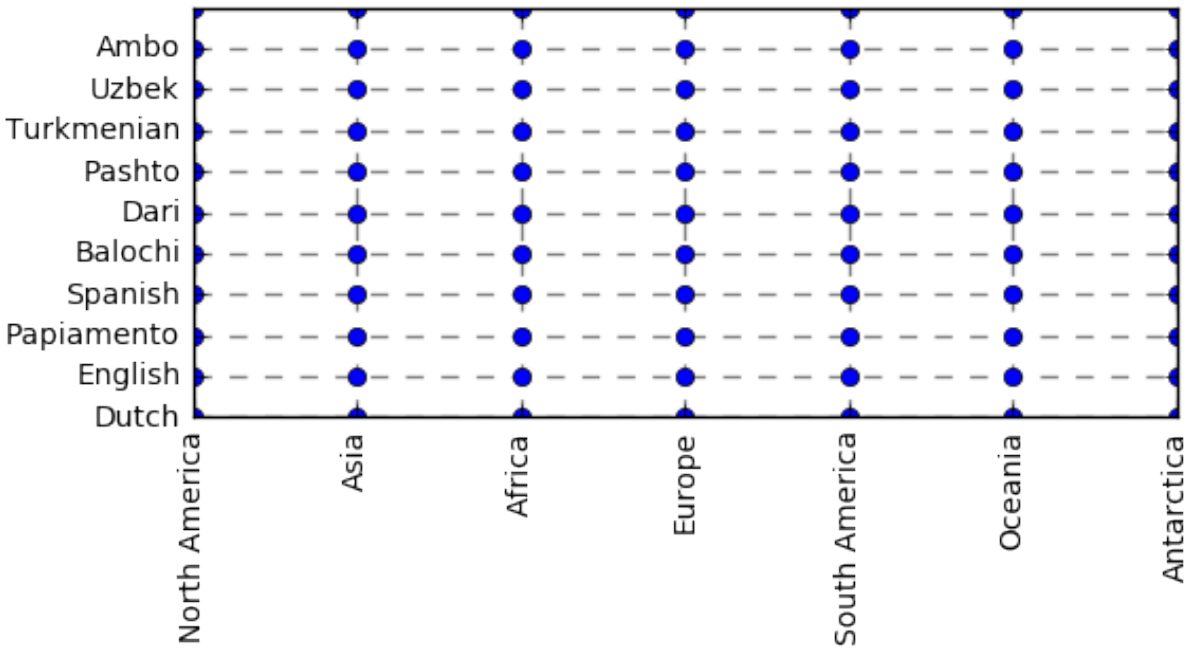
Prodotto cartesiano (2)

È più facile da visualizzare graficamente...



Prodotto cartesiano (3)

...soprattutto facendo uno zoom.



Selezione condizionale

Oltre a filtrare rispetto alle colonne (attributi), è possibile includere solo alcune righe (entità) specificando la parola chiave `WHERE` seguita da una condizione.

```
SELECT Name, Population FROM city WHERE Population < 500;
```

Name	Population
West Island	167
Adamstown	42
Fakaofo	300
Città del Vaticano	455

Selezione condizionale e stringhe

- Gli attributi *alfanumerici* hanno come valori successioni di generici caratteri
- In genere si parla di *stringhe*
- Per indicare una stringa in una condizione è necessario delimitarla usando dei doppi apici

```
SELECT Name FROM city WHERE CountryCode="MCO";
```

Name
Monte-Carlo
Monaco-Ville

Operatori disponibili

- relazioni aritmetiche (minore `<` , minore o uguale `<=` , uguale `=` , **diverso** `<>` , ...)
- inclusione in intervalli (`BETWEEN <lower> AND <upper>`), *anche temporali*

Operatori disponibili

```
SELECT Name, LocalName FROM country WHERE Name <> LocalName LIMIT 10
;
```

Name	LocalName
Afghanistan	Afganistan/Afqanestan
Albania	Shqipëria
Netherlands Antilles	Nederlandse Antillen
United Arab Emirates	Al-Imarat al-´Arabiya al-Muttahida
Armenia	Hajastan
American Samoa	Amerika Samoa
Antarctica	–
French Southern territories	Terres australes françaises
Austria	Österreich
Azerbaijan	Azärbaycan

Operatori disponibili

```
SELECT Name, Population FROM city WHERE Population BETWEEN 500 AND 1
000;
```

Name	Population
South Hill	961
The Valley	595
Flying Fish Cove	700
Bantam	503
Yaren	559
Alofi	682
Kingston	800

Operatori disponibili

```
SELECT Name, Population FROM city WHERE Population > 500 AND Populat
ion < 1000;
```

Name	Population
South Hill	961
The Valley	595
Flying Fish Cove	700
Bantam	503
Yaren	559
Alofi	682
Kingston	800

Operatori disponibili

- inclusione in insiemi (`IN ( <valore>, ..., <valore>)`)
- operatori logici (`AND` , `OR` , `NOT`)

```
SELECT Name FROM country WHERE Continent IN ("Africa", "Asia") AND G
overnmentForm = "Monarchy";
```

Name
Bhutan
Qatar
Saudi Arabia
Swaziland

Operatori disponibili

- *pattern matching* (`LIKE <expr>`)
  - `%` indica una qualsiasi sequenza di caratteri

```
SELECT Name FROM country WHERE Name LIKE "D%";
```

Name
Djibouti
Dominica
Denmark
Dominican Republic

Operatori disponibili

- *pattern matching* (`LIKE <expr>`)
  - `%` indica una qualsiasi sequenza di caratteri

```
SELECT Name FROM country WHERE Name LIKE "%q%";
```

Name
Equatorial Guinea
Iraq
Mozambique
Martinique
Qatar
Saint Pierre and Miquelon

Operatori disponibili

- *pattern matching* (`LIKE <expr>`)
  - `%` indica una qualsiasi sequenza di caratteri
  - `_` indica un qualsiasi carattere

```
SELECT Name FROM country WHERE Name LIKE "E_UA%";
```

