Basi di dati

(aiuto!!!)

Matteo Balzerani

October 2025

Contents

1 Descrizione corso

La prof di basi di dati è disponibile a fare un preappello l'ultimo mercoledì di lezione di gennaio (quindi nella prima metà di gennaio secondo le sue previsioni)

1.1 Modalità d'esame

L'esame è scritto e si compone di più parti:
Domande di teoria (3 domande)

- Sono considerate bloccanti.
- Significa che, se non si raggiunge la sufficienza in questa sezione, l'intero esame non viene superato.
- La docente corregge comunque la prova, ma se le risposte teoriche non sono sufficienti, l'esame non è valido.
- Durante il corso verranno indicati chiaramente i possibili argomenti di domanda, e prima della fine del corso si farà anche una simulazione d'esame.

Esercizio di progettazione concettuale

- Viene fornita la descrizione di un contesto in linguaggio naturale, da trasformare nel modello concettuale.
- Questo esercizio ha un peso rilevante (circa 15 punti su 33).
- Durante il corso saranno svolti molti esercizi in aula, per imparare a ragionare in questa prospettiva.

Esercizio di progettazione logica

- Si tratta della traduzione del modello concettuale in uno schema logico relazionale.

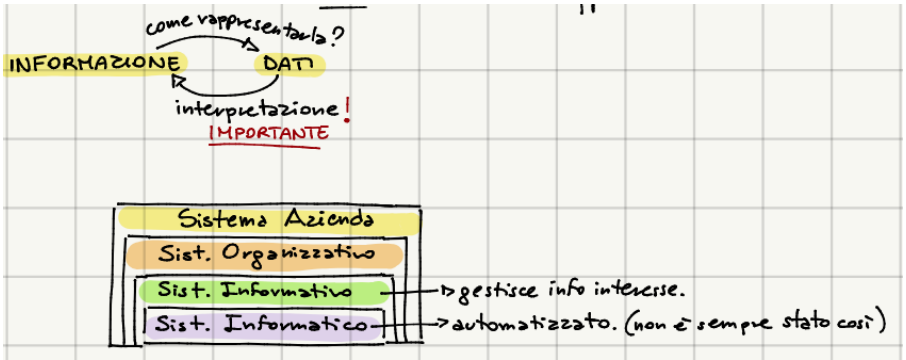
Esercizi di interrogazione

- Riguardano sia l'algebra relazionale sia il linguaggio SQL.
- Anche questi esercizi verranno affrontati numerose volte durante il corso, così che gli studenti arrivino preparati.

2 Introduzione

2.1 Possibili domande di teoria:

- **Cos'è un sistema informatico? Un dato? Informazione?** Un *sistema informativo* è la componente di un'organizzazione che **gestisce** (acquisisce, elabora, conserva e produce) le **informazioni di interesse**, cioè le informazioni utilizzate per il perseguimento degli scopi dell'organizzazione. **Il contesto descrive le informazioni di interesse.** Informazione $\neq$ dato. $\rightarrow$ Le **informazioni** vengono rappresentate attraverso i dati, mentre un **dato** è il modo in cui rappresento l'informazione. **Gestire delle informazioni** vuol



dire raccolta, acquisizione, archiviazione, conservazione, elaborazione, scambio, sincronizzazione, etc...

- **Che cos'è una base di dati?** È un insieme **organizzato** di dati utilizzati per il **supporto allo svolgimento** delle attività di un ente ed è gestito da un Sistema di Gestione di Basi di Dati o **DataBase Management System (DBMS)**. Una base di dati può essere vista in due modi: In senso generico: è un insieme organizzato di dati, utilizzato a supporto delle attività di un'organizzazione. *(Esempio: per un'università, la base di dati riguarda le carriere degli studenti, i voti, le iscrizioni; per un ospedale, i pazienti, le diagnosi, le terapie.)* ; In senso tecnologico: la base di dati è il sistema che permette di gestire quei dati, cioè il software e l'infrastruttura che ne consentono utilizzo, interrogazione e conservazione.

- **Cos'è un DBMS?** È un sistema informatico che gestisce collezioni di dati **di grandi dimensioni (numerosi e complessi), persistenti (memorizzati in maniera durevole), condivisi (accessibili da più utenti con diversi livelli di visibilità e autorizzazione)** garantendo **privatezza (accesso regolato ai dati), affidabilità (sicuri, integri e consistenti in qualsiasi condizione), efficienza (utilizzo di tempo e memoria ottimali), efficacia (deve farmi risparmiare tempo).** Si basa sul concetto di **transazione**, che deve essere corretta, ed atomica, ovvero che a fronte di un malfunzionamento ci assicura che i dati non vengano persi. Le transazioni, inoltre, sono **concorrenti**, cioè l'effetto deve essere coerente (equivalente) per ciascuna transazione che avvenga in contemporanea. Una transazione deve essere con *effetti definitivi.* (per i posteri, guardare i criteri ACID)

- **Sistema organizzativo e informativo**: ogni organizzazione ha un **Sistema Organizzativo** che è l'insieme di **risorse** e regole per lo svolgimento coordinato delle attività (processi) al fine del perseguimento degli scopi. Le risorse di un'azienda sono persone, denaro, materiali, informazioni. Ogni organizzazione ha un **Sistema Informativo**, eventualmente non esplicitato nella struttura. Quasi sempre, il sistema informativo è di supporto ad altri sottosistemi, e va quindi studuiato nel contesto in cui è inserito. Il sistema informativo è di solito suddiviso in sottosistemi (in modo gerarchico o decentrato), più o meno fortemente integrati. Il sistema informativo è parte del sistema organizzativo, ed esegue e gestisce i processi informativi (cioè che coinvolgono le informazioni). **Il concetto centrale che va ricordato è che il sistema informativo è la parte dell'organizzazione che gestisce le informazioni di interesse.** Quando però questa gestione delle informazioni avviene in modo automatizzato, cioè tramite un elaboratore, allora si parla di **sistema informatico.** Il concetto di sistema informativo è indipendente da qualsiasi automatizzazione; esistono organizzazioni la cui ragion d'essere è la gestione di informazioni e che operano da secoli. Il termine "*automatizzato*" significa infatti che non è più l'uomo, manualmente, a raccogliere, scrivere e organizzare i dati, ma che tale compito è affidato a un sistema tecnologico, un calcolatore. Oggi, praticamente sempre, quando si parla di sistema informativo, si intende di fatto un sistema informatico: la gestione manuale è stata sostituita quasi del tutto da quella automatizzata. In passato però non era così. Un esempio importante è l'anagrafe. L'anagrafe ha sempre avuto il compito di gestire informazioni fondamentali come nascite, decessi, residenze, matrimoni, ecc. In origine non esistevano sistemi informatici, e tutto questo veniva registrato a mano in grandi registri cartacei. Chiunque abbia avuto modo di vederli, sa che si trattava di veri e propri mega-libri, compilati a penna, nei quali si scrivevano i dati con dovizia di particolari. Quella era già gestione dell'informazione, solo che avveniva senza sistemi informatici. Questo esempio mostra chiaramente che un sistema informativo può esistere anche senza strumenti tecnologici; semplicemente, oggi, tali strumenti sono diventati indispensabili e quindi i termini "sistema informativo" e "sistema informatico" tendono ad essere usati quasi come sinonimi. **Sistema informatico:** è la porzione automatizzata del sistema informativo. È la parte del sistema informativo che gestisce informazioni con tecnologia informatica.

2.2 Modellazione dei dati

Una **base di dati** è un insieme organizzato di dati; per organizzare i dati serve un **modello di dati**. Un **modello dei dati** è un insieme di concetti utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che essa risulti comprensibile ad un elaboratore.

Modelli logici: organizzano i dati in modo strutturato ma ancora astratto. Esempio: il modello relazionale, che rappresenta i dati con tabelle e relazioni. **Modelli concettuali:** ancora più astratti, descrivono i concetti fondamentali di un dominio indipendentemente dal modello logico. *Esempio: modello entità-relazione (E-R), che rappresenta concetti come Studente, Corso e la relazione frequenta.* Serve per dare una visione ad alto livello, più intuitiva, del dominio di interesse.

2.2.1 Schema, Istanza e Architettura delle basi di dati

Schema:

- Lo schema descrive la struttura della base di dati.
- È sostanzialmente invariante nel tempo: quando progetto uno schema cerco che sia definitivo. Non è "scolpito nella pietra" (può cambiare), ma la buona progettazione serve proprio a ridurre al minimo le modifiche. Perché? Perché se cambio lo schema, devo poi aggiornare tutte le istanze e adattare tutte le query collegate. *Esempio: se aggiungo una colonna "secondo nome" alla tabella degli studenti, devo aggiornare ogni riga della tabella. Con basi di dati grandi, ciò diventa costoso.*
- In sintesi: lo schema è la descrizione formale della struttura dei dati (es. attributi delle tabelle, tipi di dato, relazioni tra entità).

Istanza: L'istanza è l'insieme dei valori attuali contenuti nella base di dati. Sono i dati veri e propri, che possono cambiare nel tempo. Esempio: aggiungere uno studente, modificare un voto, cancellare un record. Le istanze cambiano quotidianamente, mentre lo schema dovrebbe rimanere stabile.
Schema = struttura dei dati. Istanza = valori memorizzati.
Il concetto di schema e istanza non vale solo per il modello logico (relazionale), ma anche per il livello concettuale. In un diagramma E-R (Entità-Relazioni) lo **schema** è costituito dalle **entità** (rettangoli) e dalle **relazioni** (rombi) che collegano le entità. L'**istanza** è costituita dalle **occorrenze** delle entità e dalle **coppie** (o tuple) che formano le **relazioni**.

Studente	Matricola	Cognome	Nome	DataNascita
	VR001122	Rossi	Lorenzo	16/04/1998
	VR123456	Verdi	Camilla	23/11/1998
	VR098765	Bianchi	Giacomo	21/07/1998

- Lo schema è:

Studente	Matricola	Cognome	Nome	DataNascita

- L'istanza è:

	VR001122	Rossi	Lorenzo	16/04/1998
	VR123456	Verdi	Camilla	23/11/1998
	VR098765	Bianchi	Giacomo	21/07/1998

Esempi:

- Entità Studente → ciascuno studente reale è un'istanza.
- Entità Docente → ciascun professore è un'istanza.
- Relazione Relatore di tesi → è rappresentata dalle coppie (Docente, Studente) che concretamente svolgono quella relazione.

2.2.2 Architettura semplificata di un DBMS

- **Utente: interroga la base di dati o inserisce nuovi dati.** Esempio: uno studente si iscrive a un appello → inserimento di dati. Il docente controlla la lista degli iscritti → interrogazione della base di dati.
- **Livello logico (schema logico):** L'utente interagisce con le tabelle (modello relazionale). Scrive query SQL che operano sui dati organizzati in tabelle. Non si preoccupa di come i dati siano memorizzati fisicamente.

- **Livello fisico (schema interno):** Descrive come i dati sono realmente memorizzati nei dispositivi (file, puntatori, strutture di archiviazione). Questo livello è gestito dal DBMS, non dall'utente.
- **Indipendenza dei dati:** il livello logico è indipendente da quello fisico: una tabella è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica. L'utente e il progettista interagiscono solo col modello logico (tabelle, attributi, tuple). Non devono preoccuparsi di come i dati siano fisicamente memorizzati. È il DBMS che si occupa di gestire la memorizzazione fisica. Per questo motivo, nello studio ci si concentra soprattutto sul modello relazionale e sulle query SQL: è il livello con cui realmente si interagisce.

2.2.3 Utente, sistema informativo e modello relazionale

- **Interazione dell'utente con il DBMS:** L'utente che utilizza un DBMS (Database Management System) non interagisce con il modello concettuale in modo diretto. Può usarlo come documentazione per capire quali sono i concetti importanti. Quando interagisce, lo fa direttamente a livello logico, cioè lavora con le tabelle, le relazioni, i dati organizzati secondo lo schema logico. Il modello concettuale serve al progettista per rappresentare i concetti rilevanti del dominio, ma l'utente finale non deve preoccuparsene.
- **Sistema informativo e dati**: Un sistema informativo esprime l'informazione di interesse. Differenza tra dato e informazione: Dato → valore grezzo, singolo elemento. Informazione → interpretazione dei dati in un contesto applicativo, utile per prendere decisioni o rispondere a domande concrete. Contesto applicativo: determina quali dati sono rilevanti. *Esempio: memorizzare i dati degli studenti. Il numero di scarpe non è rilevante, a meno che non serva per un contesto particolare.*
- **Modello di dati:** La base di dati è un insieme organizzato di dati, gestito da un DBMS. L'organizzazione dei dati è rappresentata tramite modelli di dati, che forniscono costrutti concettuali per strutturare le informazioni.Tipi di modelli di dati: **Concettuali** → rappresentano concetti importanti (E-R). **Logici** → rappresentano la struttura reale delle tabelle e delle relazioni (modello relazionale).
- **Schema e istanza Schema** → organizzazione dei dati secondo il modello scelto.
  - ◇ Relazionale: struttura delle tabelle, attributi, tipi di dati.
  - ◇ Concettuale: entità e relazioni.

  Istanza → dati effettivi memorizzati.
  - ◇ Relazionale: tuple della tabella.
  - ◇ Concettuale: occorrenze delle entità e delle relazioni.

  L'utente interagisce con lo schema logico, non con quello fisico, garantendo indipendenza dei dati.
- **Modello relazionale di dati**: Proposto negli anni '60 per favorire l'indipendenza dei dati: gli utenti non devono preoccuparsi di file o puntatori. Pubblicato negli anni '70 e diventato operativo negli anni '80 (1981) È il modello più utilizzato nella pratica (quasi il 99% dei DB reali). Si basa su due concetti principali:
  1. Relazione matematica → insieme di tuple su uno o più domini.
  2. Tabella → rappresentazione concreta delle relazioni, con righe e colonne.

2.2.4 Concetto di relazione

La parola relazione ha tre accezioni:

1. Matematica → sottoinsieme del prodotto cartesiano di domini.
2. E-R (Entità-Relazione) → associazione tra entità (es. relatore di tesi e studente).
3. Modello relazionale dei dati → tabella organizzata secondo domini e tuple.

