Cultura Tecnologica di Progetto

Politecnico di Milano
Facoltà di Disegno Industriale

DBMS
Progettazione di Basi di Dati
Introduzione all' SQL

Autore: Filippo Naggi

Sommario

1. Dati e Archivi
2. I DBMS
3. Il modello relazionale
4. Il linguaggio SQL
5. Esempi di progettazione con il modello ER

Dati volatili e dati persistenti

- I dati **volatili** sono quelli che esistono solo durante l'esecuzione del programma che li crea.

- I dati **persistenti** sono quelli che sopravvivono all'esecuzione del programma che li crea.

Archivi

Un archivio è una struttura di dati persistenti memorizzata su memoria di massa

- Esistono diversi tipi di di accesso agli archivi
  - sequenziale, con indice ...
- Gli archivi sono organizzati in record logici
  - a loro volta strutturati in sequenze di campi (unità elementari di informazione)
- Per ogni archivio sono definite delle operazioni fondamentali sui relativi dati:
  - inserimento
  - cancellazione
  - modifica
  - ricerca e lettura

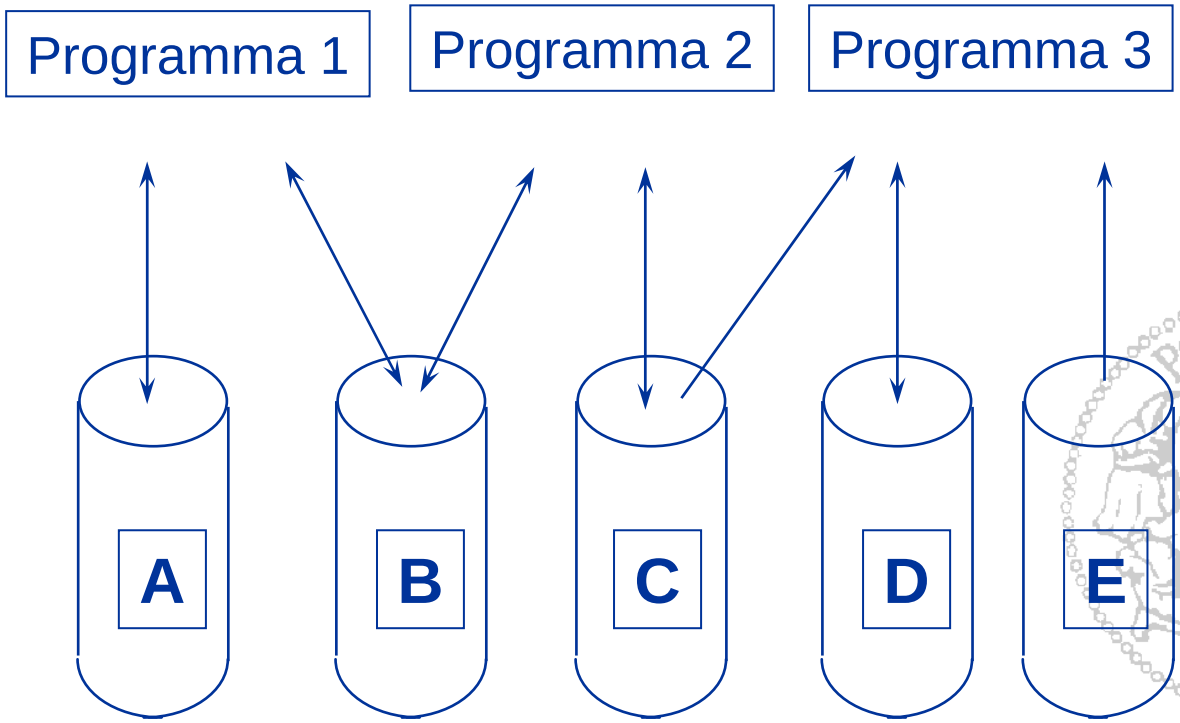
Accesso agli archivi

E' possibile accedere agli archivi in due modi:

- Accesso esclusivo: quando ad un archivio può accedere soltanto un programma
- Accesso condiviso: quando un ad un archivio può accedere più di un programma

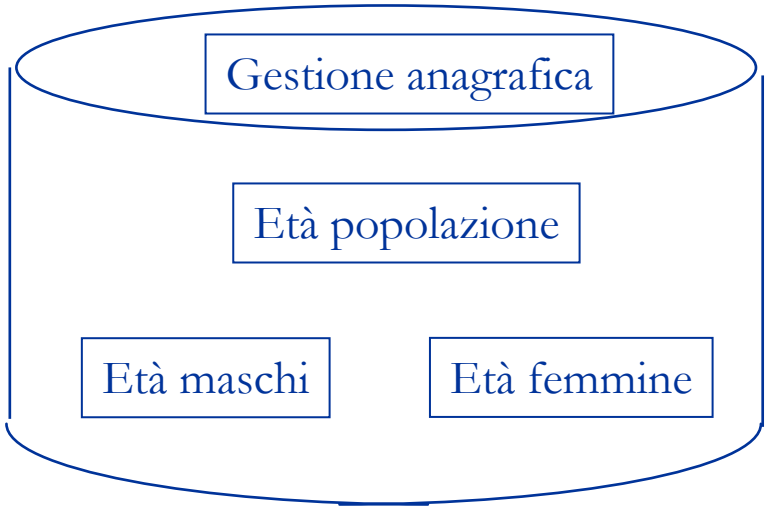
Uso degli archivi

- Diversi programmi usano diversi archivi, alcuni condivisi, altri esclusivi:



Limiti degli archivi: Ridondanza

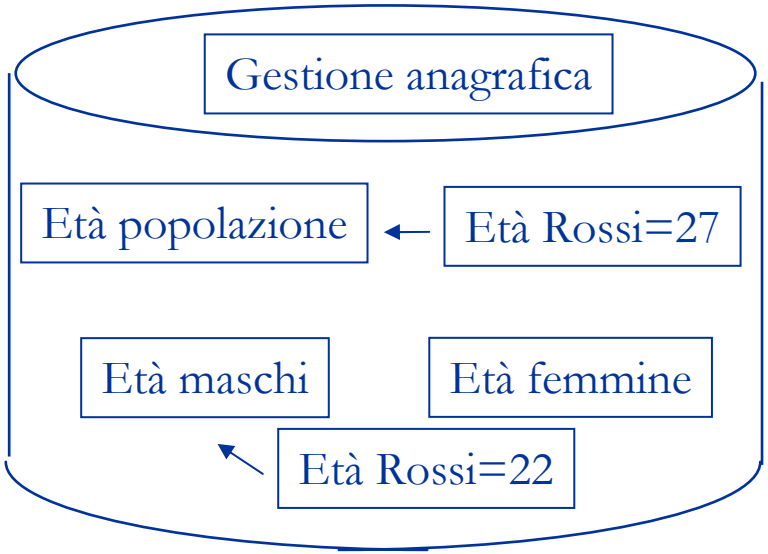
Un dato potrebbe comparire più volte nello stesso archivio. Si crea in questo modo un problema di ridondanza.