Name
Ecuador
Equatorial Guinea

Esecuzione di operazioni (1)

È possibile indicare delle espressioni che contengono uno o più attributi. Per esempio si può calcolare come è variato il prodotto interno in due momenti successivi.

```
SELECT Name, GNP/GNPOld FROM country WHERE Population > 1000000 LIMI
T 3;
```

Name	GNP/GNPOld
Afghanistan	None
Angola	0.832665
Albania	1.282

Valori mancanti

- La parola chiave `None` viene utilizzata dalla libreria che utilizziamo per indicare dei valori mancanti
- A seconda dello strumento che usiamo per interagire con il DB potrebbe essere utilizzato un altro simbolo (per esempio `NONE`, oppure `NA`)

Valori mancanti

- Nell'esempio di prima, il risultato ha un valore indefinito in quanto almeno uno degli operandi nell'espressione è un valore mancante.

```
SELECT GNP, GNPOld FROM country WHERE Name="Afghanistan";
```

GNP	GNPOld
5976.0	None

Valori mancanti

- Perché non si conosce il valore dell'attributo per una tupla
- Perché questo valore, pur esistendo, non è disponibile
- Perché per la tupla di questione il valore dell'attributo non ha senso (è indefinito)

Esecuzione di operazioni (2)

Esprimere la popolazione in milioni di abitanti.

```
SELECT Name, Population/1000000 FROM city WHERE Population > 1000000
LIMIT 3;
```

Name	Population/1000000
Kabul	1.7800
Alger	2.1680
Luanda	2.0220

Uso di alias

La parola chiave `AS` permette di introdurre un *alias* ovvero un nome alternativo per un'espressione

- è comodo per mostrare i risultati senza scrivere l'espressione come intestazione di colonna
- (anche per accedere ai risultati in modo programmatico e in query più complicate)

```
SELECT Name, Population/1000000 AS Millions FROM city WHERE Populati
on > 1000000 LIMIT 3;
```

Name	Millions
Kabul	1.7800
Alger	2.1680
Luanda	2.0220

Selezionare da più tabelle

Se in una query compaiono più tabelle e vi sono attributi con lo stesso nome, è possibile risolvere l'omonimia usando la sintassi `tabella.attributo`

```
SELECT city.Name, country.Name  FROM city, country LIMIT 10;
```

Name	Name
Kabul	Aruba
Kabul	Afghanistan
Kabul	Angola
Kabul	Anguilla
Kabul	Albania
Kabul	Andorra
Kabul	Netherlands Antilles
Kabul	United Arab Emirates
Kabul	Argentina
Kabul	Armenia

Selezionare da più tabelle

In casi di ambiguità gli alias permettono di visualizzare i risultati rinominando le colonne coinvolte

```
SELECT country.Name as Country, city.Name as City FROM city, country
LIMIT 10;
```

Country	City
Aruba	Kabul
Afghanistan	Kabul
Angola	Kabul
Anguilla	Kabul
Albania	Kabul
Andorra	Kabul
Netherlands Antilles	Kabul
United Arab Emirates	Kabul
Argentina	Kabul
Armenia	Kabul

Operazioni di join

- La query dell'esempio precedente restituisce il prodotto cartesiano delle tabelle city e country
- ovviamente nella maggior parte delle righe i due valori visualizzati rappresentano stati e città non necessariamente collegati tra loro

Operazioni di join

- Per ottenere la lista degli stati e delle corrispondenti capitali si può tenere conto del fatto che
  - la tabella city contiene un attributo countrycode
  - la tabella country contiene un attributo code
  - i due attributi hanno come valori delle sigle internazionali che individuano gli stati
- quindi si può utilizzare un'opportuna clausola `WHERE`

Operazioni di join

```
SELECT country.Name, country.Code, city.Name FROM city, country WHER
E city.CountryCode = country.Code LIMIT 10;
```

Name	Code	Name
Aruba	ABW	Oranjestad
Afghanistan	AFG	Kabul
Afghanistan	AFG	Qandahar
Afghanistan	AFG	Herat
Afghanistan	AFG	Mazar-e-Sharif
Angola	AGO	Luanda
Angola	AGO	Huambo
Angola	AGO	Lobito
Angola	AGO	Benguela
Angola	AGO	Namibe

Operazioni di join

Ovviamente se lo scopo del codice che individua lo stato è solo quello di mettere in relazione città con stati, è possibile eliminarlo dai risultati grazie a una proiezione

```
SELECT country.Name, city.Name FROM city, country WHERE city.Country
Code = country.Code LIMIT 10;
```

Name	Name
Aruba	Oranjestad
Afghanistan	Kabul
Afghanistan	Qandahar
Afghanistan	Herat
Afghanistan	Mazar-e-Sharif
Angola	Luanda
Angola	Huambo
Angola	Lobito
Angola	Benguela
Angola	Namibe

Operazioni di join

In casi come questo l'uso di alias permette di dare dei nomi non ambigui alle colonne

```
SELECT country.Name AS stato, city.Name as Citta FROM city, country
WHERE city.CountryCode = country.Code LIMIT 10;
```

stato	Citta
Aruba	Oranjestad
Afghanistan	Kabul
Afghanistan	Qandahar
Afghanistan	Herat
Afghanistan	Mazar-e-Sharif
Angola	Luanda
Angola	Huambo
Angola	Lobito
Angola	Benguela
Angola	Namibe

Operazioni di join

Ricapitolando:

- si è partiti dal prodotto cartesiano di due tabelle
- tramite `WHERE` si sono isolate le righe in cui i codici dello stato coincidevano
- si è effettuata una proiezione per visualizzare solo i nomi di stati e città
- sono stati usati due alias per avere intestazioni di colonne significative

Operazioni di join

L'operazione descritta prende il nome di *join*, ed è così importante da meritare una sintassi speciale:

```
SELECT <attributi> FROM <tabella> JOIN <tabella> ON <campi uguali>;
```

```
SELECT country.Name as Stato, city.Name as Citta FROM city JOIN coun
try ON city.CountryCode = country.Code LIMIT 10;
```

