

Informatica Grafica  
Corso di Laurea in Ingegneria Edile – Architettura

## **Basi di dati**

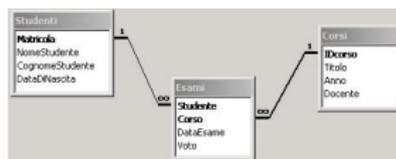
Michele Lombardi  
su materiale originario di Paolo Torroni

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)  
Università degli Studi di Bologna

Anno Accademico 2010/2011

# Basi di dati

	Matricola	NomeStudente	CognomeStudente	DataDiNascita
	1111	Mario	Neri	02/10/1983
	2222	Elena	Rossi	05/07/1982
	3333	Mario	Bianchi	24/01/1983
	4444	Giuseppe	Neri	15/11/1981
▶				



## ▶ Introduzione

- ▶ basi di dati vs. Data Base Management Systems (DBMS) – ragioni per l'uso di DBMS, tipi di DBMS

## ▶ Modello relazionale dei dati

- ▶ relazione, schema, istanza – altri modelli dei dati – vantaggi del modello relazionale

## ▶ Definizione di una base di dati relazionale

- ▶ Data Definition Languages – tipi di dato – creazione di tabelle

## ▶ Vincoli di integrità

- ▶ di dominio, di enunzia, di chiave, di integrità referenziale

## ▶ Progetto di una base di dati relazionale

- ▶ progetto logico – progetto fisico

Parte I

Introduzione

## Concetti di base

- ▶ **Sistema informativo:** sistema di supporto ai processi informativi di un'organizzazione.
  - ▶ Non è legato in alcun modo all'Informatica
- ▶ **Sistema informatico:** porzione di sistema informativo gestita in modo automatico mediante tecnologie informatiche.
  - ▶ Solo una porzione di gran parte dei sistemi informativi
  - ▶ L'informazione da gestire deve essere strutturata ed adattata per la manipolazione automatica
- ▶ **Informazioni e dati**
  - ▶ Informazioni: conoscenza acquisita, che ha per noi un valore
  - ▶ Dati: una collezione di "fatti grezzi"
  - ▶ I **dati**, che devono essere **interpretati** per fornire informazione

# Sistemi software dedicati alla gestione delle informazioni

- ▶ **Sistema di archiviazione:** memorizzazione e ricerca di informazioni invarianti nel tempo.
  - ▶ Enciclopedia, dizionario, ... (CD, DVD, ...)
- ▶ **Basi di dati:** collezione di dati.
  - ▶ Accesso ai dati offerto come servizio
  - ▶ Persistenza: dati vivono più a lungo dei programmi che li gestiscono.
  - ▶ Condivisione: garantire accesso coerente di più applicazioni e utenti.
- ▶ **Database Management System (DBMS):** sistema software per gestire collezioni (basi di dati) grandi, condivise e persistenti di dati



## Uno scenario d'uso

- ▶ Andrea, Bianca e Carlo ricevono la commessa di rilevare l'area di via Righi, via Capo di Lucca e via Delle Moline intorno al canale di Reno
- ▶ In particolare, il committente è interessato a rilevare le caratteristiche degli edifici nella zona e l'ubicazione di alcuni servizi in epoca storica (es. i mulini)
- ▶ I ragazzi decidono di spartirsi i compiti: Andrea e Bianca si occuperanno del rilievo degli edifici, mentre Carlo si curerà dei servizi



## I dati raccolti dai ragazzi

- ▶ Supponiamo che i ragazzi abbiano raccolto le informazioni in tabelle (es. fogli di calcolo o documenti di testo):
- ▶ Il risultato del lavoro di **Andrea e Bianca**:

Edificio	Ubicazione	Affaccio	Periodo	...
Pal. Gini	Righi 5	50mt	1500	...
Casa A	Capo di Lucca 2	18mt	1300	...
Casa B	Capo di Lucca 4	22mt	1400	...
...				

- ▶ Le informazioni raccolte da **Carlo**:

Servizio	Ubicazione	Periodo	...
Mulino A	Capo di Lucca 2	1300	...
Mulino B	Capo di Lucca 4	1400	...
Fabbro A	Capo di Lucca 2	1400	...
...			

# Alcuni problemi

1. Come fanno A. e B. a mettere insieme le loro informazioni?
    - ▶ Un solo documento? Due documenti separati? Chi aggrega i risultati?
    - ▶ Cosa succede se i due ragazzi scoprono indipendentemente informazioni sullo stesso edificio?
    - ▶ In generale: problemi di **accesso condiviso** alle informazioni
  2. Come vengono integrate le informazioni di C.?
    - ▶ Alcuni dati ripetono informazioni già scoperte da A. e B.
    - ▶ Come capire che la “Casa A” di A. e B. è un mulino (poi fabbro) di C.?
    - ▶ In generale: problemi di **ridondanza** ed **integrità** dei dati
- ▶ **i DBMS nascono per rispondere (anche) a queste problematiche**

# Caratteristiche di un DBMS

- ▶ Caratteristiche di un **Database Management System (DBMS)**:
  - ▶ **Affidabilità**: conservazione dei dati a fronte di guasti. Meccanismi di *backup* e *recovery*.
  - ▶ **Privatezza**: più utenti, meccanismi di autorizzazione.
  - ▶ **Efficienza**: utilizzo delle risorse accettabile dall'utente.
  - ▶ **Efficacia**: capacità di rendere produttivi gli utenti.
- ▶ Un DBMS fornisce meccanismi e tecnologie per offrire queste caratteristiche

# DBMS Esistenti

- ▶ Non esiste una sola tipologia di DBMS
- ▶ Scelta dipende da alcuni parametri di utilizzo previsto

Tabella: Esempi di tipologie di basi di dati

<b>Tipologia</b>	<b>Dimensioni</b>	<b>Utenti</b>	<b>Oper/s</b>	<b>DBMS</b>
Personalì	10MB-100MB	1-3	< 3	OpenOffice.org, Access, MySQL, PostgreSQL
Gruppo/Dip	100MB-10GB	3-100	< 300	SQLserver, MySQL, PostgreSQL
Aziendali	10GB-100GB	100-1000	< 30.000	SQLServer, Oracle, DB2
Grandi Aziende	100GB-10TB	> 1000	< 300.000	Oracle, DB2

# Modello dei dati

- ▶ Diversi **tipi** di DBMS si distinguono per il **modello dei dati** adottato
- ▶ **Modello dei dati**: insieme dei **concetti** utilizzati per
  - ▶ **organizzare** i dati di interesse e
  - ▶ **descrivere la struttura**, in modo che risulti
  - ▶ **comprensibile ad un computer**
- ▶ Vari **tipi di modelli dei dati**:
  - ▶ **Gerarchico**: uso di **strutture ad albero** (file system)
  - ▶ **Reticolare**: uso di **grafi** (come nel Web)
  - ▶ **Relazionale**: basato sul costrutto di **relazione**, rappresentato mediante una **tabella**.