In questo archivio esiste un problema di ridondanza in quanto i dati relativi all'età della popolazione sono replicati in più locazioni

Limiti degli archivi: Inconsistenza

Un dato ridondante potrebbe essere modificato differentemente nelle varie locazioni dove compare. Si crea in questo modo un problema di inconsistenza dei dati.



In questo archivio esiste un problema di inconsistenza in quanto l'età di Rossi compare con valori differenti

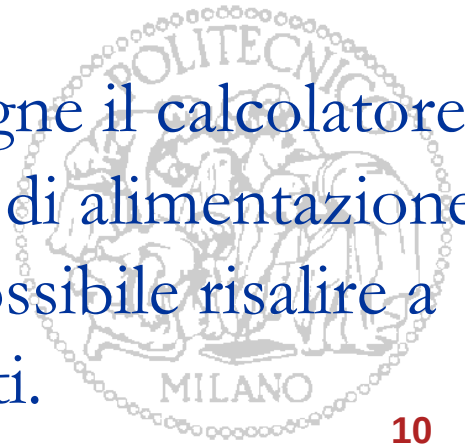
Limiti degli archivi:

Utilizzo dei servizi del Sistema Operativo

- Gli archivi sono normalmente realizzati semplicemente mediante i servizi messi a disposizione dal file system del S.O.
- Ogni archivio è costituito da un insieme di file, cui si accede attraverso le funzioni del file system.
- Quindi gli archivi sono soggetti ai limiti causati da questa particolare implementazione

Problemi archivi

- **Sicurezza fisica:** se si rompe il supporto di memoria di massa sul quale risiede l'archivio si perdono i dati
- **Privatezza:** il livello di sicurezza offerto è quello fornito dal sistema operativo. Ad esempio non è possibile consentire ad un utente l'accesso solamente ad una parte di un file.
- **Atomicità delle operazioni:** se si spegne il calcolatore (ad esempio per un guasto o per mancanza di alimentazione) le operazioni restano incomplete. Non è possibile risalire a quale stadio del calcolo corrispondono i dati.



Limiti archivi: Accessi concorrenti

Transazioni contemporanee sullo stesso archivio possono generare uno stato finale non uguale alla combinazione delle transazioni.

Supponiamo per esempio che due programmatori abbiamo scritto ciascuno un programma per incrementare una variabile X.

Programma 1
leggi il valore di X in x
y=x+1
scrivi in X il valore di y

Programma 2
leggi il valore di X in t
z=t+1
scrivi in X il valore di z

Limiti archivi: Accessi concorrenti

Supponiamo ora di eseguire parallelamente i due programmi:

- 1. leggi il valore di X in x [x = 4]
- 2. y=x+1 [y = 5]
- 3. leggi il valore di X in t [t = 4]
- 4. z=t+1 [z = 5]
- 5. scrivi in X il valore di z [X = 5]
- 6. scrivi in X il valore di y [X = 5]

Due programmi incrementano X di 1, ma l'effetto finale è un **unico** incremento!

Limiti degli archivi: Integrità dei dati

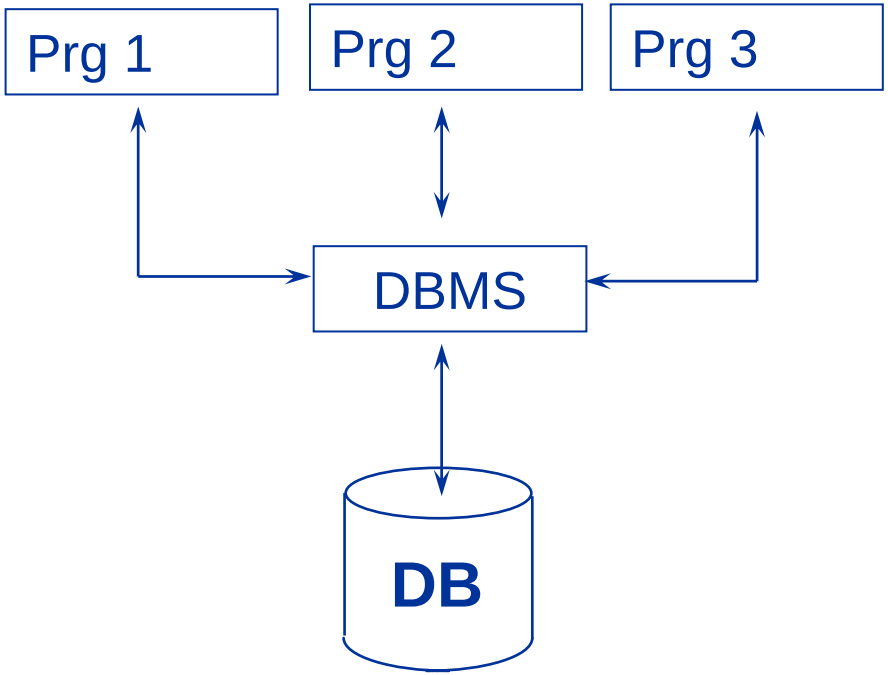
I dati di un archivio devono sempre essere consistenti con un insieme di vincoli logici. Se uno o più di questi vincoli non sono soddisfatti, esiste un problema

- Ad es. in un'anagrafe ogni dato riguarda una persona, che ha un padre e una sequenza di figli. Ovviamente se esiste la persona P avente padre P1 occorre che P compaia tra i figli di P1.

  Se il programma che aggiorna i padri è diverso da quello che registra le nascite si possono avere momenti in cui i dati non sono consistenti.

I sistemi per la gestione di basi di dati

- Nascono per ovviare ai limiti degli archivi
- L'idea fondamentale è di centralizzare la gestione dei dati, affidandola ad un unico programma: il DBMS (data base management system, o sistema per la gestione di basi di dati).



Benefici derivanti dall'uso dei DBMS

Se ben usati, i DBMS forniscono i seguenti vantaggi:

- unica rappresentazione dei dati (meno possibilità di ridondanza e inconsistenza)
- accesso disciplinato attraverso il DBMS (privatezza, ...)
- gestione centralizzata dei vincoli di consistenza (si possono impedire le operazioni che violerebbero i vincoli)
- gestione della memoria di massa (all'utente viene presentata una visione logica dei dati, nascondendo i dettagli, con conseguente portabilità della base di dati.)
- gestione degli accessi concorrenti (l'accesso ad un dato è negato se c'è già un programma che sta manipolando quel dato)

I modelli dei dati

Il modello dei dati indica:

- come sono organizzate, strutturate e presentate le informazioni agli utenti del DB.
- quali operazioni sono disponibili sui dati

Esistono diversi modelli dei dati usati nei DB:

- Gerarchico (metà anni sessanta)
- Reticolare (1973, 1978)
- Relazionale (inizio anni '80)
- Object-oriented (inizio anni '90)

