



UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

Corso 3° anno - 6 CFU

Basi di dati

Professore:
Prof. Riccardo Guidotti

Autore:
Filippo Ghirardini

Anno Accademico 2024/2025

Contents

1	Introduzione	4
1.1	Sistema informativo	4
1.2	Sistema informatico	5
1.2.1	Sistema informatico operativo	5
1.2.2	Sistema informatico direzionale	5
1.2.3	Big Data	6
1.3	Base di dati	6
1.3.1	Sistemi per BD	7
1.3.2	Linguaggi	7
1.3.3	Controllo di BD	8
1.3.4	Vantaggi e svantaggi	8
2	Modelli di dati	9
2.1	Ruoli	9
2.2	Fasi	9
2.3	Aspetto ontologico	9
2.3.1	Conoscenza concreta	10
2.3.2	Conoscenza astratta	11
2.4	Modello entità-relazione	13
2.5	Modello relazionale	14
3	Progettazione	15
3.1	Documentazione	15
4	Modello relazionale	17
4.1	Matematica	17
4.2	Modello	17
4.2.1	Tabella	18
4.3	Valori	18
4.4	Vincoli di integrità	18
4.4.1	Vincoli di ennupla	18
4.5	Chiave	19
4.6	Integrità referenziale	19
4.6.1	Integrità referenziale	20
5	Trasformazione di schemi	21
5.1	Rappresentazioni	22
5.1.1	Uno a molti/uno	22
5.1.2	Molti a molti	23
5.1.3	Gerarchie tra classi	23
5.1.4	Chiavi primarie	24
5.1.5	Attributi multivalore	25
5.1.6	Attributi composti	25
6	Algebra relazionale	26
6.1	Notazione	26
6.2	Operatori primitivi	27
6.2.1	Ridenominazione	27
6.2.2	Unione	27
6.2.3	Differenza	27
6.2.4	Proiezione	28
6.2.5	Restrizione o selezione	28
6.2.6	Prodotto	29
6.3	Operatori derivati	29
6.3.1	Intersezione	29

6.3.2	Inner Join	29
6.3.3	Theta Join	30
6.3.4	Natural Join	30
6.3.5	Semi Join	31
6.3.6	External Join	31
6.3.7	Self Join	31
6.4	Proprietà algebriche degli operatori	32
6.5	Quantificatori	33
6.6	Operatori non insiemistici	33
6.6.1	Group By	33
6.6.2	Proiezione generalizzata	34
6.6.3	Proiezione multi-insiemistica	34
6.6.4	Ordinamento	34
6.7	Calcolo relazionale	34

Basi di dati

Realizzato da: Ghirardini Filippo

A.A. 2024-2025

1 Introduzione

Definizione 1.0.1 (Base di dati). *Insieme **strutturato** e **organizzato** di dati omogenei utilizzati per il supporto allo svolgimento di attività (ente, azienda, ufficio, persona).*

Definizione 1.0.2 (Database Management System). *Sistema software progettato per consentire la **creazione**, **manipolazione** e **interrogazione** di uno o più database in modo corretto ed efficiente.*

Le figure coinvolte nella costruzione di una *base di dati* sono:

- **Committente**
 - Dirigente
 - Operatore
- **Fornitore**
 - Direttore del progetto
 - Analista
 - Progettista di BD
 - Programmatore di applicazioni che usano BD
- **Manutenzione** e messa a punto
 - Gestione del DBMS
 - Amministratore del DBMS

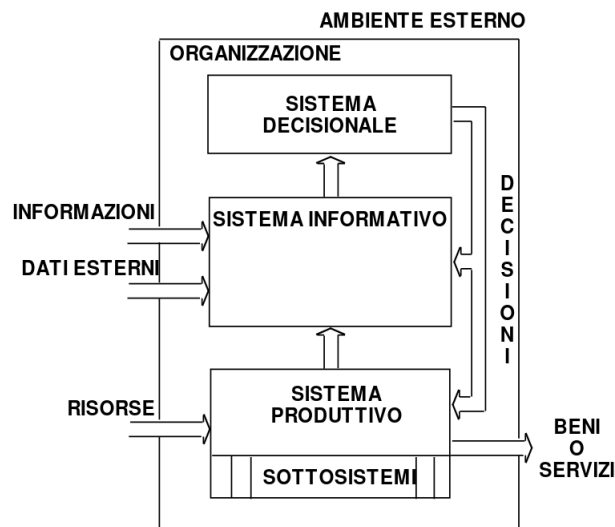
1.1 Sistema informativo

Definizione 1.1.1 (Sistema informativo). *Una combinazione di **risorse**, umane e materiali, e di procedure organizzate allo scopo di:*

- ***raccogliere***
- ***archiviare***
- ***elaborare***
- ***scambiare***

le informazioni necessarie alle attività:

- ***operative** (servizio)*
- ***di programmazione e controllo** (gestione)*
- ***di pianificazione strategica** (governo)*

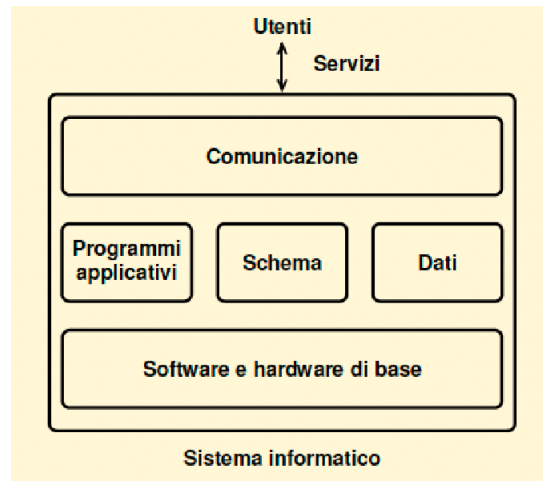


Esempio 1.1.1 (Azienda manifatturiera). Un'azienda manifatturiera avrà un sistema informativo che prevede la **gestione** di ordini (a clienti e fornitori), pagamenti e del magazzino, nonché la **pianificazione** e il controllo dei costi.

1.2 Sistema informatico

Nello specifico, si utilizza un **sistema informatico** per eseguire parte delle operazioni del sistema informativo in maniera **automatizzata**.

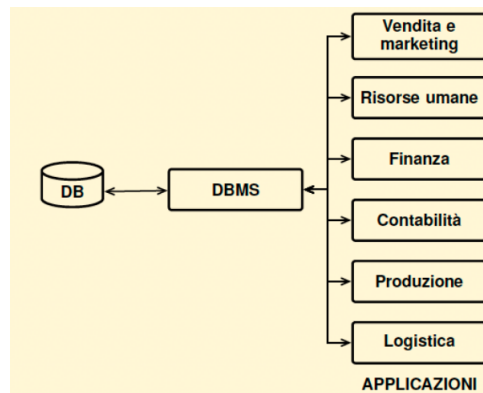
Definizione 1.2.1 (Sistema informatico). *Insieme delle tecnologie informatiche e della comunicazione (ICT) a supporto delle attività di un'organizzazione.*



1.2.1 Sistema informatico operativo

In questo tipo di sistemi i dati sono organizzati in BD (gestite a loro volta da DBMS) e le applicazioni sono usate per svolgere attività **strutturate** e **ripetitive** (e.g. amministrazione, vendite, produzione, HR), ovvero **On-Line Transaction Processing** (operazioni **semplici** e con **pochi dati**).

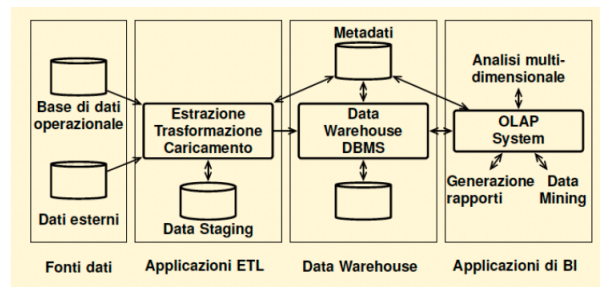
Definizione 1.2.2 (Transaction Processing System). *Sistema basato su transazioni: serie di azioni S che se non andate a buon fine lasciano la BD nello stesso stato in cui era prima che S iniziasse.*



1.2.2 Sistema informatico direzionale

In questo tipo di sistemi i dati sono organizzati in una **Data Warehouse** e gestiti da un opportuno sistema. A differenza di quello operativo, dove i dati sono aggiornati tempestivamente, in questo l'aggiornamento avviene in maniera periodica. Inoltre l'uso principale è per **On-Line Analytical Processing**, ovvero analisi dei dati a supporto delle decisioni (operazioni complesse e con molti dati).

Definizione 1.2.3 (Business Intelligence). *Le applicazioni di Business intelligence sono strumenti di supporto ai processi di controllo delle prestazioni aziendali e di decisione manageriale.*



Di seguito una tabella che riassume le differenze tra **OLTP** e **OLAP**:

	OLTP	OLAP
Scopi	Supporto operatività	Supporto decisioni
Utenti	Molti, esecutivi	Pochi, dirigenti e analisti
Dati	Pochi, analitici, relazionali	Molti, sintetici, multidimensionali
Usi	Noti a priori	Poco prevedibili
Orientamento	Applicazione	Soggetto
Aggiornamento	Frequente	Raro
Visione dei dati	Corrente	Storica
Ottimizzati per	Transazioni	Analisi

1.2.3 Big Data

Con questo termine ci si riferisce alle situazioni in cui DB o DW sono troppo lenti o restrittivi a causa della natura dei dati: **Volume**, **Varietà** e **Velocità**. In questo caso si usano approcci come:

- NoSQL
- **Data mining**: fase di un processo interattivo e iterativo che cerca di estrarre modelli utili per prendere decisioni da un insieme di dati. Il modello sarà una rappresentazione concettuale che evidenzia alcune caratteristiche implicite
- Machine learning
- Data Lake

1.3 Base di dati

Definizione 1.3.1 (Base di dati). Una *BD* è una raccolta di **dati permanenti**, gestiti da un elaboratore elettronico, suddivisi in:

- **Metadati** o schema: definizioni che **descrivono** i dati, pongono **restrizioni** su di essi, indicano le possibili **relazioni** ed **operazioni**
- **Dati**: rappresentazione di fatti conforme allo schema. Sono organizzati in insiemi **strutturati** ed **omogenei** fra i quali sono definite **relazioni**. Hanno le seguenti caratteristiche:
 - **Molti** rispetto ai metadati
 - **Permanenti** fino ad esplicita cancellazione
 - Accessibili tramite **transazioni** (unità atomiche che non possono avere effetti parziali)
 - **Protetti** da utenti ed errori
 - Utilizzabili in **parallelo**

1.3.1 Sistemi per BD

Il DBMS si occupa di garantire le caratteristiche della BD, controllandone i dati e gestendone l'accessibilità.