# Modello dei dati

- ▶ Esempio di base di dati universitaria con tre tabelle:
  - ▶ Studenti, Esami, Corsi.

Tabella: Studenti

Matricola	Cognome	Nome	DataDiNascita
6545	Rossi	Maria	15/10/1978
8678	Pinti	Paola	13/12/1976
4567	Verdi	Luigi	02/09/1979
3465	Rossi	Mario	11/12/1978

Tabella: Corsi

Codice	Titolo	Docente
01	Matematica	Rossi
02	Storia	Bruni
03	Inglese	Verdi

Tabella: Esami

Studente	Voto	Corso
3465	29	03
3465	28	02
6545	30	02
8678	18	01

# Modello dei dati

- ▶ Esempio di base di dati universitaria con tre tabelle:
  - ▶ Studenti, Esami, Corsi.

Tabella: Studenti

Matricola	Cognome	Nome	DataDiNascita
6545	Rossi	Maria	15/10/1978
8678	Pinti	Paola	13/12/1976
4567	Verdi	Luigi	02/09/1979
3465	Rossi	Mario	11/12/1978

Tabella: Corsi

Codice	Titolo	Docente
01	Matematica	Rossi
02	Storia	Bruni
03	Inglese	Verdi

Tabella: Esami

Studente	Voto	Corso
3465	29	03
3465	28	02
6545	30	02
8678	18	01

- ▶ Dati correlati tra di loro attraverso **valori comuni**

# Schema e Istanza

- ▶ **Schema**, invariante nel tempo: **caratteristiche** dei dati

Tabella: Studenti

⇒

<b>Matricola</b>	<b>Cognome</b>	<b>Nome</b>	<b>DataDiNascita</b>
6545	Rossi	Maria	15/10/1978
8678	Pinti	Paola	13/12/1976
4567	Verdi	Luigi	02/09/1979
3465	Rossi	Mario	11/12/1978

# Schema e Istanza

- ▶ **Schema**, invariante nel tempo: **caratteristiche** dei dati
- ▶ **Istanza** o **stato** della base di dati: **valori effettivi**

Tabella: Studenti

	<b>Matricola</b>	<b>Cognome</b>	<b>Nome</b>	<b>DataDiNascita</b>
⇒	6545	Rossi	Maria	15/10/1978
⇒	8678	Pinti	Paola	13/12/1976
⇒	4567	Verdi	Luigi	02/09/1979
⇒	3465	Rossi	Mario	11/12/1978

## Parte II

### Modello relazionale dei dati

# Modello relazionale

Basato sul concetto matematico di **relazione**

⇒ Una base di dati è un **insieme di relazioni**

Vantaggi rispetto a modelli gerarchico e reticolare:

- ▶ Nessun riferimento esplicito tra strutture di memorizzazione per esprimere **dipendenze** tra dati, mentre:
  - ▶ nel modello gerarchico: uso di percorsi per individuare i componenti della base di dati
  - ▶ nel modello reticolare: uso di sequenze di puntatori per individuare i componenti della base di dati
- ▶ Basato unicamente sui **valori**
  - ▶ Valori numerici, o sequenze di caratteri (matricola)
  - ▶ Struttura della base di dati = una collezione di tabelle

# Esempio

Gestione di dati su libri e sui loro autori

- ▶ Due classi di oggetti, con specifiche **proprietà**, o **attributi**
  - ▶ **Libro**: titolo e autore
  - ▶ **Autore**: nome, cognome e anno di nascita
- ▶ Una **tabella (relazione)** per ciascuna classe
  - ▶ Struttura **logica**, indipendente dai meccanismi di memorizzazione adottati
  - ▶ Identificata da un **nome**
  - ▶ Colonne  $\Rightarrow$  **attributi**, righe  $\Rightarrow$  **istanza**
  - ▶ Accesso alle istanze effettuato tramite **valori** degli attributi

Tabella: Libri

<b>Titolo</b>	<b>Editore</b>	<b>Autore</b>
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

# Definizioni formali

Tabella: Libri

Titolo	Editore	Autore
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

## Definizione (Dominio di un attributo)

Per ogni attributo, è definito un **dominio**, cioè l'insieme di tutti i suoi possibili valori

- ▶  $\mathcal{D}(\mathbf{Titolo})$  = stringhe di 50 caratteri
- ▶  $\mathcal{D}(\mathbf{Disponibile})$  = { Sì, No }

# Definizioni formali

Tabella: Libri

Titolo	Editore	Autore
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

## Definizione (Schema di relazione)

Data una relazione, lo **schema della relazione** è una rappresentazione della sua **struttura**

- ▶ Nome della relazione, attributi, domini.
- ▶  $R(A_1, \dots, A_n)$

Libri

Titolo	Editore	Autore
--------	---------	--------

# Definizioni formali

Tabella: Libri

Titolo	Editore	Autore
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

## Definizione (Ennupla)

L'**ennupla** è una **funzione** che associa a ciascun attributo un valore preso dal suo dominio

- ▶ È una possibile scelta di valori (uno per attributo).
- ▶  $a_1 \in \mathcal{D}(A_1), a_2 \in \mathcal{D}(A_2), \dots \Rightarrow (a_1, \dots, a_n)$
- ▶ Corrisponde alla **riga** di una tabella

Sunset limited	Einaudi	Cormac McCarthy
----------------	---------	-----------------

