MODELLO ENTITY RELATIONSHIP (ER)

Peter Chen 1976

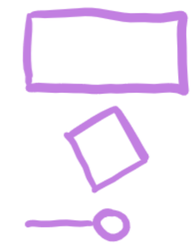
25/02/25

modello concettuale → quali sono le nostre risorse?

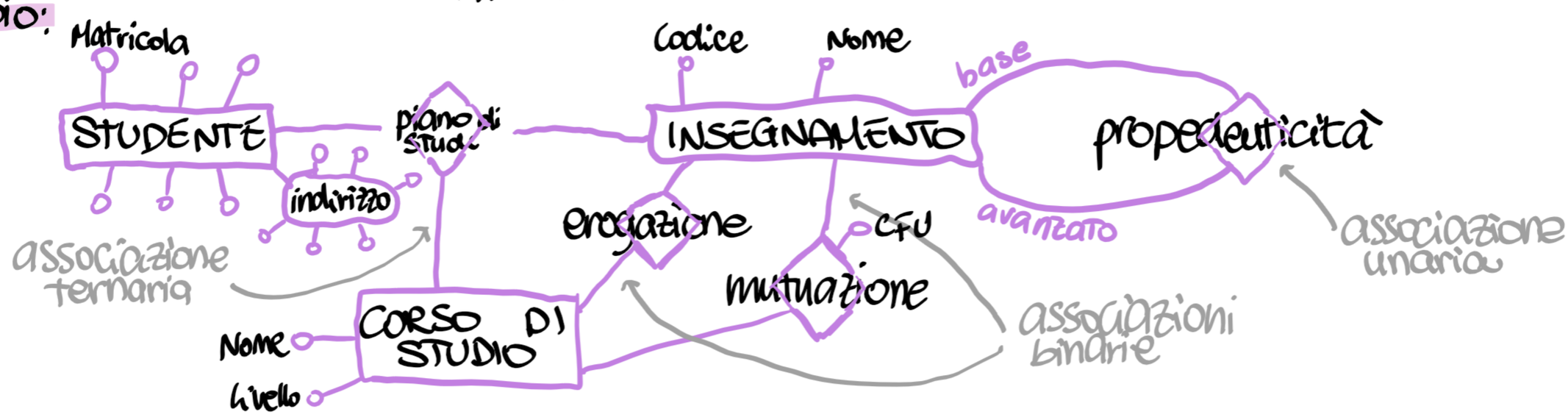
insieme di costrutti →

- Entità → collezioni di oggetti con stesse proprietà
- Associazioni → legami tra istanze di entità
- Attributi

rappresentati con



esempio:



in un diagramma non avremmo mai un ▭ e un ◇ da soli!!!
Un ◇ non può collegare un altro ◇, e un ▭ non può collegare un altro ▭.

VINCOLI: es: per ogni propedeuticità, l'insegnamento di base e quello avanzato devono essere distinti

diversi tipi di vincoli: vincoli di dominio → rappresentazione: dom(Livello, CDS) = {T, M, U}

Gli attributi possono avere dei sovrattributi? ⬭ (pallozzo id K) ATTRIBUTO COMPOSTO

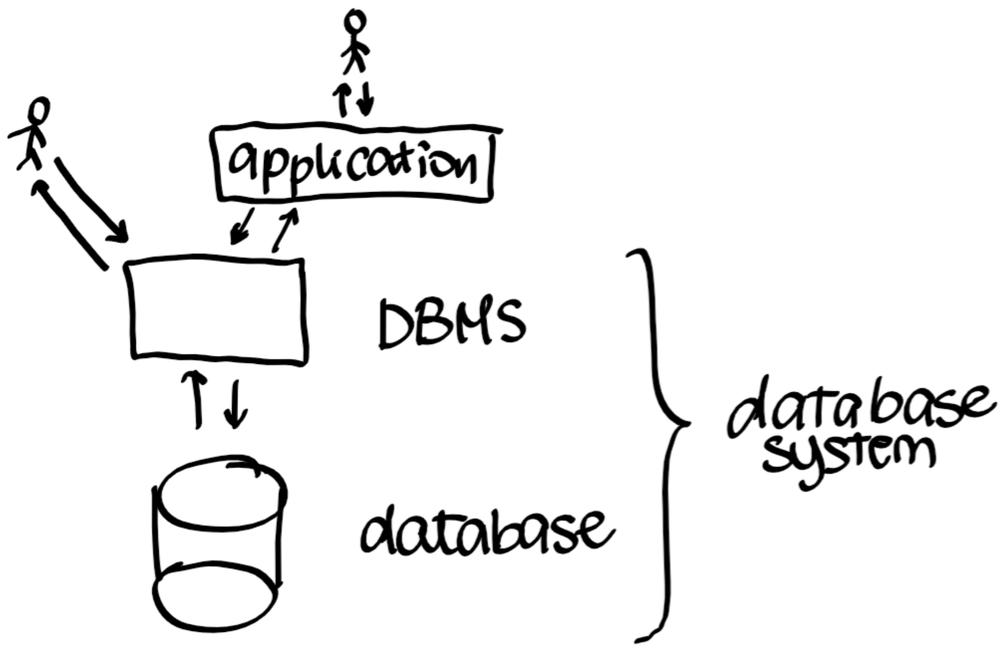
DIFFERENZA TRA BASI DI DATI E SISTEMI DI GESTIONE DI DATI

27/02/25

base di dati → collezione di dati tra loro correlati, informazioni di interesse in un sistema informativo

sistema di gestione di basi di dati (DBMS) → sistema software che fornisce gli strumenti necessari a gestire basi di dati

una base di dati è una collezione di dati gestita da un DBMS.



PROBLEMI:

1) rindondanza e inconsistenza
2) difficoltà di accesso ai dati
3) integrità dei dati
4) accesso concorrente
5) protezione dei dati

LA "CARTA VINCENTE" DEL DBMS

- schema logico della base di dati
  - descrive la struttura della base di dati all'interno del DBMS
  - astrae dai dettagli dell'implementazione fisica
  - descrizione in termini di un modello dei dati
- unica rappresentazione centralizzata e condivisa
  - ridurre le rindondanze e le inconsistenze
  - permette controllo centralizzato
- cambia raramente

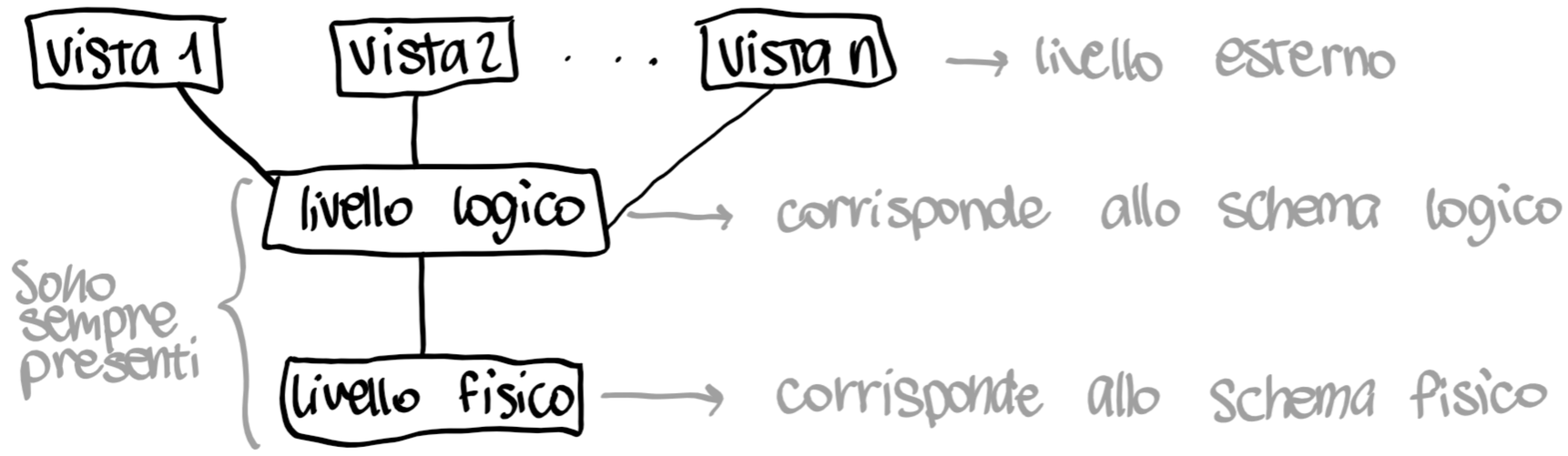
ISTANZA DELLA BASE DI DATI: insieme dei dati presenti in un dato momento in una base di dati

MODELLO RELAZIONALE tabella = relazione riga = tupla

SECONDA "CARTA VINCENTE" DEL DBMS

- mette a disposizione dei linguaggi (dichiarativi) per agire sui dati rappresentati secondo il modello