Definizione 1.3.2 (DBMS). *Un **Database Management System** è un sistema centralizzato o distribuito che offre opportuni linguaggi per:*

- *Definire lo **schema***
- *Scegliere le **strutture dati** per memorizzazione e accesso*
- *Memorizzare, recuperare e modificare i dati interattivamente o da programmi*

Il modello più diffuso è quello **relazionale** che utilizza come meccanismo principale di astrazione la **tabella**, ovvero un insieme di **record** con campi elementari (definiti assieme al nome dallo schema). Le principali funzionalità del DBMS sono:

- **Definizione e uso della BD**
- **Controllo dei dati**
- **Amministrazione:** definizione e modifica degli schemi, controllo e messa a punto del sistema, gestione dei diritti di accesso, strumenti di ripristino
- **Sviluppo:** strumenti per creare applicazioni. E.g. per produrre grafici, rapporti o GUI

1.3.2 Linguaggi

Per **definire** e **usare** una BD esistono due tipi di linguaggi:

- **Data Definition Language:** per la definizione dello *schema*
- **Data Manipulation Language:** permette agli operatori di accedere ai dati e modificarli

Dato che gli utenti sono di diversi tipi un DBMS deve prevedere più modalità d'uso: **GUI**, linguaggio di **querying** per non esperti, linguaggio di **programmazione** per integrarsi con applicazioni e linguaggio per lo **sviluppo di interfacce**.

Un esempio di linguaggio interattivo è **SQL** mentre un linguaggio ad-hoc è **PL/SQL**.

Livelli di descrizione Ci sono tre livelli a cui descrivere i dati:

- **Logico:** descrive la struttura degli insiemi di dati e delle relazioni fra loro, secondo un certo modello dei dati, senza nessun riferimento alla loro organizzazione fisica nella memoria permanente
- **Fisico:** descrive come vanno organizzati fisicamente i dati nelle memorie permanenti e quali strutture dati ausiliarie prevedere per facilitarne l'uso
- **Esterno:** descrive come deve apparire la struttura della base di dati ad una certa applicazione

È necessario che ci sia **indipendenza** logica e fisica:

- **Logica:** i programmi applicativi non devono essere modificati in seguito a modifiche dello schema logico
- **Fisica:** i programmi applicativi non devono essere modificati in seguito a modifiche dell'organizzazione fisica dei dati

1.3.3 Controllo di BD

Una base di dati deve sempre garantire **integrità** (mantenimento delle proprietà specificate nello schema), **sicurezza** (chi e come può accedere ai dati) e **affidabilità**.

Affidabilità La BD deve garantire la protezione dei dati da malfunzionamenti HW, SW e da interferenze dovute ad accesso parallelo.

Definizione 1.3.3 (Malfunzionamento). *Evento a causa del quale la BD si può trovare in uno stato scorretto.*

Le transazioni permettono di garantire affidabilità.

Definizione 1.3.4 (Transazione). *Una sequenza di azioni di lettura e scrittura in memoria permanente e di elaborazioni di dati in memoria temporanea, con le seguenti proprietà:*

- **Atomicità:** *le transazioni che terminano prematuramente sono trattate dal sistema come se non fossero mai iniziate ed eventuali effetti sono annullati*
- **Persistenza:** *le modifiche di una transazione terminata normalmente non sono alterabili da eventuali malfunzionamenti*
- **Serializzabilità:** *nel caso di transazioni concorrenti l'effetto è quello di una esecuzione seriale*

In caso di malfunzionamento rilevato si procede con:

1. Interruzione della transazione o del sistema
2. Messa in atto di procedure di recupero

I tipi di malfunzionamento sono:

Tipo	Perdita dati
Transaction	Nessuna
System	Memoria non permanente
Media	Memoria permanente

1.3.4 Vantaggi e svantaggi

I DBMS garantiscono:

- **Indipendenza, integrità e sicurezza** dei dati
- Gestione degli accessi **concorrenti** e **interattiva** in maniera sicura
- **Amministrazione** dei dati
- Riduzione di **tempi** e **costi** di sviluppo

Al contrario però sono **complessi** e costosi da gestire perché rendono necessaria la definizione di uno **schema** e possono solo gestire dati **strutturati** ed **omogenei**.

2 Modelli di dati

Progettare una BD significa progettare la struttura dei dati e le applicazioni. Per farlo al meglio è fondamentale rappresentare in modo **astratto** e simbolico il dominio del discorso tramite la **modellazione**.

Definizione 2.0.1 (Modello astratto). *Un modello astratto è la **rappresentazione formale** di idee e conoscenze relative ad un fenomeno tramite un **linguaggio formale** a seguito di una **interpretazione soggettiva**.*

2.1 Ruoli

I ruoli principali nella modellazione sono:

- **Committente:** persona con l'esigenza
- **Progettista o analista:** crea un progetto concettuale
- **Programmatore:** sviluppano la BD e le applicazioni
- **DB Administrator:** gestisce gli utenti e il sistema

2.2 Fasi

Le fasi della progettazione sono:

1. **Analisi dei requisiti:** definizione dei bisogni del committente
2. **Progettazione concettuale:** traduzione dei requisiti in un progetto, struttura concettuale dei dati, dei vincoli e delle operazioni
3. **Progettazione logica:** traduzione dello *schema concettuale* nello schema logico, che è espresso nel modello dei dati del sistema scelto
4. **Progettazione fisica:** produce lo schema fisico che arricchisce quello logico con specifiche sull'organizzazione fisica dei dati



2.3 Aspetto ontologico

Questo aspetto si concentra su quale conoscenza del dominio deve essere rappresentata. In particolare la conoscenza può essere **astratta**, **concreta** o **procedurale** (operazioni di base e comunicazione).

2.3.1 Conoscenza concreta

La conoscenza concreta riguarda i **fatti** specifici che si vogliono rappresentare: **entità, collezioni e associazioni**.

Entità e proprietà Le **entità** sono ciò di cui ci interessa rappresentare alcuni fatti o proprietà (e.g. un libro) mentre le **proprietà** sono fatti che descrivono caratteristiche di determinate entità (e.g. titolo).

Note 2.3.1. Un'entità non coincide con l'insieme dei valori assunti dalle sue proprietà, in quanto queste possono cambiare nel tempo o possono essere identiche pur essendo due entità diverse. E.g. una persona la cui età aumenta ogni anno o due persone con stesso nome, età e indirizzo.

Una **proprietà** è una coppia *nome e valore*, dove quest'ultimo è di un certo tipo e all'interno di un **dominio** di possibili valori. Si possono classificare come:

- **Atomica** se il suo valore non è scomponibile, altrimenti **strutturata**
- **Univoca** se il suo valore è unico, altrimenti **multivalore**
- **Totale o obbligatoria** se ogni entità nell'universo assume un valore per essa, altrimenti **parziale o opzionale**
- **Costante o variabile**
- **Calcolata o non calcolata**

Un **tipo di entità** è una descrizione astratta di ciò che accomuna un insieme di entità omogenee, esistenti o possibili. È quindi un insieme infinito. E.g. persona, auto, esame.

Collezione Una collezione è un insieme **variabile** nel tempo di entità omogenee interessanti nel dominio. L'insieme degli elementi di una collezione in un dato momento è detto **estensione** della collezione. A differenza del *tipo di entità*, questi insiemi sono finiti.

Associazione

Definizione 2.3.1 (Istanza di associazione). *Un'istanza di associazione è un fatto che correla due o più entità stabilendo un legame logico tra di loro.*

Definizione 2.3.2 (Associazione). *L'associazione $R(X, Y)$ fra due collezioni di entità X e Y è quindi l'insieme di istanze di associazione tra gli elementi di X e Y che varia nel tempo.*

Definizione 2.3.3 (Prodotto cartesiano). *Il prodotto cartesiano $(X \times Y)$ è il dominio dell'associazione.*

Osservazione 2.3.1. Se vediamo due collezioni X e Y come due insiemi, un'istanza di associazione tra di loro può essere vista come una coppia di elementi $(x; y)$, con $x \in X$ e $y \in Y$, e quindi un'associazione R tra X e Y può essere vista come un sottoinsieme del prodotto $X \times Y$, ovvero come una relazione matematica tra tali insiemi.

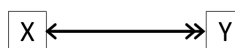
Le associazioni hanno due caratteristiche strutturali:

- **Molteplicità**

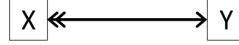
Definizione 2.3.4 (Vincolo di univocità). *Un'associazione $R(X, Y)$ è **univoca** rispetto ad X se per ogni elemento $x \in X$ esiste al più un elemento di Y che è associato ad x . Altrimenti è **multivalore** rispetto ad X .*

Questo ci porta alla **cardinalità** di un'associazione, che può essere:

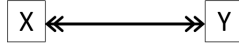
- **Uno a molti:** $R(X, Y)$ è $(1 : N)$ se essa è multivalore su X ed univoca su Y



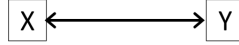
- **Molti a uno:** $R(X, Y)$ è $(N : 1)$ se essa è univoca su X e multivalore



- **Molti a molti:** $R(X, Y)$ è $(N : M)$ se essa è multivalore su X e Y

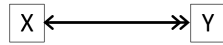


- **Uno a uno:** $R(X, Y)$ è $(1 : 1)$ se essa è univoca su X e Y

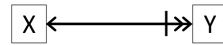


• Totalità

Definizione 2.3.5 (Vincolo di totalità). Un'associazione $R(X, Y)$ è **totale** (o **surgettiva**) su X se per ogni elemento $x \in X$ esiste almeno un elemento di Y che è associato ad x . Altrimenti è **parziale** rispetto ad X .



Totale



Parziale

Un'associazione si rappresenta come visto e con l'aggiunta di un'**etichetta** con il suo nome, scelto di solito con un *predicato*.

2.3.2 Conoscenza astratta

La conoscenza astratta riguarda i fatti generali che descrivono la **struttura** della conoscenza concreta, le **restrizioni** sui valori possibili della conoscenza concreta e sui **vincoli d'integrità** e le **regole** per dedurre nuovi fatti.

Definizione 2.3.6 (Modello dei dati). Un *modello dei dati* è un insieme di meccanismi di astrazione per descrivere la struttura della conoscenza concreta.