# Definizioni formali

Tabella: Libri

Titolo	Editore	Autore
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

## Definizione (Istanza di una relazione)

L'**istanza di una relazione** (o semplicemente **relazione**) è un insieme di ennuple

- ▶  $istanza(R(A_1, \dots, A_n)) \subseteq \mathcal{D}(A_1) \times \dots \times \mathcal{D}(A_n)$

Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

# Condizioni perché una tabella rappresenti una relazione

Tabella: Libri

Titolo	Editore	Autore
Delitto e Castigo	Einaudi	Fëdor Dostoevskij
Sunset limited	Einaudi	Cormac McCarthy

- ▶ In realtà, una tabella non è necessariamente una relazione
- ▶ L'identità vale se e solo se:
  1. ogni **colonna** della tabella ha una diversa **intestazione**;
  2. i **valori** presenti in ogni **colonna** sono **omogenei** fra di loro;
  3. l'**ordinamento** tra le colonne è **irrilevante**.
  4. le **righe** sono **diverse** fra loro;
  5. l'**ordinamento** tra le righe è **irrilevante**.
- ▶ Nel seguito: relazione  $\leftrightarrow$  tabella

# Basi di dati

## Definizione (Schema della base di dati)

Uno **schema di una base di dati** è costituito da un insieme di schemi di relazione. A tale insieme è dato un nome (nome della base di dati)

## Definizione (Istanza di base di dati)

Dato uno schema di una base di dati  $R$ , un'**istanza di una base di dati** su tale schema è costituita da un insieme di istanze di relazioni  $I$ , dove  $I$  contiene un'istanza di relazione per ogni schema di relazione presente in  $R$ .

- ▶ Lo schema rappresenta le intestazioni delle tabelle.
- ▶ L'istanza contiene i dati veri e propri (righe di tutte le tabelle).

# Perché il modello relazionale?

- ▶ **Basato unicamente sui valori**
- ▶ Notevoli vantaggi rispetto all'organizzazione reticolare (tipo Web):
  1. Schema indipendente da come vengono memorizzati i dati
  2. Trasporto agevole dei dati da un sistema a un altro

vs: Cosa succede se cerco di copiare una parte di un DB reticolare?

  3. Rappresentate solo le informazioni rilevanti per l'applicazione

vs: Nella tipologia reticolare i puntatori (link) fanno parte della base di dati

  4. Associazioni bidirezionali tra dati

## Parte III

### Definizione di una Base di Dati relazionale

## Definizione una base di dati relazionale

Definire una base di dati (database) relazionale significa innanzitutto definirne lo schema (**schema logico**):

- ▶ quali **relazioni** (tabelle)
- ▶ quali **attributi** (intestazioni)
- ▶ che **dominio** per ogni attributo (tipi di dato)

Definire una base di dati (o database) = definirne lo schema logico:

- ▶ si fa solo riferimento al modello dei dati
- ▶ prescinde dal software (DBMS) che si vuole usare

Invece, nel contesto di uno **specifico DBMS**:

- ▶ si usa un **Data Definition Language (DDL)**
- ▶ insieme di costrutti per definire i dati (tabelle)



## Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**

# Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**

## 1. **testo**: singoli caratteri e stringhe

- ▶ CHARACTER: singolo carattere, es: Sesso (M/F)
- ▶ CHARACTER(16): stringa di lunghezza fissa, es: CodiceFiscale
- ▶ VARCHAR(30): stringa con lunghezza massima, es: Cognome, Indirizzo

## Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**
- 1. **testo**: CHARACTER, CHARACTER(16), VARCHAR(30)
- 2. **flag**: la presenza di una o più proprietà
  - ▶ BIT: solo valori '0' e '1', es: Lode
  - ▶ BIT(10): stringa di bit di lunghezza fissa, es: '0010011101'
  - ▶ BIT VARYING(25): stringa di bit con lunghezza massima

## Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**
- 1. **testo**: CHARACTER, CHARACTER(16), VARCHAR(30)
- 2. **flag**: BIT, BIT(10), BIT VARYING(25)
- 3. **tipi numerici**
  - ▶ INTEGER: interi 32 bit, dominio:  $[-2G, +2G-1]$
  - ▶ NUMERIC(5,2): numeri decimali, es 12345.12
  - ▶ REAL, FLOAT, DOUBLE PRECISION: numeri in virgola mobile

## Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**
- 1. **testo**: CHARACTER, CHARACTER(16), VARCHAR(30)
- 2. **flag**: BIT, BIT(10), BIT VARYING(25)
- 3. **tipi numerici**: INTEGER, NUMERIC(5,2), REAL, FLOAT, DOUBLE PRECISION
- 4. **informazioni temporali**
  - ▶ DATE, strutturato in yyyy-mm-dd
    - ▶ Es: '1973-04-03'
  - ▶ TIME, strutturato in hh:mm:ss
    - ▶ Es: '15:05:59'
  - ▶ TIMESTAMP: comprende tutti i campi di DATE e TIME
    - ▶ Es: '1973-04-03 15:05:59'

## Tipi di dato (domini)

- ▶ Tipi di dati memorizzati in un db  $\Rightarrow$  **domini**
- 1. **testo**: CHARACTER, CHARACTER(16), VARCHAR(30)
- 2. **flag**: BIT, BIT(10), BIT VARYING(25)
- 3. **tipi numerici**: INTEGER, NUMERIC(5,2), REAL, FLOAT, DOUBLE PRECISION
- 4. **informazioni temporali**: DATE, TIME, TIMESTAMP

## Informazione incompleta

- ▶ Data una ennupla, non è detto che abbiamo tutte le informazioni su tutti gli attributi.
- ▶ Possibili tre situazioni diverse:
  - ▶ Si sa che per l'ennupla esiste il valore di un attributo, ma non si sa qual è: es. *non so chi ha scritto il libro "Delitto e Castigo"* ⇒ Valore **sconosciuto**
  - ▶ Per l'ennupla si sa che non è definito un certo attributo: es. *nessun editore ha pubblicato il libro "Delitto e Castigo"* ⇒ Valore **inesistente**
  - ▶ Non si è sicuri se per l'ennupla sia definito o meno l'attributo: es. *non so se esista un editore per "Delitto e Castigo"* ⇒ Valore **senza informazione**
- ▶ Necessaria estensione della definizione di ennupla
  - ▶ Valore **NULL** (nessun dominio)
  - ▶ Unico valore convenzionale per i tre casi sopra