È importante capire il contesto per sapere a quale significato ci si riferisce.

- **Relazione matematica:** una relazione matematica è *per definizione* il **prodotto cartesiano dei domini degli insieme in relazione.** *Ad esempio:*

  Dominio $D_1 = \{1, 2\}, D_2 = \{x, y, z\}$

  Prodotto Cartesiano: $D_1 \times D_2 = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}$

  Una relazione $R \subseteq D1 \times D2 \rightarrow R = \{(1, x), (2, x), (2, z)\}$

  L'ordine dei valori nel prodotto cartesiano è posizionale.

- **Relazione nel modello relazionale (Codd):** Codd introduce una variante della relazione matematica per rendere la relazione non posizionale: Nelle tabelle, **i valori non dipendono dalla posizione della colonna.** Ogni colonna ha un ruolo distinguibile tramite il nome, non tramite l'ordine. Questo semplifica le interrogazioni e la gestione dei dati: non occorre ricordare quale colonna rappresenta quale informazione.
  Esempio tabella "partite":
  - Colonne: squadra in casa, squadra ospite, goal squadra in casa, goal squadra ospite
  - Ogni dominio ha un ruolo distinto, identificabile dalla colonna, non dalla posizione.

Relazione nel modello relazionale: relazione matematica con una variante che mira a rendere la notazione NON posizionale.
Esempio: Partite $\subseteq$ string $\times$ string $\times$ int $\times$ int $\leftarrow$ A ciascun dominio viene associato un nome (**attributo**) che è unico nella relazione e descrive il "ruolo" (il significato) del dominio.

Squadra casa	Squadra fuori	Reti casa	Reti fuori
Milan	Inter	2	0
Roma	Lazio	0	0
Inter	Roma	2	0
Lazion	Milan	0	0

Table 1: Esempio non posizionale di relazione

Il significato della colonna è descritto dal nome della colonna stessa, quindi l'ordine della riga non è più rilevante nell'informazione $\rightarrow$ **Relazione NON posizionale.**
Tabelle e relazioni (nel modello relazionale dei dati): ¡Domanda di Esame¿ quando una tabella rappresenta una relazione nel modello relazionale dei dati? *(per i posteri, 1 Forma Normale)*

- Una tabella rappresenta una relazione *se* i valori di ogni colonna sono fra loro **omogenei**, perché appartengono allo stesso dominio.
- le righe della tabella sono tutte diverse tra loro, perché la relazione è un insieme, quindi ogni elemento(riga) è diverso tra loro.
- Le intestazioni di ogni colonna sono diverse perché se dò nomi uguali si torna ad una notazione posizionale, mentre io ne voglio una NON posizionale.

Se una tabella rappresenta una relazione, abbiamo che l'ordinamento tra le righe è **irrilevante**, perché *essendo un insieme, non importa l'ordine.* Anche l'ordinamentro tra le colonne è **irrilevante**, siccome vogliamo una notazione non posizionale, quindi quando ogni colonna ha il suo nome posso mescolarle e non cambia nulla.
RICORDA: *la relazione è un insieme.*
Nel modello relazionale ogni riga si chiama ***tupla***, ed ogni valore nella tupla lo recupero indicando il valore dell'attributo. Ad esempio, nella tabella 1, se t è la prima tupla, allora *t[squadraCasa]* = "Milan".
In una relazione un **collegamento è basato sui valori**, infatti, ad esempio, posso collegare la prima squadra della tabella 1 con la tabella

NomeSquadra	Anno	Città
Milan	1899	Milano
...	...	...

Table 2: Squadre

2.3 DEFINIZIONI relative al MODELLO RELAZIONALE DEI DATI

- **Grado:** numero di colonne di una relazione;
- **Cardinalità:** numero di tuple dell'istanza di una relazione.
- **Schema di relazione:** (*Schema:* struttura, organizzazione dei dati.) Descrive la STRUTTURA e si indica $R(X)$ dove R è il nome della relazione, X è l'insieme degli attributi, quindi $X = A_1, A_2, ..., A_n$; Ad ogni attributo è associato un dominio.

  Esempio: $Studente(Matricola, Cognome, Nome, DataNascita)$
- **Schema di base di dati:** un'insieme di schemi di relazione con nomi diversi.

$$\boldsymbol{R} = \{R_1(X_1), R_2(X_2), ..., R_n(X_n)\}$$

Esempio:

$$\begin{aligned}\boldsymbol{R} = \{&Studente(Matricola, Cognome, Nome, Datanascita),\\ &Docente(Codice, Cognome, Nome, NumTel),\\ &Supervisore(MatrStCodDoc, DataInizio)\}\end{aligned}$$

- **Tupla:** una tupla su in insieme di attributi X è una *funzione* che associa a ciascuno degli attributi A in X un valore del dominio di A. *Esempio:* $t[A]$ denota il valore della tupla t sull'attributo a
- **(ISTANZA di) RELAZIONE:** Quando parliamo di relazione spesso parliamo di istanza di relazione. È data dal contenuto della relazione, cioè è formata dall'*insieme delle tuple della relazione.*
- **Istanza di una relazione su uno schema** $R(X)$: insieme r di tuple su X. Esempio: dato lo schema di relazione Studente(Matricola, Cognome, Nome, DataNascita), l'istanza potrebbe essere

$$\begin{aligned}r = \{&(VR000001, Rossi, Mario, 12/04/2005),\\ &(VR000002, Gialli, Lucia, 27/0/2004),\\ &(VR000003, Verdi, Luca, 21/08/2006)\}\end{aligned}$$

- **(ISTANZA di) BASE di DATI:** istanza di basi di dati su uno schema

$$\boldsymbol{R} = \{R_1(X_1), R_2(X_2), ..., R_n(X_n)\}$$

è data dall'insieme di (istanze di) relazioni $r = \{r_1, r_2, ..., r_n\ \}$

$$\begin{aligned}\boldsymbol{R} = \{&Studente(Matricola, Cognome, Nome, Datanascita),\\ &Docente(Codice, Cognome, Nome, NumTel),\\ &Supervisore(MatrStCodDoc, DataInizio)\}\end{aligned}$$

quindi

$$\begin{aligned}r = \{\quad &\{(VR000001, Rossi, Mario, 12/12/2005), \quad (VR00002, Gialli, Lucia, 16/01/2005)\},\\ &\{(DO1, Oliboni, Barbara, 045...), \quad (DO2, Quintarelli, Elisa, 045...)\},\\ &\{VR00001, DO1, 01/10/2025), \quad (VR005, DO2, 20/09/2025)\} \quad \}\end{aligned}$$

Avremmo quindi le seguenti tabelle:

Matricola	Cognome	Nome	DataNascita
VR00001	Rossi	Mario	12/12/2005
VR00002	Gialli	Lucia	16/01/2005

Table 3: Studente

Codice	Cognome	Nome	NumTel
DO1	Oliboni	Barbara	045 ...
DO2	Quintarelli	Elisa	045...

Table 4: Docente

MatrSt	CodDoc	DataInizio
VR00001	DO1	01/10/2025
VR00002	DO2	20/09/2025

Table 5: Supervisore

Rappresentante
VR000002

Table 6: Rappresentante

Le informazioni delle tabelle sono collegate, e le relazioni possono essere anche con un solo attributo e si basano sui valori.

Lo schema della Base di Dati impone una struttura rigida definita a priore. Tutti i dati inseriti nella base di dati infatti devono rispettare quello schema, una volta deciso. Questo significa che le tuple che posso inserire nella mia base di dati devono rispettare la struttura che ho definito.

Per alcune tuple, potrei non avere il valore di qualche attributo. → **Informazione incompleta**
Ad esempio:

Matricola	Cognome	PrimoNome	SecondoNome	DataNascita
VR00001	Rossi	Mario	Francesco	12/12/2005
VR00002	Gialli	Lucia	***NULL***	16/01/2005

Table 7: Studente

Se non so un valore non metto testo "Placeholder" perché una tupla potrebbe aver bisogno di usare proprio quel valore, quindi non è consigliato cercare una stringa. La soluzione è scegliere un **VALORE NULLO:** denota l'assenza di un valore nel dominio, NON È UN VALORE DEL DOMINIO. Quindi in una colonna o abbiamo un valore del dominio, o valore nullo.
Tipi di valori nulli:

- **Valore sconosciuto:** il valore esiste, ma a noi non è noto.
- **Valore inesistente:** il valore non esiste nella realtà modellata.
- **Valore senza informazione:** il valore può esistere o non esistere e se esiste non sappiamo quale sia. (Caso più generale). *Nei DBMS un valore NULL è interpretato come questo caso, ovvero l'* ***"OR"*** *logico dei due precedenti casi.*

Esempio:

Codice	Nominativo	Telefono	DataNascita
NULL	A.A	045 ...	12/06/2000
PO2	B.B	NULL	15/05/2005
PO3	C.C	045...	NULL

Table 8: Paziente

Codice	Nominativo	Specializzazione
MO1	NULL	NULL
MO2	E.E	Ortopedia
MO3	F.F	Chirurgia
NULL	E.E	NULL

Table 9: Medico

CodP	CodM	Motivo
PO2	MO2	NULL
NULL	MO1	Artrite
PO3	NULL	NULL

Table 10: In cura

Osservazioni relazione "paziente"

Se non ha un codice la prima tupla in paziente, non potrò mai connetterlo con la base di dati, quindi è un problema; non posso lasciare NULL il codice. Il numero di telefono, invece, non crea problemi; tantomeno la data di nascita; non creano problemi nello schema attuale della base di dati e delle sue istanze se assume valore NULL.

Osservazioni relazione "medico"

Nel caso della prima tupla, la coppia null nominativo-specializzazione è un problema, perché **ci sono così tanti valori nulli che la tupla non esprime alcuna informazione!** NEL CASO DELLA SECONDA E QUARTA TUPLA IL NULL CREA PROBLEMI PERCHÉ POTREBBE ESSERE LA STESSA TUPLA.

Si possono/devono imporre restrizioni sulla presenza di valori nulli. Nella definizione dello schema della relazione è possibile/auspicabile specificare su quali attributi non sono (o sono) ammessi valori nulli. Ad esempio, se decido che posso NON avere il numero telefono, potrebbe essere

$$PAZIENTE(Codice, Nominativo, Telefono^*, DataNascita)$$

Se su un attributo metto un asterisco, significa che sono ammessi valori nulli, altrimenti NON sono ammessi. Secondo quest'ultimo schema, nella tabella paziente l'unica tupla ammessa è la seconda.

3 Modello relazionale

3.1 Vincoli di integrità:

Vincoli di integrità: descrivono le proprietà che devono essere soddisfatti dai dati (cioé le sue istanze) all'interno della nostra base di dati per rappresentare **informazioni corrette**. Ad esempio scrivere

$$STUDENTE(Matricola, Cognome, Nome, DataN, NumTel, Indirizzo*)$$

significa affermare che nell'istanza di uno studente *solo l'indirizzo* può essere non inserito, cioè NULL, mentre tutti gli altri devono avere un valore per essere corretti. Un **VINCOLO** è una funzione booleana che associa ad ogni istanza il valore *vero o falso.* Quando paarlo di integrità, parlo di **qualità dei dati**, cioè di rappresentare in maniera **corretta e completa** i dati memorizzati. Un altro valore di integrità indica quale può essere il valore assunto da un determinato attributo. *Ad esempio, un voto di un esame può andare da 18 a 30.* Quindi il vincolo di integrità può **specificare anche il dominio.**

Tipi di vincoli di integrità

Posso classificarli sulla base degli elementi coinvolti, e abbiamo due categorie:

- **Vincoli INTRA-relazionali:** cioè sono vincoli riferiti ad una singola tabella alla volta, cioè su *singole relazioni.* Tra questi abbiamo:

  1. **Vincolo di TUPLA:** la valutazione avviene su ciascuna tupla indipendentemente dalle altre. Specifico una proprietà che vado a verificare su ogni singola tupla. La validazione della tupla e l'accettazione di questa quindi dipende singolarmente dai valori contenuti in essa.
     Ad esempio: $voto \geq 18 \land voto \leq 30 \land EsameSuperato = SI$.
  2. **Vincolo su VALORI (di DOMINIO):** vincolo riferito ad un singolo valore.
     Esempio: $Settimana \geq 1 \land Settimana \leq 40$.
  3. **Vincolo di CHIAVE:** permette di specificare qual è l'insieme di attributi che identificano in maniera univoca ogni singola tupla. La chiave primaria è l'insieme *minimo* di attributi, che non possono assumere valore NULL. **La chiave identifica UNIVOCAMENTE ogni tupla.**
     Esempio: $STUDENTE(\underline{matricola}, cognome, nome, ...)$

  Esempio: $CORSO(\underline{codice}, titolo, descrizione*, CFU)$ dove $CFU \geq 2 \land CFU \leq 12$.

Codice	**Titolo**	**Descrizione**	**CFU**
BD1	Basi di Dati	...	6
ALGO	Algoritmi	*Descrizione*	12

Table 11: Tabella Corso: esempio vincoli intrarelazionali

- **Vincoli INTER-relazionali:** fanno riferimento a più relazioni. **Vincoli di integrità referenziale.**
  Esempio sulla tabella indicata sopra, non esiste l'esame BD2, quindi non posso referenziarlo.

Matricola	**Codice Corso (FK)**	**Voto**	**Data**
VR01	BD1	30	02/09/2025
VR02	~~BD2~~	26	03/09/2025

Table 12: Tabella Esame: esempio vincoli interrelazionali

Possibile domanda d'esame
Si dica che cosa sono i vincoli di integrità specificando brevemente le due categorie.

3.2 Chiavi

Vincolo di chiave: intuitivamente una **CHIAVE** è uno o un insieme di attributi utilizzato per **identificare univocamente le tuple di una relazione.**

Esempio con paziente dell'ospedale:

Codice	**Cognome**	**Nome**	**CittaNascita**	**DataNascita**
PAZ011	Verdi	Giulia	Verona	02/03/1990
PAZ012	Neri	Giulia	Milano	18/12/1989
PAZ013	Verdi	Greta	Verona	02/03/1990

Table 13: Tabella Paziente: esempio di vincoli di chiave.