Definizione 2.3.7 (Schema). Uno *schema* è la descrizione della struttura della conoscenza concreta e dei vincoli di integrità usando un particolare modello di dati.

Note 2.3.2. Come notazione grafica per lo schema usiamo una **variante** del modello ER.

Oggetti Ad ogni **entità** del dominio corrisponde un oggetto che può rispondere a dei **messaggi** (anche chiamati **attributi**), restituendo valori memorizzati o calcolati tramite procedure.

Definizione 2.3.8 (Oggetto). Un *oggetto* è un'entità software che modella un'entità dell'universo e che ha:

- **Stato:** modellato da un insieme di costanti o variabili con valori di qualsiasi complessità
- **Comportamento:** un insieme di procedure locali chiamate **metodi**, che modellano le operazioni di base che riguardano l'oggetto e le proprietà derivabili da altre
- **Identità**

Il **tipo oggetto** definisce l'insieme dei messaggi a cui può rispondere un insieme di possibili oggetti. Tra i tipi oggetto può essere definita una **relazione di sottotipo** che ha le seguenti proprietà:

- È **asimmetrica**, **riflessiva** e **transitiva** (relazione di *ordine parziale*)
- **Sostitutività:** se T è sottotipo di T' allora gli elementi di T possono essere usati in ogni contesto i cui possano apparire quelli di T' . In particolare:
 - Gli elementi di T hanno tutte le **proprietà** di quelli di T'
 - Per ogni **proprietà** $p \in T'$, il suo tipo T è un sottotipo del suo tipo in T'

Classe Una classe è un insieme di oggetti dello stesso tipo, modificabile con operatori per includere o estrarre elementi dall'insieme.

Spesso le classi sono organizzate in una **gerarchia** di **specializzazione** o **generalizzazione** (**sottoclassi** e **superclassi**). Queste hanno due caratteristiche:

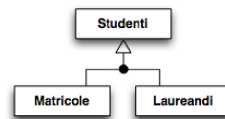
- **Ereditarietà** delle proprietà che permette di definire:
 - un tipo oggetto a partire da un altro
 - l'implementazione di un tipo oggetto a partire da un'altra implementazione. In questo caso gli attributi possono solo essere **aggiunti** o **ridefiniti** solo specializzandone il tipo
- Gli elementi di una sottoclasse sono un **sottoinsieme** di quelli della superclasse. Questa relazione ha le seguenti proprietà:
 - È **asimmetrica, riflessiva e transitiva**
 - **Vincolo intensionale**: se C è sottoclasse di C' allora il tipo degli elementi di C è sottotipo degli elementi di C'
 - **Vincolo estensionale**: se C è sottoclasse di C' allora gli elementi di C sono un sottoinsieme degli elementi di C'

I vincoli sugli insieme di sottoclassi possono essere di **disgiunzione** e di **copertura**. Questo porta ad avere quattro tipi di relazioni tra sottoinsiemi:

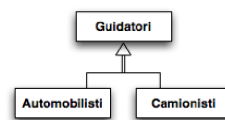
- **Scorrelate**: non richiedono nessun vincolo e possono essere rappresentate in due modi



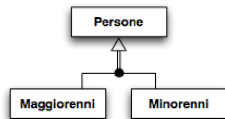
- **Disgiunte**



- **Copertura**

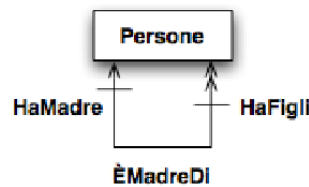


- **Partizione**



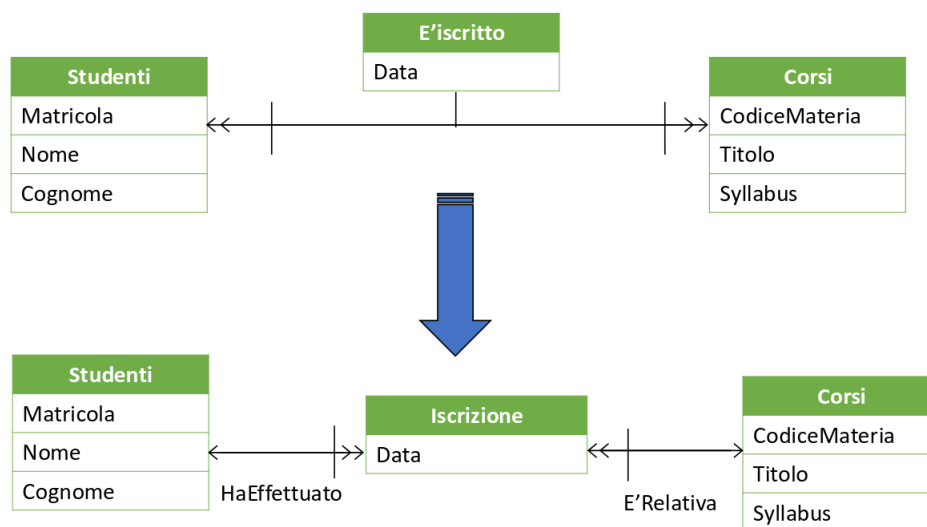
È possibile avere l'**ereditarietà multipla** definendo un tipo per ereditarietà da più supertipi. Bisogna prestare attenzione quando un attributo viene ereditato da più antenati.

Associazioni Le associazioni si modellano con un costrutto apposito e possono avere delle **proprietà** ed essere **ricorsive**. L'ultimo caso si presenta quando abbiamo relazioni binarie fra gli elementi di una stessa collezione. In questo caso bisogna etichettare anche i ruoli agli estremi della freccia.

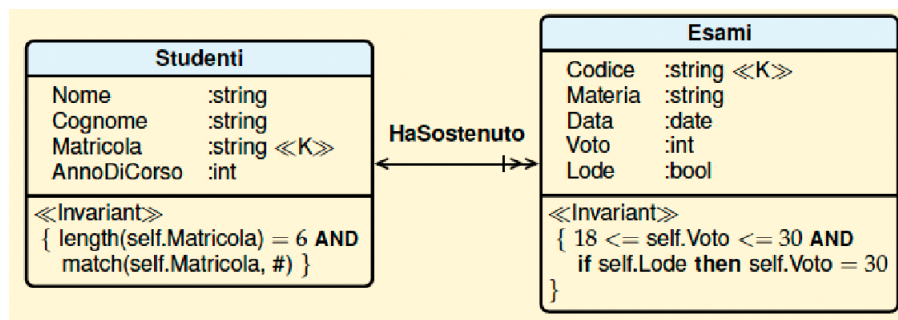


Note 2.3.3. È possibile avere più associazioni tra classi diverse che rappresentano informazioni diverse.

Osservazione 2.3.2 (Reificazione). È possibile trasformare un'associazione tra due classi in una situazione con tre classi e due associazioni. Ad esempio:



Restrizioni I **vincoli d'integrità** impongono restrizioni sui possibili valori della conoscenza concreta. Possono essere **statici** o **dinamici** e arricchiscono la descrizione di una classe.



Note 2.3.4. Gli attributi marcati con << K >> sono una **chiave**.

2.4 Modello entità-relazione

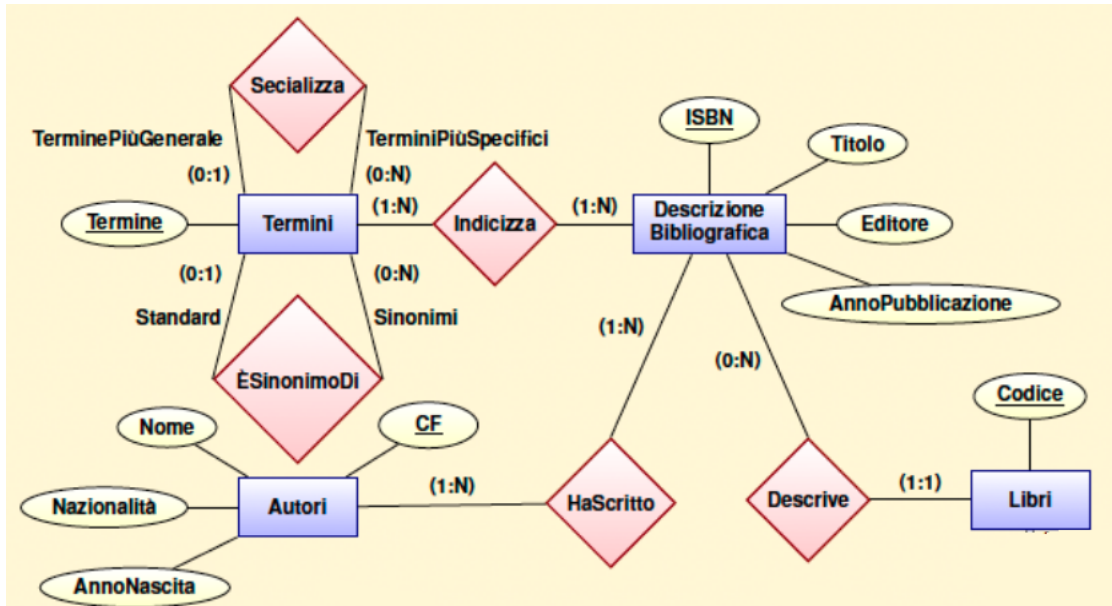
È Il modello più popolare per il disegno concettuale di BD. Non tratta gerarchie di inclusione tra collezioni, non distingue collezioni e tipi e non supporta alcun meccanismo di ereditarietà. Definisce un meccanismo per modellare direttamente le associazioni non binarie o con proprietà.

Prevede due meccanismi di **astrazione**:

- Modellare **insiemi di entità**, con le relative proprietà

- Modellare le associazioni (chiamate **relazioni**).

Le collezioni sono chiamate **tipi di entità**, e gli attributi dei loro elementi possono assumere solo valori di tipo primitivo.