I linguaggi dei DB

- Lo *schema* di un DB è la descrizione dei dati e della loro struttura nel DB, definito attraverso l'attività di progettazione del DB
- Il DB contiene *istanze o occorrenze* dei dati.
- Ogni DBMS mette a disposizione
  - un *DDL* (data definition language) per la descrizione dello schema
    - contenuta nel *Data Dictionary*
  - un *DML* (data manipulation language) per la manipolazione delle istanze
    - query (interrogazioni)
    - modifiche al contenuto del DB (inserimenti, cancellazioni, aggiornamenti)

Programmazione ed uso di un DB

- Ogni DB viene usato da diverse applicazioni, ciascuna avente caratteristiche ed esigenze diverse.
- Un DBA (data base administrator) è responsabile della definizione e del mantenimento di uno schema in grado di soddisfare le diverse esigenze.
- Il DB può essere usato da applicazioni anche complesse che usano i dati del DB (e che contengono quindi istruzioni scritte nel DML del DB)
- Il DB può essere usato anche da utenti casuali attraverso interfacce che consentono di esprimere interrogazioni ad-hoc.

Il modello relazionale

Oggi è decisamente il più diffuso, grazie alla sua semplicità e alla sua sua flessibilità.

Per il modello relazionale:

- Un database è un insieme di relazioni (o tabelle).
- Ciascuna tabella è un insieme di tuple.
- Ciascuna tupla è una sequenza di attributi.
- L'attributo è l'unità elementare di informazione, contraddistinto dal **dominio**, cioè dall'insieme predefinito di valori che può assumere.

Esempio di tabella

Conto_corrente

Num_CC	Nome	Indirizzo	Saldo
1	Rossi	Via Roma	5800
2	Bianchi	Via Firenze	2700
3	Gialli	Via Milano	1300

Attributi : (Num_CC: intero; Nome:stringa;
Indirizzo:stringa;Saldo:intero)

Grado della relazione (numero di attributi) : 4

Cardinalità della relazione (numero di tuple): 3

Lo schema di un DB relazionale

- Lo schema di un database relazionale è dato dall'insieme degli schemi delle tabelle appartenenti al DB.
- Lo schema di una tabella è semplicemente l'elenco degli attributi, ciascuno col suo tipo (o dominio).
- Oltre ai soliti tipi, nei DB si trovano spesso i tipi date, time e money.

Esempio di schema relazionale

- Relation Conto_corrente

  Num_CC: integer,
  Nome: char(20),
  Indirizzo: char(20),
  Saldo: integer

- Relation Movimento

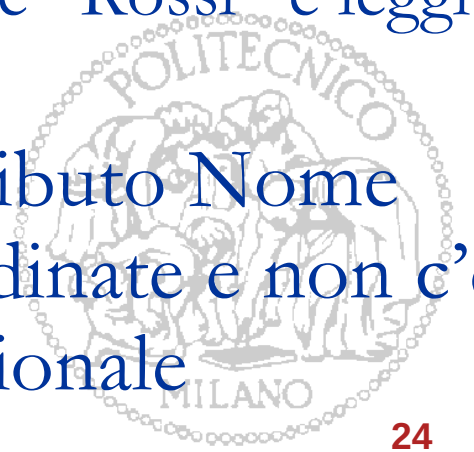
  Num_CC: integer,
  Data_mov: date,
  Num_mov: integer,
  Importo: integer,
  Causale: char(1)

Considerazioni sul modello relazionale

- Quando utilizziamo un modello relazionale, siamo vincolati ad introdurre informazioni che soddisfino lo schema.
  Ad es. nella relazione Conto_corrente possiamo introdurre solo il nome e l'indirizzo del cliente, non il suo numero di telefono o il codice fiscale. Per inserire queste informazioni dovremmo prima modificare lo schema.
- Partendo dalle relazioni esistenti è possibile ricavare informazioni non direttamente disponibili nel database.
  Ad es. l'elenco dei clienti che abitano in una certa zona e hanno versato più di un milione nel 1995.
- Questo tipo di operazioni viene fatto attraverso il query language (o linguaggio di interrogazione del DB).

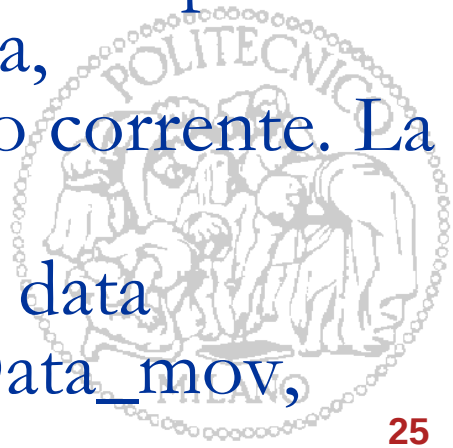
Accesso alle tuple

- L'accesso ad una certa tupla (o ad un insieme di tuple) di una relazione è sempre ed esclusivamente di tipo associativo (non posizionale): avviene in base al valore contenuto nella tupla.
- Ovvero, posso chiedere al DB il saldo di Rossi
  - trova la tupla in cui l'attributo Nome vale "Rossi" e leggi l'attributo Saldo
- Non posso chiedere il valore dell'attributo Nome della terza tupla: le tuple non sono ordinate e non c'è modo di fare un accesso diretto posizionale



Chiavi delle relazioni

- A causa dell'accesso associativo, è spesso utile dotare le relazioni di "chiave". Una chiave è un **insieme minimo** di attributi il cui valore identifica univocamente una tupla.
- La chiave serve per poter accedere ad una singola tupla.
  Ad es. nel caso del Conto_corrente si vuole poter aggiornare il Saldo di una specifica tupla, corrispondente ad un ben preciso conto corrente. La chiave sarà pertanto Num_CC.
- Nel caso del Movimento, la chiave sarà data dall'insieme degli attributi Num_CC, Data_mov, Num_mov.

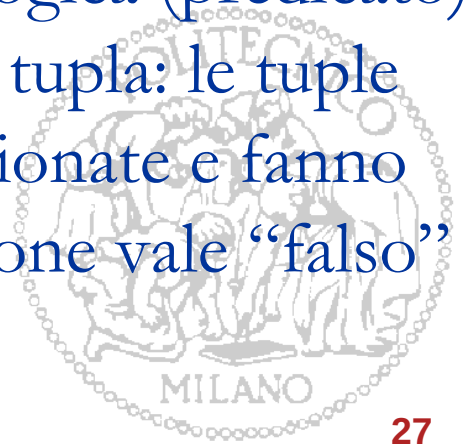


Le operazioni relazionali

- Ci sono operazioni che servono per combinare relazioni, limitandosi a leggere il contenuto del DB; restituiscono sempre una relazione
  - Operazioni unarie
    - hanno come operando un'unica relazione
  - Operazioni binarie
    - hanno come operando due relazioni
  - Operazioni insiemistiche
    - corrispondono alle solite operazioni di unione, differenza e intersezione
- Ci sono altre operazioni che servono a modificare il contenuto (istanza) del DB
- Ci sono operazioni che servono a modificare lo schema del DB

Le operazioni unarie - Selezione

- La selezione restituisce una relazione che è strutturalmente identica all'operando (ha lo stesso schema), ma contiene un sottoinsieme delle tuple dell'operando.
- Questa operazione effettua una selezione delle tuple della relazione "operando", utilizzando un criterio di selezione:
- Il **criterio di selezione** è una espressione logica (predicato) che viene valutata (vero/falso) su ciascuna tupla: le tuple per cui l'espressiona vale "vero" sono selezionate e fanno parte del risultato, le altre, per cui l'espressione vale "falso" sono scartate.



