

Basi di Dati

Angelo Passarelli

October 17, 2023



Appunti basati sulle lezioni e dispense della professoressa Giovanna Rosone ¹

¹<https://pages.di.unipi.it/rosone/index.html>

Contenuti

0	Introduzione	3
1	DBMS e Linguaggi	4
1.1	Funzionalità del DBMS	5
1.2	Proprietà dei Database	6
1.3	Transazioni	6
2	Modellazione	7
2.1	Fasi della Modellazione	8
2.2	Analizzare il Dominio	8
2.3	Oggetti e Classi	9
2.4	Associazioni	10
2.5	Gerarchie	12
2.5.1	Tipi Oggetto	12
2.5.2	Classi	12
2.5.3	Ereditarietà Multipla	13
3	Progettazione Logica	14
3.1	Relazioni	14
3.2	Tabelle	14
3.3	Vincoli di Integrità	15
3.4	Chiavi	15
3.5	Rappresentazione delle Associazioni	16
3.6	Rappresentazione delle Gerarchie	18
3.7	Rappresentazione Campi Multivalore	19
4	Algebra Relazionale	19
4.1	Operatori Insiemistici	19
4.1.1	Join	21

0 Introduzione

Definizione (Base di Dati). Una base di dati è un insieme organizzato di dati utilizzati per il supporto allo svolgimento di attività.

Struttura dei Dati I dati sono organizzati in insiemi strutturati che possono presentare fra loro delle relazioni. Tuple che rappresentano dati nello stesso insieme devono essere omogenee ed univoche.

Definizione (Sistema Informativo). Un sistema informativo è una combinazione di risorse umane e/o materiali e procedure organizzate per la **raccolta**, l'**archiviazione**, l'**elaborazione** e lo **scambio** di informazioni necessarie ad un'attività, le quali possono essere classificate in:

- Informazioni di servizio (operative).
- Informazioni di controllo (pianificazione e gestione).
- Informazioni di governo (pianificazione strategica).

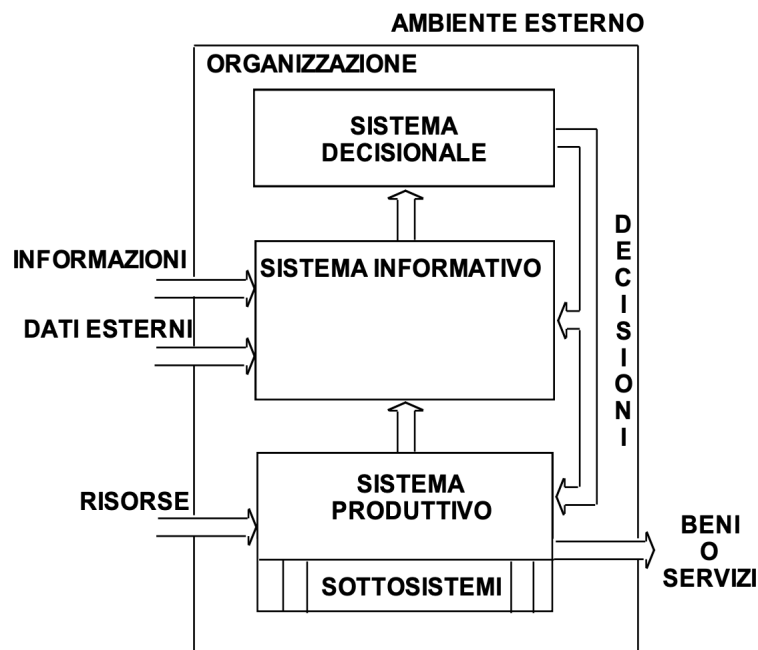


Figura 1: Esempio di sistema informativo

Definizione (Sistema Informativo Automatizzato). Un sistema informativo automatizzato è una parte del sistema informativo che permette di implementare le procedure che si occupano della gestione delle informazioni usando un sistema informatico.