Il **CODICE** identifica univocamente i pazienti in questo esempio. Anche i dati anagrafici { Cognome, Nome, DataNascita } identificano i pazienti.

DEFINIZIONE di CHIAVE: la chiave di una relazione è un *insieme di attributi* che serve ad identificare **UNIVOCAMENTE** le tuple della relazione e che garantisce le seguenti proprietà:

1. **UNICITÀ:** in una qualunque istanza della relazione non possono esistere due tuple distinte, la cui restrizione alla chiave sia uguale, cioè che abbiano gli stessi valori sugli attributi della chiave.

$$t_1[k] = t_2[k] \Rightarrow t_1[x] = t_2[x] \tag{1}$$

2. **MINIMALITÀ:** non deve essere possibile sottrarre alla chiave un attributo senza che la condizione di unicità cessi di valere. Cioè, *una chiave non contiene un'altra chiave.*

CHIAVE E SUPERCHIAVE:

- Un insieme K di attributi è **SUPERCHIAVE** per r se r NON contiene due tuple distinte t_1 e t_2 con $t_1[k] = t_2[k]$.
- Un insieme K di attributi è **CHIAVE** per r se è una **SUPERCHIAVE MINIMALE** per r (cioè non contiene un'altra superchiave).

Esempio con paziente dell'ospedale:

Codice	**Cognome**	**Nome**	**CittaNascita**	**DataNascita**
PAZ011	Verdi	Giulia	Verona	02/03/1990
PAZ012	Neri	Giulia	Milano	18/12/1989
PAZ013	Verdi	Greta	Verona	02/03/1990

Table 14: Tabella Paziente: esempio di chiave e superchiave.

- In questa tabella {*Codice*} è una chiave e una superchiave, e contiene un solo attributo, cioè è un attributo minimale, quindi è chiave.
- Mentre {*Codice, DataNascita*} non è una chiave perché è superchiave, ma non è minimale.
- È una superchiave anche {*Cognome, Nome, DataNascita*} , ma non è minimale, perché posso togliere, ad esempio, la data di nascita, ed ho ancora valori univoci; non è minimale e quindi *non è chiave.*
- {*Cognome, Nome*} è superchiave e minimale, quindi è chiave.

VERO O FALSO riassuntivo:

- **Ogni relazione ha almeno una chiave**, **VERO**, poiché una relazione è un insieme, quindi male che vada ho tutti gli attributi come chiave.
- Ogni relazione ha *esattamente* una chiave? **FALSO**.
- Ogni attributo della appartiene al massimo ad una chiave. FALSO. Una chiave può essere sottoinsieme di un'altra. **FALSO**.
- Può esistere una chiave che coinvolge tutti gli attributi. **VERO.**

CHIAVE PRIMARIA: chiave su cui *non sono ammessi valori nulli.*

Si indica sottolineando l'insieme di attributi.

$$STUDENTE(\underline{Matricola}, Cognome, Nome, DataNascita)$$
$$PRODOTTO(\underline{Nome, Linea}, Descrizione, Prezzo)$$
$$ESAME(\underline{MatricolaStudente, CodiceCorso}, Voto, Data)$$

Quando come chiave primaria ho una coppia di attributi, è la coppia di valori che non si può ripetere, mentre i singoli sì. Cioé, nell'esempio sopra, nel caso dell'esame si può ripetere la matricola poiché si verbalizzano più esami, ma lo stesso esame è verbalizzato solo una volta per lo stesso studente. Cioè, data la la chiave primaria ($\underline{MatricolaStudente, CodiceCorso}$), le istanze $\{(ST01, BD1), (ST01, ALGO), (ST02, BD1)\}$ sono valide, mentre non lo sarebbe ripetere $\{(ST01, BD1), (ST01, BD1)\}$.

3.3 Vincoli di integrità referenziale

Lo scopo è quello di rappresentare informazioni **coerenti.** Vengono usati in particolare i valori delle chiavi primarie.
Esempio:

$$PAZIENTE(\underline{Codice}, Cognome, Nome, DataNascita, CodMedico)$$
$$MEDICO(\underline{Codice}, COgnomeNome, Ospedale, Reparto)$$
$$REPARTO(\underline{Ospedale, Reparto}, Indirizzo, Telefono, COD\_Primario)$$

Codice	**Cognome**	**Nome**	**DataNascita**	**CodiceMedico**
PAZ01	Verdi	Giulia	12/12/2004	MED03
PAZ02	Neri	Giulia	10/12/2005	MED02
PAZ03	Verdi	Greta	12/12/2004	MED03

Table 15: Paziente

Codice	**Cognome**	**Nome**	**Ospedale**	**Reparto**
MED01	Rossi	Mario	BR	Ostetricia
MED02	Bronchi	Lucia	BT	Ostetricia
MED03	Neri	Sara	BT	Ostetricia
MED04	Gialli	Andrea	Negrar	Ostetricia

Table 16: Medico

Ospedale	**Reparto**	**Indirizzo**	**Telefono**	**COD_Primario**
BR	Ostetricia	...	...	MED01
BT	Ostetricia	...	...	MED03
Negrar	Ostetricia	...	...	MED04

Table 17: Reparto

I vincoli di integrità referenziale in questo esempio sono:
TABELLE INTERNE → TABELLE ESTERNE

- $PAZIENTE.CodMedico \rightarrow MEDICO.Codice$
- $MEDICO.[Ospedale, Reparto] \rightarrow REPARTO.[Ospedale, Reparto]$
- $REPARTO.COD\_Primario \rightarrow MEDICO.Codice$

VINCOLO DI INTEGRITÀ REFERENZIALE (O DI FOREIGN KEY):
Un vincolo di integrità referenziale fra gli attributi X, di una relazione R_1 e un'altra relazione R_2 impone ai valori su X in R_1 di comparire come valori della chiave primaria di R_2.

- $R_1 \rightarrow$ **TABELLA INTERNA**, in cui specifico il vincolo di integrità referenziale.
- $R_2 \rightarrow$ **TABELLA ESTERNA**

N.B.1 nei vincoli su più attributi conta l'ordine.
N.B.2 *Il valore dell'attributo Foreign Key può anche assumere valore NULL (se concesso dallo schema) qualora non vi sia ancora un collegamento con un altro valore. Deve rispettare i vincoli di integrità, infatti, solo se non è NULL.*

A livello di SQL, si possono scegliere politiche di reazione alla violazione dei vincoli di integrità referenziale. Ad esempio, qualora una tupla esterna viene referenziata, bisogna aggiornare le tuple interne che prima la referenziavano. Le modifiche che violano l'integrità referenziali possono essere sia della tabella interna, che di quella esterna.

Esercizio 1: *definire un modello relazionale dei dati per organizzare i dati di una clinica privata in cui lavorano degli infermieri. Ogni infermiere è associato ad un reparto ed ha un codice fiscale, nome, cognome data di nascita. Per i reparti si memorizzino il codice (identificativo) il nome, il piano, il caporeparto (che è un infermiere) e il primario che è un medico. Di un medico si conoscono il codice (identificato), cognome, nome e specializzazione.*

$$\begin{aligned}
\textbf{INFERMIERE}(\quad & \underline{CF_Infermiere}, Nome, Cognome, DataNascita, \\
& CodiceReparto(FK) \rightarrow REPARTO.CodiceReparto \quad) \\
\textbf{REPARTO}(\quad & \underline{CodiceReparto}, Nome, Piano, Caporeparto(FK \rightarrow Infermiere.CF_Infermiere), \\
& CodicePrimario(FK \rightarrow Medico.CodiceMedico) \quad) \\
& \textbf{MEDICO}(\underline{CodiceMedico}, Cognome, Nome, Specializzazione)
\end{aligned}$$

Esercizio 2: *Ogni infermiere è associato ad un reparto ed ha CF, nome cognome e data di nascita.*
Per i reparti si memorizzano il codice, il nome, il piano e il minimo di posti letto.
Per ogni letto si memorizza il reparto in cui si trova, il numero della stanza, il numero di letto e il suo tipo.
Per ogni medico si memorizzano il reparto in cui lavora e le info personali tra cui codice, nome, cognome , la specializzazione, indirizzo e il numero di telefono (fisso e mobile). Il lavoro del reparto è organizzato in turni che hanno una data e un orario di inizio.
Ogni reparto ha un medico come primario ed un infermiere come caporeparto (il caporeparto cambia ad ogni turno).

$$\begin{aligned}
INFERMIERE(&\underline{CF}, Nome, Cognome, DataNascita, CodReparto \rightarrow REPARTO.CodReparto) \\
REPARTO(&\underline{CodReparto}, Nome, Piano, NumeroLetti, CodPrimario \rightarrow MEDICO.CodMedico) \\
MEDICO(&\underline{CodMedico}, Nome, Cognome, Specializzazione, Indirizzo, TelFisso, TelMob, \\
&CodReparto \rightarrow REPARTO.CodReparto) \\
LETTO(&\underline{CodReparto \rightarrow REPARTO.CodReparto, NumStanza, NumLetto}, Tipo) \\
TURNO(&\underline{Data, Ora, CodReparto \rightarrow REPARTO.CodReparto,}, CodInfermiere \rightarrow INFERMIERE.CF)
\end{aligned}$$

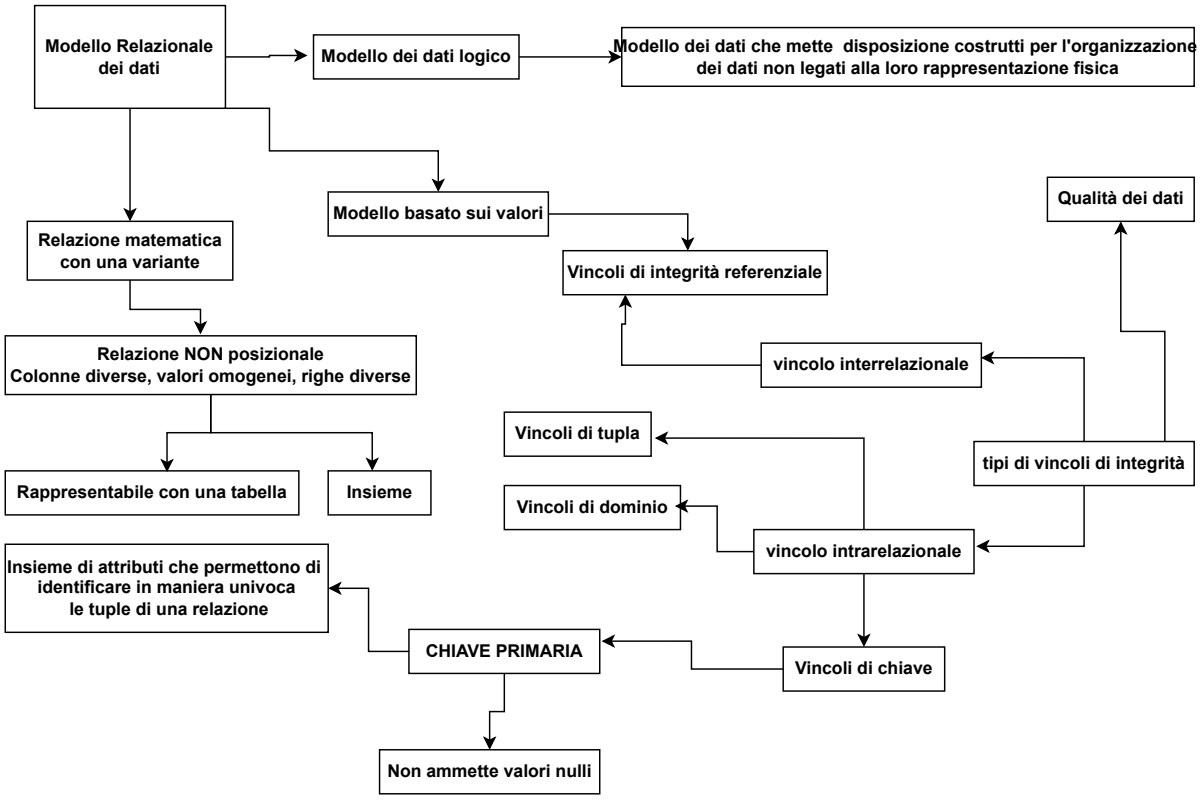


Figure 1: Schema riassuntivo del modello relazionale

4 Algebra relazionale

Classificazione dei linguaggi:

- **Linguaggi FORMALI:** Come l'algebra relazionale.
- **Linguaggi COMMERCIALI:** Come l'SQL (Structured Query Language).

Inoltre i linguaggi possono essere:

- **DICHIARATIVI:** specificano le proprietà del risultato, cioè "che cosa" voglio ottenere. SQL è un linguaggio dichiarativo.
- **PROCEDURALI:** specificano le modalità di generazione del risultato, cioè "come" ottenerlo, quali sono i passaggi. L'algebra relazionale è un linguaggio procedurale.

L'Algebra Relazionale quindi è un linguaggio *formale e procedurale*, composto da un insieme di **operatori**, che producono **relazioni** e che possono essere composti.

4.1 Operatori dell'algebra relazionale

- **OPERATORI INSIEMISTICI:**

  Le relazioni sono insiemi, si possono applicare solo a relazioni definite sugli stessi attributi. R_1 e R_2 devono essere **COMPATIBILI** quindi devono avere lo **stesso grado** e **domini ordinatamente dello stesso tipo.**

  - **UNIONE** $R_1 \cup R_2$. Mette insieme tutte le tuple.
    L'unione di due relazioni r_1 ed r_2 (istanze) definite sullo stesso insieme di attributi X, $r_1 \cup r_2$ è una relazione ancora su X contenente tutte le tuple che appartengono ad r_1, oppure ad r_2, oppure ad entrambe.
    Esempio:

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Table 18: Laureato

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Table 19: Quadro

Matricola	Nome	Età
9297	Neri	33
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Table 20: Laureato ∪ Quadro

  - **Intersezione** $R_1 \cap R_2$ mette insieme le tuple comuni.
    L'intersezione di due relazioni r_1 ed r_2 (istanze) definite sullo stesso insieme di attributi X, $r_1 \cap r_2$ è una relazione ancora su X contenente tutte le tuple che appartengono sia ad r_1 che ad r_2.