I DATI VENGONO RAPPRESENTATI SU PIÙ LIVELLI



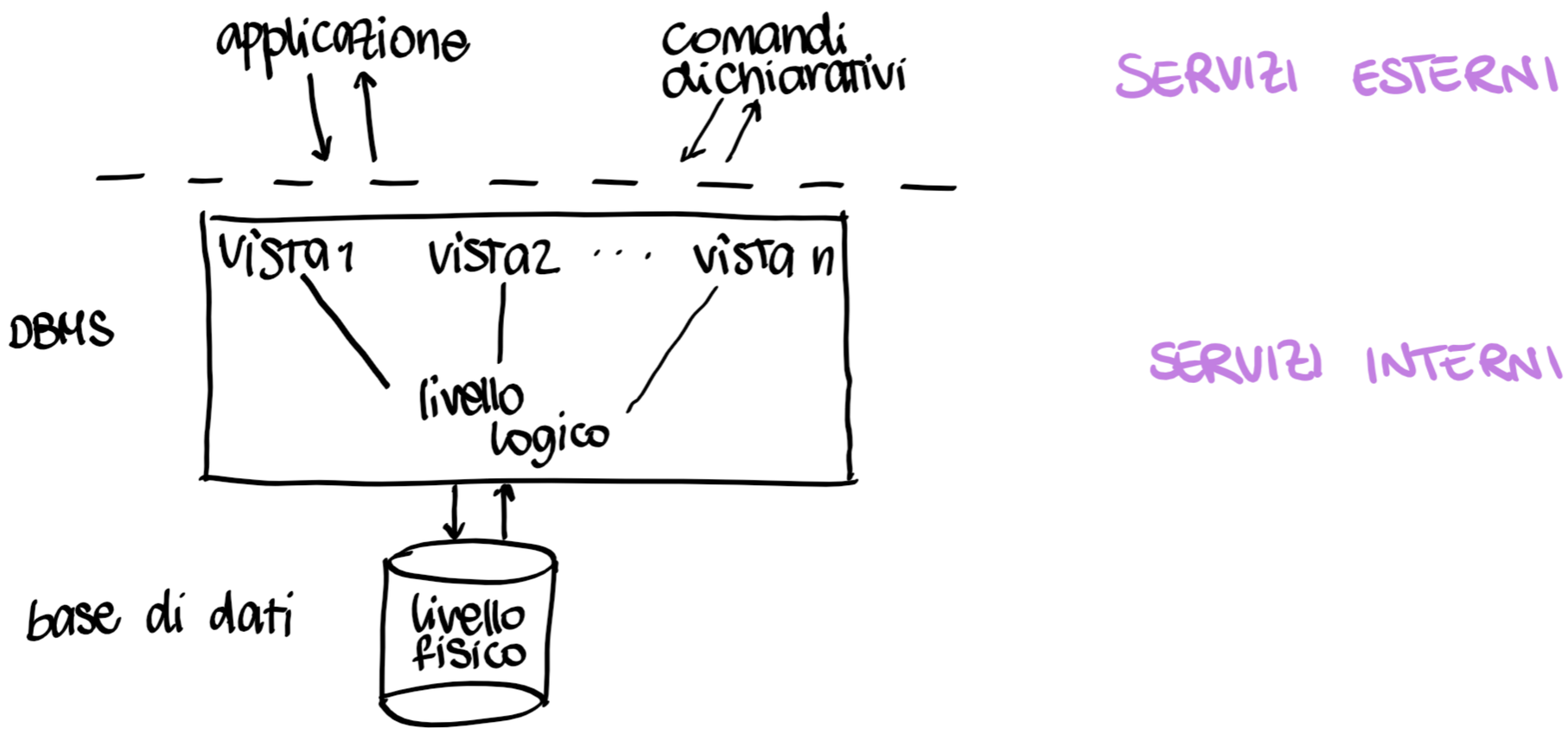
VANTAGGI DI QUESTO APPROCCIO A TRE LIVELLI:

- indipendenza fisica → cambiano le strutture di memorizzazione, non il livello logico
- indipendenza logica → cambia lo schema logico (aggiunta di un attributo), solo una parte delle viste cambia

I LINGUAGGI:

- Data Definition Language (DDL): permette di specificare e modificare lo schema della base di dati e lo schema delle viste
  LIVELLO LOGICO ED ESTERNO
- Data Manipulation Language (DML): permette di creare, modificare e interrogare l'istanza della base di dati
  LIVELLO LOGICO (ED ESTERNO)
- Storage Definition Language (SDL): definisce lo schema fisico del DB
  LIVELLO FISICO

il DBMS risponde alle domande precedenti mediante servizi



SERVIZI ESTERNI AL DBMS

- Gli utenti comunicano con il DBMS, tramite comandi in linguaggi dichiarativi, invocando servizi esterni
- Manipolazione dei dati: inserire nuovi dati, oppure accedere, modificare o cancellare dati esistenti
- Possono essere invocati tramite:
  - interfacce (testuali o grafiche)
  - tramite API per l'esecuzione dei comandi dichiarativi all'interno di programmi (applicazioni) scritti in linguaggi general purpose

SERVIZI INTERNI AL DBMS

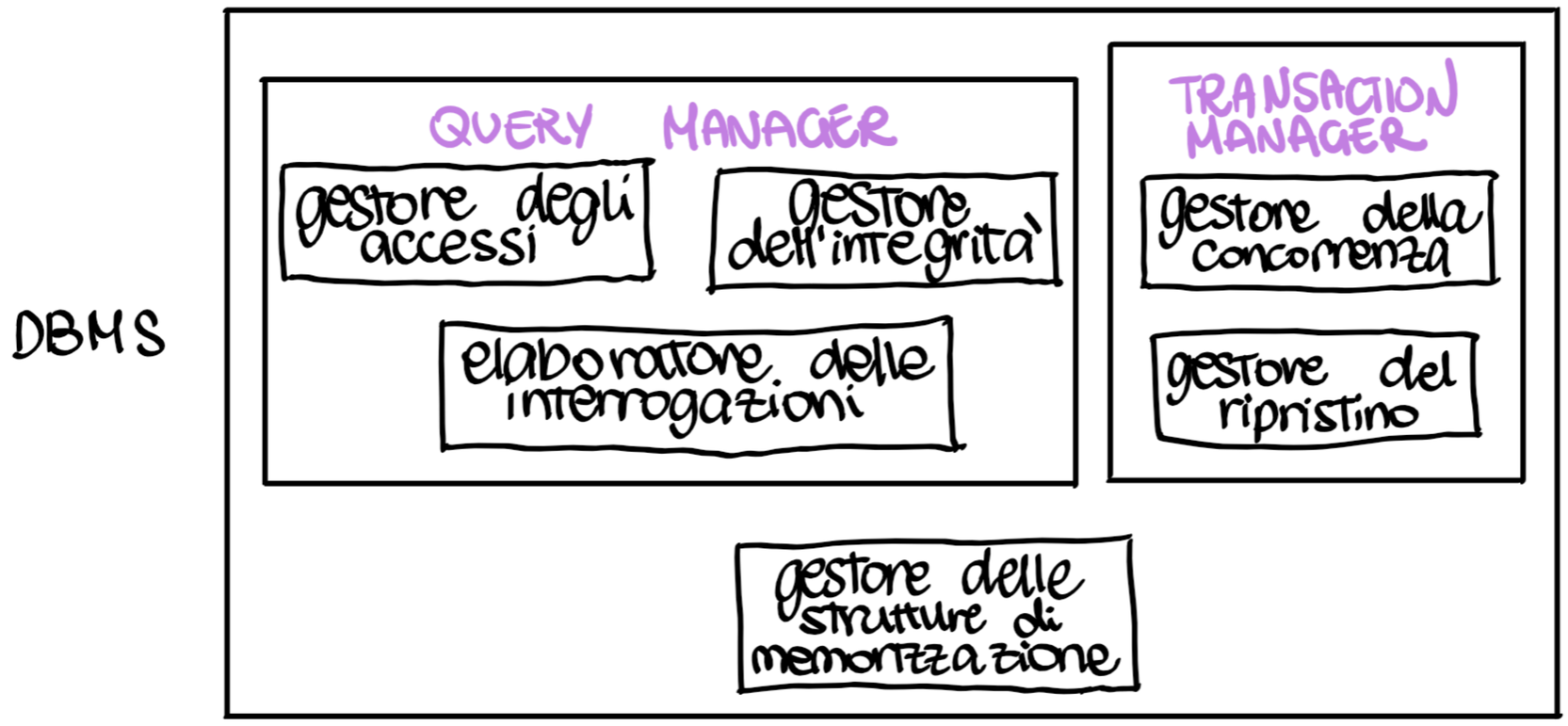
Il DBMS implementa i servizi esterni mediante molteplici servizi interni

Un DBMS è un sistema software che offre molteplici servizi al fine di garantire:

- controllo di integrità
- strutture di memorizzazione (grandi quantità di dati persistenti e condivisi)
- ottimizzazione di interrogazioni (efficienza)
- protezione dei dati (gestione degli accessi)
- ripristino della base di dati
- controllo della concorrenza

(esecuzione coordinata delle richieste e protezione da malfunzionamenti)

Ogni servizio corrisponde ad uno o più componenti software !!!



COMPONENTI DI UN DBMS

28/02/25

Tutte le diverse componenti hanno bisogno di accedere ai dati utente e/o dati di sistema per funzionare.

I DATI DI SISTEMA vengono memorizzati nella base di dati insieme ai dati utente; sono gestiti dal sistema come i dati utente, disponibili a livello logico come tabelle e memorizzati su disco come file.