# Creazione di una tabella: CREATE TABLE Studenti

Tabella: Studenti

<u>Matricola</u>	Nome	Cognome	DataNascita
1234	Mario	Rossi	02/02/1990
0142	Giulia	Bianchi	28/10/1991
8392	Giulia	Bianchi	17/10/1943

- ▶ Studenti( Matricola, Nome, Cognome, DataNascita )
- ▶ Domini delle colonne:
  - ▶ **Matricola**: testo (4 caratteri)
  - ▶ **Nome, Cognome**: testo (al più 50 caratteri)
  - ▶ **DataNascita**: data

# Creazione di una tabella: CREATE TABLE Studenti

Tabella: Studenti

<u>Matricola</u>	Nome	Cognome	DataNascita
1234	Mario	Rossi	02/02/1990
0142	Giulia	Bianchi	28/10/1991
8392	Giulia	Bianchi	17/10/1943

- ▶ Studenti( Matricola, Nome, Cognome, DataNascita )
- ▶ Domini delle colonne:
  - ▶ **Matricola**: testo (4 caratteri) CHARACTER(4)
  - ▶ **Nome, Cognome**: testo (al più 50 caratteri) VARCHAR(50)
  - ▶ **DataNascita**: data DATE

## Creazione di una tabella: CREATE TABLE Corsi

Tabella: Corsi

<b>ID</b>	<b>Denominazione</b>	<b>Anno</b>	<b>Docente</b>
145	Analisi Matematica 1	1	Arcozzi
317	Informatica grafica	1	Montanari
161	Sociologia	1	Piccoli
957	Estimo	2	Minghini

- ▶ Corsi( **ID**, **Denominazione**, **Anno**, **Docente** )
- ▶ Domini delle colonne:
  - ▶ **ID**: intero
  - ▶ **Denominazione**: testo (al più 50 caratteri)
  - ▶ **Anno**: intero
  - ▶ **Docente**: testo (al più 20 caratteri)

## Creazione di una tabella: CREATE TABLE Corsi

Tabella: Corsi

<u>ID</u>	Denominazione	Anno	Docente
145	Analisi Matematica 1	1	Arcozzi
317	Informatica grafica	1	Montanari
161	Sociologia	1	Piccoli
957	Estimo	2	Minghini

- ▶ Corsi( ID, Denominazione, Anno, Docente )
- ▶ Domini delle colonne:
  - ▶ **ID**: intero `INTEGER`
  - ▶ **Denominazione**: testo (al più 50 caratteri) `VARCHAR(50)`
  - ▶ **Anno**: intero `INTEGER`
  - ▶ **Docente**: testo (al più 20 caratteri) `VARCHAR(20)`

## Parte IV

### Vincoli di integrità

## Errori nei dati

- ▶ Possibile immettere dati sintatticamente corretti, ma semanticamente privi di senso

Tabella: Esami

Studente	Corso	Voto	Lode
Rossi	Informatica	32	No
Verdi	Matematica	18	Sì
Neri	Giardinaggio applicato	25	No

- ▶ Prima ennupla: **Voto** 32  $\notin$  [18..30]
- ▶ Seconda ennupla: **Voto** e **Lode** non correlati
- ▶ Terza ennupla: **Corso** (probabilmente) errato

# Vincoli di integrità

- ▶ Si usano per limitare l'immissione di dati scorretti
- ▶ Sono formulazioni di **proprietà** che i dati devono soddisfare
- ▶ Definiti **a livello di schema**
  - ▶ Devono essere soddisfatti da tutte le possibili istanze corrette dello schema
- ▶ Modellano alcune caratteristiche rilevanti della realtà che si vuole rappresentare
- ▶ Quattro tipi di vincoli di integrità:
  - ▶ vincoli su **valori** (o di dominio)
  - ▶ vincoli di **ennupla** (o di riga)
  - ▶ vincoli di **chiave**
  - ▶ vincoli di **integrità referenziale**

# Vincoli di integrità

- ▶ Si usano per limitare l'immissione di dati scorretti
- ▶ Sono formulazioni di **proprietà** che i dati devono soddisfare
- ▶ Definiti **a livello di schema**
  - ▶ Devono essere soddisfatti da tutte le possibili istanze corrette dello schema
- ▶ Modellano alcune caratteristiche rilevanti della realtà che si vuole rappresentare
- ▶ Quattro tipi di vincoli di integrità:
  - ▶ vincoli su **valori** (o di dominio) **intra-relazionali**
  - ▶ vincoli di **annupla** (o di riga) **intra-relazionali**
  - ▶ vincoli di **chiave** **intra-relazionali**
  - ▶ vincoli di **integrità referenziale**

# Vincoli di integrità

- ▶ Si usano per limitare l'immissione di dati scorretti
- ▶ Sono formulazioni di **proprietà** che i dati devono soddisfare
- ▶ Definiti **a livello di schema**
  - ▶ Devono essere soddisfatti da tutte le possibili istanze corrette dello schema
- ▶ Modellano alcune caratteristiche rilevanti della realtà che si vuole rappresentare
- ▶ Quattro tipi di vincoli di integrità:
  - ▶ vincoli su **valori** (o di dominio) **intra-relazionali**
  - ▶ vincoli di **annupla** (o di riga) **intra-relazionali**
  - ▶ vincoli di **chiave** **intra-relazionali**
  - ▶ vincoli di **integrità referenziale** **inter-relazionali**

## Vincoli di dominio

- ▶ Coinvolgono solo gli attributi di una singola relazione
- ▶ **Vincoli intra-relazionali**
- ▶ Esprimono condizioni sui valori
  - ▶ di **un singolo** attributo (colonna)
  - ▶ di una singola ennupla (riga)

Tabella: Esami

Studente	Corso	Voto	Lode
Rossi	Informatica	32	No
Verdi	Matematica	18	Sì
Neri	Oroscopia	25	No