Definizione (Sistema Informatico). Un sistema informatico è l'insieme delle tecnologie a supporto per le attività di un'organizzazione. Si possono classificare in:

- **Sistemi informatici operativi**: questi sistemi si utilizzano per svolgere le normali attività dell'azienda per la fruizione del suo bene o servizio, e per la gestione interna dei singoli reparti dell'azienda. Le operazioni sui dati in questo sistema sono di tipo **OLTP** (*On-Line Transaction Processing*) e prevedono elaborazioni semplici che coinvolgono pochi dati che vengono aggiornati molto frequentemente.
- **Sistemi informatici direzionali**: i dati sono organizzati in *Data Warehouse* che consentono di aiutare l'azienda nei processi di controllo delle prestazioni e di decisione manageriale. Le elaborazioni su questo tipo di sistema si chiamano **OLAP** (*On-Line Analytical Processing*) e prevedono l'utilizzo di una grande mole di dati che sono per lo più storici. In questo caso i dati vengono aggiornati molto raramente, ma su di essi vengono svolte molte operazioni, anche da un punto di vista multidimensionale, ovvero vengono incrociati più dati per analizzare le informazioni ottenute sotto molteplici punti di vista.

Definizione (DBMS). Un *Database Management System* è un sistema che garantisce il controllo e la gestione di dati per renderli accessibili agli utenti opportuni in base ai loro privilegi. Il DBMS fornisce anche dei linguaggi che permettono di definire lo **schema** di un database, di scegliere le **strutture dati** opportune per la memorizzazione dei dati, di rispettare i **vincoli** per ogni tipo di dato e di poter **modificare** e **interrogare** il database.

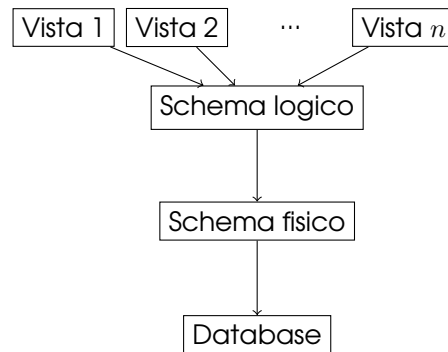
Metadati All'interno del database sono anche memorizzati dei metadati che si riferiscono agli utenti e allo schema utilizzato dal database stesso. Anche i metadati possono essere interrogati e modificati.

1 DBMS e Linguaggi

Si distinguono tre diversi livelli di descrizione dei dati:

- A livello di **vista logica**: descrive come deve apparire il database a seconda dell'utente che lo usa, in base ai suoi permessi.

- A livello **logico**: descrive la struttura degli insiemi dei dati e le relazioni fra essi, senza doversi occupare della loro organizzazione nella memoria.
- A livello **fisico**: viene descritto come sono organizzati fisicamente i dati nella memoria e vengono riportate quali strutture dati ausiliare vengono utilizzate.



Quest approccio permette di garantire le proprietà di indipendenza logica e fisica:

- **Indipendenza Logica**: gli applicativi non necessitano modifiche in seguito a variazioni dello schema logico.
- **Indipendenza Fisica**: gli applicativi non necessitano modifiche in seguito a cambiamenti dell'organizzazione fisica dei dati.

Per quanto riguarda i linguaggi di interrogazione, questi possono essere distinti in:

- **DML** (Data Manipulation Language): per l'interrogazione e l'aggiornamento dei dati.
- **DDL** (Data Definition Language): per la definizione di schemi, sia logici che fisici, ed altre operazioni.

1.1 Funzionalità del DBMS

Un DBMS deve prevedere più modalità d'uso per soddisfare le esigenze di più categorie di utenti che accedono al database. Deve poter offrire:

- Un'interfaccia grafica per accedere ai dati.
- Un linguaggio di interrogazione per gli utenti inesperti (non programmatori).

- Un linguaggio di programmazione per chi sviluppa applicazioni, nello specifico deve prevedere l'integrazione del *DDL* e del *DML* nel linguaggio ospite.
- Un linguaggio per lo sviluppo di interfacce per le applicazioni.
- Predisporre per l'**amministratore** strumenti per stabilire i diritti d'accesso ai dati, per il ripristino del sistema e per la modifica e la definizione degli schemi logici (sia interno che esterno).