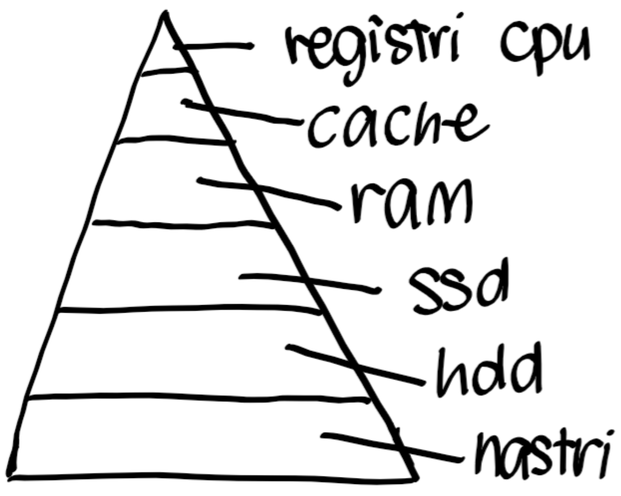
ARCHITETTURA CLIENT-SERVER

- I servizi interni ed esterni vengono forniti da una singola macchina, il server
- le applicazioni risiedono e vengono eseguite su macchine client, dalle quali vengono richiesti i servizi esterni

ARCHITETTURA DEI DBMS

1) GESTORE DELLE STRUTTURE DI MEMORIZZAZIONE

le memorie si possono classificare in base alla loro capacità e al loro tempo per trovare certi dati



prestazioni di una memoria:

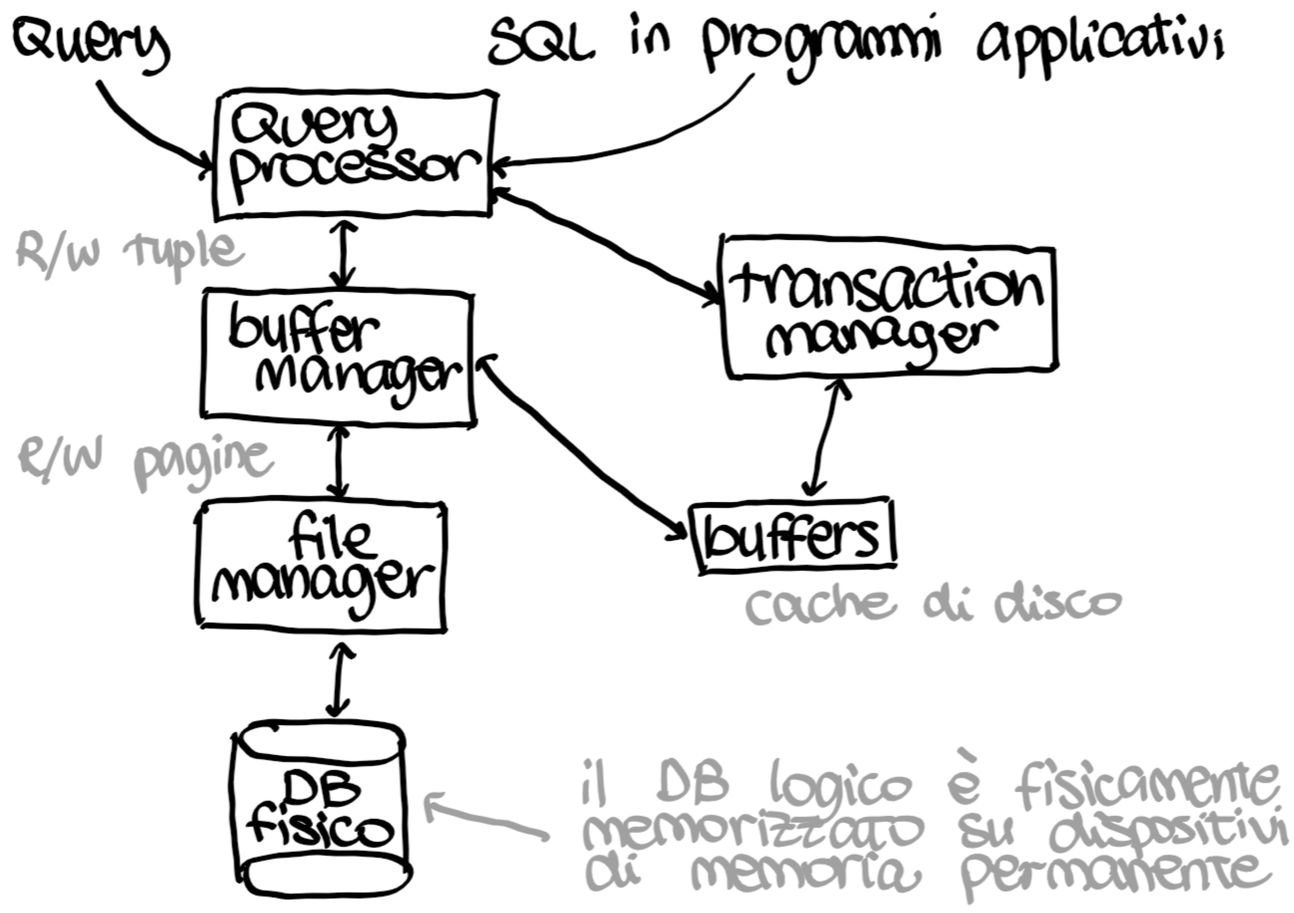
tempo di accesso ottenuto dalla somma di:

- latenza: tempo necessario per accedere al primo byte di interesse
- tempo di trasferimento: tempo necessario per spostare i dati tra i livelli della gerarchia

$$\text{tempo di accesso} = \text{latenza} + \frac{\text{dimensione dati da trasferire}}{\text{velocità di trasferimento}}$$

I dati devono essere trasferiti da memoria non volatile a memoria centrale (RAM, cache) per essere elaborati dal DBMS (trasferimento a blocchi/pagine)

GESTORE DELLE STRUTTURE DI MEMORIZZAZIONE



TIPI DI DISPOSITIVI:

- Hard Disk (HD): dispositivo elettro-meccanico per la conservazione di informazioni sotto forma magnetica, su supporto rotante a forma di piatto su cui agiscono delle testine di lettura/scrittura. Le informazioni sono memorizzate su una superficie del disco in cerchi concentrici detti tracce, divisa in settori. Per leggere o scrivere un settore si usa un braccio con testina magnetica: i dati vengono letti/scritti quando il settore passa sotto la testina. Nei dischi a piatti multipli, le tracce con lo stesso diametro sulle varie superfici sono dette cilindro; i dati sullo stesso cilindro possono essere recuperati molto più velocemente che nei dati distribuiti su diversi cilindri.

COMMAND OVERHEAD TIME: tempo necessario a impartire comandi al drive → trascurabili
SETTLE TIME: tempo richiesto per la stabilizzazione del braccio
SEEK TIME: tempo impiegato dal braccio a posizionarsi sulla traccia desiderata
ROTATIONAL LATENCY: tempo di attesa del primo settore da leggere
tempo di trasferimento: velocità con cui si trasferiscono byte dai piatti sulla cache del controller.

TEMPO DI TRASFERIMENTO DI UN BLOCCO:

tempo impiegato dalla testina per trasferire un blocco nel buffer, una volta posizionata all'inizio del blocco. Questo tempo è molto più breve del tempo di seek e dipende dalla dimensione del blocco (P) e della velocità di trasferimento (Tr)

$$Tt = P/Tr$$

Il gestore delle strutture di memorizzazione deve cercare di ridurre il tempo di latenza

IL DB FISICO

A livello fisico un DB consiste in un insieme di file, ognuno dei quali viene visto come una collezione di pagine/blocchi, di dimensione fissa. Ogni pagina/blocco memorizza più record. A sua volta un record consiste di più campi, di lunghezza fissa e/o variabile, che rappresentano gli attributi (le "colonne" della tabella)

- ogni relazione del DB è memorizzata in un proprio file
- tutto il DB è memorizzato in un singolo file

RECORD:

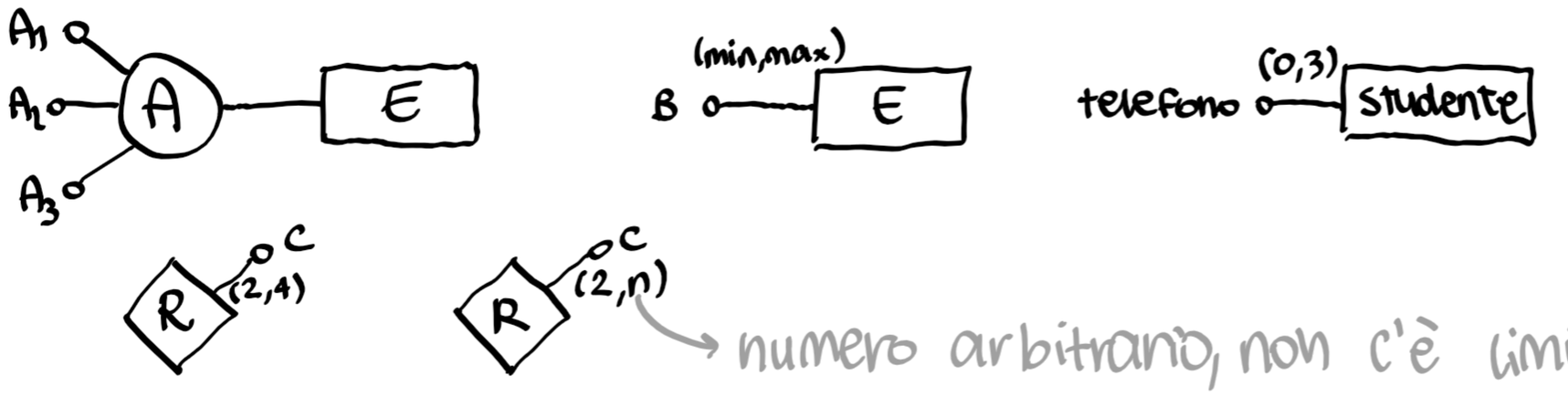
I dati sono memorizzati in forma di record. Un record è costruito da un insieme di valori collegati, ognuno formato da uno o più byte e corrisponde ad un campo del record. Una collezione di nomi di campi a cui sono associati i tipi corrispondenti costituisce un tipo di record. Per ogni tipo di record nel DB deve essere definito uno schema (fisico) che permetta di interpretare correttamente il significato dei byte che costruiscono il record. Quindi esiste un mapping tra lo schema delle tuple della relazione e un tipo di record.