Stato	Citta
Aruba	Oranjestad
Afghanistan	Kabul
Afghanistan	Qandahar
Afghanistan	Herat
Afghanistan	Mazar-e-Sharif
Angola	Luanda
Angola	Huambo
Angola	Lobito
Angola	Benguela
Angola	Namibe

Equijoin

- L'operazione di join appena vista è incardinata sul fatto che i due attributi considerati devono assumere lo stesso valore
- In simboli, se $S(A, B)$ e $T(B, C)$ indicano le due relazioni di partenza, il risultato dell'operazione di join è la relazione $S \bowtie T(A, B_S, B_T, C)$ contiene una tupla (a, b, b, c) se e solo se $(a, b) \in S$ e $(b, c) \in T$
- In casi come questo si parla di *equijoin*

Equijoin e join naturale

- Siccome di norma ha poco senso ripetere l'attributo comune b, si introduce l'operazione di *join naturale* ottenuta dalla equijoin ed eliminando uno dei doppioni
- In teoria è possibile considerare altri tipi di relazioni di join, non necessariamente basata sulla relazione di uguaglianza, ma noi non le considereremo

Un esempio (1)

Che cosa succederebbe se eseguissimo il join sui nomi di città e stati?

```
SELECT city.Name, country.Name FROM city JOIN country ON city.Name =
country.Name;
```

Name	Name
Djibouti	Djibouti
Mexico	Mexico
Gibraltar	Gibraltar
Armenia	Armenia
Kuwait	Kuwait
Macao	Macao
San Marino	San Marino
Singapore	Singapore

Un esempio (2)

Per capire...

```
SELECT city.Name, city.CountryCode, country.Code, country.Name FROM
city JOIN country ON city.Name=country.Name;
```

Name	CountryCode	Code	Name
Djibouti	DJI	DJI	Djibouti
Mexico	PHL	MEX	Mexico
Gibraltar	GIB	GIB	Gibraltar
Armenia	COL	ARM	Armenia
Kuwait	KWT	KWT	Kuwait
Macao	MAC	MAC	Macao
San Marino	SMR	SMR	San Marino
Singapore	SGP	SGP	Singapore

Raggruppamento e aggregazione

- Come possiamo sommare assieme il numero di abitanti di tutte le città di uno stesso stato?
- Operazioni di questo tipo prendono il nome di *raggruppamento e aggregazione*
  - si specifica un attributo G si cui effettuare raggruppamento
  - si specifica un'operazione di aggregazione γ e un attributo A
  - l'operazione γ viene applicata a tutti i valori dell'attributo A in tuple per cui G assume uno stesso valore

Raggruppamento e aggregazione

- Nel nostro caso
  - raggruppamento su countrycode
  - quindi vengono considerate insieme tutte le tuple che corrispondono a città di uno stesso stato
  - aggregazione su population
  - operatore di aggregazione: somma

Raggruppamento e aggregazione

```
SELECT CountryCode, SUM(Population) FROM city GROUP BY CountryCode L
IMIT 5;
```

CountryCode	SUM(Population)
ABW	29034
AFG	2332100
AGO	2561600
AIA	1556
ALB	270000

Raggruppamento e aggregazione

Un esempio più complicato: associare a ogni stato il numero di abitanti in città

- Join tra country e city
- raggruppamento su countrycode
- aggregazione tramite somma su city.population
- (esiste anche un attributo population in country, ma fa riferimento all'intero stato)

Raggruppamento e aggregazione

```
SELECT country.Name, SUM(city.Population) FROM country JOIN city ON
Code=CountryCode GROUP BY CountryCode LIMIT 10;
```

Name	SUM(city.Population)
Aruba	29034
Afghanistan	2332100
Angola	2561600
Anguilla	1556
Albania	270000
Andorra	21189
Netherlands Antilles	2345
United Arab Emirates	1728336
Argentina	19996563
Armenia	1633100

Raggruppamento e aggregazione

Operatori di aggregazione

- somma (`SUM`)
- minimo e massimo (`MIN` e `MAX`)
- conteggio (`COUNT`)
- media (`AVG`)
- ...

Un esempio

Selezioniamo per ogni nazione il linguaggio ufficiale

```
SELECT  country.Name, countrylanguage.Language FROM country JOIN cou
ntrylanguage ON country.Code=countrylanguage.CountryCode LIMIT 10;
```

Name	Language
Aruba	Dutch
Aruba	English
Aruba	Papiamento
Aruba	Spanish
Afghanistan	Balochi
Afghanistan	Dari
Afghanistan	Pashto
Afghanistan	Turkmenian
Afghanistan	Uzbek
Angola	Ambo

Un esempio

Selezioniamo per ogni nazione il linguaggio ufficiale e la percentuale di diffusione

```
SELECT Name, Language, Percentage FROM country JOIN countrylanguage
ON Code=CountryCode WHERE IsOfficial="T" LIMIT 10;
```

Name	Language	Percentage
Aruba	Dutch	5.3
Afghanistan	Dari	32.1
Afghanistan	Pashto	52.4
Anguilla	English	0.0
Albania	Albaniana	97.9
Andorra	Catalan	32.3
Netherlands Antilles	Dutch	0.0
Netherlands Antilles	Papiamento	86.2
United Arab Emirates	Arabic	42.0
Argentina	Spanish	96.8

Un esempio

Selezioniamo per ogni nazione il linguaggio più parlato

```
SELECT Name, Language, MAX(Percentage) FROM country JOIN countrylang
uage ON Code=CountryCode WHERE IsOfficial="T" GROUP BY Name, Languag
e LIMIT 10;
```

Name	Language	MAX(Percentage)
Afghanistan	Dari	32.1
Afghanistan	Pashto	52.4
Albania	Albaniana	97.9
Algeria	Arabic	86.0
American Samoa	English	3.1
American Samoa	Samoan	90.6
Andorra	Catalan	32.3
Anguilla	English	0.0
Antigua and Barbuda	English	0.0
Argentina	Spanish	96.8

Funzioni

Oltre agli operatori di aggregazione esistono altre funzioni che permettono di scrivere espressioni complesse nelle operazioni di selezione