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Table 21: Laureato

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Table 22: Quadro

Matricola	Nome	Età
7432	Neri	54
9824	Verdi	45

Table 23: Laureato ∩ Quadro

  - **Differenza** $R_1 - R_2$ Prendo le tuple di R_1 che non sono in comune con R_2.
    La differenza di due relazioni r_1 ed r_2 (istanze) definite sullo stesso insieme di attributi X, $r_1 - r_2$ è una relazione ancora su X contenente tutte le tuple che appartengono ad r_1, ma che non appartengono ad r_2.

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Table 24: Laureato

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

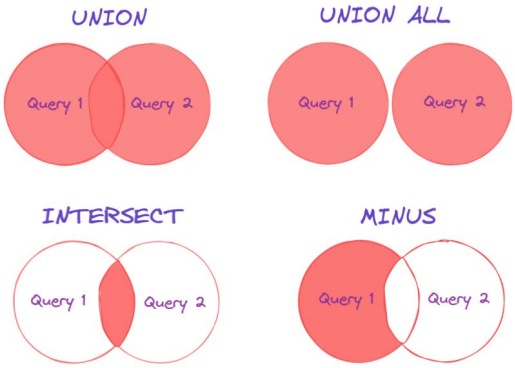
Table 25: Quadro

Matricola	Nome	Età
7274	Rossi	42

Table 26: Laureato − Quadro

Matricola	Nome	Età
9297	Neri	33

Table 27: Quadro − Laureato



- **OPERATORI DELL'ALGEBRA RELAZIONALE:**
  - **Ridenominazione ρ**
  - **Selezione σ**
  - **Proiezione Π**
- **OPERATORE FONDAMENTALE:** JOIN
  - JOIN naturale
  - Prodotto cartesiano $\times$
  - Theta-Join

Padre	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco

Table 28: Paternità

Madre	Figlio
Eva	Abele
Eva	Caino
Sara	Isacco

Table 29: Maternità

Genitore	Figlio
Adamo	Abele
Adamo	Caino
Abramo	Isacco
Eva	Abele
Eva	Caino
Sara	Isacco

Table 30: Unione dopo la ridenominazione

Non posso fare Paternità $\cup$ Maternità perché non sono definite sullo stesso insieme di attributi. Per risolvere il problema uso l'operatore di **RIDENOMINAZIONE**.

$$\rho_{Genitore \leftarrow Padre}(PATERNITA)$$
$$\rho_{Genitore \leftarrow Madre}(MATERNITA)$$

dove ρ è l'operatore e nelle parentesi ci va la relazione.

Quindi ora ho $PATERNITA(Genitore, Figlio)$ e $MATERNITA(Genitore, Figlio)$ e quindi

$$\rho_{Genitore \leftarrow Padre}(PATERNITA) \cup \rho_{Genitore \leftarrow Madre}(MATERNITA)$$

4.1.1 Ridenominazione ρ

- Operatore unario;
- "Modifica lo schema" lasciando inalterata l'istanza;
- Permette di cambiare il nome agli attributi, per rendere compatibili domini diversi.

Definizione

1. *Sia r una relazione definita sull'insieme di attributi X e sia Y un altro insieme di attributi con la stessa cardinalità.* Ad esempio:

$$X = \{Padre, Figlio\}$$
$$Y = \{Genitore, Figlio\}$$

2. Siano $A_1, A_2, ...A_k$ e $B_1, B_2, ..., B_k$ rispettivamente un ordinamento per gli attributi in X e un ordinamento per quelli in Y.
3. Allora la **ridenominazione** contiene una tupla t' per ogni tupla t in r. $\rho_{B_1,B_2,...,B_k} \leftarrow_{A_1,A_2,...,A_k} (r)$

Ad esempio, data la tabella $STUDENTE(Matricola, Conome, Nome, Ind, Tel)$, se volessi cambiare solo il numero di telefono scrivo $\rho_{TelPer} \leftarrow_{Tel} (STUDENTE)$.
Se volessi cambiare tutti gli attributi posso fare $\rho_{\overline{x}'} \leftarrow \overline{x}(STUDENTE)$.

4.1.2 Selezione σ

- Operatore unario;
- Produce un risultato che ha lo stesso schema dell'operando;
- contiene un sottoinsieme delle tuple;
- genera una *decomposizione orizzontale*, perché la relazione avrà ancora gli stessi attributi, ma con solo le tuple vere.
- il risultato di una selezione è una relazione con lo stesso schema, quindi lo **stesso grado**, **cardinalità** $\leq$ di quella di partenza.

Definizione Data una relazione $r(x)$, una formula proposizionale F su x

$$\sigma_F(r) \rightarrow risultato* \tag{2}$$

dove σ è l'operatore di selezione, F è una formula su x e r è la relazione su cui applico la selezione.
$\rightarrow risultato*$: il risultato contiene tutte le tuple di r per cui F è vera. Quindi F descrive la *condizione di selezione.*

Esempio:

$$STUDENTE(\underline{Matricola}, Cognome, Nome, DataNascita, CittaResidenza, Tel)$$

Matricola	Cognome	Nome	DataNascita	CittaResidenza	Tel
VR01	Rossi	Giulia	12/12/2005	Verona	347...
VR02	Gialli	Luca	01/01/2006	Milano	343...
VR04	Gialli	Mario	03/03/2004	Verona	329...

Table 31: STUDENTE

Se ora io applico l'operazione sotto descritta sulla tabella soprastante

$$\sigma_{CittaResidenza="Verona"}(STUDENTE)$$

Matricola	Cognome	Nome	DataNascita	CittaResidenza	Tel
VR01	Rossi	Giulia	12/12/2005	Verona	347...
VR04	Gialli	Mario	03/03/2004	Verona	329...

Table 32: STUDENTE dopo l'operazione $\sigma_{CittaResidenza="Verona"}(STUDENTE)$

Otterrò il seguente risultato
Generalmente il risultato di una selezione è un sottoinseme della relazione su cui viene applicata. Potrebbe essere anche un insieme vuoto o lo stesso insieme, poiché la condizione di selezione potrebbe applicarsi a tutte le tuple presenti in precedenza.

Una condizione di selezione può prevedere:

- **Confronto tra attributo e costante** $A_1\Theta$ *costante* dove Θ è un operatore $\{=, >, <, ...\}$
  Esempio: $\sigma_{CittaResidenza="Verona"}(STUDENTE)$
- **Confronto tra due attributi** $A_1\Theta A_2$
  Esempio: $\sigma_{Cognome=CittaResidenza}(STUDENTE)$
- **Combinazione complessa:** ottenuta combinando condizioni semplici con i connettivi logici.
  - ◇ **AND** $COND_1 \wedge COND_2$
  - ◇ **OR** $COND_1 \vee COND_2$
  - ◇ **NOT** $\neg COND_1$

  Esempio: $\sigma_{Cognome="Gialli" \wedge CittaResidenza="Verona"}(STUDENTE)$

Esempio:

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5993	Neri	Milano	64
9553	Milano	Milano	44
5698	Neri	Napoli	64

Table 33: IMPIEGATO

Effettuare le seguenti query:

1. **Trovare gli impiegati che guadagnano più di 50:**
   $\sigma_{Stipendio>50}(IMPIEGATO)$
   Risultato:

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5993	Neri	Milano	64
5698	Neri	Napoli	64

2. **Trovare gli impiegati che guadagnano più di 50 e lavorano a Milano:**
   $\sigma_{Stipendio>50 \wedge Filiale="Milano"}(IMPIEGATO)$
   Risultato:

Matricola	Cognome	Filiale	Stipendio
5993	Neri	Milano	64

3. **Trovare gli impiegati che hanno lo stesso nome della filiale in cui lavorano:**
   $\sigma_{Cognome=Filiale}(IMPIEGATO)$
   Risultato:

Matricola	Cognome	Filiale	Stipendio
9553	Milano	Milano	44

4.1.3 Proiezione π

- Operatore unario (lavora su una singola relazione)
- Produce un risultato che a differenza della selezione **non ha lo stesso schema dell'operando.**
- Il risultato ha lo schema definito su un sottoinsieme degli attributi dello schema dell'operando.
- Contiene **TUTTE le tuple** eliminando i duplicati, effettua una **decomposizione verticale.**
- Tuple uguali collassano in un'unica tupla.
- **Grado** $\leq$ del grado della relazione di partenza.
- **Cardinalità** $\leq$ della cardinalità della relazione di partenza.

Si indica con

$$\Pi_{listaAttributi}(OPERANDO)$$

dove Π è l'operatore e al pedice ho la lista degli attributi che voglio come risultato e "OPERANDO" è la tabella a cui applico la proiezione.

Ad esempio, facendo $\Pi_{Matricola,Cognome}(IMPIEGATO)$ otterrò:

Matricola	**Cognome**
7309	Rossi
5993	Neri
9553	Milano
5698	Neri

Se io scrivessi $\Pi_{Filiale}(IMPIEGATO)$ otterrei:

Filiale
Roma
Milano
Napoli

non ripetendo però la tupla duplicata singleton contenente il valore "Milano".

Data la proiezione $\Pi_Y(r)$ il risultato contiene lo stesso numero di tuple di r se e solo se Y è una **SUPERCHIAVE** di r. *In generale, quando si effettua una proiezione su una relazione, la cardinalità è certo che rimanga uguale, cioè rimane lo stesso numero di tuple, se e solo se nella lista degli attributi dell'operatore vi è contenuto anche l'attributo (o l'insieme di attributi) chiave*

Possibile domanda d'esame: Data una relazione $R_1(\underline{A}, B, C, D)$ e la relazione $R_2(\underline{D}, \underline{E}, F)$ dove $|R_1| = N_1$ e $|R_2| = N_2$.

- Se effettuo $\Pi_A(R_1) \rightarrow |\Pi_A(R_1)| = N_1$ cioè stessa cardinalità.
- Se effettuo $\Pi_B(R_1) \rightarrow |\Pi_B(R_1)| \leq N_1$
- Se effettuo $\Pi_{D,E}(R_2) \rightarrow |\Pi_{D,E}(R_2)| = N_2$
- Se effettuo $\Pi_D(R_2) \rightarrow |\Pi_{D,E}(R_2)| \leq N_2$
- Se effettuo $\sigma_{B="Verona"}(R_1) \rightarrow |\sigma_{B="Verona"}(R_1)| \leq N_1$

Ad esempio, se volessi trovare matricola e cognome degli impiegati che guadagnano più di 50 scriverò

$$\Pi_{Matricola,Cognome}(\sigma_{Stipendio>50}(IMPIEGATO))$$

Otterrò come risultato

Matricola	**Cognome**
7309	Rossi
5993	Neri
5698	Neri

ESERCIZIO:
Data la tabella IMPIEGATO seguente:

Matricola	**Nome**	**Eta**	**Stipendio**
7309	Rossi	34	45
5998	Bianchi	37	38
9553	Neri	42	35
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

Table 34: IMPIEGATO

Effetture le seguenti query:

1. **Trovare la matricola, nome, età e stipendio degli impiegati che guadagnano più di 40:**
   $\sigma_{Stipendio>40}(IMPIEGATO)$

   Risultato:

Matricola	**Nome**	**Eta**	**Stipendio**
7309	Rossi	34	45
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

   Non ho messo la proiezione perché gli attributi che voglio ottenere sono gli stessi della tabella di partenza.

2. **Trovare matricola, nome ed età degli impiegati che guadagnano più di 40**:
   $\Pi_{Matricola,Nome,Eta}(\sigma_{Stipendio>40}(IMPIEGATO))$

   Risultato:

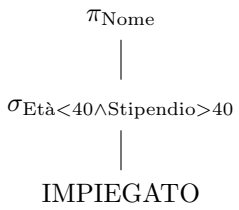
Matricola	**Nome**	**Eta**
7309	Rossi	34
5698	Bruni	43
4076	Mori	45
8123	Lupi	46

3. **Trovare nome degli impiegati con un'età minore di 40 che guadagnano più di 40:**
   $\Pi_{Nome}(\sigma_{Eta<40 \wedge Stipendio>40}(IMPIEGATO))$

   Risultato:

Nome
Rossi

Si può indicare anche così:



4.1.4 JOIN

Operatore dell'algebra relazionale che permette di correlare i dati contenuti in relazioni diverse, confrontando i valori. Ne abbiamo due versioni: join naturale e theta-join.

- **Join naturale:** $R_1 \bowtie R_2$
  - ◇ Operatore **binario** (generalizzante)
  - ◇ Correla dati in relazioni diverse sulla base di valori uguali in attributi con lo stesso nome.
  - ◇ Produce un risultato sull'unione degli operandi con tuple costruite ciascuna a partire da una tupla di ciascuno degli operandi combinando le tuple con valori uguali su attributi comuni.
  - ◇ Il **grado** della relazione ottenuta come risultato del join è minore o uguale (caso del prodotto cartesiano) della somma dei gradi dei due operandi.

Esempio:

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Table 35: R1

Reparto	Capo
A	Mori
B	Bruni

Table 36: R2

Impiegato	Reparto	Capo
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni

Table 37: $R_1 \bowtie R_2$

$$|R_1| = 3; \quad |R_2| = 2; \quad |R_1 \bowtie R_2| = 3.$$

Dimensione del risultato di un join:

1. **Join completo:** ciascuna tupla dei due operandi contribuisce ad *almeno una* tupla del risultato. Quindi, date le due tabelle di partenza, ogni tupla delle due tabelle di partenza finisce nel risultato. Ad esempio, le tabelle sopra presenti con il loro join rappresentano un *join completo.* La cardinalità del join completo sarà $\geq$ del massimo della cardinalità delle due relazioni:

$$|R_1 \bowtie R_2| \geq max(|R_1|, |R_2|)$$

2. **Join non completo:** alcune tuple (definite *dangling*)degli operandi NON contribuiscono al risultato perché non trovano corrispondenza di valore negli attributi comuni nell'altra relazione.
Esempio:

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Table 38: R1

Reparto	Capo
B	Mori
C	Bruni

Table 39: R2

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Table 40: $R_1 \bowtie R_2$

Non trovo quindi la tupla $\{Rossi, A\}$ e $\{C, Bruni\}$ e $|R_1| = 3; \quad |R_2| = 2; \quad |R_1 \bowtie R_2| = 2$..