## Vincoli di dominio

- ▶ Coinvolgono solo gli attributi di una singola relazione
- ▶ **Vincoli intra-relazionali**
- ▶ Esprimono condizioni sui valori
  - ▶ di **un singolo** attributo (colonna)
  - ▶ di una singola ennupla (riga)

Tabella: Esami

⇒

Studente	Corso	Voto	Lode
Rossi	Informatica	32	No
Verdi	Matematica	18	Sì
Neri	Oroscopia	25	No

- ▶ Prima ennupla:  $32 \notin \mathcal{D}(\mathbf{Voto})$

## Vincoli di ennupla

- ▶ Coinvolgono solo gli attributi di una singola relazione
- ▶ **Vincoli intra-relazionali**
- ▶ Esprimono condizioni sui valori
  - ▶ di **più attributi** (colonne)
  - ▶ di una singola ennupla (riga)

Tabella: Esami

Studente	Corso	Voto	Lode
Rossi	Informatica	32	No
Verdi	Matematica	18	Sì
Neri	Oroscopia	25	No

- ▶ Se **Lode=Sì** allora **Voto=30**

## Vincoli di ennupla

- ▶ Coinvolgono solo gli attributi di una singola relazione
- ▶ **Vincoli intra-relazionali**
- ▶ Esprimono condizioni sui valori
  - ▶ di **più attributi** (colonne)
  - ▶ di una singola ennupla (riga)

Tabella: Esami

⇒

Studente	Corso	Voto	Lode
Rossi	Informatica	32	No
Verdi	Matematica	<b>18</b>	<b>Sì</b>
Neri	Oroscopia	25	No

- ▶ Se **Lode=Sì** allora **Voto=30**
- ▶ Seconda ennupla: vincolo violato

# Chiave

- ▶ È un attributo che permette di **identificare univocamente le ennuple** di una tabella
  - ▶ Esempi: ISBN, CF, Matricola (Studenti), Codice (Corsi)
  - ▶ Anche più attributi: (Studente, Corso) nella tabella Esami
- ▶ Costituita da insieme **minimale** di attributi
  - ▶ Esempio: (CF, ~~Nome~~) in una tabella Contribuenti

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	<u>Voto</u>	<u>Lode</u>
Rossi	Informatica	32	No
Verdi	Matematica	18	Sì
Rossi	Informatica	25	No

# Chiave

- ▶ È un attributo che permette di **identificare univocamente le ennuple** di una tabella
  - ▶ Esempi: ISBN, CF, Matricola (Studenti), Codice (Corsi)
  - ▶ Anche più attributi: (Studente, Corso) nella tabella Esami
- ▶ Costituita da insieme **minimale** di attributi
  - ▶ Esempio: (CF, ~~Nome~~) in una tabella Contribuenti

Tabella: Esami

	<u>Studente</u>	<u>Corso</u>	<u>Voto</u>	<u>Lode</u>
⇒	Rossi	Informatica	32	No
	Verdi	Matematica	18	Sì
⇒	Rossi	Informatica	25	No

- ▶ Prima e terza ennupla: stessa chiave!

# Vincoli di chiave

- ▶ Proprietà della chiave:
  - ▶ Definita considerando insiemi **minimali** di attributi
  - ▶ Garantisce accessibilità a ciascun dato della base di dati
    - ▶ **Non sono permessi duplicati**
  - ▶ Attributo **essenziale** perché accesso solo tramite valori
  - ▶ **Chiave primaria**, scelta tra tutte le chiavi:
    - ▶ **Non sono permessi valori nulli**
- ▶ Vincoli di chiave:
  - ▶ Coinvolgono solo gli attributi di una singola relazione
  - ▶ **Vincoli intra-relazionali**
  - ▶ Vincoli di **unicità** e **presenza** della chiave nella relazione  
⇒ “UNICO”, “NON NULLO”

## Creazione di una tabella: CREATE TABLE Corsi

Tabella: Corsi

<u>ID</u>	Denominazione	Anno	Docente
145	Analisi Matematica 1	1	Arcozzi
317	Informatica grafica	1	Montanari
161	Sociologia	1	Piccoli
957	Estimo	2	Minghini

- ▶ Corsi( ID, **Denominazione**, **Anno**, **Docente** )
- ▶ Domini delle colonne:
  - ▶ **ID**: intero
  - ▶ **Denominazione**: testo (al più 50 caratteri)
  - ▶ **Anno**: intero
  - ▶ **Docente**: testo (al più 20 caratteri)

# Creazione di una tabella: CREATE TABLE Corsi

Tabella: Corsi

<u>ID</u>	Denominazione	Anno	Docente
145	Analisi Matematica 1	1	Arcozzi
317	Informatica grafica	1	Montanari
161	Sociologia	1	Piccoli
957	Estimo	2	Minghini

- ▶ Corsi( ID, Denominazione, Anno, Docente )
- ▶ Domini delle colonne:
  - ▶ **ID**: intero
  - ▶ **Denominazione**: testo (al più 50 caratteri)
  - ▶ **Anno**: intero
  - ▶ **Docente**: testo (al più 20 caratteri)
- ▶ Requisiti aggiuntivi:
  - ▶ Non possono esistere due corsi con lo stesso titolo
  - ▶ Anno di corso  $\in [1..5]$
  - ▶ Il docente deve essere specificato obbligatoriamente

# Creazione di una tabella: CREATE TABLE Corsi

Tabella: Corsi

<b>ID</b>	<b>Denominazione</b>	<b>Anno</b>	<b>Docente</b>
145	Analisi Matematica 1	1	Arcozzi
317	Informatica grafica	1	Montanari
161	Sociologia	1	Piccoli
957	Estimo	2	Minghini

- ▶ Corsi( **ID**, **Denominazione**, **Anno**, **Docente** )
- ▶ Domini delle colonne:
  - ▶ **ID**: intero CHIAVE PRIMARIA
  - ▶ **Denominazione**: testo (al più 50 caratteri)
  - ▶ **Anno**: intero  $\geq 0$  e  $\leq 5$
  - ▶ **Docente**: testo (al più 20 caratteri) NON NULLO
- ▶ Requisiti aggiuntivi:
  - ▶ Non possono esistere due corsi con lo stesso titolo
  - ▶ Anno di corso  $\in [1..5]$
  - ▶ Il docente deve essere specificato obbligatoriamente