Esempio di Selezione

Selezioniamo dalla relazione Conto_corrente le tuple in cui Saldo > 2.000

Num_CC	Cognome	Indirizzo	Saldo
1	Rossi	Via Roma	2300
2	Verdi	Via Milano	2100
3	Bianchi	Via Firenze	1700

Selezione Saldo > 2000

Num_CC	Cognome	Indirizzo	Saldo
1	Rossi	Via Roma	2300
2	Verdi	Via Milano	2100

Le operazioni unarie - Proiezione

- L'operazione di proiezione consente di effettuare la scelta di particolari attributi di una relazione.
- Mentre la selezione elimina delle righe della tabella operando, la proiezione elimina delle colonne.
- Contrariamente al caso della selezione il criterio di eliminazione non dipende da un'epressione da valutare: infatti busogna specificare direttamente l'insieme degli attributi che vanno

Esempio di Proiezione

Proiettiamo la relazione Movimento sugli attributi Data_mov e Importo :

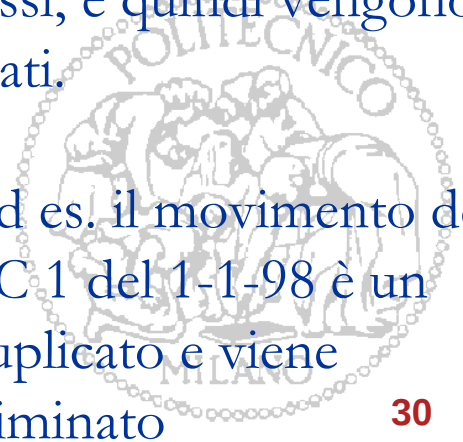
Num_CC	Data_Mov	Importo	Causale
1	1/1/1998	200	P
1	3/2/1999	300	V
2	1/1/1998	200	V
2	12/4/1999	300	V

Proiezione Data_mov e Importo

Data_Mov	Importo
1/1/1998	200
3/2/1999	300
12/4/1999	300

Nota: eliminando delle colonne può darsi che tra le tuple così ottenute esistano dei duplicati. Poichè le relazioni sono insiemi, i duplicati non sono ammessi, e quindi vengono eliminati.

Ad es. il movimento del CC 1 del 1-1-98 è un duplicato e viene eliminato



Il linguaggio SQL

- I DB relazionali dispongono di un linguaggio standard per le interrogazioni, quindi anche per fare selezioni e proiezioni.
- Questo linguaggio si chiama SQL (Structured Query Language)
- Il costrutto più comune è il blocco SELECT:

  SELECT lista di attributi

  FROM relazione

  WHERE predicato

Esempi di operazioni SQL

- Selezione delle tuple con Saldo > 2000 dalla relazone Conto

```
SELECT *
FROM Conto
WHERE Saldo > 2.000
```

- Proiezione degli attributi Data_mov e Importo dalla relazione Movimento:

```
SELECT Data_mov, Importo
FROM Movimento
```

- Nota: nei DB commerciali i duplicati sono ammessi. Per eliminarli occorre indicarlo esplicitamente, attraverso la parola chiave DISTINCT:

```
SELECT DISTINCT Data_mov, Importo
FROM Movimento
```

Operazioni combinate in SQL

- Una singola espressione SQL può indicare selezione e proiezione insieme:

```
SELECT Num_CC, Importo
FROM Movimento
WHERE Data_mov = 1-1-96
```

- L'operazione descritta seleziona le tuple in cui l'attributo Data_mov ha il valore prescritto, poi sopprime gli attributi diversi da Num_CC e Importo

Le operazioni binarie: Prodotto cartesiano

- Il prodotto cartesiano crea una relazione avente per tuple tutte le possibili combinazioni ottenibili combinando una tupla del primo operando con una tupla del secondo.

Prodotto cartesiano (**T1** , **T2**) =

T1
A
B
C

T2
1
2

T3
A1
A2
B1
B2
C1
C2

Le operazioni binarie: (equi-natural) Join

- Il prodotto cartesiano raramente è utile, perché generalmente si vogliono ottenere solo le combinazioni di tuple tra le quali esiste una certa corrispondenza, o vale una certa proprieta'.
- equi-natural Join: si combinano tra loro solo le tuple in cui valori delle relazioni in due attributi aventi dominio uguale verificano la proprieta' di uguaglianza; nella tabella risultante, si considera la colonna uguale una volta sola.

Join in SQL

- Per conoscere i nomi dei correntisti interessati da ciascun movimento devo combinare le tuple di Conto_corrente con le tuple di Movimento aventi uguale Num_CC:

```
SELECT Nome
FROM Conto_corrente, Movimento
WHERE
Conto_corrente.Num_CC=Movimento.Num_CC
```

Join in SQL

- Il join è combinabile con la selezione e la proiezione in un'unica operazione SQL: semplicemente il predicato non indicherà solo la corrispondenza tra tuple, ma anche un criterio cui le tuple del risultato dovranno essere conformi
- Voglio conoscere importo e causale dei movimenti di Rossi del 1-1-96:

```
SELECT Importo, Causale
FROM Conto_corrente, Movimento
WHERE  Conto_corrente.Num_CC=
       Movimento .Num_CC AND
       Nome = "Rossi" AND
       Data_mov = 1-1-96
```

Le operazioni insiemistiche

- Corrispondono alle normali operazioni sugli insiemi
- Occorre notare che per avere come risultato dei veri insiemi occorre sempre indicarlo esplicitamente usando la keyword DISTINCT

  Altrimenti ad es. l'unione darà semplicemente un risultato contenente le tuple del primo operando e quelle del secondo, duplicati compresi.
- In SQL queste operazioni si chiamano rispettivamente
  - UNION
  - MINUS
  - INTERSECT

Esempio di interrogazione

- Estraiamo i CC che hanno un saldo >2.000 e per i quali non è stato fatto alcun movimento per un importo >1.000

```
SELECT Num_CC
FROM Conto_corrente
WHERE Saldo > 2.000
MINUS
SELECT Num_CC
FROM Movimento
WHERE Importo > 1.000
```

Stili di interrogazione imperativo e dichiarativo

- L'SQL supporta uno stile di interrogazione dichiarativo, nel senso che le query specificano le caratteristiche del risultato ma non indicano come estrarlo.
  - Ad es. nelle selezioni non diciamo se la tabella deve essere scandita dall'alto o dal basso, o quale parte della condizione deve essere valutata per prima.
- Le query vengono interpretate dal DBMS, che provvede anche a ottimizzarle.
  - Ad es. dovendo fare un join e una selezione conviene fare la selezione prima, per avere meno tuple su cui valutare la condizione del join.