2.5 Modello relazionale

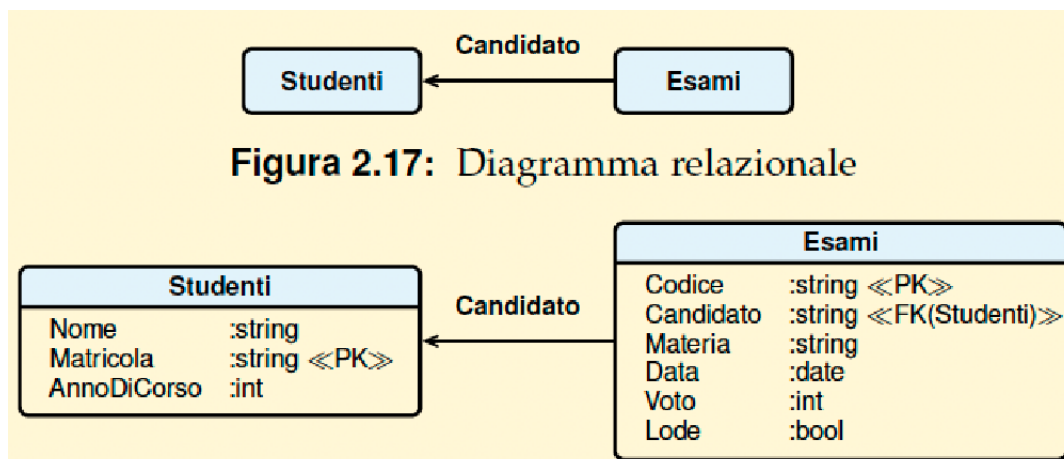
E' il modello dei dati usato dagli attuali sistemi commerciali. I meccanismi per definire una base di dati con questo modello sono l'**ennupla** e la **relazione**.

Definizione 2.5.1 (Ennupla). *Un tipo ennupla è un insieme di coppie (attributo, tipo primitivo) ed un valore di tipo ennupla è un insieme di coppie (attributo, valore), dette anche campi, con gli stessi attributi del tipo e in cui il valore di ogni attributo appartiene al corrispondente tipo primitivo.*

Definizione 2.5.2 (Relazione). *Una relazione è un insieme di ennuple con lo stesso tipo.*

Definizione 2.5.3 (Superchiave e chiave). *Un insieme di attributi i cui valori determinano univocamente un'ennupla di una relazione R è una **superchiave** per R . Una superchiave tale che togliendo un qualunque attributo essa non sia più una superchiave è una **chiave** per R . Tra le chiavi di R ne viene scelta una come chiave **primaria**.*

Le **associazioni** tra i dati sono rappresentate attraverso opportuni attributi, chiamati **chiavi esterne**, che assumono come valori quelli della chiave primaria di un'altra relazione.



3 Progettazione

Progettare una BD significa definire lo **schema globale** dei dati, i **vincoli di integrità** e le **operazioni** delle applicazioni allo scopo di prepararsi alla realizzazione. Si articola in tre fasi:

1. Analisi dei requisiti:

- Analizza il sistema esistente e raccoglie **requisiti informali**
- Elimina **ambiguità, imprecisioni e disuniformità** cercando sinonimi e omonimi e unificandoli
- Raggruppa le frasi relative a diverse categorie di **dati, vincoli, e operazioni**
- Definisce un **glossario**
- Disegna lo **schema di settore**
- Specifica le **operazioni** e ne verifica la **coerenza** con i dati

2. Progettazione:

- Concettuale, logica e fisica dei dati. Identificare **classi** (e gli attributi e tipi), associazioni (e le loro proprietà), elencare le **chiavi**, individuare le **sottoclassi** e le **generalizzazioni**
- Delle applicazioni

3. Realizzazione

3.1 Documentazione

Dato che il linguaggio naturale è pieno di **ambiguità** è fondamentale evitarle. Questo è possibile con l'aiuto delle seguenti regole per scrivere una buona **documentazione**:

- Studiare e comprendere il sistema informativo e i bisogni di tutti i settori dell'organizzazione
- Scegliere il corretto **livello di astrazione**
- **Standardizzare** la struttura delle frasi
- **Suddividere** le frasi articolate
- **Separare** le frasi sui dati da quelle sulle funzioni

Esempio 3.1.1 (Società di formazione). Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei **partecipanti** ai corsi e dei docenti. Per gli **studenti** (circa 5000), identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, il **luogo** di nascita, il nome dei loro attuali datori di lavoro, i **posti** dove hanno lavorato in precedenza insieme al periodo, l'indirizzo e il **numero di telefono**, i corsi che hanno frequentato (i corsi sono in tutto circa 200) e il giudizio finale.

Rappresentiamo anche i **seminari** che stanno attualmente frequentando e, per ogni giorno, i **luoghi** e le ore dove sono tenute le lezioni. I **corsi** hanno un codice, un titolo e possono avere varie **edizioni** con date di inizio e fine e numero di partecipanti. Se gli **studenti** sono liberi professionisti, vogliamo conoscere l'area di interesse e, se lo possiedono, il titolo. Per quelli che lavorano alle dipendenze di altri, vogliamo conoscere invece il loro livello e la posizione ricoperta.

Per gli insegnanti (circa 300), rappresentiamo il cognome, l'età, il **posto** dove sono nati, il nome del corso che insegnano, quelli che hanno insegnato nel passato e quelli che possono insegnare. Rappresentiamo anche tutti i loro **recapiti telefonici**. I docenti possono essere dipendenti interni della società o collaboratori esterni.

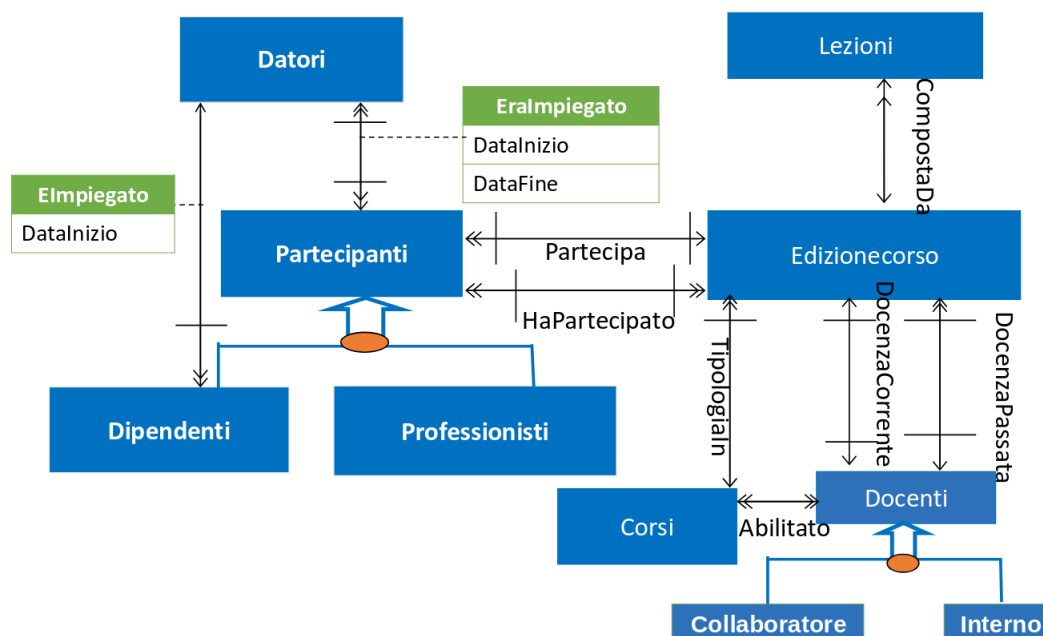
Per prima cosa abbiamo evidenziato dello stesso colore i **sinonimi** e le parole utilizzate in più **contesti** diversi in modo da poterli unificare e costruire il glossario.

Termine	Descrizione	Sinonimi	Collegamenti
Partecipante	Persona che partecipa ai corsi	Studente	Corso, società
Docente	Docente dei corsi. Può essere esterno	Insegnante	Corso
Corso	Corso organizzato dalla società. Può avere più edizioni.	Seminario	Docente
Datori	Ente presso cui i partecipanti lavorano o hanno lavorato.	Posti	Partecipante

Procediamo poi trovando gli **attributi** per ogni classe.

Partecipanti	Docenti	Corsi	DatoriLavoro
Codice	CodiceFiscale	Codice	CodiceFiscale
CodiceFiscale	Cognome	Titolo	Cognome
Cognome	Nome		Nome
Nome	Genere		Indirizzo
Genere	CittaNascita		Citta
CittaNascita	DataNascita		Telefono
DataNascita	Recapiti		

Infine definiamo le **relazioni** tra le classi.



4 Modello relazionale

Il modello relazionale fu presentato da E. F. Codd nel 1970 per favorire l'**indipendenza** dei dati. Si basa sul concetto di **relazione**, la quale ha come rappresentazione la **tabella**.

In ogni base di dati distinguiamo lo **schema relazionale**, invariato nel tempo e che descrive la struttura) e l'**istanza**, ovvero i valori attuali.

4.1 Matematica

In matematica definiamo una **relazione** come l'insieme dei **domini**

$$D_1, \dots, D_n$$

Inoltre il **prodotto cartesiano** $D_1 \times \dots \times D_n$ è l'insieme di tutte le **n-uple** (d_1, \dots, d_n) tali che

$$d_1 \in D_1, \dots, d_n \in D_n$$

Esempio 4.1.1 (Relazioni in matematica). Dati i domini

$$D_1 = \{a, b\} \quad D_2 = \{x, y, z\}$$

il prodotto cartesiano è

$$D_1 \times D_2 = \{(a, x), (a, y), (a, z), (b, x), (b, y), (b, z)\}$$

mentre una relazione possibile è

$$r \subseteq D_1 \times D_2 = \{(a, x), (a, z), (b, y)\}$$

4.2 Modello

I meccanismi per definire una BD con il modello relazionale sono la **ennupla** (insieme finito di coppie (Attributo, Tipo elementare)) e la **relazione**.

Definizione 4.2.1 (Schema di relazione). *Uno schema di relazione $R(T)$ è una coppia formata da un nome R e da un tipo T definito come segue:*

- *int, real, boolean e string sono tipi **primitivi***
- *$T = (A_1 : T_1, \dots, A_n : T_n)$ è un tipo **ennupla** di **grado** n se T_1, \dots, T_n sono tutti tipi primitivi e se A_1, \dots, A_n sono etichette distinte dette **attributi***
- *Due ennuple sono uguali se hanno uguale il grado, gli attributi e il tipo degli attributi con lo stesso nome*
- *L'ordine degli attributi non importa*
- *Se T è tipo ennupla allora $\{T\}$ è un insieme di ennuple o tipo **relazione***
- *Due tipi relazione sono uguali se hanno lo stesso tipo ennupla*

Definizione 4.2.2 (Schema relazionale). *Uno schema relazionale è costituito da schemi di relazione $R_i : \{T_i\}$ $i = 1, \dots, k$ e da un insieme di relativi **vincoli di integrità**.*

Definizione 4.2.3 (Ennupla). *Un'ennupla $t = (A_1 : V_1, \dots, A_n : V_n)$ di tipo $T = (A_1 : T_1, \dots, A_n : T_n)$ è un insieme di coppie (A_i, V_i) con V_i di tipo T_i . Due ennuple sono uguali se hanno lo stesso insieme di coppie.*

Definizione 4.2.4 (Istanza). *Un'istanza dello schema $R_i : \{T_i\}$ o **relazione** è un insieme finito di ennuple $\{t_1, t_2, \dots, t_k\}$, con t_i di tipo T_i . La sua **cardinalità** è il numero delle sue ennuple. L'istanza di uno schema relazionale è formata da un'istanza di ogni suo schema di relazione.*

4.2.1 Tabella

Una tabella rappresenta una **relazione** se:

- I valori di ogni colonna sono fra loro omogenei
- Le righe sono diverse fra loro
- Le intestazioni delle colonne sono diverse tra loro

L'ordinamento di righe e colonne non è importante.