## Vincoli di integrità referenziale

- ▶ Coinvolgono gli attributi di più relazioni
- ▶ **Vincoli inter-relazionali**
- ▶ Permettono di **correlare** i dati memorizzati in **tabelle diverse**
- ▶ Impongono che i valori degli attributi di **una relazione** compaiano nella **chiave primaria di un'altra relazione**
- ▶ Un vincolo di integrità referenziale:
  - ▶ fra gli attributi  $X$  di una relazione  $R_1$  e un'altra relazione  $R_2$
  - ▶ impone ai valori di  $X$  in  $R_1$
  - ▶ di comparire come valori della chiave primaria di  $R_2$
- ▶ Esempio: vincolo tra **Studente (Esami)** e **Matricola (Studenti)**

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	<u>Voto</u>
0142	Matematica	28
8392	Informatica	30L
1022	Informatica	23

Tabella: Studenti

<u>Matricola</u>	<u>Nome</u>	<u>Cognome</u>
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

## Vincoli di integrità referenziale

- ▶ Coinvolgono gli attributi di più relazioni
- ▶ **Vincoli inter-relazionali**
- ▶ Permettono di **correlare** i dati memorizzati in **tabelle diverse**
- ▶ Impongono che i valori degli attributi di **una relazione** compaiano nella **chiave primaria di un'altra relazione**
- ▶ Un vincolo di integrità referenziale:
  - ▶ fra gli attributi  $X$  di una relazione  $R_1$  e un'altra relazione  $R_2$
  - ▶ impone ai valori di  $X$  in  $R_1$
  - ▶ di comparire come valori della chiave primaria di  $R_2$
- ▶ Esempio: vincolo tra Studente (Esami) e Matricola (Studenti)

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	<u>Voto</u>
0142	Matematica	28
8392	Informatica	30L
<b>1022</b>	Informatica	23

Tabella: Studenti

<u>Matricola</u>	<u>Nome</u>	<u>Cognome</u>
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

⇒

# Creazione di tabelle con integrità referenziale

Tabella: Studenti

<u>Matricola</u>	<u>Nome</u>	<u>Cognome</u>
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	<u>DataEsame</u>	<u>Voto</u>
8392	318	09/12/2009	28
0989	317	09/12/2009	30
0142	145	14/12/2009	24
8392	161	12/01/2010	23

- ▶ Esistono **correlazioni** tra campi di **tabelle diverse**
  - ▶ Esami.Studente ↔ Studenti.Matricola
  - ▶ Esami.Corso ↔ Corsi.ID
- ▶ Possibili problemi (da evitare):
  - ▶ In Esami compare un codice di un **corso** sconosciuto
  - ▶ In Esami compare un codice di uno **studente** sconosciuto
- ▶ Soluzione: **vincoli di integrità referenziale**

# Creazione di tabelle con integrità referenziale

Tabella: Studenti

<u>Matricola</u>	<u>Nome</u>	<u>Cognome</u>
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	<u>DataEsame</u>	<u>Voto</u>
8392	<b>318</b>	09/12/2009	28
<b>0989</b>	317	09/12/2009	30
0142	145	14/12/2009	24
8392	161	12/01/2010	23

- ▶ Occorre distinguere tra 2 ruoli di una tabella
  - ▶ Tabella **interna** (es: Esami)
  - ▶ Tabella **esterna** (es: Studenti, Corsi)
- ▶ Vincolo di integrità referenziale:
  - ▶ per ogni ennupla della **tabella interna**,
  - ▶ il valore dell'**attributo specificato**, se non nullo,
  - ▶ deve trovarsi in almeno una ennupla della **tabella esterna**
  - ▶ tra i valori del **corrispondente attributo**.
- ▶ CHIAVE ESTERNA(attr. interno)  
RIFERIMENTO tabella esterna(attr. esterno)

# Creazione di tabelle con integrità referenziale

Tabella: Studenti

<u>Matricola</u>	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

<u>Studente</u>	<u>Corso</u>	DataEsame	Voto
8392	<b>318</b>	09/12/2009	28
<b>0989</b>	317	09/12/2009	30
0142	145	14/12/2009	24
8392	161	12/01/2010	23

- ▶ Esami( Studente, Corso, DataEsame, Voto )
- ▶ Domini delle colonne:
  - ▶ **Studente**: testo (4 caratteri)
  - ▶ **Corso**: intero
  - ▶ **DataEsame**: data
  - ▶ **Voto**: intero
- ▶ Vincoli:
  - ▶  $Voto > 17$  e  $Voto \leq 30$
  - ▶ Chiave primaria: (Studente, Corso)
  - ▶ CHIAVE ESTERNA(Corso) RIFERIMENTO Corsi(ID)

## Parte V

### Progettazione dei sistemi informativi

## Ciclo di vita dei sistemi informativi

Insieme di attività: da **specifica informale** delle caratteristiche desiderate a **realizzazione** del S.I.

1. **Studio di fattibilità.** Definire **costi** delle alternative e **priorità** delle componenti.
2. **Raccolta e analisi dei requisiti.** Individuazione di **proprietà** e **funzionalità** del futuro sistema.
  - ▶ Fonti: **utenti, documentazione, realizzazioni preesistenti.**
3. **Progettazione.** Individuazione di struttura e organizzazione dei **dati**, e caratteristiche degli **applicativi** che li useranno.
  - ▶ Produce uno **schema relazionale** della base di dati.
4. **Implementazione.** Realizzazione del S.I., seguendo uno **schema fisico** dei dati.
5. **Validazione e collaudo.** Verifica di **funzionamento** e **qualità** del S.I.
6. **Funzionamento.** Il S.I. è operativo: **gestione, manutenzione, formazione.**

# Progettazione Logica di un DB Relazionale

Il **progetto** di un database relazionale consiste nella specifica di:

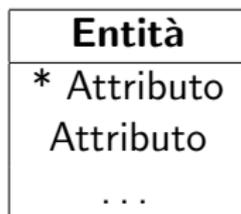
- ▶ quali **relazioni** contiene (tabelle)
- ▶ quali **attributi** (intestazioni)
- ▶ che **dominio** per ogni attributo (tipi di dato)
- ▶ **vincoli intra-relazionali**
- ▶ **vincoli inter-relazionali**

E' conveniente descrivere la "parte di mondo" di cui vogliamo memorizzare informazioni in termini di:

- ▶ **entità**
- ▶ **relazioni**

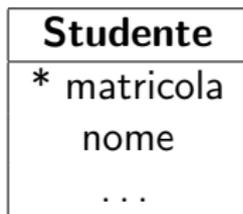
# Diagrammi Entity-Relationship

- ▶ Modello **Entity-Relationship**.
- ▶ **Entità**: una persona, luogo o cosa di cui si vogliono raccogliere e conservare istanze multiple di dati.
  - ▶ Ha un **nome** (un sostantivo) e degli **attributi** che descrivono i dati di interesse.
  - ▶ Ha un **identificatore** per individuare le istanze in modo univoco.
  - ▶ L'attributo (o gli attributi) che fanno da identificatore sono segnati da asterischi.



# Diagrammi Entity-Relationship

- ▶ Modello **Entity-Relationship**.
- ▶ **Entità**: una persona, luogo o cosa di cui si vogliono raccogliere e conservare istanze multiple di dati.
- ▶ **Relazione**: una associazione tra due entità.
  - ▶ Ha un **nome** (un verbo).

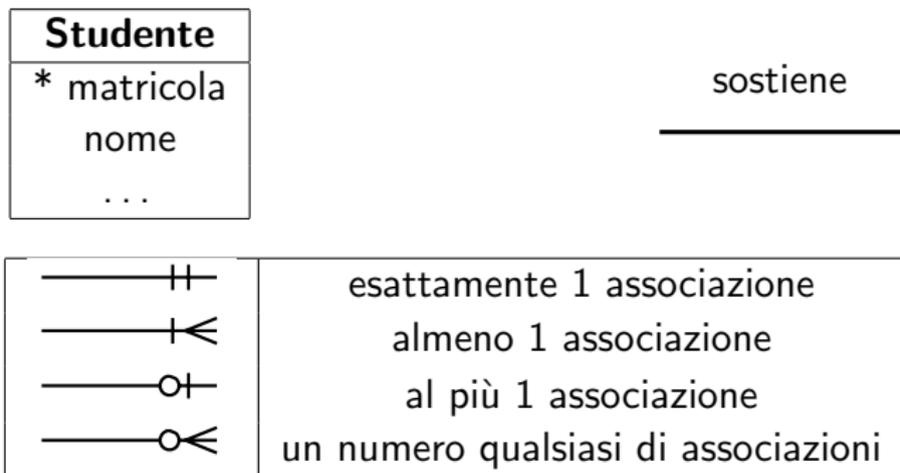


sostiene

---

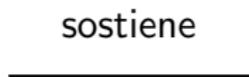
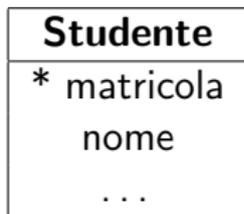
# Diagrammi Entity-Relationship

- ▶ Modello **Entity-Relationship**.
- ▶ **Entità**: una persona, luogo o cosa di cui si vogliono raccogliere e conservare istanze multiple di dati.
- ▶ **Relazione**: una associazione tra due entità.
  - ▶ Ha un **nome** (un verbo).
  - ▶ Ha **cardinalità** (min) e **modalità** (max).



# Diagrammi Entity-Relationship

- ▶ Modello **Entity-Relationship**.
- ▶ **Entità**: una persona, luogo o cosa di cui si vogliono raccogliere e conservare istanze multiple di dati.



Anche le relazioni possono avere attributi

- ▶ Esempio: **Studente** supera **Esame**  
(Voto, Data)



## Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

1. **Fattibilità:** OK
2. **Analisi dei requisiti:**
3. **Progetto:**
4. **Implementazione:**
5. **Validazione e collaudo:**
6. **Funzionamento & Manutenzione:**

# Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

## 1. **Fattibilità:**

## 2. **Analisi dei requisiti:**

- ▶ capacità di registrare dati presi da SQLzoo per eseguire semplici esperimenti
- ▶ dati a disposizione: paesi, regioni, area, popolazione, PIL, paesi confinanti
- ▶ entità dei dati: poche centinaia di record

## 3. **Progetto:**

## 4. **Implementazione:**

## 5. **Validazione e collaudo:**

## 6. **Funzionamento & Manutenzione:**

# Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

## 1. Fattibilità:

## 2. Analisi dei requisiti:

## 3. Progetto:

- ▶ organizzazione e struttura dei dati:
  - ▶ **Entità:** paesi, regioni
  - ▶ **Relazioni:** paese appartiene a regione, paese confina con paese
  - ▶ **Proprietà:**
    - area, popolazione, PIL → paese
    - nome regione → regione
- ▶ **Schema relazionale** (Entity-Relationship)
- ▶ **Schema fisico** (insieme di tabelle)

## 4. Implementazione:

## 5. Validazione e collaudo:

## 6. Funzionamento & Manutenzione:

# Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

1. **Fattibilità:**
2. **Analisi dei requisiti:**
3. **Progetto:**
4. **Implementazione:**
  - ▶ DBMS: MySQL 5.1.49
  - ▶ piattaforma: Ubuntu Linux 10.10
  - ▶ implementazione delle tabelle segue lo schema fisico
5. **Validazione e collaudo:**
6. **Funzionamento & Manutenzione:**

## Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

1. **Fattibilità:**
2. **Analisi dei requisiti:**
3. **Progetto:**
4. **Implementazione:**
5. **Validazione e collaudo:**
  - ▶ Non ora!
6. **Funzionamento & Manutenzione:**

## Esercizio: DB di paesi e regioni geografiche

DB della BBC (“country profile”) accessibile su <http://www.sqlzoo.net/> (leggermente modificato)