Manipolazione dei dati

- In SQL esistono tre operazioni fondamentali per manipolare le tuple:
  - INSERT: inserimento di nuove tuple
  - UPDATE: modifica del valore di attributi di tuple esistenti
  - DELETE: cancellazione di tuple

L'operazione INSERT

- Inserisce un insieme di tuple in una relazione
- Le tuple da inserire possono essere date esplicitamente:

```
INSERT INTO Conto_corrente
(4, "Conti", "V. Piave, 9", 0),
(5, "Russo", "V. Zara, 19", 100)
```

- Le tuple da inserire possono essere il risultato di una interrogazione:

```
INSERT INTO Conto_corrente
SELECT *
FROM Altri_conti
WHERE Saldo > 1500
```

L'operazione DELETE

- Cancella da una relazione le tuple corrispondenti ad una data condizione

```
DELETE FROM Movimento
WHERE Causale = "S"
```

L'operazione UPDATE

- Modifica tutte le tuple che soddisfano il predicato dato nel modo indicato.
  Ad es. per incrementare del 1 per mille il saldo dei conti correnti con saldo > 50.000:

```
UPDATE Conto_corrente
SET Saldo = Saldo*1.001
WHERE Saldo > 50.000
```

- Se si vuole aggiornare solo una tupla alla volta, occorre che il predicato sia espresso sugli attributi che compongono una chiave, e che sia soddisfatto per un'unica combinazione di valori:

```
UPDATE Conto_corrente
SET Saldo = Saldo*1.07
WHERE Num_CC = 123
```

Definizione di relazioni: CREATE

Il comando Create permette di creare una relazione, specificando gli attributi con i relativi domini.

```
CREATE TABLE Conto_corrente
    (Num_CC: integer,
     Nome: char(20),
     Indirizzo: char(20),
     Saldo: integer)
```

Progettazione di un DBMS

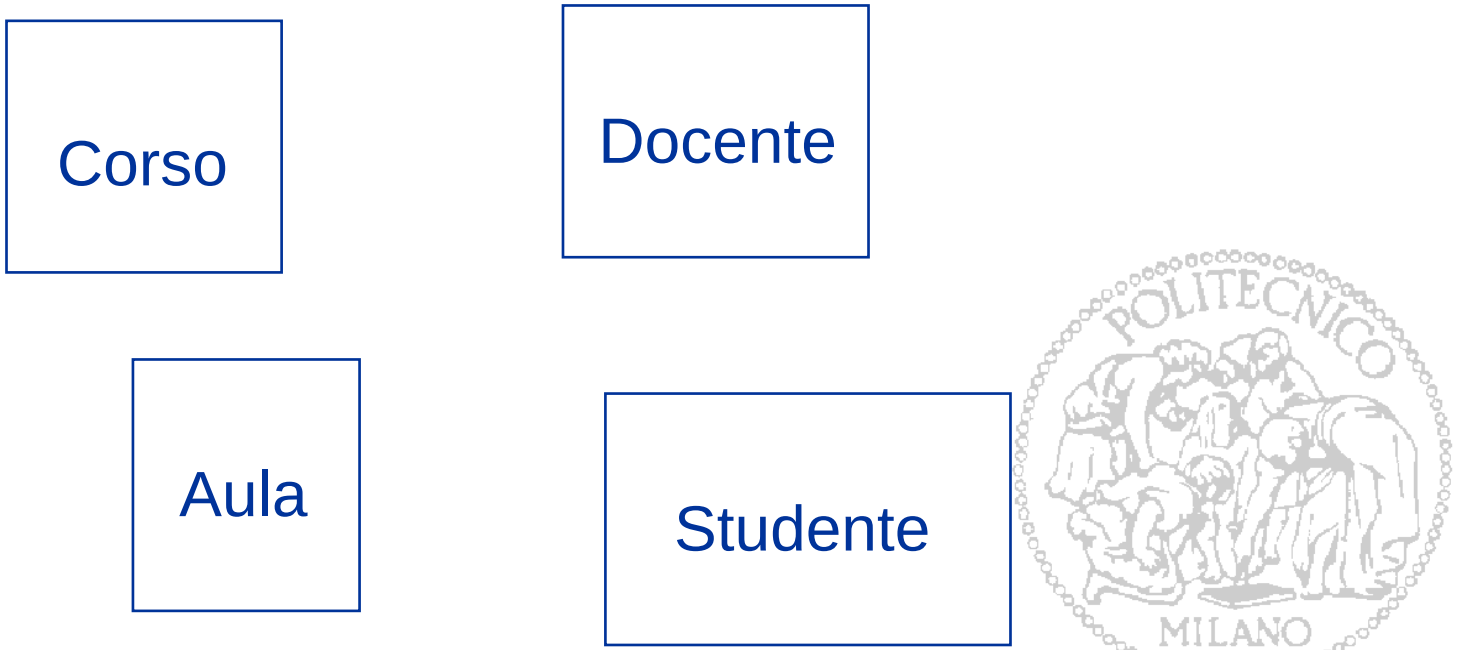
- Progettare un DBMS vuol dire essenzialmente definirne lo schema
  - capire quali tabelle servono
  - definire lo schema di ciascuna tabella
  - creare le chiavi
- La progettazione delle query e delle modalità di interazione fa invece parte della progettazione delle applicazioni che devono sfruttare il DBMS

Procedura di progettazione di un DB

- Innanzitutto si cerca di produrre un modello concettuale dei dati da gestire
- Il modello è concettuale in quanto prescinde dal modello logico dei dati adottato dal particolare DBMS scelto
- Un formalismo molto usato per la descrizione concettuale dei dati è costituito dai diagrammi **Entity/Relationship** (Chen)
  - Entità: ciò che è di interesse per il sistema
  - Relazioni: legami di diversa natura tra entità
  - Attributi: caratteristiche (proprietà) delle entità e delle relazioni