La cardinalità massima quindi del join non completo sarà:

$$|R_1 \bowtie R_2| < |R_1| \times |R_2|$$

3. **Join vuoto:** nessuna tupla degli operandi trova corrispondenza. $|R_1 \bowtie R_2| = 0$
Esempio:

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Table 41: R1

Reparto	Capo
D	Mori
C	Bruni

Table 42: R2

Impiegato	Reparto	Capo
...	...	...

Table 43: $R_1 \bowtie R_2$

In questo caso quindi $|R_1 \bowtie R_2| = 0$.

4. ***Caso particolare di join completo:*** tutte le tuple di una relazione si combinano con tutte le altre tuple dell'altra relazione.
Esempio:

Impiegato	Reparto
Rossi	F
Neri	F

Table 44: R1

Reparto	Capo
F	Mori
F	Bruni

Table 45: R2

Impiegato	Reparto	Capo
Rossi	F	Mori
Rossi	F	Bruni
Neri	F	Mori
Neri	B	Bruni

Table 46: $R_1 \bowtie R_2$

In questo caso $|R_1 \bowtie R_2| = |R_1| \times |R_2|$. Ogni tupla di R_1 è combinabile con ogni tupla di R_2.

Dimensione del risultato del join naturale: Il join $R_1 \bowtie R_2$ contiene un numero di tuple compreso tra 0 e $|R_1| \times |R_2|$. Quindi $0 \leq |R_1 \bowtie R_2| \leq |R_1| \times |R_2|$.
Inoltre, se il join di R_1 e R_2 è completo, allora contiene *almeno* un numero di tuple pari al massimo tra la cardinalità di R_1 ($|R_1|$) e la cardinalità di R_2. ($|R_2|$)

Considerazioni sui join:

Date $R_1(A, B)$ e $R_2(B, C)$ in generale $0 \leq |R_1 \bowtie R_2| \leq |R_1| \times |R_2|$.

- ◇ **Se B è chiave** in R_2 allora $0 \leq |R_1 \bowtie R_2| \leq |R_1|$

A	**B**
...	x
...	x
...	z

Table 47: R1

B	**C**
x	...
y	...

Table 48: R2

A	**B**	**C**
...	x	...
...	x	...

Table 49: $R_1 \bowtie R_2$

- ◇ **Se B è chiave** in R_2 **ed esiste un vincolo di integrità referenziale** tra $R_1.B$ ed R_2. Allora $|R_1 \bowtie R_2| = |R_1|$.

A	**B**
...	x
...	x
...	y
...	$\not$

Table 50: R1

B	**C**
x	...
y	...

Table 51: R2

A	**B**	**C**
...	x	...
...	x	...
...	y	...

Table 52: $R_1 \bowtie R_2$

Se io volessi trovare tutti gli impiegati con l'indicazione del capo, se noto, un join non basterebbe perché vedrei solo le tuple con corrispondenza. Utilizzo quindi un **JOIN ESTERNO** che estende le tuple anche a quelle che non hanno corrispondenza.

JOIN ESTERNO:

Prevede che tutte le tuple degli operandi diano un contributo al risultato. Estende con valori NULL le tuple che verrebbero tagliate fuori da un join interno. Abbiamo **tre versioni del join esterno:**

- ◇ **Join esterno SINISTRO:** mantiene tutte le tuple del primo operando, estendendole con valori nulli se necessario.

  Esempio:

Impiegato	**Reparto**
Rossi	A
Neri	B
Bianchi	B

Table 53: R1

Reparto	**Capo**
B	Mori
C	Bruni

Table 54: R2

Impiegato	**Reparto**	**Capo**
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL

Table 55: $R_1 \bowtie_{LEFT} R_2$

- ◇ **Join esterno DESTRO:** mantiene tutte le tuple del secondo operando (a destra), estendendole con valori nulli se necessario.

Impiegato	**Reparto**
Rossi	A
Neri	B
Bianchi	B

Table 56: R1

Reparto	**Capo**
B	Mori
C	Bruni

Table 57: R2

Impiegato	**Reparto**	**Capo**
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni

Table 58: $R_1 \bowtie_{RIGHT} R_2$

- ◇ **Join esterno COMPLETO:** mantiene tutte le tuple di entrambi gli operandi, estendendole con valori nulli se necessario.

Impiegato	**Reparto**
Rossi	A
Neri	B
Bianchi	B

Table 59: R1

Reparto	**Capo**
B	Mori
C	Bruni

Table 60: R2

Impiegato	**Reparto**	**Capo**
Rossi	A	NULL
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni

Table 61: $R_1 \bowtie_{FULL} R_2$

- **Join naturale N-ario:**

  Proprietà del join naturale: (alcuni non valgono per i join esterni)

  - ◇ Il join naturale è **commutativo**. Non lo è il left/right-join, mentre il full join sì.
  - ◇ Il join naturale è **associativo**. Cioè $R_1 \bowtie (R_2 \bowtie R_3) = (R_1 \bowtie R_2) \bowtie R_3 = R_1 \bowtie R_2 \bowtie R_3$

    Esempio: dato lo schema

$$IMPIEGATO(NomeImpiegato, NomeReparto)$$
$$REPARTO(NomeReparto, NomeCapo)$$
$$CAPOPROGETTO(NomeCapo, NomeProgetto)$$
$$PROGETTO(NomeProgetto, Descrizione)$$

    Allora $IMPIEGATO \bowtie REPARTO \bowtie CAPOPROGETTO \bowtie PROGETTO$ avrà come schema:

$$RIS(NomeImpiegato, NomeReparto, NomeCapo, NomeProgetto, Descrizione)$$

ATTENZIONE: dati $R_1(X_1) \bowtie R_2(X_2)$ fin'ora abbiamo visto che $|X_1 \cap X_2| > 0$, e nel nostro caso degli esempi $X_1 \cap X_2 = \{B\}$.
Abbiamo due **casi estremi**:

- ◇ $X_1 = X_2$. Allora $R_1 \bowtie R_2 \equiv R_1 \cap R_2$

  Esempio: $R_1(A, B) \bowtie R_2(A, B)$

A	B
1	x
2	x
3	z

Table 62: R1

A	B
1	x
2	z
3	z

Table 63: R2

A	B
1	x
3	z

Table 64: $R_1 \bowtie R_2$

- ◇ $X_1 \cap X_2 = \emptyset$. Allora $R_1 \bowtie R_2 \equiv R_1 \times R_2$

  Poiché X_1 e X_2 NON hanno attributi in comune, la condizione che le due tuple devono avere gli stessi valori sugli attributi comuni degenera in una condizione sempre verificata. Quindi il risultato contiene tutte le tuple ottenute dalla combinazione in tutti i modi possibili delle tuple degli operandi.

  Esempio: $R_1(A, B) \bowtie R_2(C, D)$

A	B
1	x
2	x
3	z

Table 65: R1

C	D
a	3
x	4
w	5

Table 66: R2

A	B	C	D
1	x	a	3
1	x	x	4
1	x	w	5
2	x	a	3
2	x	x	4
2	x	w	5
3	z	a	3
3	z	x	4
3	z	w	5

Table 67: $R_1 \bowtie R_2$

Un join naturale su relazioni che non hanno attributi in comune è un **prodotto cartesiano**. Il risultato contiene sempre un numero di tuple pari al prodotto della cardinalità dei due. $|R_1 \bowtie R_2| = |R_1| \times |R_2|$

- **Prodotto cartesiano $R_1 \times R_2$:** il risultato di $R_1(X_1) \bowtie R_2(X_2)$ con $X_1 \cap R_2 = \emptyset$ è una tabella con schema formato da tutti gli attributi di X_1 e X_2, il cui $grado(R_1 \times R_2) = grado(R_1) + grado(R_2)$.

  L'istanza, invece, è formata da *tutte le possibile coppie di tuple* di R_1 e R_2. Quindi $|R_1 \bowtie R_2| = |R_1| \times |R_2|$

Esempio: in questo esempio avviene un prodotto cartesiano perché, anche se Reparto e Codice indicano la stessa cosa, non posseggono lo stesso nome dell'attributo. Questa condizione viene risolta nel seguente punto col *theta-join* (vedi punto elenco successivo).

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Table 68: IMPIEGATO

Codice	Capo
A	Mori
B	Bruni

Table 69: REPARTO

Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

Table 70: IMPIEGATO $\bowtie$ REPARTO
(o meglio IMPIEGATO $\times$ REPARTO)

- **Theta-join:** operatore definito come **prodotto cartesiano seguito da una selezione**.

  Data la formula proposizionale F (congiunzione *AND* di atomi di confronto, cioè F può essere $A = B, A > B, A < B, etc...$) e due relazioni $R_1(X_1)$ e $R_2(X_2)$ con $X_1 \cap X_2 = \emptyset$, si definisce

  Theta-join:

$$R_1 \bowtie_F R_2 = \sigma_F(R_1 \bowtie R_2)$$

Esempio:

$$\sigma_{Reparto=Codice}(IMPIEGATO \bowtie REPARTO) = IMPIEGATO \bowtie_{Reparto=Codice} REPARTO$$

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Table 71: IMPIEGATO

Reparto	Capo
A	Mori
B	Bruni

Table 72: REPARTO

Impiegato	Reparto	Capo
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni

Table 73: $IMPIEGATO \bowtie_{Reparto=Codice} REPARTO$

Esempio[2]: dato il seguente schema

$$IMPIEGATO(\underline{Matricola}, Nome, Eta, Stipendio)$$
$$SUPERVISIONE(\underline{Impiegato}, Capo)$$

Osservazione: Scrivere

$$IMPIEGATO(\underline{Matricola}, Nome, Eta, Stipendio)$$
$$SUPERVISIONE^1(\underline{Impiegato}, Capo)$$

è molto diverso da scrivere

$$IMPIEGATO(\underline{Matricola}, Nome, Eta, Stipendio)$$
$$SUPERVISIONE^2(\underline{Impiegato}, \underline{Capo})$$

Poiché $SUPERVISIONE^1$ permette una relazione 1-N tra impiegato e capo, mentre $SUPERVISIONE^2$ permette una relazione N-N tra impiegato e capo. *(verrà approfondito in seguito)*

- ◇ Cioè, $SUPERVISIONE^1$ permette che per ogni impiegato vi sia *uno ed un solo* capo, infatti, la tupla che contiene lo stesso impiegato non si può ripetere; dunque, un impiegato in questo schema non può avere due capi.
- ◇ Se invece specifico $SUPERVISIONE^2$, sto permettendo che ogni impiegato abbia *almeno uno o più* capi e che, quindi, ciascun capo abbia almeno uno o più impiegati. Infatti, la condizione di super-chiave è sulla coppia Impiegato-Capo. Non potrò ripetere due volte che lo stesso impiegato ha lo stesso capo (ad esempio dire "A è impiegato di B e A è impiegato di B"), ma invece potrò dire che lo stesso impiegato ha più capi (ad esempio, "A è impiegato di B e A è impiegato di C e D è impiegato di B").

Esercizio: *Trovare le matricole dei capi degli impiegati che guadagnano più di 40*

Matricola	Nome	Eta	Stipendio
7309	Rossi	34	45
5998	Bianchi	37	38
9553	Neri	42	35
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

Table 74: IMPIEGATO

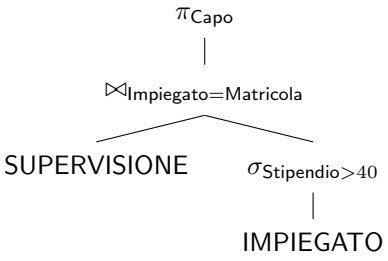
Impiegato	Capo
7309	5698
5998	5698

Table 75: SUPERVISIONE

Soluzione:

$$\Pi_{Capo}(SUPERVISIONE \bowtie_{Impiegato=Matricola} (\sigma_{Stipendio>40}(IMPIEGATO)))$$

Rappresentazione ad albero della soluzione:



4.2 Rappresentazione delle espressioni tramite gli alberi

Ogni espressione dell'algebra relazionale si può rappresentare mediante un albero sintattico che esprime l'ordine di valutazione degli operatori.

Esempi usando un paio di query difficili:

Trovare gli impiegati che guadagnano più del proprio capo, mostrando nel risultato Matricola, Nome e stipendio dell'impiegato e del capo.