Studenti	Nome	Matricola	Provincia	AnnoNascita	Schema di relazione Istanza di Relazione o estensione della relazione
	Isaia	071523	PI	1982	
	Rossi	067459	LU	1984	
	Bianchi	079856	LI	1983	
	Bonini	075649	PI	1984	

4.3 Valori

Il modello relazionale è basato su **valori**, ovvero i riferimenti fra i dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle ennuple. Questo permette di mantenere **indipendenza** dalle strutture fisiche, si rappresenta solo i dati necessari e si garantisce **portabilità**.

Definizione 4.3.1 (Valore nullo). *Il valore nullo denota l'assenza di un valore del dominio.*

Definizione 4.3.2 (Valore). *$t[A]$, per ogni attributo A , è un valore nel dominio $\text{dom}(A)$ oppure il valore nullo $NULL$.*

4.4 Vincoli di integrità

Uno schema relazionale è costituito da un insieme di **scemi di relazione** e da un insieme di **vincoli d'integrità** su i possibili valori delle estensioni delle relazioni. Questi ultimi permettono di descrivere più accuratamente la realtà migliorando la **qualità dei dati** e aiutando nella progettazione

Definizione 4.4.1 (Vincolo d'integrità). *Un vincolo d'integrità è una proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione. È espresso mediante una funzione booleana che associa ad ogni istanza il valore vero o falso.*

Esistono due tipi di vincoli:

- **Intrarelazionali:** devono essere rispettati dai valori contenuti nella relazione considerata. Possono essere sui valori o sulle ennuple.
- **Interrelazionali:** devono essere rispettati da valori contenuti in relazioni diverse

4.4.1 Vincoli di ennupla

Questi vincoli esprimono condizioni sui valori di ciascuna ennupla, indipendentemente dalle altre. Quando coinvolgono un solo attributo sono chiamati **vincoli di dominio**.

4.5 Chiave

Una chiave è un insieme di attributi che identificano le ennuple di una relazione.

Definizione 4.5.1 (Chiave). *Un insieme K di attributi per uno schema di relazione r è:*

- **Superchiave** se r non contiene due ennuple distinte t_1 e t_2 con $t_1[K] = t_2[K]$
- **Chiave** se è una superchiave **minimale**, cioè se non contiene un'altra superchiave

Definizione 4.5.2 (Chiave primaria). *La chiave primaria di uno schema di relazione è una delle chiavi, di solito quella con il minor numero di attributi. Non ammette valori nulli ed è indicata con la sottolineatura.*

<u>Matricola</u>	Cognome	Nome	Corso	Nascita
86765	NULL	Mario	Ing Inf	5/12/78
78763	Rossi	Mario	Ing Civile	3/11/76
65432	Neri	Piero	Ing Mecc	10/7/79
87654	Neri	Mario	Ing Inf	NULL
43289	Neri	Mario	NULL	5/12/78

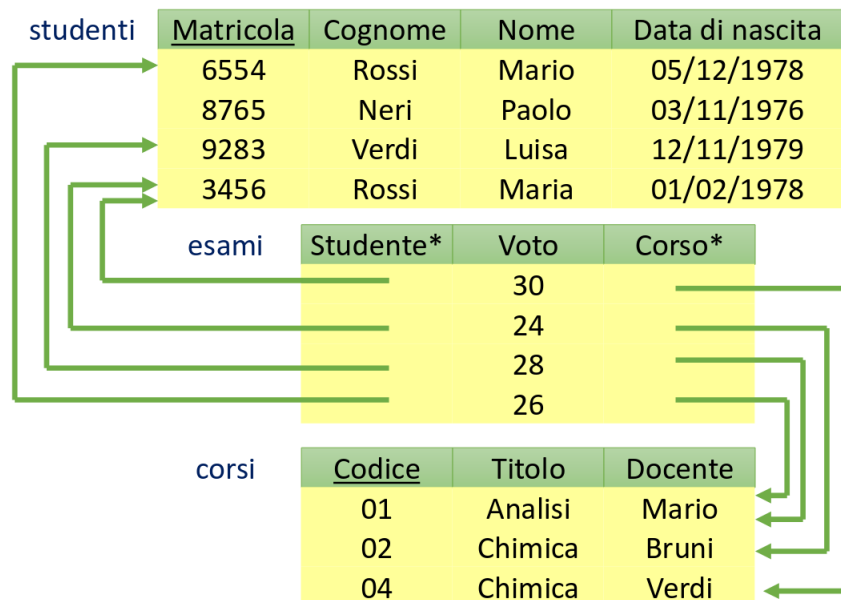
Note 4.5.1. È possibile che esistano degli insiemi di attributi che soddisfino casualmente tutti i vincoli per essere chiavi, ma questo deve succedere **sempre** per tutte le istanze.

Osservazione 4.5.1 (Esistenza). Ogni relazione ha come superchiave l'insieme di tutti gli attributi su cui è definita, quindi ha sempre almeno una chiave.

4.6 Integrità referenziale

Nel modello relazionale le informazioni in relazioni sono correlate attraverso **valori** comuni, spesso quelli delle **chiavi primarie**. Deve quindi esserci una **coerenza**.

Gli attributi che permettono le correlazioni sono indicati sia con la sottolineatura (**chiavi esterne**) che con l'asterisco.



4.6.1 Integrità referenziale

Definizione 4.6.1 (Vincolo di integrità referenziale). *Un vincolo di integrità referenziale (**foreign key**) 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 .*

In caso di violazione del vincolo di integrità (e.g. viene eliminata una ennupla dalla tabella riferita), è possibile:

- Rifiutare l'operazione
- Eliminare in cascata nelle altre tabelle
- Introdurre valori nulli

Note 4.6.1. È possibile avere più chiavi esterne e in questo caso ci sono **vincoli multipli**.

5 Trasformazione di schemi

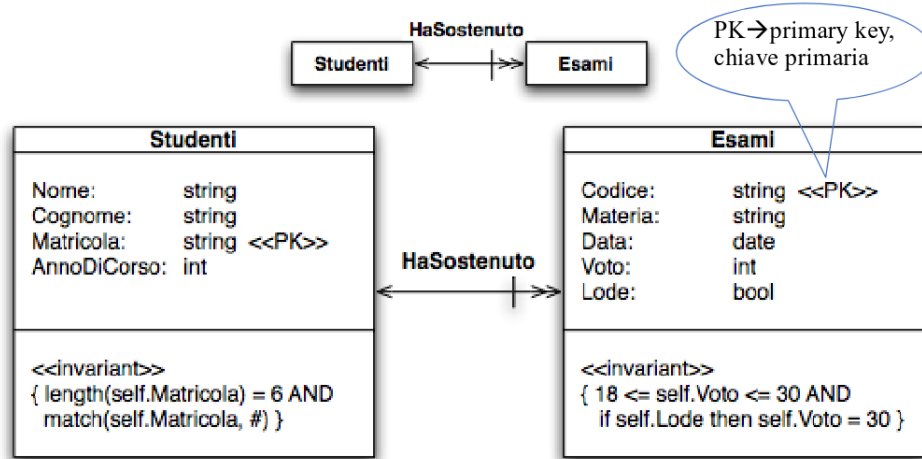
L'obiettivo della trasformazione è quello di ottenere da uno schema concettuale uno schema **logico-relazionale** che rappresenti gli stessi dati in maniera **corretta** ed **efficiente**, riducendo la ridondanza e facilitandone la comprensione.

Questa trasformazione prende in **ingresso** lo schema relazionale, il carico applicativo e il modello logico e restituisce in **uscita** uno schema logico e la documentazione.

Si compone dei seguenti passi:

1. Rappresentazione delle associazioni uno ad uno e uno a molti
2. Rappresentazione delle associazioni molti a molti o non binarie
3. Rappresentazione delle gerarchie di inclusione
4. Identificazione delle chiavi primarie
5. Rappresentazione degli attributi multivalore
6. Appiattimento degli attributi composti

Esempio 5.0.1 (Esempio di trasformazione). Dato il seguente schema concettuale



per ottenere uno schema logico introduciamo l'attributo **matricola**. Questo ci porta ad avere due relazioni collegate dal nuovo attributo.

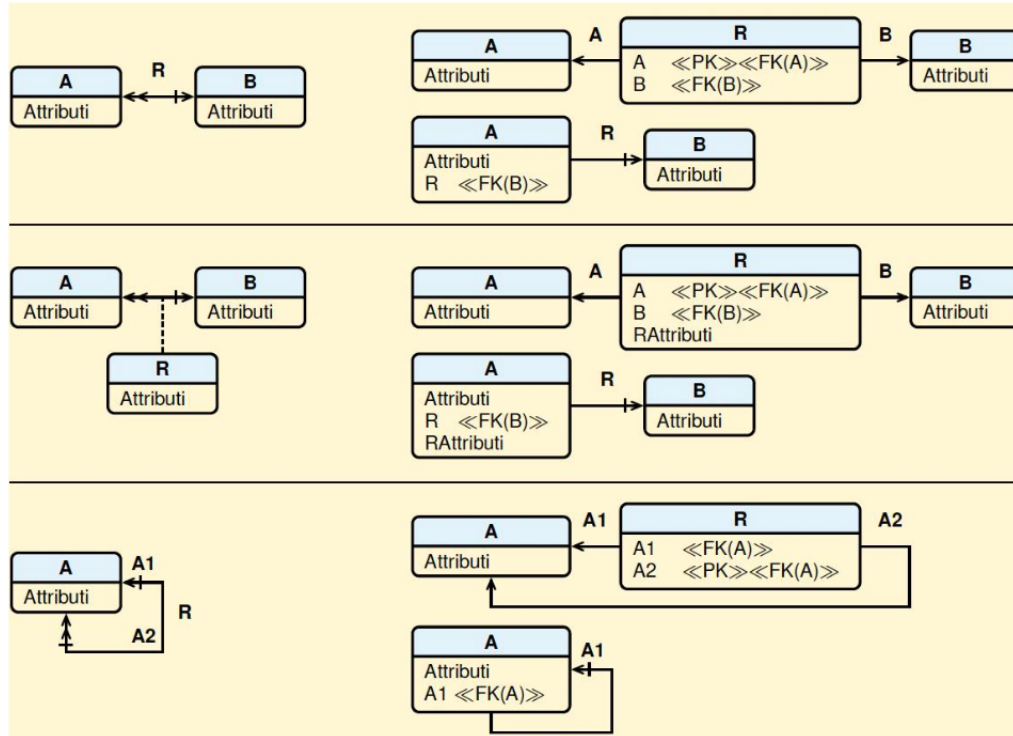
Nome	<u>Matricola</u>	Provincia	AnnoNascita
Isaia	071523	PI	1982
Rossi	067459	LU	1984
Bianchi	079856	LI	1983
Bonini	075649	PI	1984