FILE:

Un file è una sequenza di record. Un file è detto file con record a lunghezza fissa se tutti i record memorizzati nel file hanno la stessa dimensione (in byte), altrimenti parliamo di file con record a lunghezza variabile

---

03/03/25

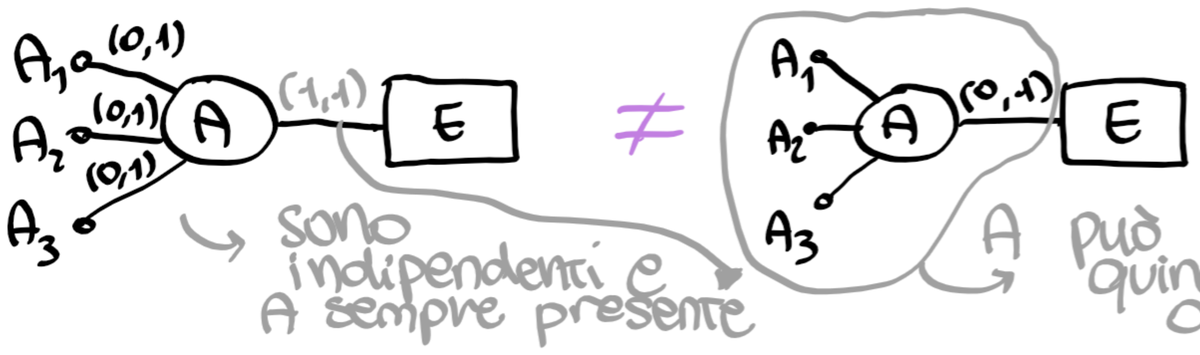


numero arbitrario, non c'è limite al numero massimo

(min, max) min ∈ ℕ , max ∈ ℕ⁺ ∪ {n}

(2, 2)

se non metto la coppia, l'attributo assume (1,1) default

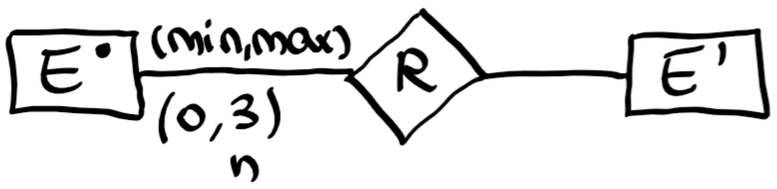


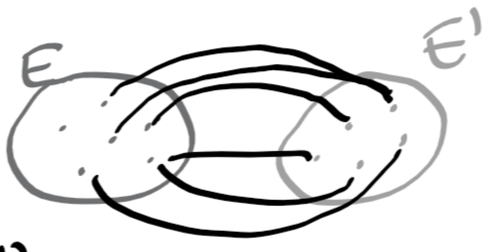
sono indipendenti e A sempre presente

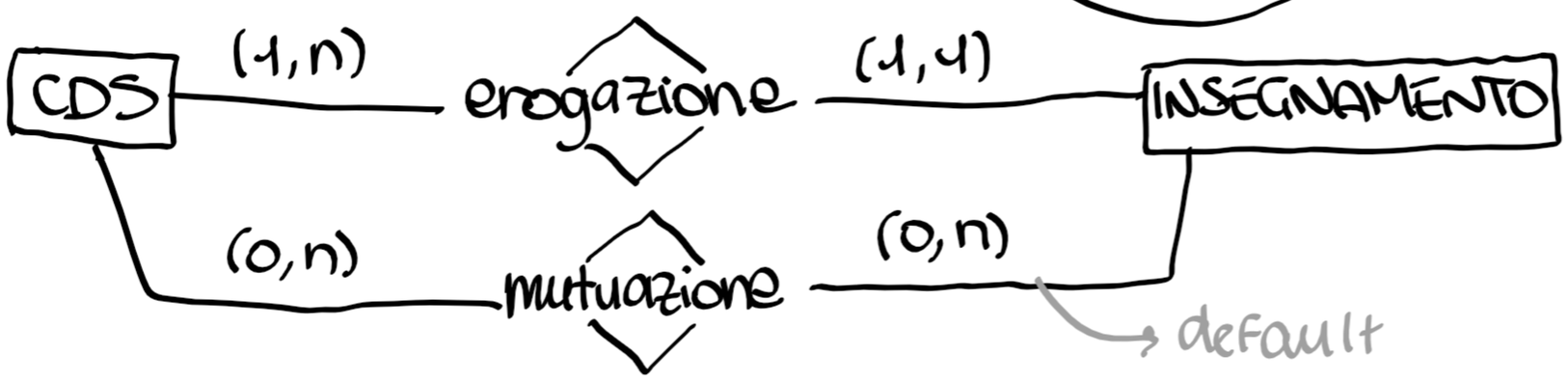
A può essere o non essere presente, quindi gli attributi o sono tutti presenti o sono tutti assenti