- somma, sottrazione, prodotto, divisione, ... (`+` , `-` , `*` , `/` , ...)
- troncamento e arrotondamento (`TRUNCATE` , `ROUND`)
- operazioni legate al tempo (calcolo di lunghezze di intervalli temporali, estrazione di elementi in una data, ...)
- operatori logici (congiunzioni, disgiunzioni, negazioni, ...)
- e tanto altro

Un esempio

```
SELECT CountryCode, ROUND(SUM(Population)/10000000, 2) AS Population
FROM city GROUP BY CountryCode LIMIT 10;
```

CountryCode	Population
ABW	0.00
AFG	0.23
AGO	0.26
AIA	0.00
ALB	0.03
AND	0.00
ANT	0.00
ARE	0.17
ARG	2.00
ARM	0.16

Estensioni del concetto di join

- Join *interne* che escludono le tuple che non soddisfano il criterio di join (sono quelle che abbiamo visto finora)
- Join *esterne* che includono nei risultati anche le tuple delle relazioni originali che non soddisfano il criterio di join
- Join esterna sinistra (include comunque le tuple della prima relazione, aggiungendo un valore nullo se queste non soddisfano il criterio)
- Join esterna destra (include comunque le tuple della seconda relazione, aggiungendo un valore nullo st queste non soddisfano il criterio)

Un esempio di join esterna sinistra

- Ci sono degli stati in corrispondenza dei quali non è indicato alcun linguaggio?
- (nota: per selezionare gli eventuali valori mancanti si usa `IS NULL`)

```
SELECT Name, Language FROM country LEFT OUTER JOIN countrylanguage O
N Code=CountryCode WHERE Language IS NULL;
```

Name	Language
Antarctica	None
French Southern territories	None
Bouvet Island	None
Heard Island and McDonald Islands	None
British Indian Ocean Territory	None
South Georgia and the South Sandwich Islands	None

Un esempio di join esterna sinistra

Non saremmo riusciti a trovare questi stati usando una join interna

```
SELECT Name, Language FROM country JOIN countrylanguage ON Code=Coun
tryCode WHERE Name="Antarctica";
```

Name	Language

Query complesse e operatori insiemistici

- `(<query 1>) UNION (<query 2>);` calcola due query e restituisce la loro *unione* (una tupla si troverà nel risultato se è stata restituita da **almeno** una query)
- `(<query 1>) INTERSECT (<query 2>);` calcola due query e restituisce la loro *intersezione* (una tupla si troverà nel risultato se è stata restituita da **entrambe** le query)
- `(<query 1>) EXCEPT (<query 2>);` calcola due query e restituisce la loro *differenza* (una tupla si troverà nel risultato se è stata restituita dalla prima query ma **non** dalla seconda)

Un esempio

Elenchiamo tutte le città che si trovano nell'isola di Malta o nel principato di Monaco.

Un esempio

- Iniziamo recuperando il codice corrispondente al principato di Monaco

```
SELECT Code FROM country WHERE Name="Monaco";
```

Code
MCO

Un esempio

- Facciamo lo stesso con Malta

```
SELECT Code FROM country WHERE Name="Malta";
```

Code
MLT

Un esempio

- Siamo ora in grado di scrivere una query che estrae le città maltesi
- una relativa a quelle monegasche
- e calcolare la loro unione

```
(SELECT Name FROM city WHERE CountryCode="MLT") UNION (SELECT Name F
ROM city WHERE CountryCode="MCO");
```

Name
Birkirkara
Valletta
Monte-Carlo
Monaco-Ville

Un esempio

Avremmo potuto ottenere lo stesso risultato usando un'operazione di selezione basata su una condizione che contiene una disgiunzione logica.

```
SELECT Name FROM city WHERE CountryCode="MLT" OR CountryCode="MCO";
```

Name
Monte-Carlo
Monaco-Ville
Birkirkara
Valletta

Un esempio

Oppure avremmo potuto usare l'operatore `IN`

```
SELECT Name FROM city WHERE CountryCode IN ("MLT", "MCO");
```

Name
Monte-Carlo
Monaco-Ville
Birkirkara
Valletta

Ordinamento dei risultati

La clausola `ORDER BY` permette di visualizzare le righe ordinandole in funzione dell'espressione che segue la clausola

```
SELECT Name FROM city WHERE CountryCode IN ("MLT", "MCO") ORDER BY N
ame;
```

Name
Birkirkara
Monaco-Ville
Monte-Carlo
Valletta

Ordinamento dei risultati

- L'ordinamento viene fatto mostrando i valori dal più piccolo al più grande
- a meno che non si specifichi la parola chiave `DESC` (abbreviazione di *descending*)
- (in teoria è anche possibile specificare `ASC` ending)

```
SELECT Name FROM city WHERE CountryCode IN ("MLT", "MCO") ORDER BY N
ame DESC;
```

Name
Valletta
Monte-Carlo
Monaco-Ville
Birkirkara

Un esempio

Qual è la città più popolata? e qual è quella meno popolata?

```
SELECT Name, Population FROM city ORDER BY Population DESC LIMIT 1;
```

Name	Population
Mumbai (Bombay)	10500000

```
SELECT Name, Population FROM city ORDER BY Population LIMIT 1;
```

Name	Population
Adamstown	42

Query di inserimento

Permette di inserire una nuova riga/tupla in una relazione/tabella

```
INSERT INTO <tabella> VALUES
            (<valore 1>, ..., <valore n>);
```

- È necessario indicare un valore per ogni attributo, nell'ordine corretto

Un esempio

Aggiungiamo una città italiana: per capire quali mancano visualizziamole ordinandole alfabeticamente (per non avere un output troppo lungo da leggere, concentriamoci sulle città che iniziano per P)

```
SELECT * FROM city WHERE CountryCode="ITA" AND Name LIKE "P%" ORDER
BY Name;
```