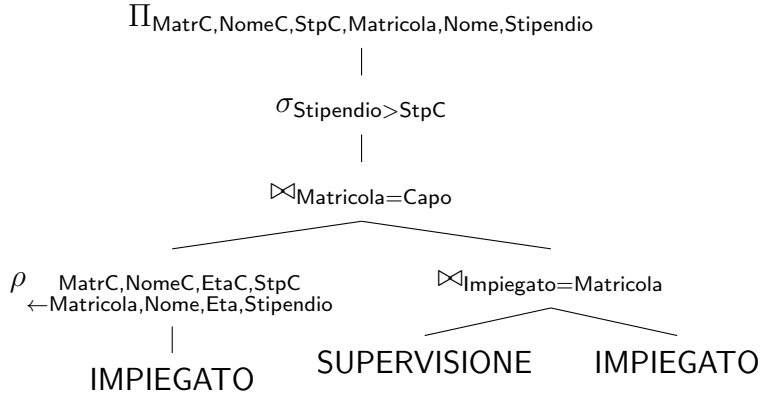
Matricola	Nome	Eta	Stipendio
7309	Rossi	34	45
5998	Bianchi	37	38
9553	Neri	42	35
5698	Bruni	43	42
4076	Mori	45	50
8123	Lupi	46	60

Table 76: IMPIEGATO

Impiegato	Capo
7309	5698
5998	5698
9553	4076
5698	4076
4076	8123

Table 77: SUPERVISIONE

Soluzione con albero:



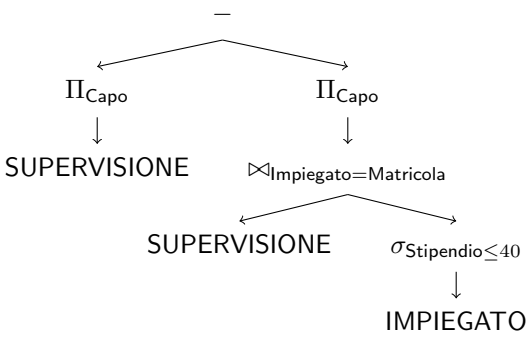
Soluzione in algebra relazionale:

$$\Pi_{MatrC,NomeC,StpC,Matricola,Nome,Stipendio}($$
$$\sigma_{Stipendio>StpC}($$
$$\rho_{MatrC,NomeC,EtaC,StpC \leftarrow Matricola,Nome,Eta,Stipendio}(IMPIEGATO)$$
$$\bowtie_{Matricola=Capo} ($$
$$SUPERVISIONE \bowtie_{Impiegato=Matricola} IMPIEGATO)))$$

Query 2: Trovare le matricole dei capi i cui impiegati guadagnano *TUTTI* più di 40.
Qua la soluzione si affida all'insiemistica, ovvero al sottrarre dall'insieme di tutti i capi quelli che hanno impiegati che guadagnano meno o uguale di 40.
Soluzione in algebra relazionale:

$$\Pi_{Capo}(SUPERVISIONE) - \Pi_{Capo}(SUPERVISIONE \bowtie_{Impiegato=Matricola} (\sigma_{Stipendio \leq 40}(IMPIEGATO)))$$

Soluzione in albero:



4.3 Equivalenza di espressioni algebriche

Equivalenza: due espressioni in algebra relazionale sono equivalenti se producono lo stesso risultato. Cioè

$$X \times (Y + Z) \equiv X \times Y + X \times Z$$

In algebra relazionale abbiamo due tipi di equivalenza:

- Equivalenza **assoluta:**

$$\Pi_{A,B}(\sigma_{A>0}(R)) \equiv \sigma_{A>0}(\Pi_{A,B}(R))$$

- Equivalenza che **dipende dallo schema:**

$$\Pi_{A,B}(R_1) \bowtie \Pi_{A,C}(R_2) \equiv_R \Pi_{A,B,C}(R_1 \bowtie R_2)$$

  che vale solo se nello schema R l'intersezione tra gli attributi di R_1 e R_2 è pari ad A.

A cosa servono le equivalenze tra espressioni algebriche?
Le espressioni equivalenti servono ai DBMS per ottimizzare le query che poniamo, affinché si trovino quelle meno costose computazionalmente, ovvero quelle che si portano dietro meno tuple. Sono interessanti le trasformazioni di equivalenze che riducono le dimensioni dei risultati intermedi.

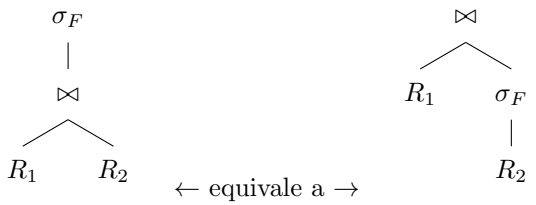
4.4 Trasformazioni di equivalenza

- **Anticipazione delle selezioni rispetto al join** (PUSHING SELECTIONS DOWN):

$$\sigma_F(R_1 \bowtie R_2) \equiv R_1 \bowtie \sigma_F(R_2)$$

se la condizione F fa riferimento solo ad attributi di R_2.

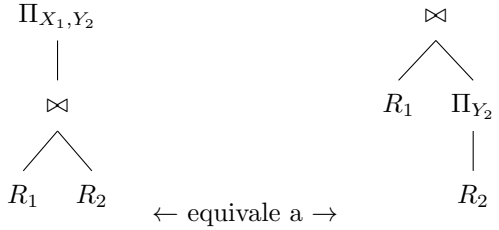
Nella rappresentazione ad albero si rappresenta così:



- **Anticipazione della proiezione rispetto al join** (PUSHING PROJECTIONS DOWN)

$$\Pi_{X_1,Y_2}(R_1 \bowtie R_2) \equiv R_1 \bowtie \Pi_{Y_2}(R_2)$$

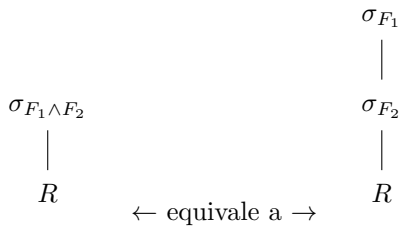
con $Y_2 \subseteq X_2$ e $Y_2 \supseteq (X_1 \cap X_2)$



La strategia in tutti questi casi è che se non mi serve quello che non appare nel risultato e quello che non è coinvolto nel join, allora lo rimuovo agli inizi della query.

- **Atomizzazione delle selezioni:**

$$\sigma_{F_1 \wedge F_2}(R) \equiv \sigma_{F_1}(\sigma_{F_2}(R))$$



- **Idempotenza delle proiezioni:** data una generica espressione algebrica E

$$\Pi_X(E) \equiv \Pi_X(\Pi_{X,Y}(E))$$

- **Distributività della selezione rispetto all'unione:**

$$\sigma_F(R_1 \cup R_2) \equiv \sigma_F(R_1) \cup \sigma_F(R_2)$$

- **Distributività della proiezione rispetto all'unione:**

$$\Pi_X(R_1 \cup R_2) \equiv \Pi_X(R_1) \cup \Pi_X(R_2)$$

Trasformazioni di equivalenza: lo scopo delle trasformazioni di equivalenza è quello di anticipare *il prima possibile* la *proiezione e selezione* all'interno di un'espressione algebrica (o interrogazione) così da ottimizzarne le prestazioni. Le prestazioni vengono ottimizzate perché anticipando queste due operazioni si "scremano", cioè si escludono, presto le colonne o le tuple che non non interessano alla risoluzione dell'interrogazione, così da ridurre il carico a tutte le operazioni che seguiranno.

Esempio: dato lo schema

$$IMPIEGATO(\underline{Matricola}, Nome, Eta, Stipendio)$$
$$SUPERVISIONE(\underline{Impiegato}, Capo)$$

E data la seguente query che richiede i capi che hanno impiegati minori di 30 anni:

$$\Pi_{Capo}(\sigma_{Matricola=Impiegato \wedge Eta<30}(IMPIEGATO \bowtie SUPERVISIONE))$$

è equivalente alla seguente query

$$\Pi_{Capo}(\Pi_{Matricola}(\sigma_{Eta<30}(IMPIEGATO)) \bowtie_{Matricola=Impiegato} SUPERVISIONE)$$

4.5 Algebra con valori nulli

Dato lo schema seguente

Matricola	**Cognome**	**Filiale**	**Eta**
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

- ◇ Se io effettuo la query $\sigma_{Eta>40}(IMPIEGATO)$ il risultato darà solo Neri.
- ◇ Mentre se effettuo la query $\sigma_{Eta\leq 40}(IMPIEGATO)$ il risultato darà solo Rossi

Come si può notare, quindi, purtroppo l'unione delle due espressioni algebriche non ci dà la totalità delle tuple della tabella. Dovremo includere la seguente espressione:

$$\sigma_{Eta\ IS\ NULL}(IMPIEGATO)$$

Quindi:

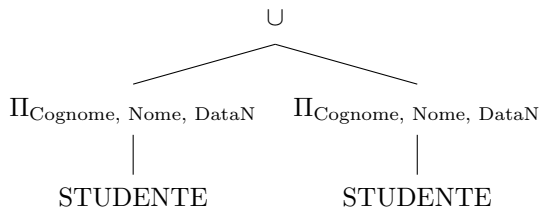
$$\sigma_{Eta>40}(IMPIEGATO) \cup \sigma_{Eta\leq 40}(IMPIEGATO) \cup \sigma_{Eta\ IS\ NULL}(IMPIEGATO) = IMPIEGATO$$

Esercizi: dato il seguente schema

$$STUDENTE(\underline{Matricola}, Cognome, Nome, DataNascita)$$
$$ESAME(\underline{MatrStud} \rightarrow STUDENTE.Matricola, \underline{CodIns} \rightarrow INSEGNAMENTO.Codice, Voto, Data)$$
$$INSEGNAMENTO(\underline{Codice}, Titolo, CFDoc \rightarrow DOCENTE.CF, AA)$$
$$DOCENTE(\underline{CF}, Cognome, Nome, DataNascita)$$

1. **Trovare cognome, nome e data di nascita di docenti e studenti.**

$$\Pi_{Cognome,Nome,DataN}(STUDENTE) \cup \Pi_{Cognome,Nome,DataN}(STUDENTE)$$

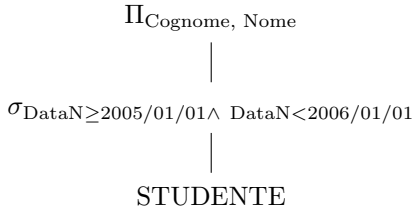


2. **Trovare cognome nome degli studenti nati nel 2005**

$$\Pi_{Cognome,Nome}(\sigma_{DataN\geq 2005/01/01}(\sigma_{DataN<2006/01/01}(STUDENTE)))$$

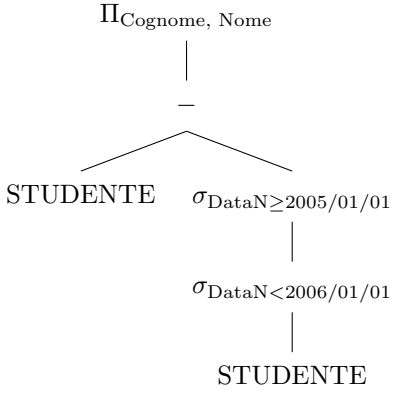
oppure

$$\Pi_{Cognome,Nome}(\sigma_{DataN\geq 2005/01/01 \wedge DataN<2006/01/01}(STUDENTE))$$



3. **Trovare cognome e nome degli studenti che NON sono nati nel 2005.**
   Ci sono diverse soluzioni: questa non va bene, devi fare prima sottrazione e poi proiezione

$$\Pi_{Cognome,Nome}(STUDENTE - \sigma_{DataN \geq 2005/01/01}(\sigma_{DataN<2006/01/01}(STUDENTE)))$$



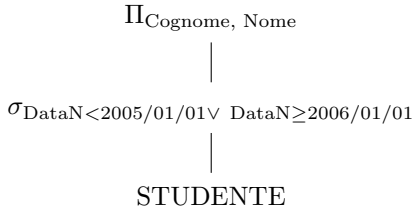
ATTENZIONE: Se scrivessi la seguente espressione, invece, che inverte ordine di proiezione e sottrazione

$$\Pi_{Cognome,Nome}(STUDENTE) - \Pi_{Cognome,Nome}(\sigma_{DataN \geq 2005/01/01}(\sigma_{DataN<2006/01/01}(STUDENTE)))$$

Questo mi escluderebbe casi di omonimia. Ad esempio, se ci fosse una "Sara Rossi" nata nel 2008 e una "Sara Rossi" nata nel 2005, la sottrazione impedirebbe di vedere quella nata nel 2008.

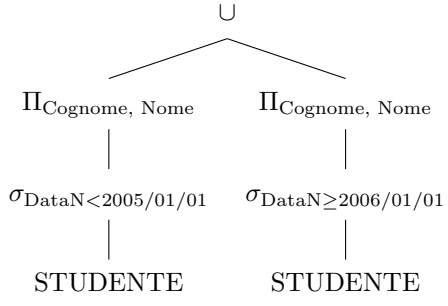
oppure un'altra soluzione è:

$$\Pi_{Cognome,Nome}(\sigma_{DataN<2005/01/01 \vee DataN \geq 2006/01/01}(STUDENTE))$$



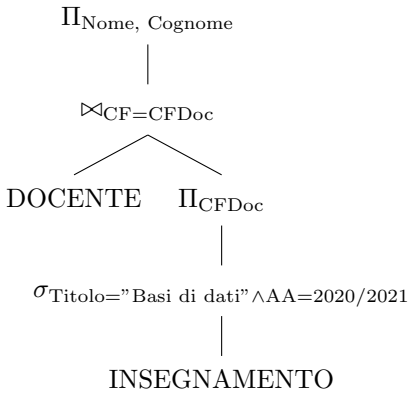
oppure

$$\Pi_{Cognome,Nome}(\sigma_{DataN<2005/01/01}(STUDENTE)) \cup \Pi_{Cognome,Nome}(\sigma_{DataN \geq 2006/01/01}(STUDENTE))$$



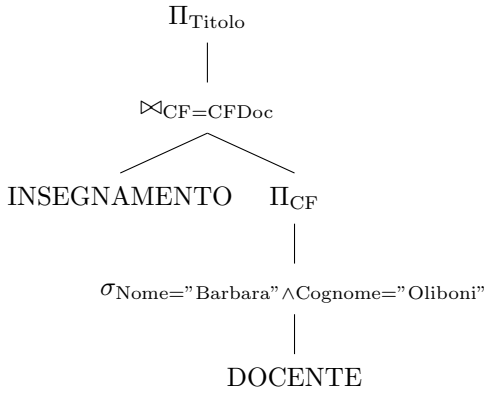
4. **Trovare cognome e nome dei docenti che hanno tenuto il corso di Basi di Dati nell'A.A. 2020/2021**

$$\Pi_{Nome,Cognome}(DOCENTE \bowtie_{CF=CF} \Pi_{CFDoc}((\sigma_{Titolo="Basi\ di\ dati"\ \wedge AA=2020/2021}(INSEGNAMENTO)))$$



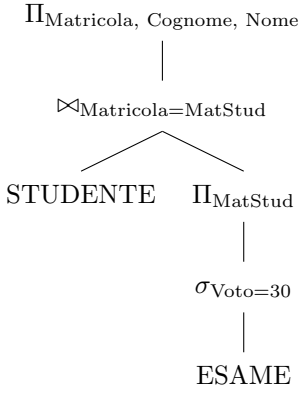
5. **Trovare il titolo degli insegnamenti tenuti da Barbara Oliboni**

$$\Pi_{Titolo}(INSEGNAMENTO \bowtie_{CF=CFDOC} \Pi_{CF}(\sigma_{Nome="Barbara"\wedge Cognome="Oliboni"}(DOCENTE)))$$