1.2 Proprietà dei Database

Il DBMS permette di garantire al database le seguenti proprietà:

- **Integrità**: mantenimento dei vincoli d'integrità dichiarati in fase di definizione dello schema.
- **Affidabilità**: protezione dei dati da parte di malfunzionamenti sia software che hardware e da anomalie indesiderate come l'accesso concorrente al database da parte di più utenti.
- **Sicurezza**: protezione dei dati da parte di utenti non autorizzati.

Inoltre un DBMS deve essere in grado di gestire collezioni di dati che siano:

- **Grandi**
- **Persistenti**: il periodo di vita dei dati è indipendente dai programmi che li utilizzano.
- **Condivise**: possono essere usati da programmi diversi.

Il DBMS deve essere anche **efficiente** (utilizzando al meglio le risorse in termini di *spazio* e *tempo*) ed **efficace**.

1.3 Transazioni

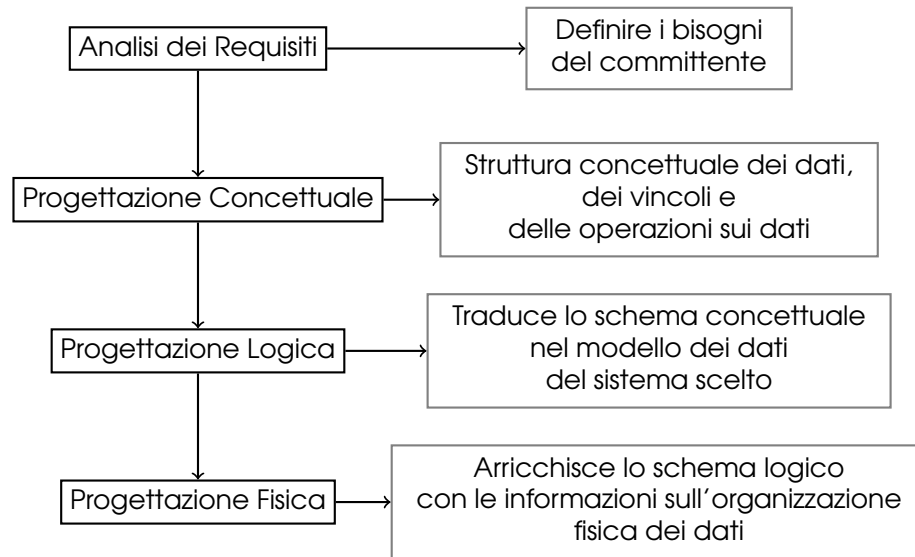
Definizione (Transazione). Una transazione è una serie di azioni di lettura e scrittura sulla memoria permanente o di elaborazione dati in memoria temporanea. Presenta le seguenti proprietà:

- **Atomicità**: le transizioni che non vanno a buon fine o che vengono abortite sono trattate come se non fossero mai state eseguite.
- **Persistenza**: le modifiche effettuate da una transazione andata a buon fine sono permanenti, ovvero non possono essere alterate da malfunzionamenti.
- **Serializzabilità**: nel caso di esecuzioni concorrenti di più transazioni, l'effetto ottenuto è quello di un'esecuzione seriale.

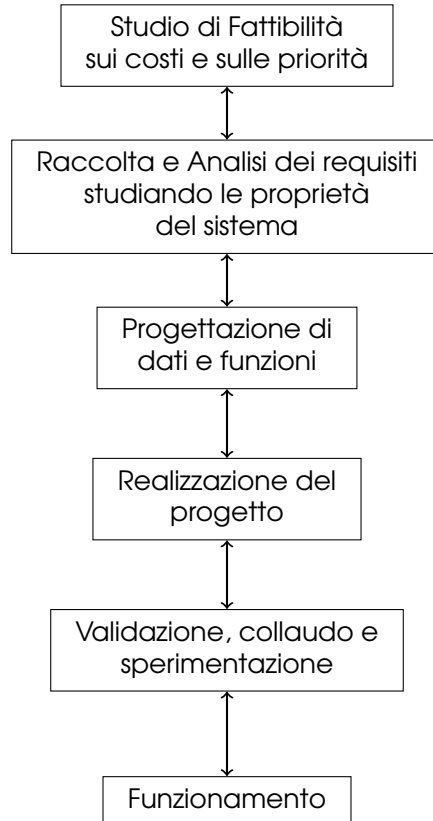
2 Modellazione

Definizione (Modello Astratto). Un modello astratto è la rappresentazione formale di idee e conoscenze relative ad un fenomeno.