VINCOLI DI CARDINALITÀ PER ATTRIBUTI

VINCOLI DI CARDINALITÀ PER ASSOCIAZIONI





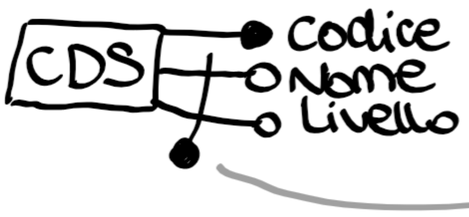


default

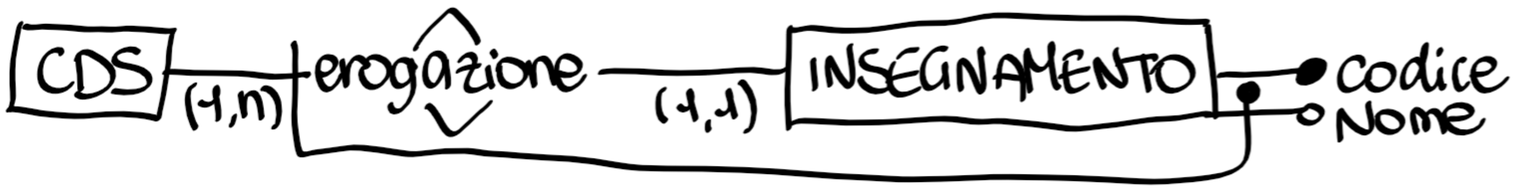
VINCOLI DI IDENTIFICAZIONE / UNICITÀ

identificatori interni semplici

elementi nell'insieme che devono essere DIVERSI

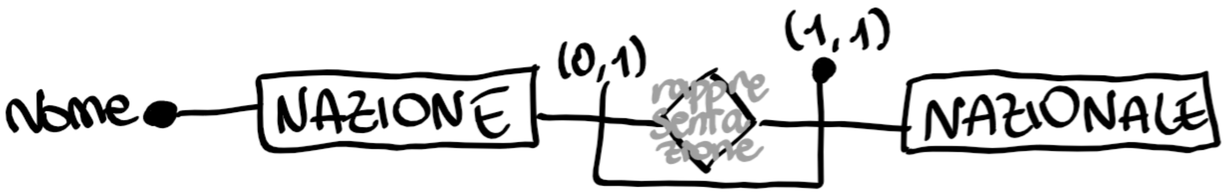


identificatore interno composto

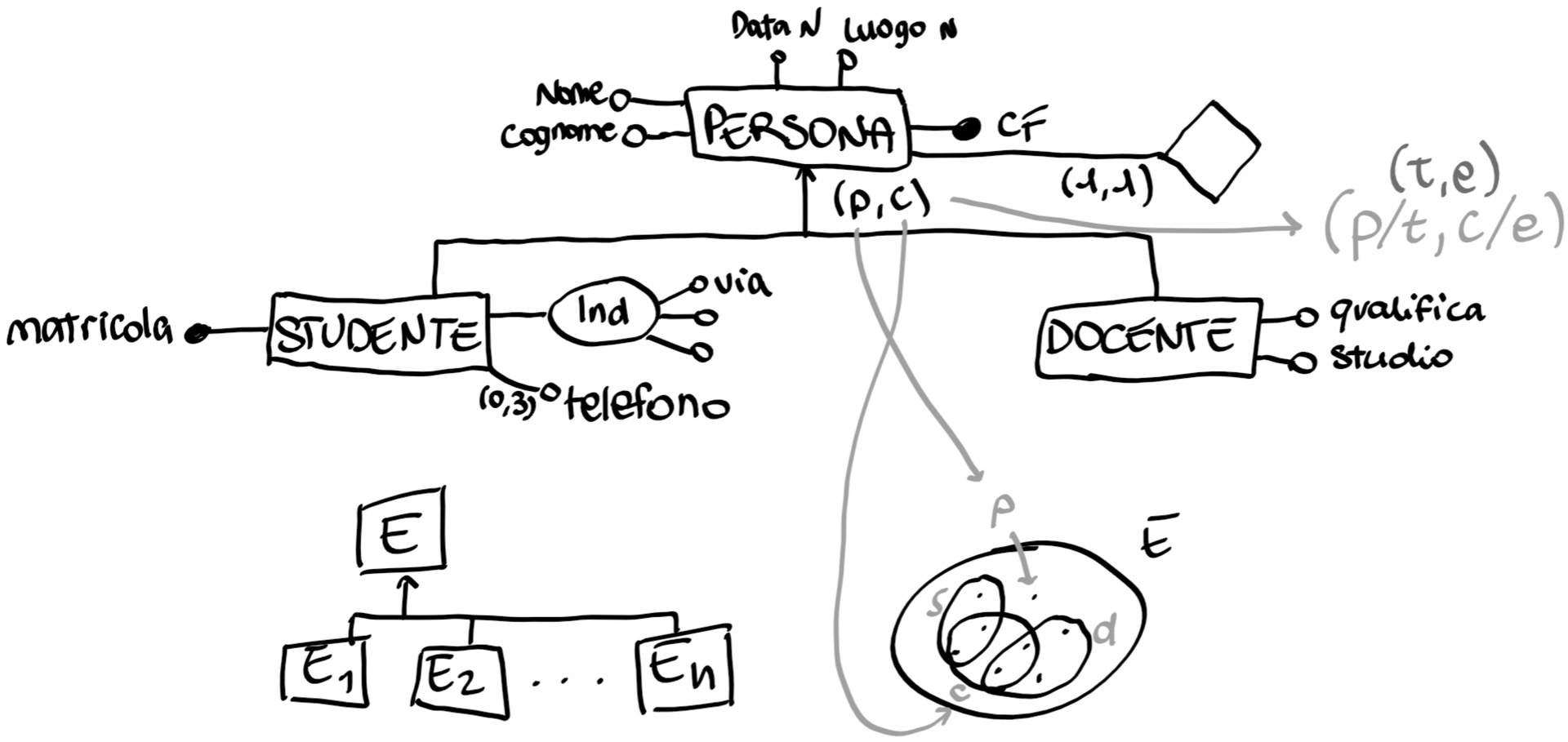


Un corso di studi non eroga due insegnamenti con lo stesso nome

identificatore misto



identificatore esterno semplice



SUPER ENTITÀ
ENTITÀ PADRE

SOTTO ENTITÀ
ENTITÀ FIGLIA

07/03/25

ORGANIZZAZIONE DEI RECORD IN BLOCCHI

normalmente la dimensione dei record è molto minore di quella di una pagina

- nel caso di record a lunghezza fissa l'organizzazione in una pagina si potrebbe presentare così:

page header	record 1	record 2	. . . .	record n	

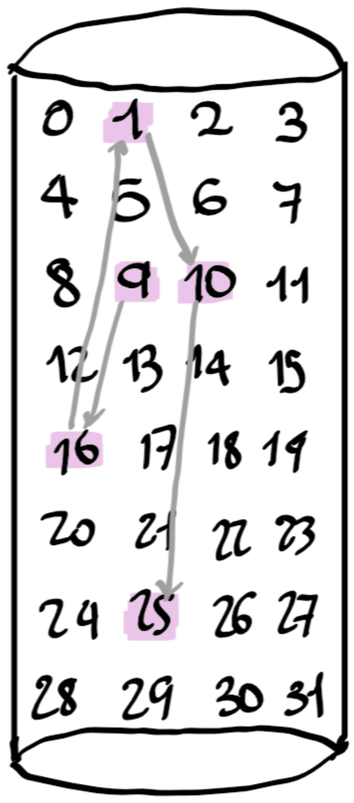
PAGE HEADER MANTIENE INFORMAZIONI QUALI:

ID della pagina del DB, timestamp che indica quando la pagina è stata modificata l'ultima volta, relazione cui le tuple nella pagina appartengono, ecc.

Normalmente un record è contenuto interamente in una pagina, quindi si può avere dello spazio sprecato

ALLOCAZIONE CONTIGUA: i blocchi del file sono allocati in blocchi di disco contigui

FILE	START	LUNGHEZZA
film	0	2
video	14	3
cliente	19	6
noleggio	28	4
f	6	2



RELAZIONE	START	END
noleggio	9	25

ALLOCAZIONE CONCATENATA: ogni blocco di un file contiene un puntatore al successivo blocco del file

ORGANIZZAZIONE DEI RECORD NEI FILE

I file che contengono i record dei dati costituiscono l'organizzazione primaria dei dati.
Il modo con cui i record vengono organizzati nei file incide sull'efficienza delle operazioni e sull'occupazione di memoria

↳ GESTORE DEL BUFFER

PROBLEMA: minimizzare gli accessi a memoria non volatile

mantenere più blocchi possibile in memoria principale in modo da evitare riletture da memoria non volatile

La lettura di un record richiede che la pagina corrispondente sia prima portata in memoria, in un'area gestita dal DBMS detta buffer.
A fronte di una richiesta di una pagina, ad esempio durante un'operazione di lettura (esecuzione di un'interrogazione) il Buffer Manager (BM) opera come segue:

- se la pagina è già nel buffer, viene fornito al programma chiamante l'indirizzo della pagina corrispondente
- se la pagina non è in memoria, il BM verificherà se c'è spazio nel buffer. Se la pagina prescelta dal buffer è occupata allora viene riscritta su disco; a questo punto il BM può leggere la pagina da disco e copiarla nella pagina del buffer selezionata, rimpiazzando così quella prima presente

---

MODELLO DEI DATI

10/03/25

- livello concettuale — umano
- livello logico — DBMS
- livello fisico — file/disco

relazionale

relazione

dominio = insieme non vuoto potenzialmente infinito di valori D

prodotto cartesiano $D_1 \dots D_n \in D$

$$D_1 \times \dots \times D_n = \{(v_1 \dots v_n) \mid v_1 \in D_1 \wedge \dots \wedge v_n \in D_n\}$$

$D_1 = \{0,1,2\}$ $\quad D_2 = \{d,v\}$

$D_1 \times D_2 = \{(0,d),(1,d),(2,d),(0,v),(1,v),(2,v)\}$

0	d
1	d
2	d
0	v
1	v
2	v

Sottoinsieme <u>finito</u> del prodotto cartesiano di n domini

$t:[1,n] \to D_1 \cup \dots \cup D_n$ t.c. $t(i) \in D_i$ $\quad \forall i = 1 \dots n$

$t(1) = 0$
$t(2) = d$ $\quad t[i]$ t.i

String x int x string x string x string x date x string x string x ... x real

R nome di relazione (stringa)

$A_1, \dots, A_n \in \mathcal{A}$ nome di attributo (stringhe) (distinti tra loro)

$dom: \mathcal{A} \to \mathcal{D}$ $\quad dom(A_1) = D_1 \quad \dots \quad dom(A_n) = D_n$

$A_1, \dots, A_n \in \mathcal{A}$ $\quad \mathbb{N} \times D_2$ nomi di attributi (stringhe)

INSEGNAMENTI (Codice, Titolo, Anno, Semestre, CFU)

Codice	Titolo	Anno	Semestre	CFU
25880	Basi di dati	2	2	12

grado = numero dei domini (delle componenti delle tuple)

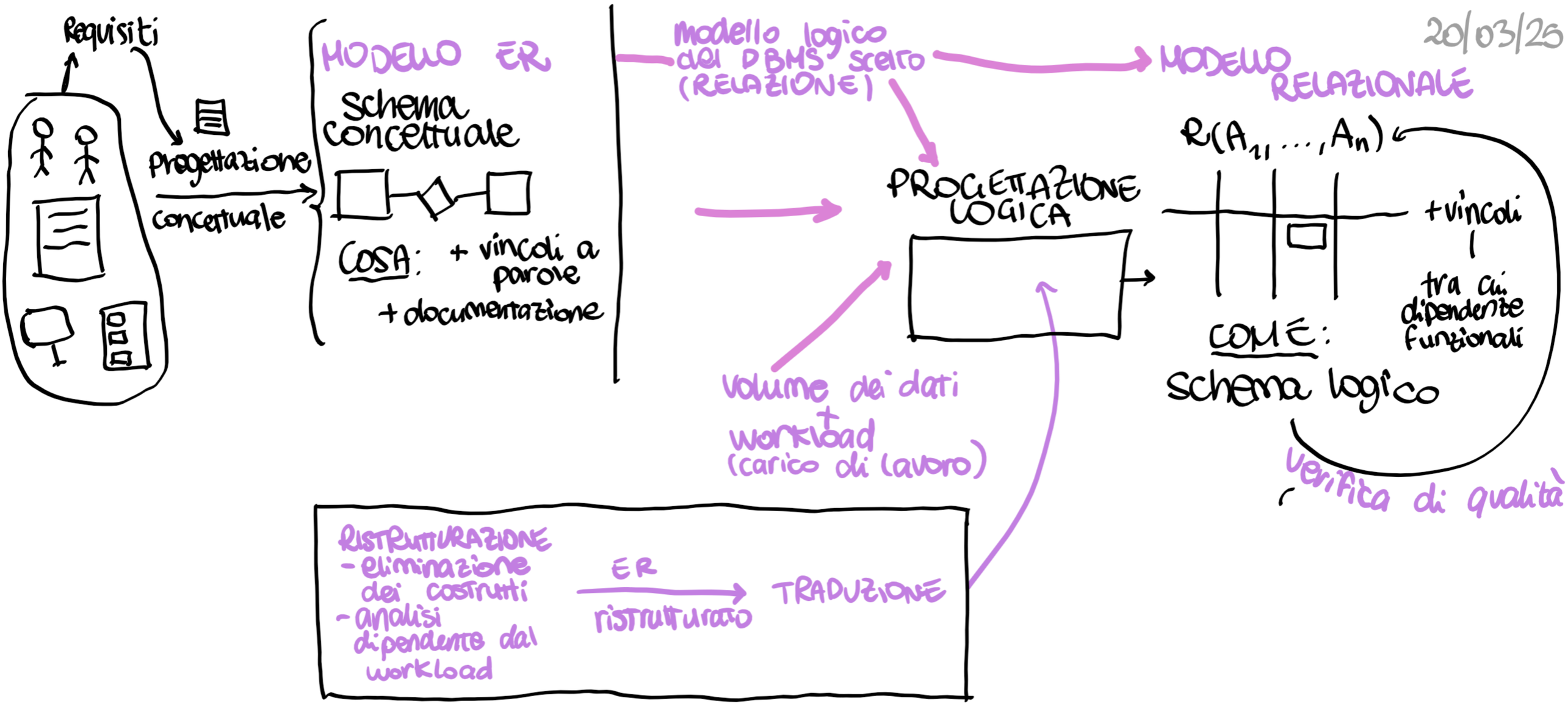
CHIAVI:

una chiave è un insieme minimale di nomi di attributi che caratterizzano le istanze di un'entità

Una relazione può avere più attributi che possono essere chiavi

chiavi candidate → chiave primaria
→ chiavi alternative

20/03/25

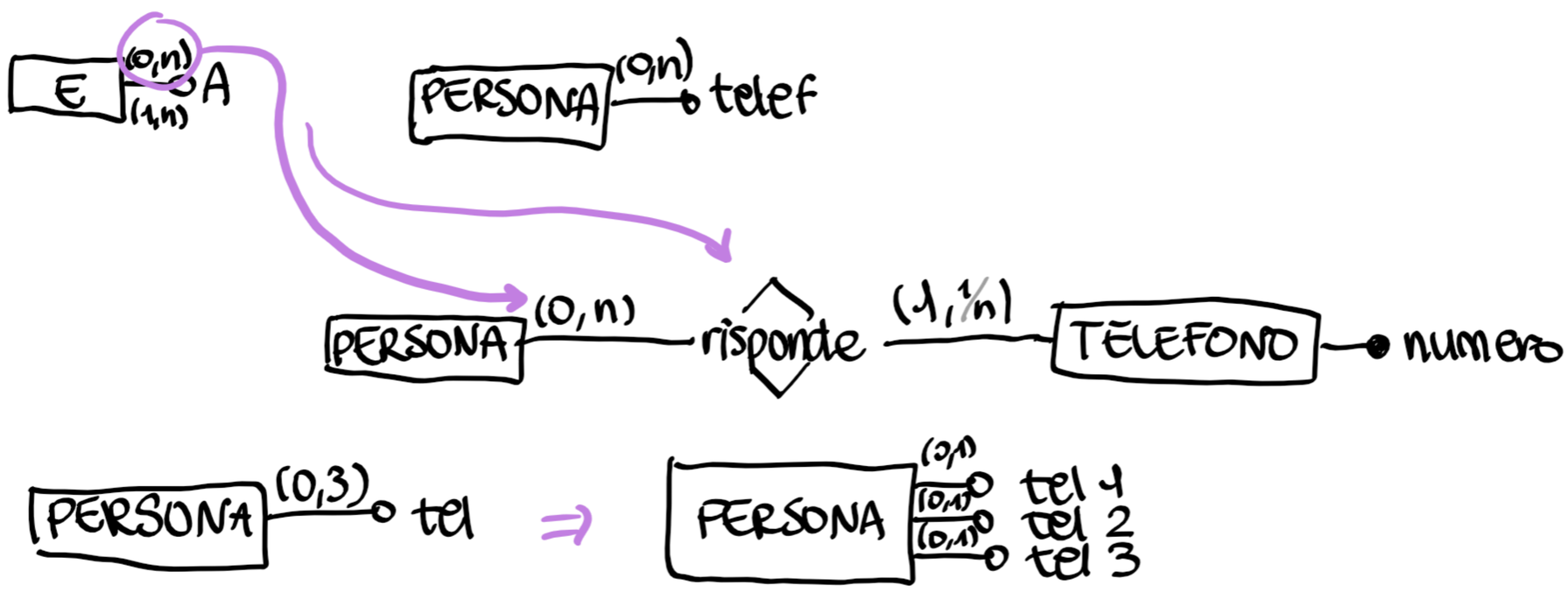


RISTRUTTURAZIONE:

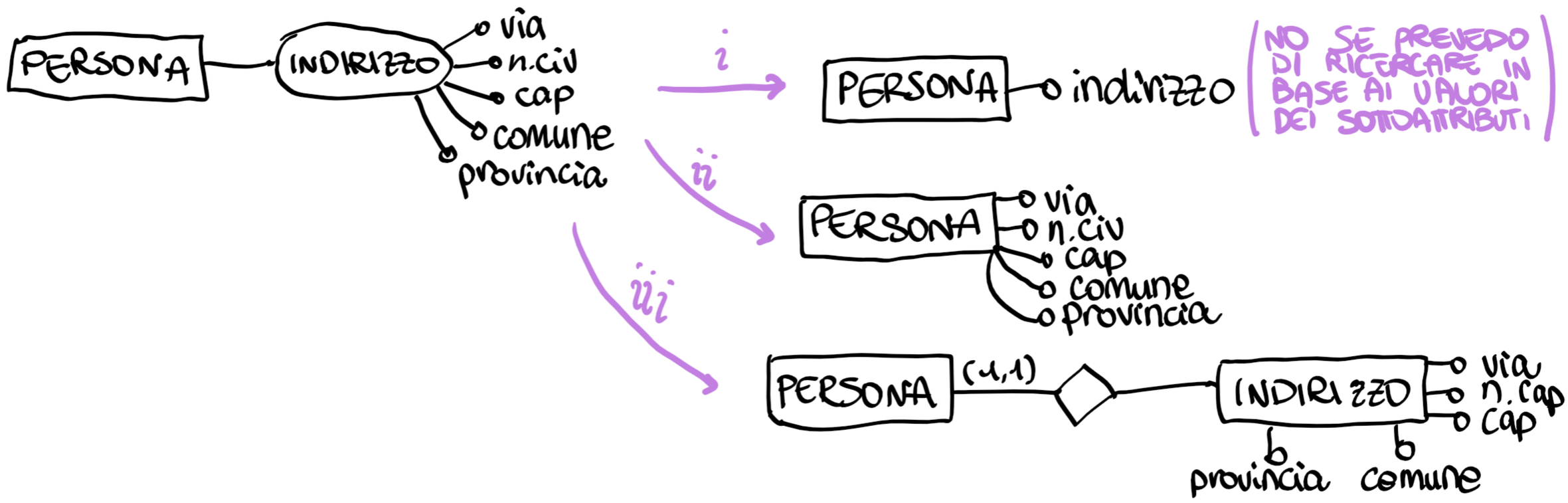
1) eliminazione costrutti:
- attributi multivalore
- attributi composti
- gerarchie di sp