<u>Materia</u>	<u>Candidato*</u>	Data	Voto
BD	071523	12/01/06	28
BD	067459	15/09/06	30
FP	079856	25/10/06	30
BD	075649	27/06/06	25
LMM	071523	10/10/06	18

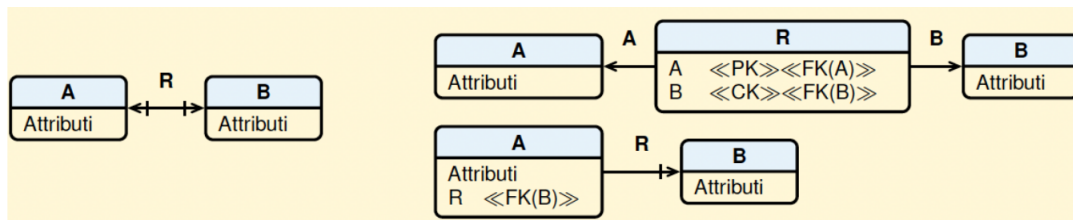
5.1 Rappresentazioni

5.1.1 Uno a molti/uno

Uno a molti Le associazioni uno a molti si rappresentano aggiungendo agli attributi della relazione rispetto a cui l'associazione è univoca una **chiave esterna** che riferisce l'altra relazione. Se l'associazione ha degli attributi, questi si aggiungono alla relazione in cui è presente la chiave esterna.



Uno ad uno Le associazioni uno a uno si rappresentano aggiungendo la **chiave esterna** ad una qualunque delle due relazioni che riferisce l'altra relazione, preferendo quella rispetto a cui l'associazione è totale, nel caso in cui esista un vincolo di totalità.



Vincoli

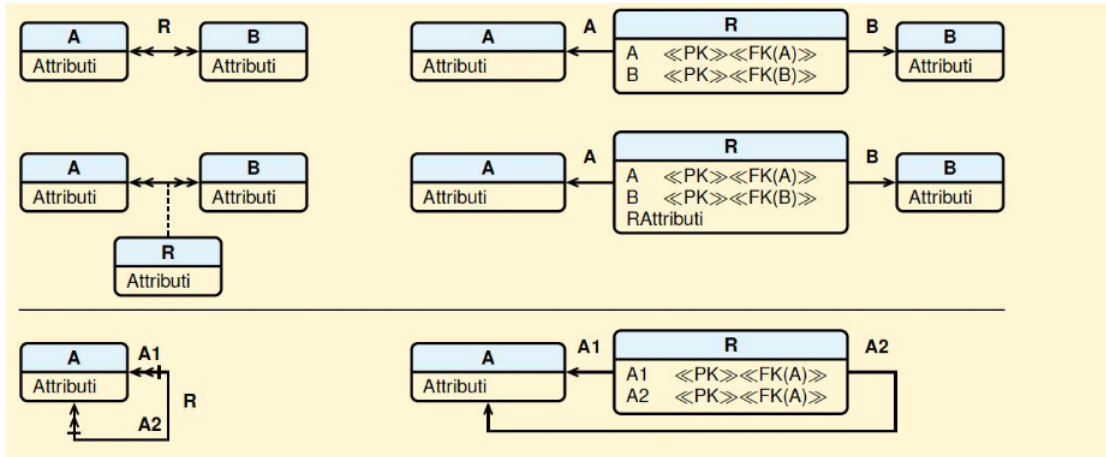
Definizione 5.1.1 (Diretta). La direzione dell'associazione rappresentata dalla chiave esterna è detta la **diretta** dell'associazione.

I vincoli sulla **cardinalità** delle associazioni **uno ad uno** e **uno a molti** sono:

- **Univocità** della diretta
- **Totalità** della diretta: vincolo *not null* sulla chiave esterna
- **Univocità** dell'inversa e **totalità** della diretta: vincolo *not null* e *di chiave* sulla chiave esterna

5.1.2 Molti a molti

Un'associazione molti a molti si rappresenta aggiungendo allo schema una nuova relazione che contiene due chiavi esterne che riferiscono le due relazioni coinvolte; la chiave primaria di questa relazione è costituita dall'insieme di tutti i suoi attributi.



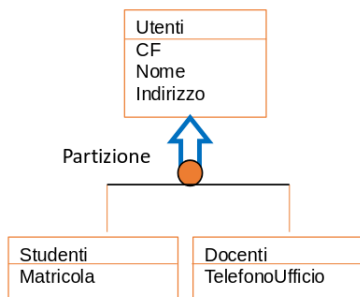
5.1.3 Gerarchie tra classi

Il modello relazionale non può rappresentare le gerarchie tra classi. Bisogna quindi eliminarle e sostituirle con classi e relazioni usando le seguenti tecniche:

- **Relazione unica:** accorpamento delle figlie nel genitore
- **Partizionamento orizzontale:** accorpamento del genitore nelle figlie
- **Partizionamento verticale:** sostituzione della gerarchia con relazioni

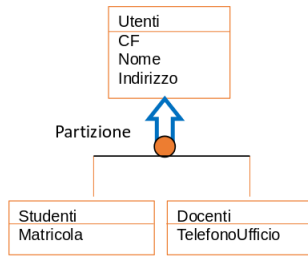
Relazione unica Si seguono i seguenti passi:

1. Se A_0 è la classe genitore di A_1 ed A_2 , queste ultime vengono eliminate ed accorpate alla prima
2. Ad A_0 viene aggiunto un attributo (**discriminatore**) che indica da quale delle classi figlie deriva una certa istanza, e gli attributi di A_1 ed A_2 vengono assorbiti dalla classe genitore, e assumono valore nullo sulle istanze provenienti dall'altra classe
3. Infine, una relazione relativa a solo una delle classi figlie viene acquisita dalla classe genitore e avrà comunque cardinalità minima uguale a 0, in quanto gli elementi dell'altra classe non contribuiscono alla relazione.



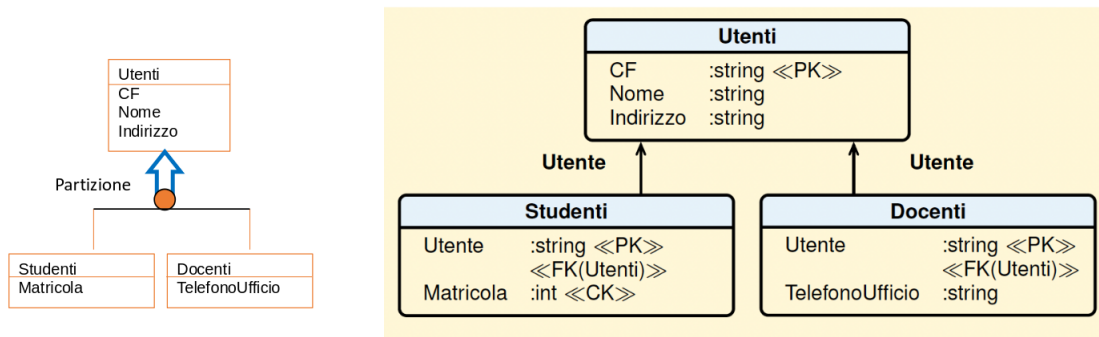
Utenti	
CF	:string <<PK>>
Nome	:string
Indirizzo	:string
Matricola	:int
TelefonoUfficio	:int
Discriminatore	:(Studente; Docente)

Partizionamento orizzontale Se A_0 è la classe genitore di A_1 ed A_2 , la classe genitore viene eliminata, e le classi figlie ereditano le proprietà (attributi, identificatore e relazioni) della classe genitore.



Note 5.1.1. Questa tecnica divide gli elementi della superclasse A_0 in più relazioni diverse, per cui non è possibile mantenere un vincolo referenziale verso A_0 . In conclusione, questa tecnica non si usa se nello schema relazionale c'è una associazione diretta verso A_0 , ovvero che entra nella superclasse.

Partizionamento verticale In questo caso non c'è un trasferimento di attributi o di associazioni e le classi figlie A_1 ed A_2 sono identificate esternamente dalla classe genitore A_0 . Nello schema ottenuto vanno aggiunti dei vincoli: ogni occorrenza di A_0 non può partecipare contemporaneamente alle due associazioni, e se la gerarchia è totale, deve partecipare ad almeno una delle due.



5.1.4 Chiavi primarie

È necessario definire per ogni relazione un insieme di attributi che funga da chiave primaria, seguendo questi passi:

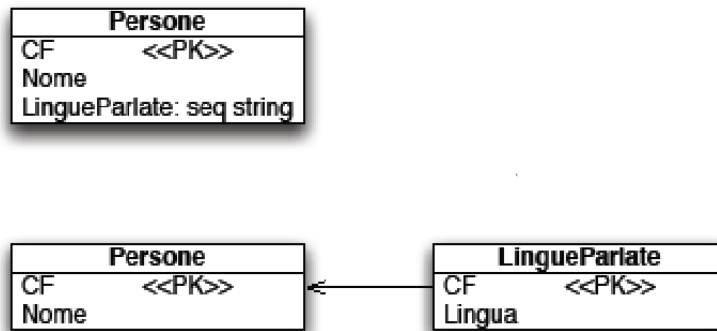
1. Si considerano le relazioni che corrispondono a classi dello schema originale che erano prive di superclassi (**classi radice**). La chiave primaria è di norma un attributo artificiale, tipicamente un numero progressivo assegnato dal sistema. E' possibile utilizzare un attributo presente nella classe, purché l'attributo sia **univoco**, **totale** e **costante**.
2. Per ogni relazione dello schema che corrisponde ad una **sottoclasse** dello schema originario, la chiave primaria sarà la stessa della superclasse.
3. Per le relazioni che corrispondono ad $N : M$ nello schema originario, la chiave primaria sarà costituita dalla concatenazione delle chiavi esterne.