La modellazione è centrale nella progettazione del database che comprende le fasi presenti in figura.



2.1 Fasi della Modellazione



2.2 Analizzare il Dominio

Il dominio può presentare più aspetti da dover analizzare:

- Aspetto *ontologico*: conoscere ciò che si suppone esista nell'universo del contesto e quindi ciò che è da modellare. Occorre analizzare tre tipi di conoscenze:
 - Conoscenza *concreta*: le entità del contesto e le associazioni fra di esse.
Definizione (Entità). Sono oggetti di cui occorre definire le proprietà.
Definizione (Proprietà). Descrivono le caratteristiche di determinate entità e sono formata da una coppia (Attributo, Valore). Ogni proprietà ha ad essa associato un dominio, quindi un insieme di valori che può assumere. Inoltre le proprietà si possono classificare in:

- * **Atomiche** o **Strutturate**: atomiche se il loro valore non è ulteriormente scomponibile.
- * **Totali** o **Parziali**: se è obbligatoria oppure opzionale.
- * **Univoche** o **Multivalore**: univoca se per ogni entità la scelta del valore è unico (es. *codice fiscale*).
- * **Costanti** o **Variabili**
- * **Calcolate** o **Non Calcolate**: calcolata se è possibile derivarla da altre proprietà.

Definizione (Tipo di un'entità). Ogni entità appartiene ad un tipo che ne indica la propria natura.

Definizione (Collezione). Insieme di entità dello stesso tipo.

- Conoscenza **astratta**: la struttura e i vincoli sulle entità.
- Conoscenza **procedurale**: le operazioni di base, sia dei singoli utenti e sia come avviene la comunicazione con il sistema informatico.
- Aspetto **logico**: meccanismi di astrazione (*modello di dati, per es. diagrammi E-R*²) con cui descrivere la struttura della conoscenza concreta.
- Aspetto **linguistico**: linguaggio formale con cui definire il modello.
- Aspetto **pragmatico**: insieme di regole da seguire in fase di modellazione.

2.3 Oggetti e Classi

Definizione (Oggetto). Un oggetto è un'entità software che presenta uno *stato*, un *comportamento* e un'identità. Lo **stato** è rappresentato da un insieme di costanti o variabili, mentre il **comportamento** è un insieme di procedure locali chiamate *metodi*. Un oggetto può rispondere a dei messaggi di input, con dei valori memorizzati nello stato o calcolandoli con un metodo.

Definizione (Classe). Una classe è un insieme di oggetti dello stesso tipo, e presenta delle operazioni per l'inserimento e la rimozione di elementi.

Definizione (Tipo Oggetto). Un tipo oggetto definisce l'insieme degli attributi a cui può combaciare un insieme di possibili oggetti. I tipi oggetto non sono presenti nei diagrammi E-R, però dagli attributi di una collezione è possibile dedurre il tipo oggetto associato.

²https://it.wikipedia.org/wiki/Modello_E-R

2.4 Associazioni

Definizione (Istanza di un'associazione). Un'istanza di un'associazione determina un legame logico tra due o più istanze.

Definizione (Associazione). Un'associazione $R(X, Y)$ fra due collezioni di entità chiamate X e Y è un insieme, che varia nel tempo, di istanze di associazione tra gli elementi delle due collezioni. Il prodotto cartesiano $(X \cdot Y)$ è chiamato **dominio dell'associazione**.

Un'associazione è caratterizzata da due proprietà: **molteplicità e totalità**.

Definizione (Vincolo di Unicità). Un'associazione $R(X, Y)$ è detta **univoca** rispetto ad X se per ogni elemento di $x \in X$ esiste al più un elemento $y \in Y$ che è associato ad x . Se questo vincolo non vale, si dice che l'associazione è **multivalore** rispetto ad X .

Cardinalità dell'Associazione

- $R(X, Y)$ è **(1:N)** se è multivalore su X ed univoca su Y .
- $R(X, Y)$ è **(N:1)** se è univoca su X e multivalore su Y .
- $R(X, Y)$ è **(N:M)** se è multivalore su X e multivalore su Y .
- $R(X, Y)$ è **(1:1)** se è univoca su X ed univoca su Y .