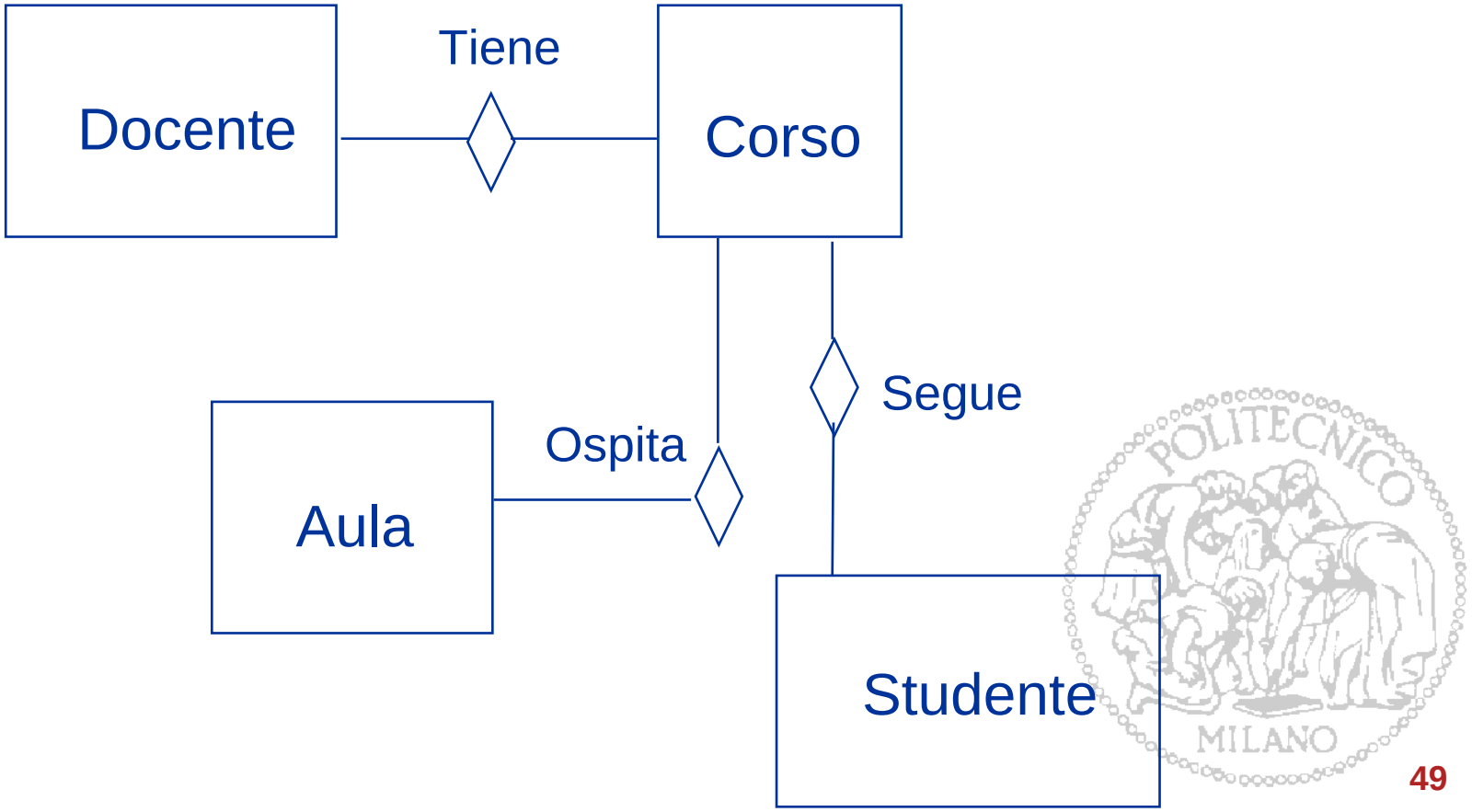
Diagrammi E/R: Entità

Esempio: definiamo le entità necessarie per la gestione di un corso di aggiornamento professionale:



Diagrammi E/R: Relazioni

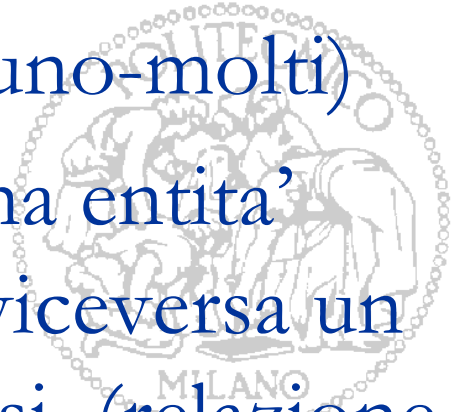
Definiamo le relazioni per il nostro esempio:



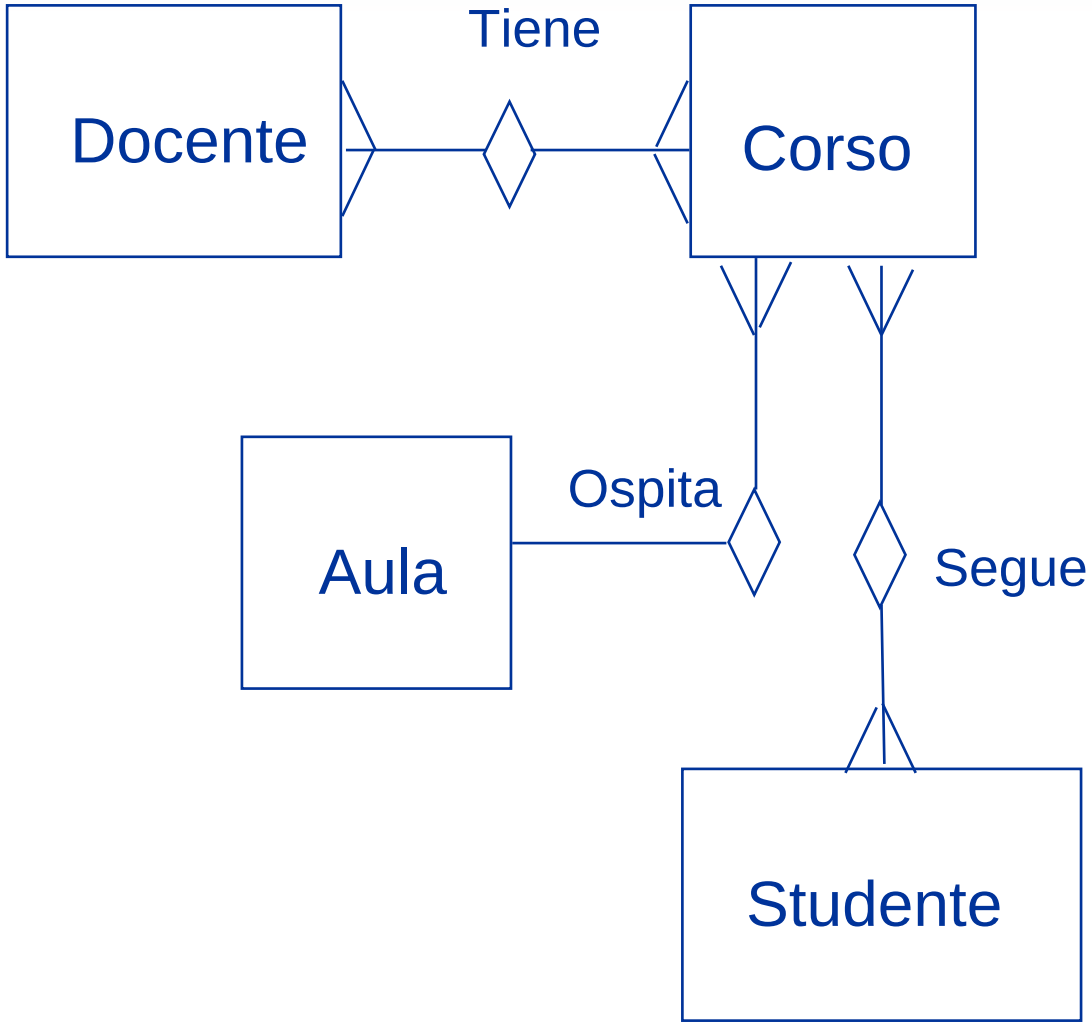
Diagrammi E/R: Molteplicità relazioni

Molteplicità delle relazioni: indicata graficamente dal "ventaglio" ad una estremita' della relazione; ad es:

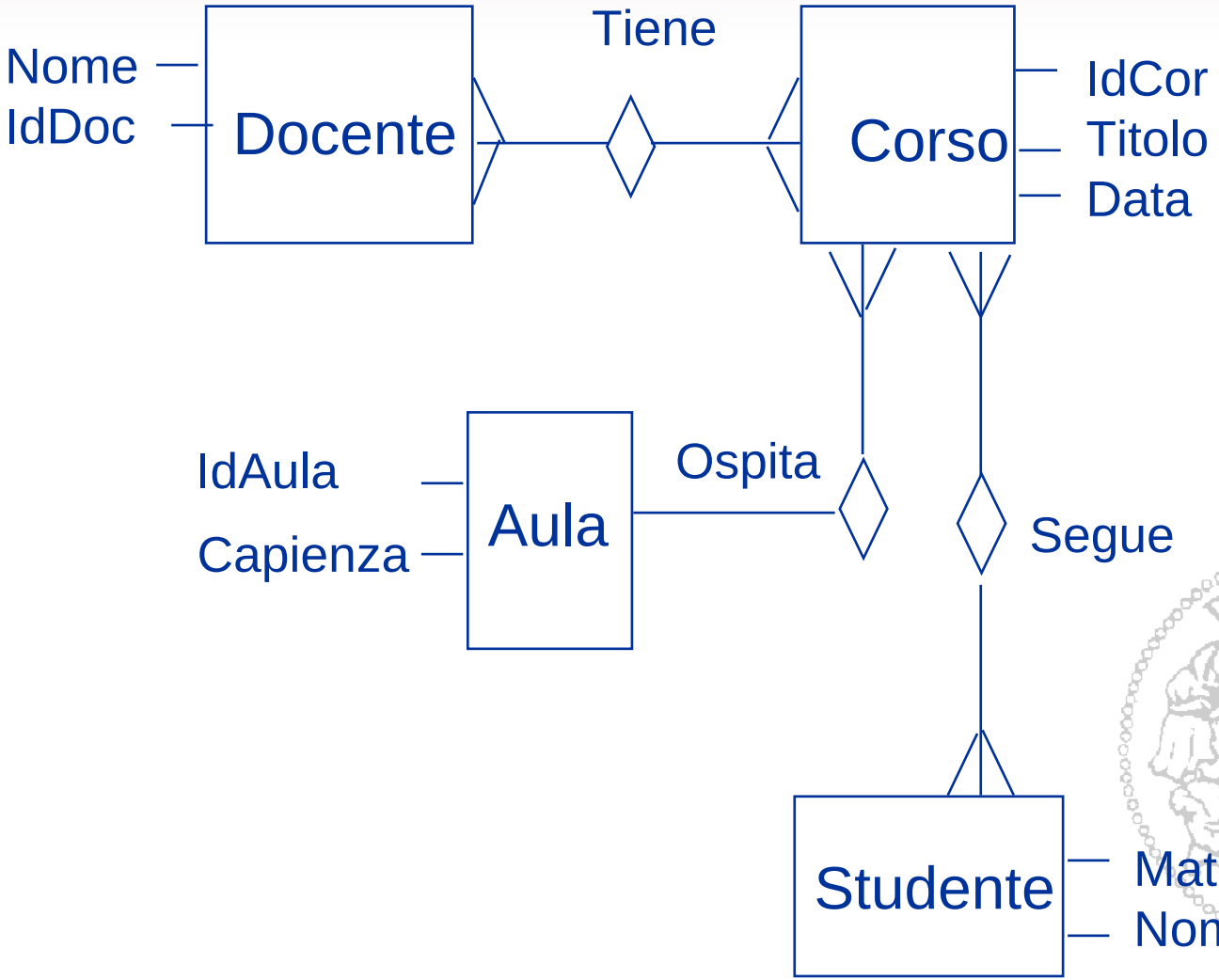
- Aula--<-Corso, indica che una entita' Aula *puo'* ospitare piu' Corsi ma un Corso puo' essere ospitato in una sola aula (relazione uno-molti)
- Corso->---<-Studente indica che una entita' Studente puo' seguire piu' Corsi, e viceversa un Corso piu essere seguito da piu' corsi (relazione



Diagrammi E/R: Molteplicità relazioni



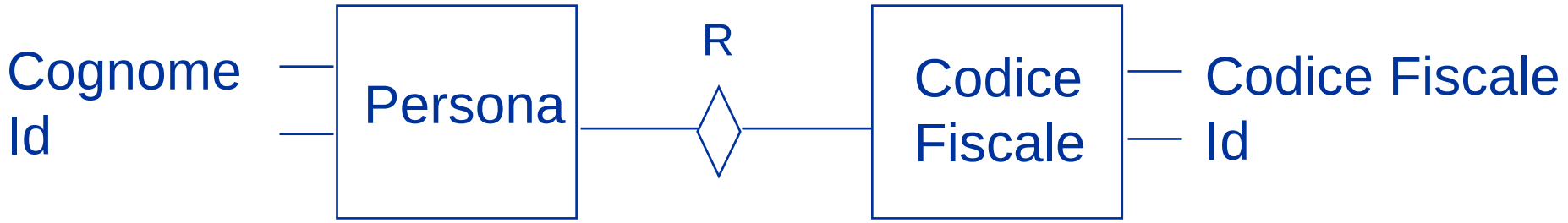
Diagrammi E/R: Attributi



Realizzare un DB partendo dal modello E/R

- Un diagramma E/R può essere "trasformato" in uno schema logico di DB
- Tuttavia, i diagrammi E/R sono particolarmente adatti al progetto di DB relazionali, perchè la trasformazione di un diagramma E/R in uno schema relazionale è molto semplice:
  - un'entità diventa una tabella
  - un attributo di un'entità diventa un attributo di una tabella
  - le relazioni tra entità possono diventare riferimenti diretti da una tupla di una tabella a tupla/e di un'altra oppure possono dar luogo ad una relazione aggiuntiva, a seconda dei casi.