ID	Name	CountryCode	District	Population
1478	Padova	ITA	Veneto	211391
1468	Palermo	ITA	Sicilia	683794
1484	Parma	ITA	Emilia-Romagna	168717
1487	Perugia	ITA	Umbria	156673
1521	Pesaro	ITA	Marche	88987
1498	Pescara	ITA	Abruzzit	115698
1505	Piacenza	ITA	Emilia-Romagna	98384
1516	Pisa	ITA	Toscana	92379
4081	Pistoia	ITA	Toscana	290000
1483	Prato	ITA	Toscana	172473

Un esempio

Manca Pavia. Per aggiungerla dobbiamo determinare un valore disponibile per l'attributo ID

```
SELECT MAX(ID) FROM city;
```

MAX(ID)
4081

Un esempio

Siamo ora in grado di inserire una nuova riga eseguendo la query seguente.

```
INSERT INTO  city VALUES (4080, "Pavia", "ITA", "Lombardia", 71000);
```

Un esempio

Verifichiamo che sia stata inserita una nuova entità.

```
SELECT * FROM city WHERE CountryCode="ITA" AND Name LIKE "P%" ORDER
BY Name;
```

ID	Name	CountryCode	District	Population
1478	Padova	ITA	Veneto	211391
1468	Palermo	ITA	Sisilia	683794
1484	Parma	ITA	Emilia-Romagna	168717
4080	Pavia	ITA	Lombardia	71000
1487	Perugia	ITA	Umbria	156673
1521	Pesaro	ITA	Marche	88987
1498	Pescara	ITA	Abruzzit	115698
1505	Piacenza	ITA	Emilia-Romagna	98384
1516	Pisa	ITA	Toscana	92379
1483	Prato	ITA	Toscana	172473

Query di inserimento

Permette di inserire una nuova riga/tupla in una relazione/tabella senza specificare tutti gli attributi.

```
INSERT INTO <tabella>
        (<attributo 1>, ..., <attributo m>)
        VALUES (valore 1>, ..., <valore m>);
```

Query di inserimento

- I valori vanno inseriti nell'ordine che corrisponde agli attributi elencati;
- Per gli attributi non specificati viene inserito un valore `NULL`
- (a meno che non sia stato specificato un valore predefinito quando è stata creata la tabella)

Un esempio

Tra le città con la P manca anche Pistoia. Aggiungiamola con una query senza dover specificare esplicitamente l'ID.

```
INSERT INTO  city (Name, CountryCode, District, Population) VALUES (
"Pistoia", "ITA", "Toscana", 290000);
```

Un esempio

Anche in questo caso, verifichiamo che sia stata inserita la nuova riga.

```
SELECT * FROM city WHERE CountryCode="ITA" AND Name="Pistoia" ORDER
BY Name;
```

ID	Name	CountryCode	District	Population
4081	Pistoia	ITA	Toscana	290000

Va notato come il campo ID sia stato automaticamente iincrementato: ciò è dovuto a come è stata definita la tabella.

Query di inserimento

È possibile effettuare query di inserimento più complesse, in cui i valori per gli attributi vengono recuperati attraverso un'altra query

Inserimenti multipli

Permette di inserire più righe/tuple in un solo colpo.

```
INSERT INTO <tabella>
            (<attributo 1>, ..., <attributo m>)
            VALUES (valore 1>, ..., <valore m>),
                   (valore 1>, ..., <valore m>),
                   (valore 1>, ..., <valore
m>);
```

È possibile usare una sintassi simile per le query che specificano i valori per tutti gli attributi.

Query di cancellazione

Permette di eliminare una o più righe/tuple da una tabella/relazione

```
DELETE FROM <tabella> WHERE <condizione>;
```

- elimina tutte le righe che soddisfano la condizione
- volendo è possibile eseguire `DELETE FROM <tabella>;`, che però elimina **tutte** le righe

Un esempio

Cancelliamo la riga corrispondente a Pavia.

```
DELETE FROM  city WHERE Name="Pavia";
```

Un esempio

Verifichiamo di avere eliminato la riga.

```
SELECT * FROM city WHERE CountryCode="ITA" AND Name LIKE "P%";
```

ID	Name	CountryCode	District	Population
1468	Palermo	ITA	Sisilia	683794
1478	Padova	ITA	Veneto	211391
1483	Prato	ITA	Toscana	172473
1484	Parma	ITA	Emilia-Romagna	168717
1487	Perugia	ITA	Umbria	156673
1498	Pescara	ITA	Abruzzit	115698
1505	Piacenza	ITA	Emilia-Romagna	98384
1516	Pisa	ITA	Toscana	92379
1521	Pesaro	ITA	Marche	88987
4081	Pistoia	ITA	Toscana	290000

Cancellazione logica

- tipicamente si evita di cancellare effettivamente (e definitivamente) delle tuple
- si preferisce una *cancellazione logica* in cui alle tabelle si aggiunge un attributo *cancellato*
- quindi per eseguire una cancellazione logica è necessario capire come aggiornare una riga

Query di aggiornamento

Permette di modificare i contenuti di una o più righe/tuple

```
UPDATE <tabella> SET <aggiornamento> WHERE <condizione>;
```

- L'aggiornamento può riguardare uno o più attributi

Un esempio

Non so se l'avete notato, ma Palermo è indicata in "Sisilia": correggiamo questo errore.

```
UPDATE city SET District="Sicilia" WHERE Name="Palermo";
```

Un esempio

Anche in questo caso possiamo verificare che l'aggiornamento sia stato effettuato.

```
SELECT * FROM city WHERE Name="Palermo";
```

ID	Name	CountryCode	District	Population
1468	Palermo	ITA	Sicilia	683794

Un esempio

In realtà un aggiornamento del genere è meglio farlo per tutte le occorrenze sbagliate del nome della regione

```
UPDATE city SET District="Sisilia" WHERE District="Sisilia";
```

Aggiornamento e cancellazione logica

Nel caso in cui nella tabella city del database considerato esistesse una colonna deleted, la cancellazione logica si potrebbe fare tramite la query seguente.

```
UPDATE city SET deleted="T" WHERE
name="Pistoia"
```

Informazioni temporali

- `DATE` per date (precise al giorno), usando il formato `"YYYY-MM-DD"`;
- `TIME` per istanti del giorno (precisi al secondo), usando il formato `"HH-MM-SS"`;
- `TIME` per durate, usando il formato `"HHH-MM-SS"`;
- `DATETIME` per istanti che comprendono data e ora, usando il formato `"YYYY-MM-DD HH-MM-SS"`.