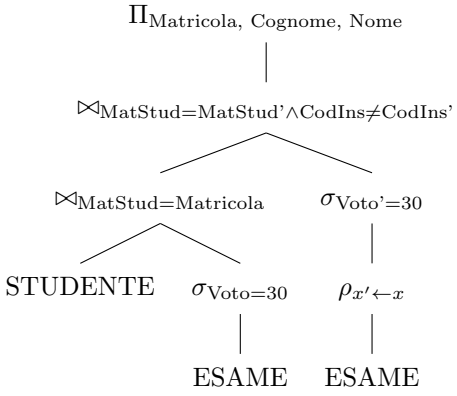
6. **Trovare matricola, cognome e nome degli studenti che hanno preso almeno un 30**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{Matricola=MatStud} (\Pi_{MatStud}(\sigma_{Voto=30}(ESAME))))$$



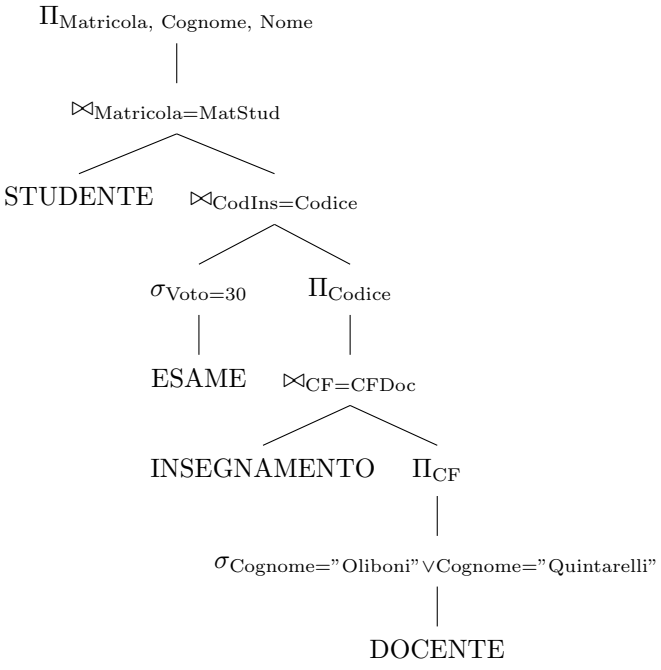
7. **Trovare matricola, cognome e nome degli studenti che hanno preso almeno due 30:**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{MatStud=Matricola} (\sigma_{Voto=30}(ESAME)) \bowtie_{MatStud=MatStud'\wedge CodIns\neq CodIns'} (\sigma_{Voto'=30}(\rho_{x'\leftarrow x}(ESAME))))$$



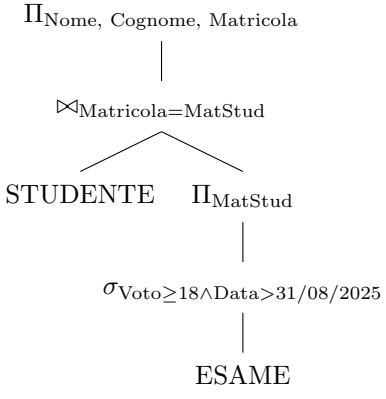
8. **Trovare matricola, cognome e nome degli studenti che hanno preso 30 in un esame di un insegnamento tenuto da Oliboni o da Quintarelli**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{Matricola=MatStud} (\sigma_{Voto=30}(ESAME) \bowtie_{CodIns=Codice} \Pi_{Codice}(INSEGNAMENTO \bowtie_{CF=CFDoc} \Pi_{CF}(\sigma_{Cognome="Oliboni" \vee Cognome="Quintarelli"}(DOCENTE)))))$$



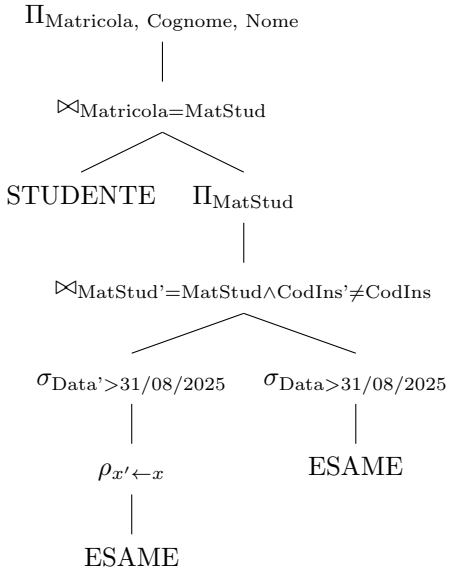
9. **Trovare matricola, cognome e nome degli studenti che hanno superato almeno un esame dopo il 31/08/2025**

$$\Pi_{Nome,Cognome,Matricola}(STUDENTE \bowtie_{Matricola=MatStud} (\Pi_{MatStud}(\sigma_{Voto \geq 18 \wedge Data > 31/08/2025}(ESAME))))$$



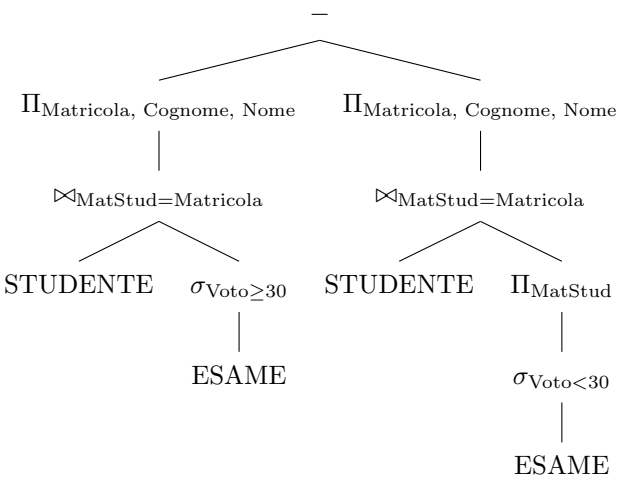
10. **Trovare matricola, cognome e nome degli studenti che hanno superato ALMENO DUE esami dopo il 31/08/2025**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{Matricola=MatStud} (\Pi_{MatStud}(\sigma_{Data'>31/08/2025}(\rho_{x' \leftarrow x}(ESAME)) \bowtie_{MatStud'=MatStud \wedge CodIns' \neq CodIns} \sigma_{Data>31/08/2025}(ESAME))))$$



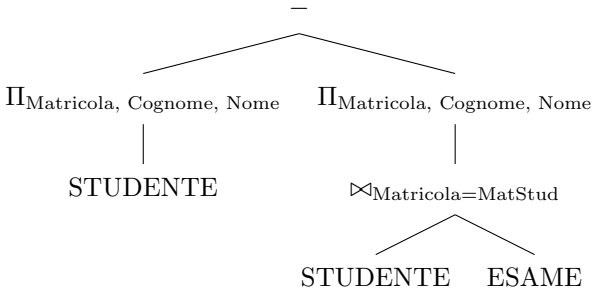
11. **Trovare matricola, cognome e nome degli studenti che hanno preso tutti trenta.**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{MatStud=Matricola} (\sigma_{Voto \geq 30}(ESAME)) - \Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{MatStud=Matricola} (\Pi_{MatStud}(\sigma_{Voto<30}(ESAME)))))$$



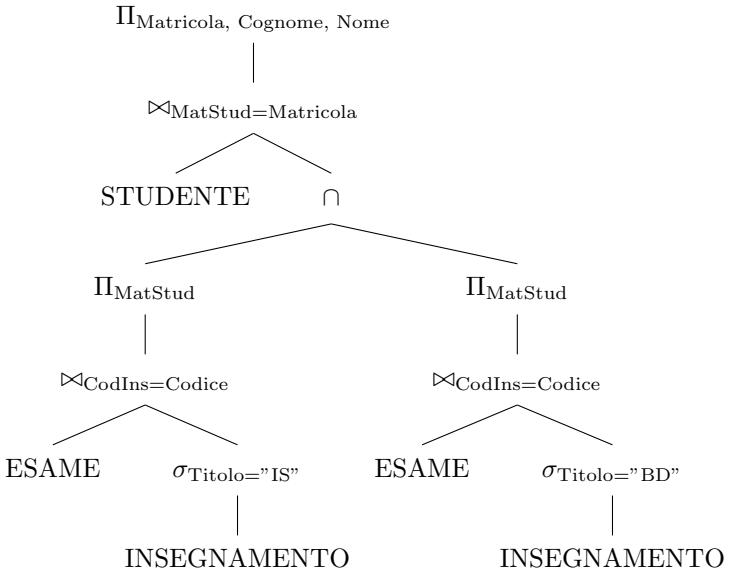
12. **Trovare matricola, cognome e nome degli studenti che non hanno fatto nessun esame:**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE) - \Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{Matricola=MatStud} ESAME)$$



13. **Trovare matricole, cognome e nome degli studenti che hanno superato sia l'esame di basi di dati (BD) che l'esame di ingegneria del software (IS)**

$$\Pi_{Matricola,Cognome,Nome}(STUDENTE \bowtie_{MatStud=Matricola} (\Pi_{MatStud}(ESAME \bowtie_{CodIns=Codice} (\sigma_{Titolo="IS"}(INSEGNAMENTO))) \cap \Pi_{MatStud}(ESAME \bowtie_{CodIns=Codice} (\sigma_{Titolo="BD"}(INSEGNAMENTO)))))$$



4.6 Esercizi da esame:

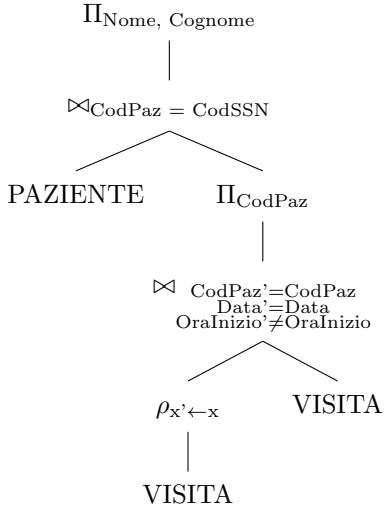
4.6.1 Algebra relazionale

Dato il seguente schema relazionale:

$$PAZIENTE(\underline{CodSSN}, Nome, Cognome, NTel, Indirizzo, Regione)$$
$$VISITA(\underline{CodPaz} \rightarrow PAZIENTE.CodSSN, \underline{CodMed} \rightarrow MEDICO.CF, \underline{Data}, \underline{OraInizio}, DurataInMinuti)$$
$$MEDICO(\underline{CF}, Cognome, Nome, Specialita)$$

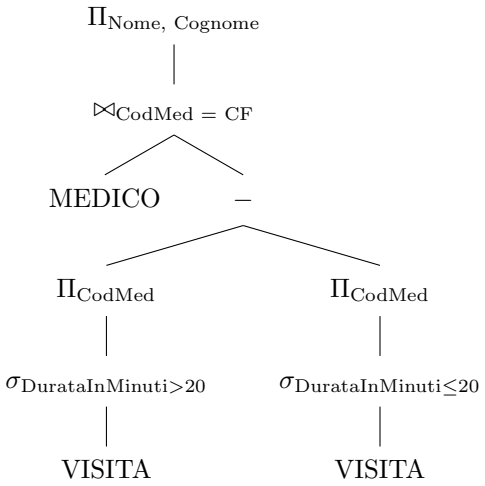
1. **Trovare nome e cognome dei pazienti che sono stati sottoposti a due visite nello stesso giorno:**

$$\Pi_{Nome,Cognome}(PAZIENTE \bowtie_{CodPaz=CodSSN} \Pi_{CodPaz}((\rho_{x' \leftarrow x} VISITA) \bowtie_{CodPaz'=CodPaz \wedge Data'=Data \wedge OraInizio' \neq OraInizio} (VISITA)))$$



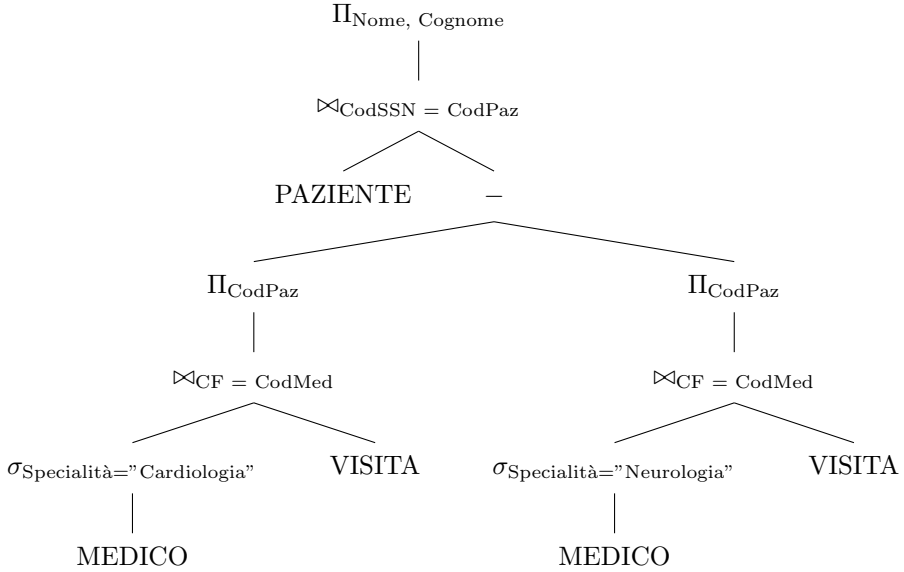
2. **Trovare nome e cognome dei medici che hanno sempre fatto visite con durata superiore a 20 minuti**

$\Pi_{Nome,Cognome}(MEDICO \bowtie_{CodMed=CF} (\Pi_{CodMed}(\sigma_{DurataInMinuti>20} VISITA) - \Pi_{CodMed}(\sigma_{DurataInMinuti\leq 20} VIS$