5.1.5 Attributi multivalore

Una proprietà multivalore di una classe C si rappresenta eliminando il corrispondente attributo da C e creando una nuova relazione N con una chiave di due attributi:

- una **chiave esterna** che fa riferimento alla chiave primaria di C
- un **attributo** che corrisponde all'attributo multivalore da trasformare

Un oggetto di C con chiave primaria k ed in cui l'attributo assume valore a_1, \dots, a_n si rappresenta poi inserendo nella relazione N le coppie $(k, a_1), \dots, (k, a_n)$



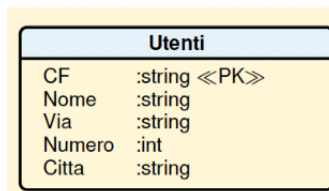
5.1.6 Attributi composti

Se un attributo A_i di uno schema di relazione è di tipo $[A_{i1} : T_{i1}, \dots, A_{ij} : T_{ij}]$, si sostituisce con gli attributi $A_{i1} : T_{i1}, \dots, A_{ij} : T_{ij}$. Se A_i faceva parte della chiave primaria dello schema di relazione, si sostituisce A_i con gli attributi A_{i1}, \dots, A_{ij} nella chiave, e poi si verifica che non esista un sottoinsieme degli attributi della nuova chiave primaria che è esso stesso una chiave.

Esempio 5.1.1. Dato l'attributo composto

[Via : string, Numero : int, Citta : string]

otteniamo



6 Algebra relazionale

L'**algebra relazionale** è l'insieme degli operatori su relazioni che danno come risultato relazioni. Viene usato come rappresentazione interna delle interrogazioni e non come linguaggio di interrogazione dei DBMS.

Il **calcolo relazionale** è invece il linguaggio dichiarativo di tipo logico dal quale è stato derivato SQL.

6.1 Notazione

Nomi di relazioni R, S, \dots

Nomi di attributi A, B, C, A_1, A_2, \dots

Insiemi di attributi X, Y, X_1, \dots

Unione di insiemi di attributi $X \cup Y = XY$

Relazione con ennuple Date le ennuple t_1, t_2, \dots, t_n la relazione è denotata con $\{t_1, t_2, \dots, t_n\}$

Relazione vuota $\{\}$

Valore attributo Data la ennupla t_k , il suo valore A_i è denotato da $t_k.A_i$

Ennupla specifica ad attributi Se X è sottoinsieme degli attributi di t , $t.X$ o $t[X]$ allora denota l'ennupla ottenuta da t considerando solo gli attributi in X

Ambiguità Se R ed S hanno lo stesso attributo A_j , in caso di ambiguità, $R.A_j$ denota l'attributo A_j della relazione R ed $S.A_j$ denota l'attributo A_j della relazione S .

6.2 Operatori primitivi

6.2.1 Ridenominazione

Viene utilizzato per cambiare il nome di una relazione e di conseguenza anche il suo **tipo**.

$$\rho_{A \leftarrow B}(R) \quad (1)$$

$$T' = \rho_{A \leftarrow A'}(T) \quad (2)$$

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

Table 1: Laureati

6.2.2 Unione

Restituisce la relazione ottenuta facendo l'unione delle ennuple di R con quelle di S dove R e S sono relazioni dello stesso tipo.

$$R \cup S \quad (3)$$

<u>A</u>	<u>B</u>	<u>C</u>		<u>A</u>	<u>B</u>	<u>C</u>		A	B	C
a1	b1	c1	∪	a1	b1	c1	→	a1	b1	c1
a1	b1	c2		a1	b2	c2		a1	b1	c2
a2	b1	c1						a1	b2	c2
a3	b1	c1						a2	b1	c1
								a3	b1	c1

6.2.3 Differenza

Restituisce la relazione contenente le ennuple di R non presenti in S .

$$R - S \quad (4)$$

<u>A</u>	<u>B</u>	<u>C</u>		<u>A</u>	<u>B</u>	<u>C</u>		A	B	C
a1	b1	c1	−	a1	b1	c1	→	a1	b1	c2
a1	b1	c2		a1	b2	c2		a2	b1	c1
a2	b1	c1						a3	b1	c1
a3	b1	c1								

6.2.4 Proiezione

Restituisce una relazione i cui elementi sono la copia delle ennuple di R proiettate (ristrette) sugli attributi A_1, \dots, A_n . Eventuali ennuple che dopo la proiezione sono uguali appaiono solo una volta.

$$\pi_{A_1, \dots, A_n}(R) \quad (5)$$

Data la tabella usata negli esempi precedenti, la proiezione è:

$$\pi_A(R) =$$

A
a1
a2
a3

Cardinalità Una proiezione conterrà al più tante ennuple quanto l'operando:

- Se X è una **superchiave** di R , allora $\pi_X(R)$ contiene esattamente tante ennuple quante R
- **Altrimenti** potrebbero esistere valori ripetuti su quegli attributi, che quindi vengono rappresentati una sola volta

6.2.5 Restrizione o selezione

Restituisce una relazione dello stesso tipo (schema) di R i cui elementi sono la copia delle ennuple di R (un sottoinsieme) che soddisfano la condizione ϕ definita come:

- $A_i \phi A_j$ con A_i e A_j attributi di R e θ un operatore di confronto $\{<, >, =, \neq, \leq, \geq\}$
- $A_i \theta c$ oppure $c \theta A_i$, con θ operatore di confronto e c costante nel dominio di A_i
- Se ϕ e ψ sono formule, allora lo sono anche $\phi \wedge \psi$, $\phi \vee \psi$ e $\neg \psi$

$$\sigma_{\phi}(R) \quad (6)$$

Data la tabella usata negli esempi precedenti abbiamo

$$\sigma_{A=a_1}(R) =$$

A	B	C
a1	b1	c1
a1	b1	c2

Osservazione 6.2.1 (Valori nulli). La presenza di valori **nulli** negli attributi usati per la restrizione portano all'assenza di **atomicità** nell'operazione.

$$\sigma_{\text{Età} > 30}(\text{Persone}) \cup \sigma_{\text{Età} \leq 30}(\text{Persone}) \neq \text{Persone}$$

Per mantenerla è quindi necessario utilizzare i costrutti *IS NULL* e *IS NOT NULL*:

$$\sigma_{\text{Età} > 30}(\text{Persone}) \cup \sigma_{\text{Età} \leq 30}(\text{Persone}) \cup \sigma_{\text{Età IS NULL}}(\text{Persone}) = \text{Persone}$$

6.2.6 Prodotto

Con R e S relazioni con attributi distinti, il loro prodotto è una relazione con elementi ottenuti concatenando ogni ennupla di R con ogni ennupla di S . La relazione risultante ha grado uguale alla somma dei gradi degli operandi e cardinalità uguale al prodotto delle cardinalità degli operandi.

$$R \times S \quad (7)$$

<u>A</u>	<u>B</u>	<u>C</u>		<u>A'</u>	<u>D</u>	
a1	b1	c1		a1	d1	
a1	b1	c2		a1	d1	
a2	b1	c1		a1	d1	
a3	b1	c1		a2	d2	

 \times

<u>A'</u>	<u>D</u>
a1	d1
a2	d2

 \longrightarrow

A	B	C	A'	D
a1	b1	c1	a1	d1
a1	b1	c1	a2	d2
a1	b1	c2	a1	d1
a1	b1	c2	a2	d2
a2	b1	c1	a1	d1
a2	b1	c1	a2	d2
a3	b1	c1	a1	d1
a3	b1	c1	a2	d2

Osservazione 6.2.2. Il prodotto di due relazioni è un'operazione che di solito non si usa da sola. Pertanto per concatenare ennuple in associazione, si restringe sempre il prodotto alle ennuple con valore uguale della chiave esterna e chiave primaria. Per questa si introduce un operatore derivato, chiamato **giunzione**, per riferirsi alla combinazione di queste due operazioni.

6.3 Operatori derivati

6.3.1 Intersezione

Restituisce la relazione ottenuta facendo l'intersezione delle ennuple di R con quelle di S dove R e S sono relazioni dello stesso tipo.

$$R \cap S = R - (R - S) \quad (8)$$

<u>A</u>	<u>B</u>	<u>C</u>		<u>A</u>	<u>B</u>	<u>C</u>		<u>A</u>	<u>B</u>	<u>C</u>
a1	b1	c1		a1	b1	c1		a1	b1	c1
a1	b1	c2		a1	b1	c1				
a2	b1	c1		a1	b2	c2				
a3	b1	c1								

 \cap

<u>A</u>	<u>B</u>	<u>C</u>
a1	b1	c1
a1	b2	c2

 \longrightarrow

A	B	C
a1	b1	c1

6.3.2 Inner Join

Restituisce la relazione contenente le ennuple del prodotto cartesiano di $R \times S$ con valori uguali per gli attributi A_i e A_j dove R e S sono relazioni di tipo diverso (con attributi distinti) tra i quali c'è A_i in R e A_j in S .

$$R \bowtie_{A_i=A_j} S = \sigma_{A_i=A_j}(R \times S) \quad (9)$$

Date le tabelle usate come esempio nel prodotto (R e T'), abbiamo

$$R \bowtie_{A=A'} T' =$$

A	B	C	A'	D
a1	b1	c1	a1	d1
a1	b1	c2	a1	d1
a2	b1	c1	a2	d2

6.3.3 Theta Join

È il caso più generale della *Inner Join*: qui la condizione θ è una congiunzione di termini logici di confronto ($>$, $<$, $=$, \dots).

$$R \bowtie_{\sigma} S \quad (10)$$

6.3.4 Natural Join

È una abbreviazione dell'*Inner Join* applicata a relazioni in cui l'associazione fra le ennuple è descritta con la chiave esterna e la chiave primaria costituite da attributi uguali. Restituisce la relazione contenente le ennuple di $R \times S$ con gli attributi di uguali nomi in R ed S definiti sugli stessi domini.

$$R \bowtie S = R \bowtie_{A=A} S = \sigma_{A=A}(R \times S) \quad (11)$$

<u>A</u>	<u>B</u>	<u>C</u>		<u>A</u>	<u>D</u>		A	B	C	D
a1	b1	c1		a1	d1		a1	b1	c1	d1
a1	b1	c2	\bowtie	a2	d2	\longrightarrow	a1	b1	c2	d1
a2	b1	c1					a2	b1	c1	d2
a3	b1	c1								