Valute

- `DECIMAL(<t>, <d>)` per quantità di denaro, dove `<t>` indica il numero totale di cifre e `<d>` il numero di cifre decimali;
- per esempio `DECIMAL(6, 2)` potrebbe memorizzare qualsiasi importo tra -9999,99€ e 9999,99€;
- se si accetta una minore precisione si possono usare gli usuali tipi di dati numerici.

Un esempio

- Creiamo una tabella con il seguente schema:
  - `Description` , attributo alfanumerico,
  - `From_date` , attributo temporale,
  - `To_date` , attributo temporale,
  - `Amount` , attributo decimale con sei cifre delle quali due decimali.

Un esempio

La query sarà dunque

```
CREATE TABLE events (Description VARCHAR(
100) NOT NULL,
                     From_date DATETIME N
OT NULL,
                     To_date DATETIME NOT
NULL,
                     Amount DECIMAL(6, 2)
);
```

```
CREATE TABLE events (Description VARCHAR(100) NOT NULL, From_date DA
TETIME NOT NULL, To_date DATETIME NOT NULL, Amount DECIMAL(6, 2));
```

Un esempio

Inseriamo alcune righe

```
INSERT INTO events VALUES ("Event 1", "2017-11-06 09:45:00", "2017-1
1-21 15:31:44", "349.99");
```

```
INSERT INTO events VALUES ("Event 2", "2017-10-15 19:30:20", "2017-1
0-29 08:37:12", "52.50");
```

```
INSERT INTO events VALUES ("Event 3", "2017-12-28 09:45:00", "2017-1
2-28 10:11:58", "170.00");
```

Un esempio

Controlliamo che tutto sia a posto

```
SELECT * FROM events;
```

Description	From_date	To_date	Amount
Event 1	2017-11-06 09:45:00	2017-11-21 15:31:44	349.99
Event 2	2017-10-15 19:30:20	2017-10-29 08:37:12	52.50
Event 3	2017-12-28 09:45:00	2017-12-28 10:11:58	170.00

Un esempio

Le query di selezione che riguardano l'importo monetario si eseguono come visto per i dati numerici

```
SELECT Description, Amount FROM events WHERE Amount > 100;
```

Description	Amount
Event 1	349.99
Event 3	170.00

Un esempio

Per i dati temporali si può sfruttare l'ordinamento naturale

```
SELECT Description, From_date FROM events WHERE From_date > "2017-12
-01";
```

Description	From_date
Event 3	2017-12-28 09:45:00

```
SELECT Description, From_date FROM events WHERE From_date BETWEEN "2
017-11-01" AND "2017-11-30";
```

Description	From_date
Event 1	2017-11-06 09:45:00

Un esempio

La funzione `DATEDIFF` calcola la differenza in giorni tra due date

```
SELECT description, DATEDIFF(To_date, From_date) FROM events;
```

description	DATEDIFF(To_date, From_date)
Event 1	15
Event 2	14
Event 3	0

Un esempio

La funzione `TIMEDIFF` calcola la differenza tra due istanti, espressa come durata temporale

```
SELECT Description, TIMEDIFF(To_date, From_date) FROM events;
```

Description	TIMEDIFF(To_date, From_date)
Event 1	15 days, 5:46:44
Event 2	13 days, 13:06:52
Event 3	0:26:58

Un esempio

Se volessimo cancellare la tabella appena creata, possiamo ricorrere all'istruzione `DROP`

```
DROP TABLE events;
```

Ridondanza e normalizzazione

- Si parla di ridondanza quando una stessa informazione è memorizzata in più di un punto.
- Per esempio, in due database differenti…
- oppure in più tabelle dello stesso database.
- È fonte di possibile *incoerenza* dei dati (per esempio quando non si modifica un dato nello stesso modo in tutte le sue occorrenze).
- Esistono diverse forme di *normalizzazione* che permettono di ridurre la ridondanza.

Ridondanza e normalizzazione

- Per esempio, perché non abbiamo inserito le informazioni relative al linguaggio direttamente nella tabella **country**?
- Perché è in essere una relazione *uno a molti*: fissato uno stato, posso avere uno o più linguaggi nel database.

Ridondanza e normalizzazione

- Come potevo eitare di avere due tabelle?
  - Duplicando le righe dello stato e aggiungendo le informazioni relative al linguaggio (→ ridondanza).
  - Inserendo nella tabella **country** una o più colonne in cui aggregare le informazioni relative ai linguaggi (→ complessità di gestione).
  - Prevedendo un numero massimo di linguaggi e inserendo un numero equivalente di colonne per le informazioni relative (→ complessità di gestione e potenziale spreco di spazio).

Relazioni uno a molti

Soluzione:

- nella tabella **country** è memorizzzato un *identificativo univoco* per lo stato, nel campo *code*;
- per ogni linguaggio parlato in uno stato è presente una riga nella tabella **countrylanguage**;
- il campo *countrycode* in **countrylanguage** permette di collegare le due tabelle;
- le query si possono fare "in un colpo solo" tramite una join naturale.