Definizione (Vincolo di Totalità). Un'associazione $R(X, Y)$ è detta **totale** su X se per ogni elemento $x \in X$ esiste almeno un elemento $y \in Y$ associato ad x . Se questo vincolo non vale, si dice che l'associazione è **parziale** rispetto ad X .

Rappresentazione delle Associazioni Un'associazione fra due collezioni C_1 e C_2 si rappresenta con una linea che collega le due classi. La linea si etichetta con il nome dell'associazione. L'**univocità** di una classe C_1 si rappresenta disegnando una freccia singola sulla linea che esce vada da C_1 a C_2 . Se invece l'associazione è **multivalore** si indica con una doppia freccia. La **parzialità** invece è rappresentata con un taglio sulla linea vicino alla freccia, mentre la **totalità** è rappresentata dall'assenza del taglio.



Figura 2: In questo caso abbiamo un'associazione *multivalore* da entrambe la parti, ma *parziale* per C_2 e *totale* per C_1

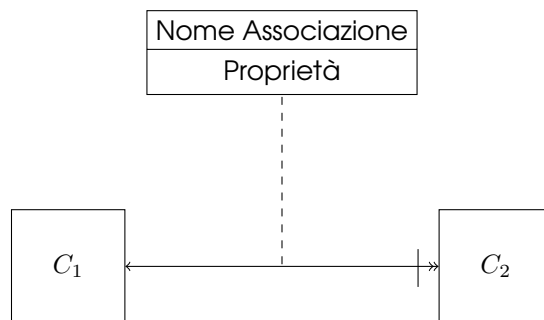


Figura 3: Le associazioni possono presentare **proprietà**

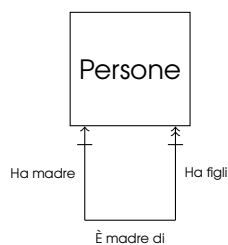


Figura 4: O possono anche essere **ricorsive**. In questo caso occorre etichettare l'associazione non solo con il proprio nome, ma anche con i nomi dei ruoli che hanno le due entità nell'associazione.

Associazione Non Binaria Le associazioni non binarie, per semplicità, non vengono rappresentate graficamente, ma ad esempio, per quanto

riguarda quelle *ternarie*, queste vengono trasformate in tre associazioni *binarie* aggiungendo un'altra collezione al posto dell'associazione *ternaria*.

2.5 Gerarchie

Le classi di entità possono essere organizzate in una gerarchia di *specializzazione*. Una classe della gerarchia minore viene chiamata **sottoclasse**, mentre le altre si chiamano **superclassi**. Gli elementi di una sottoclasse sono un sottoinsieme degli elementi della superclasse.

2.5.1 Tipi Oggetto

Fra i *tipi oggetto* viene definita una relazione di sottotipo, che comprende le seguenti proprietà:

- È una relazione **asimmetrica**, **riflessiva** e **transitiva**.
- Inoltre, se un tipo T è **sottotipo** di T' , allora tutti gli elementi di T possono essere usati in tutti i contesti in cui appaiono elementi di tipo T' . Questa proprietà è chiamata **sostituibilità** ed è data dal fatto che gli elementi di T hanno tutte le proprietà degli elementi di T' , e per ogni proprietà di T' , il suo tipo in T è un sottotipo di quello che ha in T' .

Ereditarietà L'*ereditarietà* è una proprietà delle gerarchie che permette di definire un *tipo oggetto* a partire da un altro. In quanto, nel nostro contesto, a partire da un tipo, si vuole solo definire un sottotipo, si parla di *ereditarietà stretta*, che permette solo:

- L'aggiunta di altri *attributi*.
- La ridefinizione di attributi del *supertipo*, però solo specializzando ulteriormente il tipo dell'attributo.