2) analisi di:
- rindondanza
- partizionamenti / arcopendenti

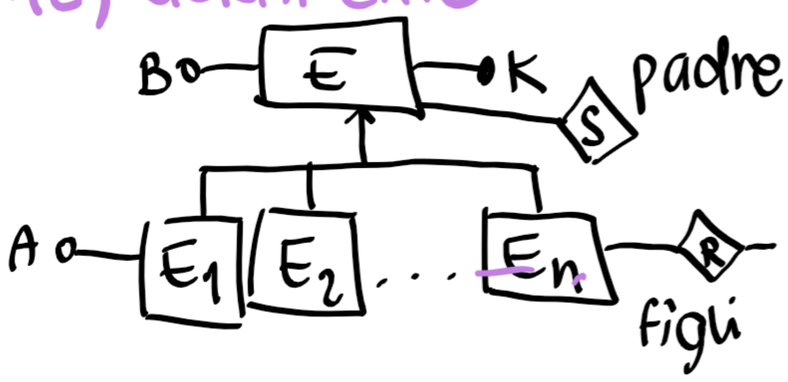
1a) ATTRIBUTI MULTIVALORE



1b) ATTRIBUTI COMPOSTI



1c) GERARCHIE



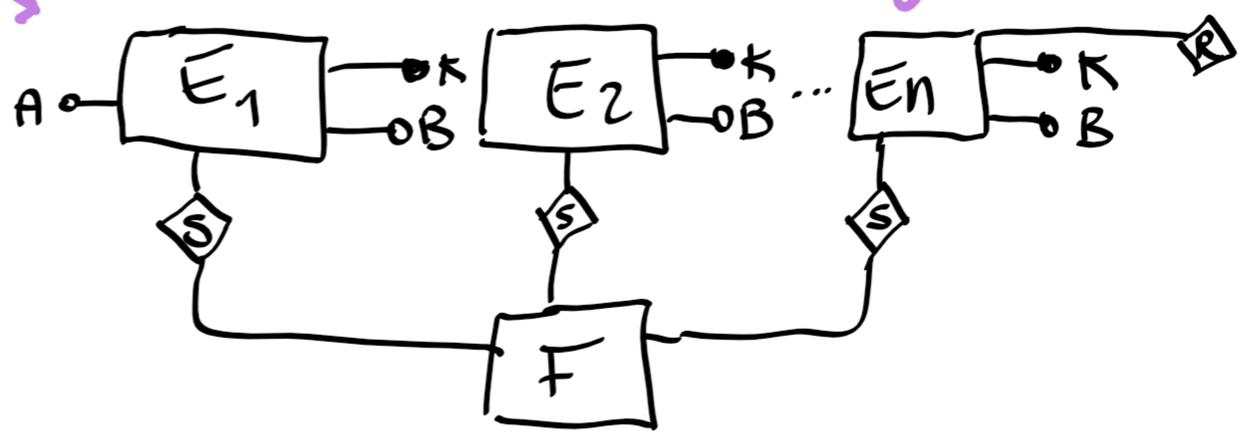
i → eliminare i figli

+ vincoli

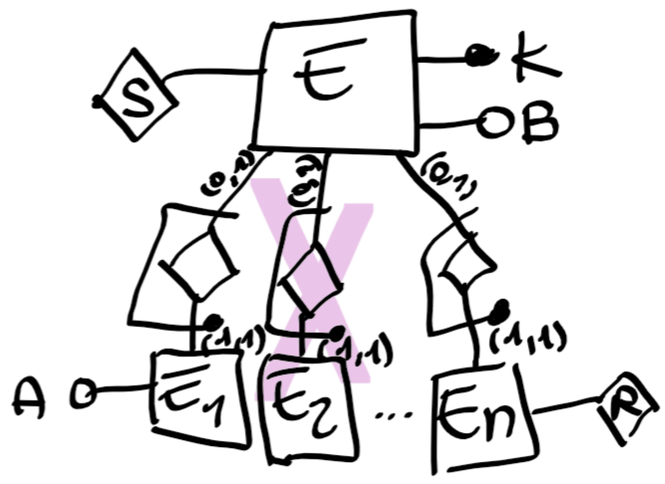
A assume valori se e solo se tipo = E_1

partecipa a associaz. R tutte e sole le istanze di tipo E_1

ii → eliminare padre solo se gerarchia totale

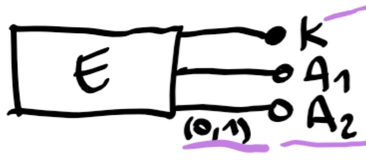


iii → trasformo gerarchia in associazione



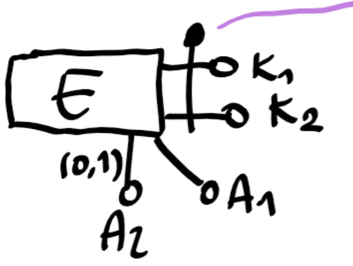
24/03/2025

ER RISTRUTTURATO

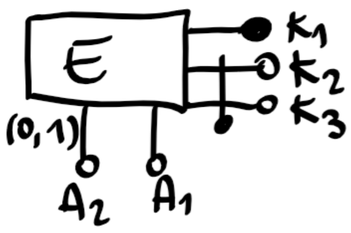
- 

$E(\underline{K}, A_1, A_2)$

(K identificatore interno semplice)

- 

$E(\underline{K_1, K_2}, A_1, A_2)$

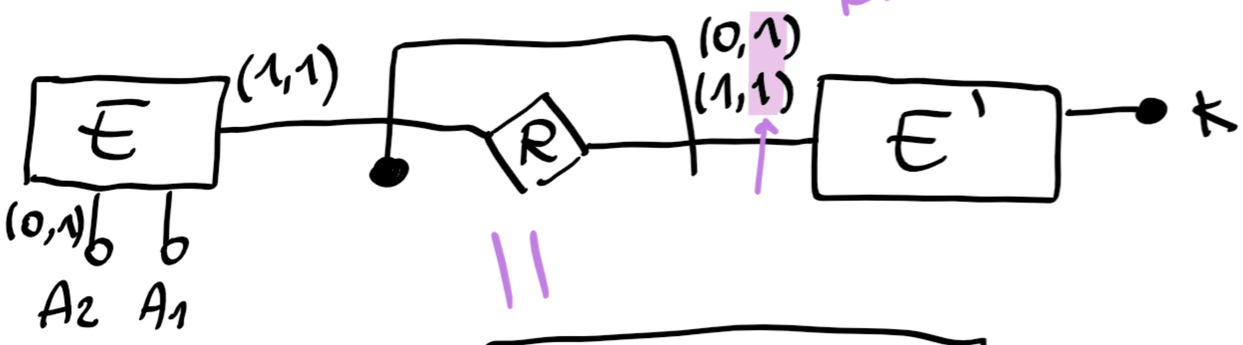
- 

$E(\underline{K_1}, \underline{K_2, K_3}, A_1, A_2)$

chiave primaria

chiavi alternative

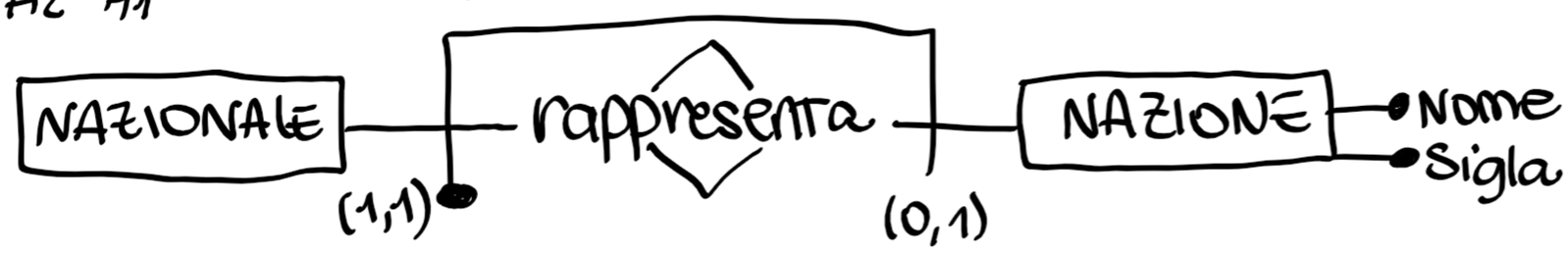
chiavi candidate: K_1, K_2, K_3

- 

$E'(\underline{K}, \ldots)$

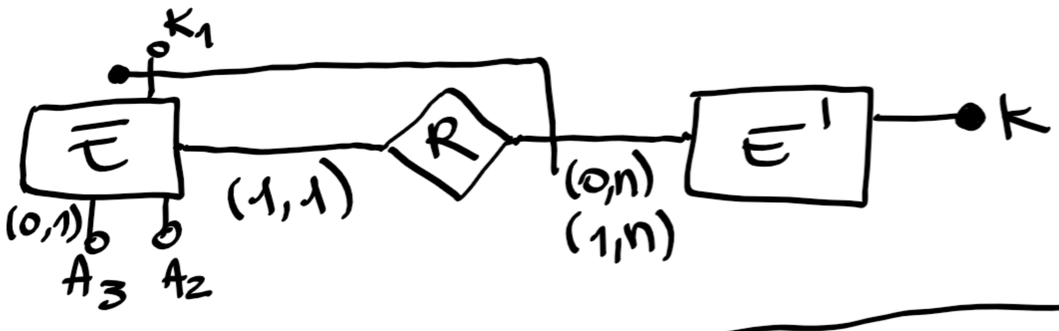
$E(\underline{K^{E'}}, A_1, A_2)$

chiave esterna da E'



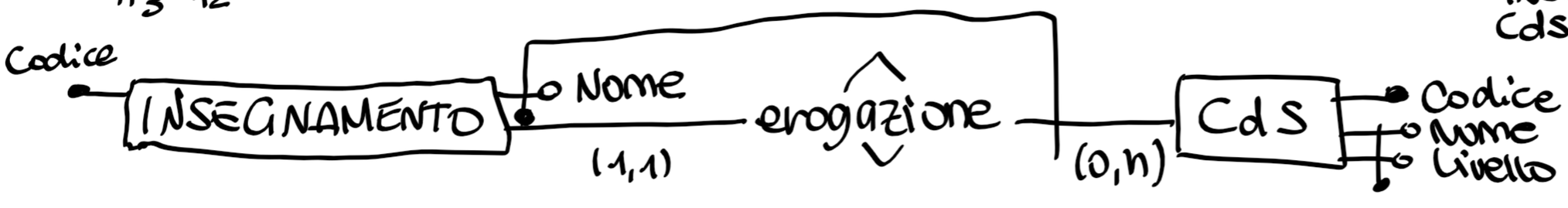
Nazione(Sigla, Nome, ...)

Nazionale(Sigla NAZIONE, ...)

- 

$E'(\underline{K}, \ldots)$

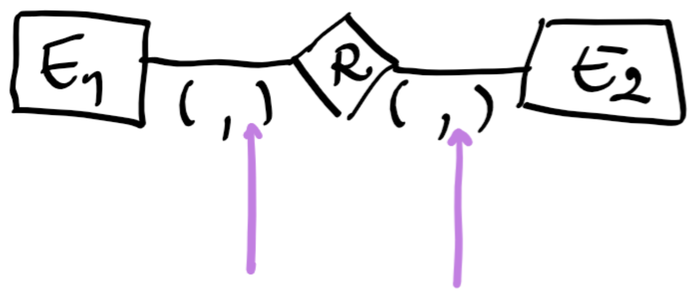
$E(\underline{K^{E'}, K_1}, A_2, A_3)$



INSEGN.(Cod., CodCds CdS, Nome, ...)