L'abbiamo già visto

```
SELECT Name, Language FROM country JOIN countrylanguage ON Code=Coun
tryCode LIMIT 5;
```

Name	Language
Aruba	Dutch
Aruba	English
Aruba	Papiamento
Aruba	Spanish
Afghanistan	Balochi

Relazioni molti a molti

In realtà in questo modo si riescono a modellare relazioni *molti a molti*:

- a ogni stato possono corrispondere più linguaggi,
- e viceversa.

```
SELECT Name FROM country JOIN countrylanguage ON Code=CountryCode WH
ERE Language="Italian";
```

Name
Argentina
Australia
Belgium
Brazil
Canada
Switzerland
Germany
France
Italy
Liechtenstein
Luxembourg
Monaco
San Marino
United States
Holy See (Vatican City State)

Raffinare le operazioni di aggregazione

- La clausola `WHERE` permette di filtrare i risultati di una query di selezione;
- `GROUP BY` permette di raggruppare e aggregare i risultati di una query di selezione;
- usando `WHERE` seguito da `GROUP BY` si raggruppano i risultati filtrati;
- per effettuare **prima** il raggruppamento e **poi** il filtraggio si utilizza la clausola `HAVING`

Raffinare le operazioni di aggregazione

Calcoliamo per ogni stato il numero di linguaggi...

```
SELECT Name, COUNT(Language) AS Num_lang FROM country JOIN countryla
nguage ON Code=CountryCode GROUP BY Name LIMIT 10;
```

Name	Num_lang
Afghanistan	5
Albania	3
Algeria	2
American Samoa	3
Andorra	4
Angola	9
Anguilla	1
Antigua and Barbuda	2
Argentina	3
Armenia	2

Raffinare le operazioni di aggregazione

...e recuperiamo i risultati solo in caso vi siano più di dieci lingue parlate.

```
SELECT Name, COUNT(Language) AS Num_lang FROM country JOIN countryla
nguage ON Code=CountryCode GROUP BY Name HAVING Num_lang > 10 ORDER
BY Num_lang DESC;
```

Name	Num_lang
India	12
Russian Federation	12
United States	12
Canada	12
China	12
Tanzania	11
South Africa	11

Raffinare le operazioni di aggregazione

Le clausole `WHERE` e `HAVING` si possono usare insieme:

- **prima** si filtrano i risultati tramite `WHERE` ,
- **successivamente** si aggregano i risultati del filtraggio,
- **infine** si effettua un secondo filtraggio sui risultati aggregati.

```
SELECT Name, COUNT(Language) AS Num_lang FROM country JOIN countryla
nguage ON Code=CountryCode WHERE Continent="Europe" GROUP BY Name OR
DER BY Num_lang DESC;
```

Name	Num_lang
Russian Federation	12
Italy	8
Czech Republic	8
Austria	8
Ukraine	7
Denmark	7
Belgium	6
Latvia	6
Hungary	6

Romania	6
Germany	6
Yugoslavia	6
France	6
Sweden	6
Estonia	5
Slovakia	5
Finland	5
Lithuania	5
Moldova	5
Norway	5
Luxembourg	5
Macedonia	5
Switzerland	4
Spain	4
Bulgaria	4
Monaco	4
Netherlands	4
Andorra	4
Belarus	4
Poland	4
Albania	3
United Kingdom	3
Liechtenstein	3
Slovenia	3
Faroe Islands	2
Croatia	2
Malta	2
Gibraltar	2
Ireland	2
Greece	2
Iceland	2
Svalbard and Jan Mayen	2
Portugal	1
San Marino	1

Holy See (Vatican City State)	1
Bosnia and Herzegovina	1

Qualche esercizio sulle query di selezione

Nota: nelle query riportate qui sotto è stata spesso usata la clausola LIMIT che, sebbene non sia strettamente necessaria, consente di non visualizzare un numero elevato di righe di output.

Tutti i linguaggi parlati in Tanzania

```
SELECT Language FROM country JOIN countrylanguage ON Code=CountryCod
e WHERE Name="Tanzania";
```

Language
Chaga and Pare
Gogo
Ha
Haya
Hehet
Luguru
Makonde
Nyakusa
Nyamwesi
Shambala
Swahili

Tutti i linguaggi parlati da più del 99% della popolazione

```
SELECT Language FROM countrylanguage WHERE Percentage>99;
```

Language
Serbo-Croatian
English
Crioulo
Spanish
Creole English
Arabic
Faroese
Creole English
Haiti Creole
Japanese
Creole English
Korean
Dhivehi
Korean
Rwanda
Spanish
Italian
Creole English
Arabic

Idem, ma con il nome della nazione corrispondente

```
SELECT Language, Name FROM countrylanguage JOIN country ON Code=Coun
tryCode WHERE Percentage>99;
```

Language	Name
Serbo-Croatian	Bosnia and Herzegovina
English	Bermuda
Crioulo	Cape Verde
Spanish	Cuba
Creole English	Dominica
Arabic	Western Sahara
Faroese	Faroe Islands
Creole English	Grenada
Haiti Creole	Haiti
Japanese	Japan
Creole English	Saint Kitts and Nevis
Korean	South Korea
Dhivehi	Maldives
Korean	North Korea
Rwanda	Rwanda
Spanish	El Salvador
Italian	San Marino
Creole English	Saint Vincent and the Grenadines
Arabic	Yemen

Tutti i linguaggi parlati in più di uno stato

```
SELECT Language, COUNT(CountryCode) FROM countrylanguage GROUP BY La
nguage HAVING COUNT(CountryCode) > 1 LIMIT 10;
```

Language	COUNT(CountryCode)
Afar	2
Afrikaans	2
Aimará	3
Akan	2
Albaniana	4
Arabic	33
Arawakan	2
Armenian	5
Asami	2
Azerbaijani	5

Calcolare la densità di popolazione di ogni stato e ordinarla in modo decrescente

```
SELECT Name, Population/SurfaceArea AS Density FROM country ORDER BY
Density DESC LIMIT 10;
```

Name	Density
Macao	26277.777778
Monaco	22666.666667
Hong Kong	6308.837209
Singapore	5771.84466
Gibraltar	4166.666667
Holy See (Vatican City State)	2499.999963
Bermuda	1226.415094
Malta	1203.164557
Maldives	959.731544
Bangladesh	896.922179

Per quali nazioni manca l'informazione sull'anno di indipendenza?

```
SELECT * FROM country WHERE IndepYear IS NULL LIMIT 10;
```