1. **Fattibilità:**
2. **Analisi dei requisiti:**
3. **Progetto:**
4. **Implementazione:**
5. **Validazione e collaudo:**
6. **Funzionamento & Manutenzione:**
  - ▶ Dati conservati nella propria installazione di MySQL
  - ▶ Eventuale aggiornamento software MySQL se necessario

# Uno sguardo ai dati

```
id, name, region, area, population, gdp, bordering countries
1 Afghanistan South Asia 652225 26000000
  China | Iran | Pakistan | Tajikistan | Turkmenistan | Uzbekistan

2 Albania Europe 28728 3200000 6656000000
  Greece | Kosovo | Macedonia | Montenegro

3 Algeria Middle East 2400000 32900000 75012000000
  Libya | Mali | Mauritania | Morocco | Niger | Tunisia | W. Sahara

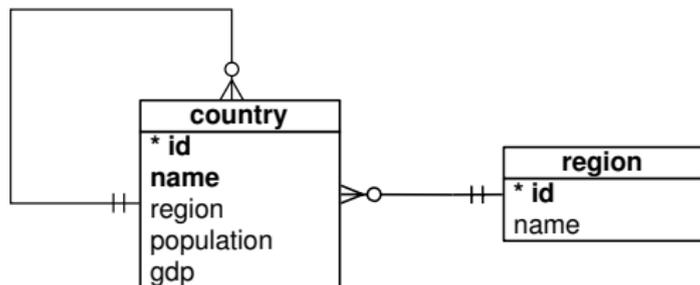
4 Andorra Europe 468 64000
  France | Spain

5 Angola Africa 1250000 14500000 14935000000
  Dem. Republic of the Congo | Namibia | Republic of the Congo | Zambia

6 Antigua and Barbuda Americas 442 77000 770000000

7 Argentina South America 2800000 39300000 146196000000
  Bolivia | Brazil | Chile | Paraguay | Uruguay
```

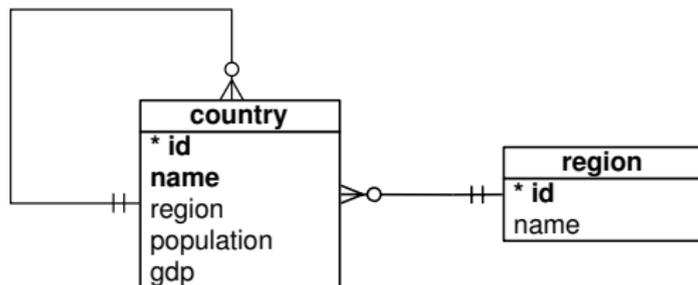
# Progetto logico del DB “country profile”



## Entità:

- ▶ regions(**id**, name)
  - ▶ Chiave primaria: **id**
  - ▶ Vincoli di unicità: name
- ▶ countries(**id**, name, region, population, gdp)
  - ▶ Chiave primaria: **id**
  - ▶ Chiave forestiera: **region** (riferimento: regions.id)
  - ▶ Vincoli di unicità: name

# Progetto logico del DB “country profile”



## Relazioni:

- ▶ ogni “country” appartiene ad esattamente una “region”
- ▶ ogni “country” confina con zero o più regioni

# Progetto fisico del DB “country profile”

Semplificando: **progetto fisico = traduzione dello schema logico in tabelle**

- ▶ Quindi il risultato del progetto fisico è il vero e proprio schema della base di dati

## **Regole generali:**

- ▶ Una tabella per ogni entità
- ▶ Ogni relazioni si realizza:
  1. aggiungendo un attributo ad una relazione esistente
  2. aggiungendo una nuova tabellas

# Progetto fisico del DB “country profile”

Semplificando: **progetto fisico = traduzione dello schema logico in tabelle**

- ▶ Quindi il risultato del progetto fisico è il vero e proprio schema della base di dati

**Per il DB “country profile”:**

- ▶ country ↔ **tabella** countries
- ▶ region ↔ **tabella** regions
- ▶ relazione “appartiene” ↔ **attributo region in** countries
- ▶ relazione “confina” ↔ **tabella** bordering

## Progetto fisico del DB “country profile”

- ▶ **regions(id, name)**
- ▶ **countries(id, name, region, population, gdp)**
- ▶ **bordering(country1, country2)**

## Progetto fisico del DB “country profile”

- ▶ **regions(id, name)**
  - ▶ Numero di regioni: < 10
  - ▶ **id**: tipo CHAR (1 carattere da 0 a 9), chiave primaria
  - ▶ **name**: tipo VARCHAR(20), *unico*,  $\neq \emptyset$
- ▶ **countries(id, name, region, population, gdp)**
- ▶ **bordering(country1, country2)**

## Progetto fisico del DB “country profile”

- ▶ **regions(id, name)**
- ▶ **countries(id, name, region, population, gdp)**
  - ▶ **id**: tipo DECIMAL(3,0) (3 cifre, da 0 a 999), chiave primaria
  - ▶ **name**: tipo VARCHAR(50) *unico*,  $\neq \emptyset$
  - ▶ **region**: chiave forestiera (rif.: regions.id, tipo CHAR),  $\neq \emptyset$
  - ▶ **area**: numerico ( $km^2$ ), val. max  $\sim 10^{10}$ , prec: 3-4 cifre  $\Rightarrow$  REAL (tipo INTEGER a 32 bit, max  $\sim 2^9 < 2^{10}$ )
  - ▶ **population**: numerico (N. abitanti), val. max  $\sim 10^8$ , prec: 3-4 cifre  $\Rightarrow$  REAL
  - ▶ **gdp**: numerico (US\$), val. max  $\sim 10^{13}$ , prec: 3-4 cifre  $\Rightarrow$  REAL
- ▶ **bordering(country1, country2)**

## Progetto fisico del DB “country profile”

- ▶ **regions**(id, name)
- ▶ **countries**(id, name, region, population, gdp)
- ▶ **bordering**(country1, country2)
  - ▶ **country1**: tipo DECIMAL(3,0), chiave forestiera (rif.: country.id)
  - ▶ **country2**: tipo DECIMAL(3,0), chiave forestiera (rif.: country.id)
  - ▶ Chiave primaria: **country1, country2**