Traduzione di relazioni 1-1



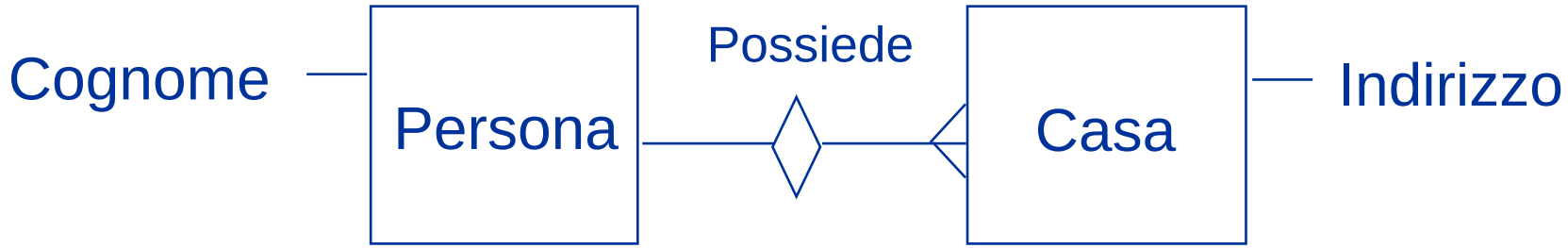
- Persona(Id: integer,Cognome: char(1),R: integer)
- CodiceFiscale(IdCF: integer, CodiceFiscale: char(16))

Una relazione 1-1 diventa una sequenza di attributi

Persona

IdP	Cognome	Codice Fiscale
1	Bocchi	BCCFTRD12342
2	Naggi	NGGFPP23836D
3	Brazov	BRZAIE2376DD

Traduzione di relazioni 1-n

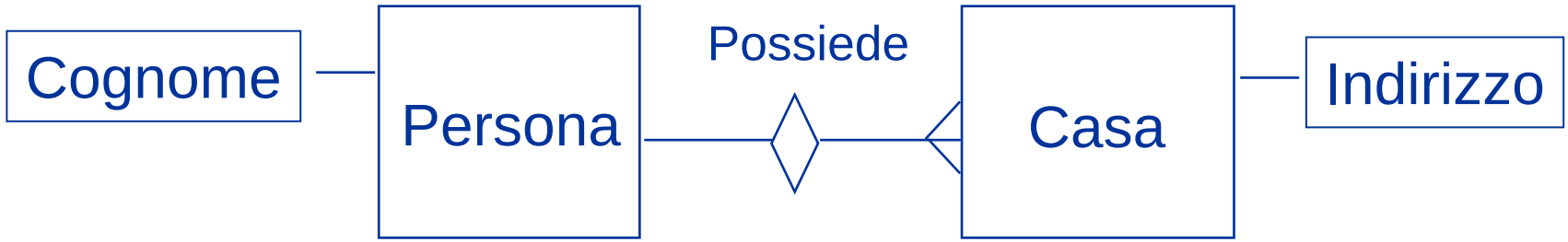


traduzione con la regola 1-1 ? NO ! (Problema delle ridondanze)

IdP	Cognome	R
1	Naggi	3
1	Naggi	4
2	Bocchi	1
3	Brazov	2
3	Brazov	5

IdC	Indirizzo
1	Via Rossini, 4
2	Via Verdi,
3	P.zza Cavour, 1
4	Corso Roma, 77
5	Via Dante, 10

Traduzione di relazioni 1-n

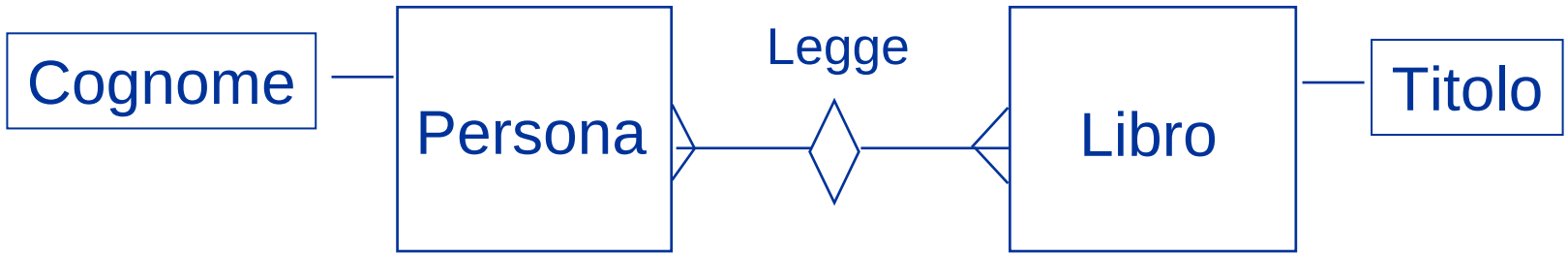


Si possono ridurre le ridondanze rappresentando la relazione (inversa) nella seconda tabella

IdP	Cognome
1	Naggi
2	Bocchi
3	Brazov

IdC	Indirizzo	R
1	Via Rossini, 4	2
2	Via Verdi,	3
3	P.zza Cavour, 1	1
4	Corso Roma, 77	1
5	Via Dante, 10	3

Traduzione di relazioni n-n



Si realizza introducendo una apposita tabella (chiamata tabella ponte) per rappresentare R

IdP	Cognome
1	Naggi
2	Bocchi
3	Brazov

IdP	IdL
1	3
1	4
2	1
3	2
3	5
2	3

IdL	Titolo
1	Architettura oggi
2	Storia del design
3	Computergrafica
4	Storia di Roma
5	Manuale del calcio

Ridondanze e inconsistenze

- Si consideri la relazione Fornitore, che lega IdFornitore a Città.
- Lo schema della relazione è tale che la coppia (IdFornitore, Città) viene ripetuta diverse volte (ridondanza)
- Se il fornitore cambia città, occorre modificare tutte le tuple relative, altrimenti si ha un'inconsistenza

IDFornitore	Città	Articolo	Pezzi
1	MI	123232	232
1	MI	324324	433
1	MI	324233	122
2	TO	434333	333
3	VE	234232	210
3	VE	543544	677

Eliminazione ridondanze

- Riunire in un'unica tabella i gruppi di informazioni ridondanti

IDFornitore	Città
1	MI
2	TO
3	VE

IDFornitore	Articolo	Pezzi
1	123232	232
1	324324	433
1	324233	122
2	434333	333
3	234232	210
3	543544	677

Altro esempio di ridondanza

- Tabella "anagrafica"

  il fatto che una persona risieda in una certa provincia dipende dalla collocazione della città in cui vive : è ridondante riportare la provincia data la città

Cognome	Città	Provincia	Data_nascita
Rossi	Cuggiono	Milano	1/1/75
Verdi	Cuggiono	Milano	2/9/73
Bianchi	Cuggiono	Milano	9/3/69
Gialli	Varese	Varese	2/5/74

Cognome	Città	Data_nascita
Rossi	Cuggiono	1/1/75
Verdi	Cuggiono	2/9/73
Bianchi	Cuggiono	9/3/69
Gialli	Varese	2/5/74

Città	Provincia
Cuggiono	Milano
Varese	Varese