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy
ABW	Aruba	North America	Caribbean	193.0	None	103000	78.4
AIA	Anguilla	North America	Caribbean	96.0	None	8000	76.1
ANT	Netherlands Antilles	North America	Caribbean	800.0	None	217000	74.7
ASM	American Samoa	Oceania	Polynesia	199.0	None	68000	75.1
ATA	Antarctica	Antarctica	Antarctica	13120000.0	None	0	None
ATF	French Southern territories	Antarctica	Antarctica	7780.0	None	0	None
BMU	Bermuda	North America	North America	53.0	None	65000	76.9
BVT	Bouvet Island	Antarctica	Antarctica	59.0	None	0	None
CCK	Cocos (Keeling) Islands	Oceania	Australia and New Zealand	14.0	None	600	None
COK	Cook Islands	Oceania	Polynesia	236.0	None	20000	71.1

Curiosità: se sommo la percentuale dei linguaggi parlati in una nazione, ottengo 100%?

```
SELECT country.Name, SUM(countrylanguage.Percentage) AS Tot FROM cou
ntry JOIN countrylanguage ON Code=CountryCode GROUP BY Code ORDER BY
Tot DESC LIMIT 10;
```

Name	Tot
Netherlands	101.0
Samoa	100.1
Guyana	100.0
Ireland	100.0
South Korea	100.0
North Korea	100.0
Palestine	100.0
Costa Rica	100.0
Bermuda	100.0
Algeria	100.0

```
SELECT country.Name, SUM(countrylanguage.Percentage) AS Tot FROM cou
ntry JOIN countrylanguage ON Code=CountryCode GROUP BY Code ORDER BY
Tot LIMIT 10;
```

Name	Tot
Montserrat	0.0
Virgin Islands, British	0.0
Svalbard and Jan Mayen	0.0
United States Minor Outlying Islands	0.0
Wallis and Futuna	0.0
Holy See (Vatican City State)	0.0
East Timor	0.0
Turks and Caicos Islands	0.0
Niue	0.0
Christmas Island	0.0

Quanti distretti ha lo stato con la maggiore popolazione?

```
SELECT COUNT(District) FROM city WHERE CountryCode = (SELECT Code FR
OM country ORDER BY Population DESC LIMIT 1);
```

COUNT(District)
363

Quale percentuale di persone vive in città in ogni stato?

```
SELECT country.Name, CountryCode, 100*SUM(city.Population)/country.P
opulation FROM city JOIN country ON Code=CountryCode GROUP BY Countr
yCode LIMIT 10;
```

Name	CountryCode	100*SUM(city.Population)/country.Population
Aruba	ABW	28.1883
Afghanistan	AFG	10.2645
Angola	AGO	19.8913
Anguilla	AIA	19.4500
Albania	ALB	7.9384
Andorra	AND	27.1654
Netherlands Antilles	ANT	1.0806
United Arab Emirates	ARE	70.8044
Argentina	ARG	53.9981
Armenia	ARM	46.3949

Quali sono le forme di governo?

```
SELECT DISTINCT GovernmentForm FROM country;
```

GovernmentForm
Nonmetropolitan Territory of The Netherlands
Islamic Emirate
Republic
Dependent Territory of the UK
Parliamentary Coprincipality
Emirate Federation
Federal Republic
US Territory
Co-administrated

Nonmetropolitan Territory of France
Constitutional Monarchy
Constitutional Monarchy, Federation
Monarchy (Emirate)
Monarchy (Sultanate)
Monarchy
Dependent Territory of Norway
Territory of Australia
Federation
People'sRepublic
Nonmetropolitan Territory of New Zealand
Socialistic Republic
Occupied by Marocco
Part of Denmark
Overseas Department of France
Special Administrative Region of China
Islamic Republic
Constitutional Monarchy (Emirate)
Socialistic State
Commonwealth of the US
Territorial Collectivity of France
Autonomous Area
Administrated by the UN
Dependent Territory of the US
Independent Church State
Parlementary Monarchy

Ordiniamole dalla meno presente alla più presente

```
SELECT GovernmentForm, COUNT(GovernmentForm) FROM country GROUP BY G
overnmentForm ORDER BY COUNT(GovernmentForm);
```

GovernmentForm	COUNT(GovernmentForm)
Parliamentary Coprincipality	1
Co-administrated	1
Monarchy (Emirate)	1

Dependent Territory of the US	1
Islamic Emirate	1
Emirate Federation	1
Federation	1
Occupied by Marocco	1
Independent Church State	1
People'sRepublic	1
Constitutional Monarchy (Emirate)	1
Autonomous Area	1
Parlementary Monarchy	1
Socialistic State	1
Administrated by the UN	1
Nonmetropolitan Territory of The Netherlands	2
Special Administrative Region of China	2
Commonwealth of the US	2
Monarchy (Sultanate)	2
Islamic Republic	2
Territorial Collectivity of France	2
Part of Denmark	2
Dependent Territory of Norway	2
Socialistic Republic	3
US Territory	3
Nonmetropolitan Territory of New Zealand	3
Territory of Australia	4
Nonmetropolitan Territory of France	4
Constitutional Monarchy, Federation	4
Overseas Department of France	4
Monarchy	5
Dependent Territory of the UK	12
Federal Republic	15
Constitutional Monarchy	29
Republic	122

In quale paese si ha un'aspettativa di vita maggiore?

```
SELECT Name, LifeExpectancy FROM country ORDER BY LifeExpectancy DES
C LIMIT 1;
```

Name	LifeExpectancy
Andorra	83.5

In quali paesi la lingua ufficiale è quella più parlata?

```
SELECT Name, Language, MAX(Percentage), IsOfficial FROM country JOIN
countrylanguage ON Code=CountryCode GROUP BY Name HAVING IsOfficial=
"T" LIMIT 10;
```

Name	Language	MAX(Percentage)	IsOfficial
Albania	Albaniana	97.9	T
Algeria	Arabic	86.0	T
American Samoa	English	90.6	T
Andorra	Catalan	44.6	T
Anguilla	English	0.0	T
Armenia	Armenian	93.4	T
Aruba	Dutch	76.7	T
Bahrain	Arabic	67.7	T
Bangladesh	Bengali	97.7	T
Belarus	Belorussian	65.6	T