Note 6.3.1. Si noti che:

- Se R ed S non hanno attributi comuni allora $R \bowtie S = R \times S$
- Se R ed S hanno lo stesso schema allora $R \bowtie S = R \cap S$

Cardinalità Date le relazioni $R_1(A, B)$ e $R_2(B, C)$:

- Il numero di ennuple della loro join è:

$$0 \leq |R_1 \bowtie R_2| \leq |R_1| \times |R_2|$$

- Se il loro join è **completo** allora contiene un numero di ennuple almeno uguale al massimo tra $|R_1|$ e $|R_2|$
- Se il join coinvolge una **chiave** B di R_2 allora:

$$0 \leq |R_1 \bowtie R_2| \leq |R_1|$$

- Se il join coinvolge una **chiave** B di R_2 e un **vincolo di integrità referenziale** tra gli attributi di R_1 e la chiave di R_2 allora:

$$|R_1 \bowtie R_2| = |R_1|$$

Osservazione 6.3.1 (Natural Join e Proiezione). Osserviamo che valgono due cose:

- Dati $R_1(X_1)$ e $R_2(X_2)$ vale

$$\pi_{X_1}(R_1 \bowtie R_2) \subseteq R_1 \quad (12)$$

- Dati $R(X)$ con $X = X_1 \cup X_2$ vale

$$R \subseteq (\pi_{X_1}(R)) \bowtie (\pi_{X_2}(R)) \quad (13)$$

6.3.5 Semi Join

Restituisce le ennuple di R che partecipano alla giunzione naturale di R ed S con A_1, A_2, \dots, A_m attributi di R .

$$RS = \pi_{A_1, A_2, \dots, A_m}(R \bowtie S) \quad (14)$$

Considerando l'esempio precedente abbiamo

$$RT =$$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1

6.3.6 External Join

Il join esterno estende con valori nulli le ennuple che verrebbero tagliate fuori da un join interno. Ne esistono tre versioni:

- **Left:** mantiene tutte le ennuple del primo operando, estendendole con valori nulli se necessario

$$R \overset{\leftarrow}{\bowtie} S \quad (15)$$

- **Right:** come left ma con il secondo operando

$$R \overset{\rightarrow}{\bowtie} S \quad (16)$$

- **Full:** come left ma con entrambi gli operandi

$$R \overset{\leftrightarrow}{\bowtie} S \quad (17)$$

6.3.7 Self Join

Questa operazione è utilizzata quando va fatta una join di una tabella con se stessa. Spesso è importante fare anche delle ridenominazioni.

Esempio 6.3.1 (Nonno nipote). Data la relazione effettuiamo le seguenti operazioni

Genitore	Figlio
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

$$\rho_{\text{Genitore, Figlio} \leftarrow \text{Nonno, Genitore}}(\text{Genitore}) \bowtie \rho_{\text{Figlio} \leftarrow \text{Nipote}}(\text{Genitore})$$

Nonno	Genitore
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

\bowtie

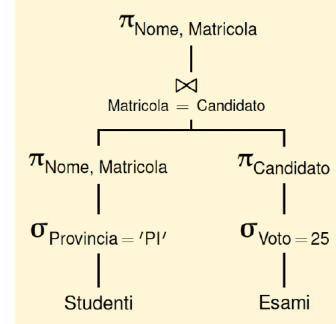
Genitore	Nipote
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

$=$

Nonno	Genitore	Nipote
Giorgio	Luca	Anna
Silvia	Maria	Anna
Enzo	Maria	Anna

6.4 Proprietà algebriche degli operatori

Un'espressione algebrica può essere rappresentata come un **albero**:

$$\pi_{\text{Nome, Matricola}}(\pi_{\text{Nome, Matricola}}(\sigma_{\text{Provincia}='PI'}(\text{Studenti})) \bowtie_{\text{Matricola}=\text{Candidato}} \pi_{\text{Candidato}}(\sigma_{\text{Voto}=25}(\text{Esami})))$$


Un'espressione dell'algebra relazionale può essere trasformata in un'altra equivalente sfruttando alcune proprietà degli operatori. Questo aiuta a ridurre il costo di esecuzione. Le proprietà più utili sono quelle che permettono di anticipare la **restrizione** e la **proiezione**:

- **Raggruppamento di restrizioni**

$$\sigma_{C_1}(\sigma_{C_2}(R)) \equiv \sigma_{C_1 \wedge C_2}(R)$$

- **Raggruppamento di proiezioni**

$$\sigma_{C_1 \wedge C_2}(R \times S) \equiv \sigma_{C_1}(R) \times \sigma_{C_2}(S)$$

- **Commutatività della restrizione, proiezione, prodotto, giunzione e degli operatori insiemistici**

$$(R \times S) \equiv (S \times R)$$

- **Anticipazione della restrizione** rispetto al prodotto e alla giunzione e rispetto agli operatori insiemistici
- **Anticipazione della proiezione** rispetto al prodotto e alla giunzione e rispetto agli operatori insiemistici
- Eliminazioni di proiezioni superflue

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

Osservazione 6.4.1 (Non distributività della proiezione rispetto alla differenza). In generale vale che

$$\pi_A(R_1 - R_2) \neq \pi_A(R_1) - \pi_A(R_2) \quad (18)$$

6.5 Quantificatori

Esistono tre tipi di quantificatori:

- **Esistenziale:** e.g. tutti gli studenti che hanno preso *almeno* un voto maggiore di 28
- **Differenza:** e.g. tutti gli studenti che NON hanno mai preso un voto maggiore di 28
- **Universale:** e.g tutti gli studenti che hanno preso SOLO voti maggiori di 28

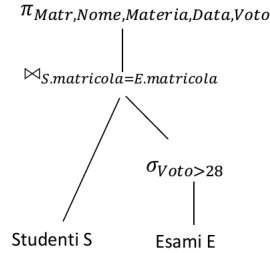


Figure 1: Esistenziale

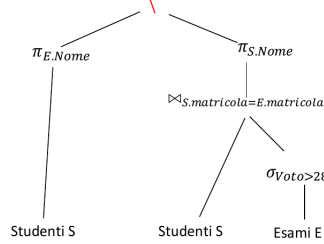


Figure 2: Differenza

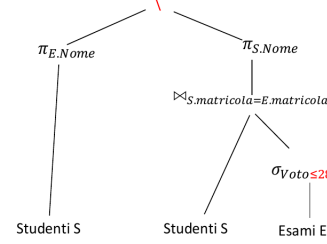


Figure 3: Universale

6.6 Operatori non insiemistici

6.6.1 Group By

Data una relazione R , i suoi attributi A_i e le espressioni che usano funzioni di aggregazione f_i (min, max, count, sum, ...), definiamo il raggruppamento come una relazione calcolata:

1. Partizionando le ennuple di R mettendo nello stesso gruppo tutte le ennuple con valori uguali degli A_i
2. Si calcolano le espressioni f_i per ogni gruppo
3. Per ogni gruppo si restituisce una sola ennupla con come attributi i valori degli A_i e delle espressioni f_i

$$\{A_i\}_{\gamma\{f_i\}}(R) \quad (19)$$

Il raggruppamento gode dell'**anticipazione** rispetto alla proiezione:

$$\sigma_C(X_{\gamma F}(R)) \equiv X_{\gamma F}(\sigma_C(R)) \quad (20)$$

Esempio 6.6.1. Vogliamo ottenere per ogni candidato il numero degli esami, il voto minimo, massimo e medio:

$$\{\text{Candidato}\}_{\gamma\{\text{count}(*), \text{min}(\text{Voto}), \text{max}(\text{Voto}), \text{avg}(\text{Voto})\}}(\text{Esami})$$

Materia	Candidato	Voto	Docente
DA	1	20	10
LFC	2	30	20
MTI	1	30	30
LP	2	20	40

↓

Candidato	Count(*)	min(Voto)	max(Voto)	avg(Voto)
1	2	20	24	22
2	2	20	30	25

6.6.2 Proiezione generalizzata

Estende la proiezione con la possibilità di usare costanti o espressioni aritmetiche nella lista degli attributi. Permette anche di aggiungere **etichette** alle espressioni tramite l'operatore **AS**:

$$\pi_{e_1 \text{ AS } ide_1, e_2 \text{ AS } ide_2, \dots}(R) \quad (21)$$

Dove e_1, e_2, \dots sono espressioni aritmetiche ottenute a partire da costanti e ide_1, ide_2, \dots sono etichette distinte.

6.6.3 Proiezione multi-insiemistica

Funziona come la proiezione ma senza eliminazione dei duplicati. Restituisce dei multi-insiemi e quindi si possono utilizzare solo come radice di un albero logico.

$$\pi_{\{A_i\}}^b(R) \quad (22)$$

6.6.4 Ordinamento

Ordina tutte le ennuple di R in ordine crescente o decrescente rispetto agli attributi A_i .

$$\tau_{\{A_i\}}(R) \quad (23)$$

6.7 Calcolo relazionale

Il calcolo relazionale è un linguaggio che permette di definire il risultato di un'interrogazione (**query**) come l'insieme di quelle ennuple che soddisfano una certa condizione ϕ .

L'algebra dà la possibilità di scrivere espressioni in cui gli operatori sono applicati al risultato di altri operatori (espressioni annidate). Il calcolo ha una **struttura piatta** ma permette di esprimere condizioni più **complesse**.

Un linguaggio che si colloca a metà tra i due stili si può ottenere:

1. Aggiungendo al calcolo la possibilità di annidare il costruttore di insiemi
2. Aggiungendo all'algebra la possibilità di avere nell'operatore di restrizione condizioni che fanno uso anche di quantificatori e di predicati di appartenenza.

Il risultato è un linguaggio che ha sia la capacità di esprimere interrogazioni in modo annidato che la possibilità di esprimere condizioni logiche complesse, come accade nel linguaggio SQL.

Esempio 6.7.1. L'insieme delle matricole degli Studenti che hanno superato qualcuno degli esami elencati nella relazione Materie, si può definire come

$$\{t.\text{Matricola} \mid t \in \text{Studenti}, \exists m \in \text{Materie}. \exists e \in \text{ProveEsami}.$$

$$e.\text{Candidato} = t.\text{Matricola} \wedge e.\text{Materia} = m.\text{Materia}\}$$

che equivale a

$$\pi_{\text{Matricola}}(\text{Studenti} \bowtie_{\text{Matricola}=\text{Candidato}} (\text{ProveEsami} \bowtie \text{Materie}))$$