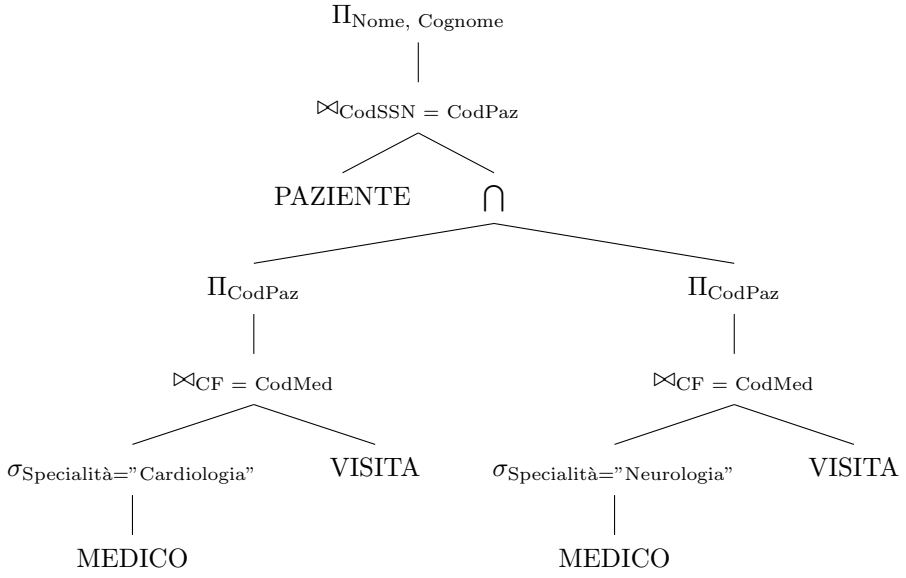
3. **Trovare nome e cognome dei pazienti che sono stati visitati tutti da un medico cardiologo, ma non sono mai stati visitati da un medico neurologo**

$$\Pi_{Nome,Cognome}($$
$$PAZIENTE \bowtie_{CodSSN=CodPaz} ($$
$$\Pi_{CodPaz}(VISITA \bowtie_{CF=CodMed} \sigma_{Specialita="Cardiologia"} MEDICO)$$
$$-\Pi_{CodPaz}(VISITA \bowtie_{CF=CodMed} \sigma_{Specialita="Neurologia"} MEDICO)))$$



4. **Trovare nome e cognome dei pazienti che sono stati visitati sia da un medico cardiologo che da un medico neurologo**.

$$\Pi_{Nome,Cognome}($$
$$PAZIENTE \bowtie_{CodSSN=CodPaz} ($$
$$\Pi_{CodPaz}(VISITA \bowtie_{CF=CodMed} \sigma_{Specialita="Cardiologia"} MEDICO)$$
$$\bigcap \Pi_{CodPaz}(VISITA \bowtie_{CF=CodMed} \sigma_{Specialita="Neurologia"} MEDICO)))$$



4.6.2 Domanda 3 sulla cardinalità

Date le relazioni $R_1(\underline{A}, B, C)$ e $R_2(\underline{D}, E, F)$ aventi rispettivamente cardinalità N_1 e N_2. Assumere che sia definito un vincolo di integrità referenziale tra $R_1.B$ e la chiave di R_2. Indicare la cardinalità del risultato delle seguenti espressioni (eventualmente specificando l'intervallo nel quale può variare):

- $\Pi_{AC}(R_1) \rightarrow |\Pi_{A,C}(R_1)| = N_1$
- $\Pi_{E}(R_2) \rightarrow 1 \leq |\Pi_{E}(R_2)| \leq N_2$
- $R_1 \bowtie_{A=D} R_2 \rightarrow 0 \leq |R_1 \bowtie_{A=D} R_2| \leq min(N_1, N_2)$

  Dove 0 è nel caso minimo dove non c'è corrispondenza , e $min(N_1, N_2)$ dove corrispondono tutti i valori.

A	B	C
X	W	2
Y	X	1
Z	X	1

Table 78: R_1

D	E	F
X	3	4
Z	3	5

Table 79: R_2

A	B	C	D	E	F
X	W	Z	X	3	4
Z	X	1	Z	3	5

Table 80: $R_1 \bowtie_{A=D} R_2$

- $R_1 \bowtie_{C=E} R_2 \rightarrow 0 \leq |R_1 \bowtie_{C=E} R_2| \leq N_1 \times N_2$

  Poiché qua abbiamo attributi non chiave potrebbe essere tra 0 e la combinazione di tutti i valori.
- $R_1 \bowtie_{B=D} R_2 \rightarrow |R_1 \bowtie_{B=D} R_2| = N_1$

  Poiché vi è un vincolo di integrità referenziale.

Dato $R_3(\underline{G}, \underline{H}, I)$, $1 \leq |\Pi_{G,I}| \leq N_3$ e non esattamente N_3 perché prendo **parte della chiave**, che si può ripetere, e non tutta. (Esempio con Nome e Cognome).

5 SQL (Structured Query Language)

Composto da diverse parti:

- **DDL (Data Definition Language):** Definizione della Base di Dati
- **DML (Data Manipulation Language):** Interrogare e modificare i dati della Base di Dati.

5.1 ISTRUZIONI DDL:

- **CREATE TABLE:** creare una tabella.

  Definizione dello schema della tabella, quindi il suo nome, gli attributi che ha e quali vincoli la compongono.

```
CREATE TABLE NomeTabella (
    Attributo Tipo [Valore Default] [Vincolo attributo]
    {, Attributo Tipo [Valore Default] [Vincolo attributo]}
    {,Vincolo Tabella}
);
```

  Esempio:

$$IMPIEGATO(Matricola, Nome, Cognome, Qualifica^*, Stipendio)$$

```
CREATE TABLE Impiegato (
    Matricola CHAR(6) PRIMARY KEY,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    Qualifica VARCHAR(20),
    Stipendio FLOAT NOT NULL,
    UNIQUE (Nome, Cognome)
);
```

  VINCOLI INTRARELAZIONALI:

  - **NOT NULL**
  - **PRIMARY KEY**
  - **UNIQUE**: valore che non si può ripetere all'interno della tabella, ad eccezione se sia chiave

  ATTENZIONE: Dire

```
    UNIQUE (Nome, Cognome)
```

  è diverso da dire

```
Nome VARCHAR(20) NOT NULL UNIQUE
Cognome VARCHAR(20) NOT NULL UNIQUE
```

  Scrviere in SQL:

```
CREATE TABLE Studente(
    Cognome VARCHAR(20),
    Nome VARCHAR(20),
    Indirizzo VARCHAR(40),
    Telefono VARCHAR(20) NOT NULL,
    PRIMARY KEY(Cognome, Nome)
);
```

  È come dire:

$$Studente(\underline{Cognome}, \underline{Nome}, Indirizzo^*, Telefono)$$

- **INSERT INTO:** inserisce i valori che si desiderano nella tabella scelta.

```
INSERT INTO Impiegato (Matricola, Cognome, Nome, Stipendio)
VALUES ("A00000", "Rossi", "Mario", 1500);
```

Darebbe come risultato:

Matricola	Cognome	Nome	Stipendio
A000000	Rossi	Mario	1500

VINCOLI INTERRELAZIONALI:

- **FOREIGN KEY:** crea un vincolo tra i valori dell'attributo della tabella corrente (INTERNA) e i valori dell'attributo di un'altra tabella (ESTERNA). L'attributo della tabella esterna deve essere soggetto a vincolo di UNIQUE o PRIMARY KEY.

ESEMPIO:

$$IMPIEGATO(\underline{Matricola}, Nome, Cognome, NomeDipartimento)$$
$$DIPARTIMENTO(\underline{NomeDip}, Sede, Tel)$$
$$ANAGRAFICA(\underline{CF}, Nome, Cognome, Indirizzo)$$
$$Nome, Cognome \leftarrow UNIQUE$$

```sql
-- tabella interna
CREATE TABLE Impiegato(
    Matricola CHAR(6) PRIMARY KEY,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    NomeDipartimento VARCHAR(15) REFERENCES Dipartimento(NomeDip),
    FOREIGN KEY(Nome, Cognome) REFERENCES Anagrafica(Nome, Cognome)
);

-- tabella esterna
CREATE TABLE Dipartimento(
    NomeDip VARCHAR(15) PRIMARY KEY,
    Sede VARCHAR(20) NOT NULL,
    Tel VARCHAR(15) NOT NULL
);

-- altra tabella esterna
CREATE TABLE Anagrafica(
    CF CHAR(11) PRIMARY KEY,
    Nome VARCHAR(20) NOT NULL,
    Cognome VARCHAR(20) NOT NULL,
    Indirizzo VARCHAR(30) NOT NULL,
    UNIQUE(Nome, Cognome)
);
```

VIOLAZIONE DEI VINCOLI: $\rightarrow$ *OPERAZIONI NON PERMESSE*

- **Operando sulla tabella interna:** modifico il valore dell'attributo della tabella referente, o inserisco una nuova riga.

Diverse alternative per rispondere a vuolazioni generate da modifiche sulla tabella esterna (o Master) La tabella interna (o tabella Slave) deve adeguarsi alle modifiche che avvengono sulla tabella Master.

La modifica che faccio sull'attributo riferito la propago. Reazione **cascade**.

Politiche di reazione per modifica attributo riferito:

- **Set null:** all'attributo referente(tabella *interna*) viene assegnato valore nullo al posto del valore modificato nella tabella *esterna*. Non voglio propagare la modifica, ma voglio farla sostare in uno stato transitorio con valore NULL. Infatti, il vincolo di integrità referenziale non è violato dal valore NULL.
- **set default:** scelgo un valore di default *che deve essere presente nella tabella riferita*. All'attributo referente viene assegnato un valore di default al posto del valore modificato nella tabella esterna.
- no action...? vincolo non rispettato?

Modificazioni tabella interna

– cascade: tutte le righe della tabella interna corrispondenti alla riga cancellata vengono cancellate.

- **UPDATE:**

```
    UPDATE Impiegato
    SET Stipendio = Stipendio +
        100
    WHERE NomeDip = "Acquisti";
```

```
UPDATE Impiegato
SET Stipendio = Stipendio + 100
```

Con

```
UPDATE Impiegato
SET Stipendio = Stipendio + 100;
```

lo dò a tutti

- **SELECT:**

```
SELECT NomeAttributo {,Attributo}
FROM NomeTabella {,Tabella}
[ WHERE Condizione ]
```

Lista degli attributi che voglio vedere nel risultato Tabelle conivolte nell'interrogazione

```
    SELECT Nome, Cognome
    FROM Impiegato
```

$\Pi_{Nome,Cognome} Impiegato$

Matricola	Nome	Cognome	NomeDIp	Stipendio
A12345	Mario	Rossi	Vendite	1600
A23456	Lucia	Verdi	Acquisti	1500
A34567	Luca	Gialli	Vendite	1300

Condizioni che i dati devono soddisfare per appartenere al risultato.

Nome	Cognome
Mario	Rossi
Lucia	Verdi
Luca	Gialli

```
    SELECT Nome, Cognome
    FROM Impiegato
    WHERE NomeDip = "Vendite"
```

Nome	Cognome
Mario	Rossi
Luca	Gialli

$\Pi_{Nome,Cognome}(\sigma_{NomeDip="Vendite"} IMPIEGATO)$

```
    SELECT Nome, Cognome
    FROM Impiegato
    WHERE NomeDip = "Vendite"
    AND Stipendio > 1500;
```

Nome	Cognome
Mario	Rossi

Vedere tutti i dati della tabella

```
    SELECT *
    FROM Impiegato
    WHERE NomeDip = "Vendite"
    AND Stipendio > 1500;
```

è come dire

```
SELECT Matricola, Nome, Cognome, NomeDip, Stipendio
FROM Impiegato
WHERE NomeDip = "Vendite"
AND Stipendio > 1500;
```

Matricola	Nome	Cognome	NomeDIp	Stipendio
A12345	Mario	Rossi	Vendite	1600

```
SELECT Nome AS NomeI, Cognome AS CognomeI
FROM Impiegato
WHERE NomeDip = "Vendite"
AND Stipendio > 1500;
```

NomeI	CognomeI
Mario	Rossi

$$STUDENTE(\underline{Matricola}, Nome, Indirizzo, Citta, CAP, Genere)$$
$$CORSO(\underline{Codice}, NomeCorso, NumCrediti)$$
$$DOCENTE(\underline{Matricola}, Nome, Telefono, Stipendio)$$

1. **Visualizzare tutte le informazioni sui corsi disponibili:**

```
SELECT *
FROM CORSO;
```

2. **Visualizzare tutte le ifnormazioni sui corsi disponibili di 6 CFU:**

```
SELECT *
FROM CORSO
WHERE NumCrediti = 6;
```

3. **Visualizzare il nome dei corsi disponibili di 6 CFU:**

```
SELECT NomeCorso
FROM CORSO
WHERE NumCrediti = 6;
```

4. **Visualizzare matricola e nome degli studenti:**

```
SELECT Matricola, Nome
FROM STUDENTE;
```

5. **Visualizzare il nome e lo stipendio settimanale dei docenti:**

```
SELECT Nome, Stipendio/4 AS StipendioSettimanale
FROM DOCENTE;
```

List degli attributi si chiama anche target list. Nella target list possono apparire espressioni generiche sul valore degli attributi di ciascuna riga selezionata.

- Visualizzare il nome e lo stipendio annuale di docenti che guadagnano più di 1600

```
SELECT Nome, Stipendio * 13 AS StipendioAnnuale
FROM DOCENTE
WHERE Stipendio > 1600;
```

WHERE Nome att OPERATORE valore

Predicati combinati mediante gli operatori AND OR NOT

- Visualizzare Nome e Indirizzo delle studentesse di Verona

```
SELECT Nome, Indirizzo
FROM STUDENTE
WHERE Citta = "Verona" AND Genere = "F"
```

- Visualizzare nome e indirizzo di tutti gli studenti di Verona o Padova

```
SELECT Nome, Indirizzo
FROM STUDENTE
WHERE Citta = "Verona" OR Citta = "Padova"
```

- Visualizzare nome e indirizzo degli studenti che non abitano a Verona

```
    SELECT Nome, Indirizzo
    FROM STUDENTE
    WHERE NOT Citta = "Verona"
```