CdS(Cod., Nome, Livello, ...)

ASSOCIAZIONI



$E_1(K_1, \dots$
$E_2(K_2, \dots$

R binaria (caso più frequente)

le associazioni si dividono in:

- uno a uno (-,1) (-,1)
- uno a molti (-,1) (-,n)
- molti a molti (-,n) (-,n)

CASO UNO A MOLTI

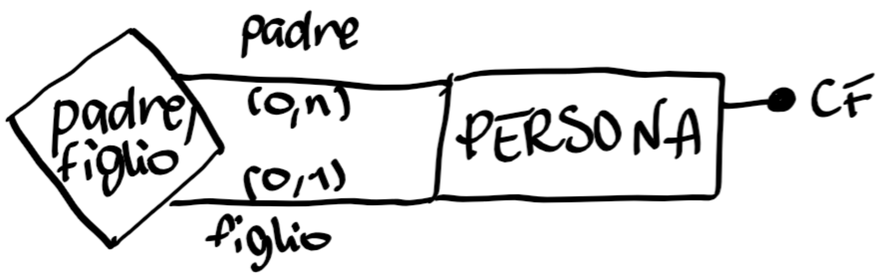
$E_1(\underline{K_1}, \dots,$
$E_2(\underline{K_2}, \dots, FK^{E_1}, A)$

MOLTI A MOLTI

nuova relazione

$E_1(\underline{K_1}$
$E_2(\underline{K_2}$
$R(\underline{K_1}^{E_1}, \underline{K_2}^{E_2}, A)$

esempio:



PERSONA (CF, ..., CF Padre $_0^{PERSONA}$)

PERSONA (xxxyyy72..., ..., zzzwww47)
PERSONA (kkkjjj05..., ..., xxxyyy72)
PERSONA (.... ..03 ..., ..., Xxxyyy72)

31/03/2025

SQL

- DDL create table / drop / alter
- DML insert / delete / update

Definition
Manipulation
Query Language

- Linguaggio dichiarativo non procedurale

$R(A_1, \dots, A_n)$ $\quad U_R = \{A_1, \dots, A_n\}$

R

```
SELECT *
FROM   R
```

* = seleziona tutto

operatori unari

$\pi_A(R)$
$\pi_{A,B,C}(R)$

- proiezione

A
A B C

$\{A,B,C\} \subseteq U_R$

```
SELECT DISTINCT A
FROM R
```

```
SELECT DISTINCT A,B,C
FROM R
```

- selezione $\sigma_F(R)$

esempio: $\sigma_{SEMESTRE=2}$ (INSEGNAMENTI)

Predicati atomici: $A_i \ \theta \ v$ $\quad A_i \ \theta \ A_j$ $\quad A_i, A_j \in U_R$

$\theta \in \{=, \neq, >, <, \geq, \leq\}$ operatori relazionali di confronto

Predicato F è:

- o un predicato atomico
- o $F_1 \wedge F_2$ $F_1 \vee F_2$ $\neg F_1$ con F_1, F_2 predicati

esempio: $\sigma_{(SEMESTRE=2 \vee CFU>6)}(INSEGNAMENTI)$

in SQL si scrive:

```
SELECT X
FROM R
WHERE F
```

- ridenominazione $\rho_{\substack{A_1,\dots,A_n \\ \leftarrow B_1,\dots,B_n}} R$

esempio: $\rho_{\substack{cds \\ \leftarrow CorsodiStudi}}(INSEGNAMENTI)$

$\sigma_{corsodistudio=3759}\left(\rho_{\substack{cds \\ \leftarrow CorsodiStudi}}(INSEGNAMENTI)\right)$

$\rho_{\substack{cds \\ \leftarrow CorsodiStudi}}\left(\sigma_{cds=3759}(INSEGNAMENTI)\right)$

QUALITÀ DI SCHEMI RELAZIONALI

PROBLEMA: come garantire che uno schema sia "di qualità"?

anomalie:

- rindondanza:
  - anomalia di aggiornamento (devo modificare tutte le istanze)

    esempio:

    ```
    UPDATE R
    SET AnnoIscrizione =3
    WHERE Matricola = ...  AND SiglaCorso = BD
    ```

    (sbagliato perche poi sarebbe associata un po' a 2 e un po' a 3)
- anomalia di inserimento

  nell'esempio puoi inserire uno studente solo se ha dato almeno un esame
- anomalie di cancellazione

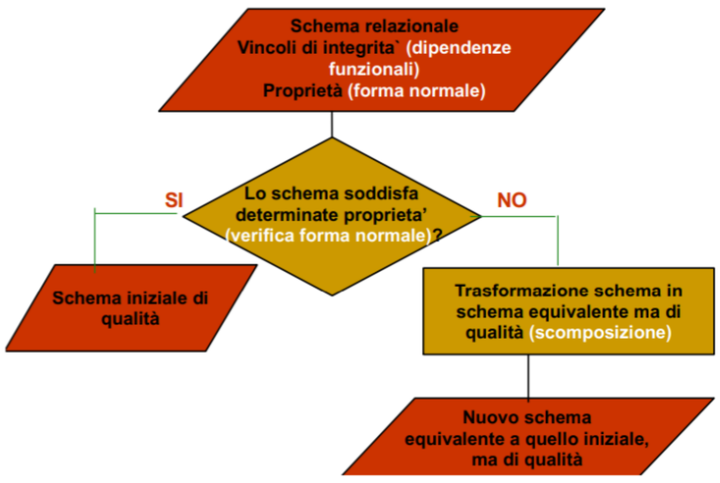
SOLUZIONI:

forma normale

⋮

NORMALIZZAZIONE

Vincolo di integrità: dipendenza funzionale



DIPENDENZE FUNZIONALI:

$R(\overbrace{A_1,\dots,A_n}^{X})$

$Y \subseteq X \quad Z \subseteq Z$

r istanza di R

in r

$Y \longrightarrow Z$ vale in r se $t_1 t_2 \in r$

se $t_1.Y = t_2.Y$ allora $t_1.Z = t_2.Z$

Y determina funzionalmente Z in $R(X)$ se qualunque istanza r di $R(X)$ soddisfa $Y \rightarrow Z$

ESEMPI

MatricolaStudente → NomeStudente
MatricolaStudente → CognomeStudente
MatricolaStudente → Anno
CodiceCorso → SiglaCorso
CodiceCorso → NomeCorso
CodiceCorso → Crediti
MatricolaStudente CodiceCorso → Voto

NomeStudente → CognomeStudente ?

non è una FD: vale solo su questa istanza, altre istanze potrebbero violare il vincolo

Matricola Studente	NomeStudente	CognomeStudente	AnnoIscr	CodiceCorso	SiglaCorso	NomeCorso	Crediti	Voto
4433038	Mario	Rossi	2	25880	BD	Basi Dati	9	30
4532453	Andrea	Verdi	3	25880	BD	Basi Dati	9	27
4532453	Andrea	Verdi	3	27054	BD2	Basi Dati 2	6	30
4446667	Bianca	Neri	2	80299	IP	Intro ...	12	24
4446667	Bianca	Neri	2	25880	BD	Basi Dati	9	27
4446667	Bianca	Neri	2	73026	EML	Elementi ...	12	21
4512345	Celeste	Mori	3	27054	BD2	Basi Dati 2	6	28
4512345	Celeste	Mori	3	25880	BD	Basi Dati	9	30
4455667	Viola	Bianchi	1	80299	IP	Intro ...	12	25
4455667	Viola	Bianchi	1	73026	EML	Elementi ...	12	26

banale: se la parte destra appartiene ad un attributo a sinistra

- MatricolaStudente CodiceCorso → CodiceCorso
- Si tratta però di una FD "banale" (sempre soddisfatta)
- Se A è un singolo attributo, Y → A è non banale se A non appartiene a Y
- Se A è un insieme di attributi, Y → Z è non banale se nessun attributo in Z appartiene a Y

MatricolaStudente → Anno
CodiceCorso → Crediti

Le parti sinistre non corrispondono a chiavi e causano anomalie

- causano anomalie

MatricolaStudente CodiceCorso → Voto

La parte sinistra corrisponde a una chiave e non causa anomalie

- non causa anomalie

Le anomalie sono legate all'eterogeneità dello schema

MatricolaStudente → Anno
CodiceCorso → Crediti
MatricolaStudente CodiceCorso → Voto

- MatricolaStudente CodiceCorso è chiave
- MatricolaStudente da solo no
- CodiceCorso da solo no

- Le anomalie sono causate dalla presenza di concetti eterogenei nella relazione:
  - proprietà degli studenti (l'anno di corso)
  - proprietà di corsi (i crediti)
  - proprietà della chiave (coppia studente-corso, il voto)

FORMA NORMALE DI BOYCE E CODD

$X \to Y$ forma normale se X contiene chiave K di R

$\forall t_1, t_2 \in r$ $t_1.K = t_2.K$ allora $t_1.X = t_2.X$ $K \to X$

<u>chiusura</u> = insieme degli attributi che dipendono funzionalmente da X

$R(A,B,C)$
F $AB \to C$
$C \to B$

C chiave?
Se è chiave $C \to ABC$ e $ABC \subseteq C^+_F$
$C^+_F = CB$
NON CHIAVE