2.5.2 Classi

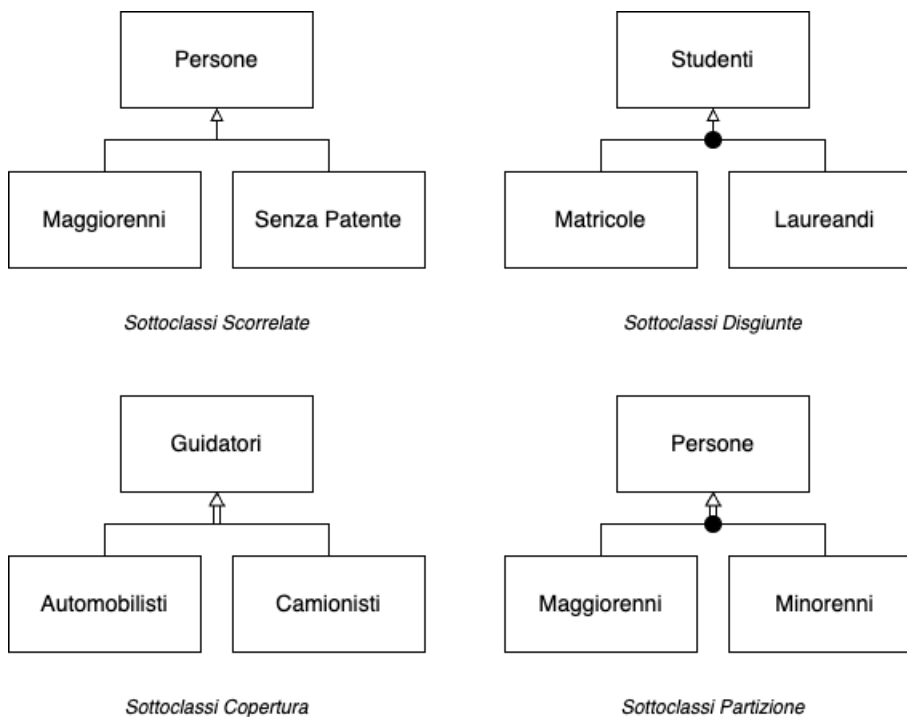
Fra le *classi*, invece, viene definita una relazione di sottoclasse, con le seguenti proprietà:

- Sempre **asimmetrica**, **riflessiva** e **transitiva**, come per i *tipi oggetto*.
- Se una classe C è **sottoclasse** di C' , allora il tipo degli elementi di C è sottotipo del tipo degli elementi di C' (**vincolo intensionale**).
- Se C è **sottoclasse** di C' , allora gli elementi di C sono un sottoinsieme degli elementi di C' (**vincolo estensionale**).

Vincoli Possiamo distinguere due tipi di vincoli su insiemi di sottoclassi:

- **Disgiunzione**: ogni coppia di sottoclassi nell'insieme è *disgiunta*, quindi è priva di elementi comuni.
- **Copertura**: l'unione degli elementi delle sottoclassi coincide con l'insieme degli elementi della *superclasse*.

Se possiedono entrambi i vincoli, allora l'insieme delle *sottoclassi* forma una **partizione** della *superclasse*. Altrimenti se nessuno dei due vincoli è rispettato, si dice che le sottoclassi sono **scorrelate**.



2.5.3 Ereditarietà Multipla

Un tipo può anche essere definito per *ereditarietà* anche a partire da più di un *supertipo*. Questo però può creare alcuni problemi quando lo stesso *attributo* viene ereditato da più di un supertipo, ma i tipi degli attributi fra loro sono diversi.

3 Progettazione Logica

L'obiettivo della **progettazione logica** è quello di ridefinire lo *schema concettuale* in uno *schema logico relazionale* che rappresenti gli stessi dati ma in una maniera più efficiente (minimizzando la **ridondanza**) e corretta, per esempio tenendo conto della dimensione dei dati e del tipo di operazioni che si effettueranno sul database. Anche perchè alcuni costrutti dello *schema relazionale* non sono rappresentabili concretamente.

3.1 Relazioni

Definizione (Relazione Matematica). Una *relazione matematica* è un insieme di n -uple ordinate (d_1, \dots, d_n) tali che $d_1 \in D_1, \dots, d_n \in D_n$.

Essendo un insieme, non c'è ordinamento fra le n -uple, che inoltre devono essere tutte distinte. Però l'ordinamento all'interno della n -upla conta, infatti l' i -esimo valore deve provenire dall' i -esimo dominio. Per questo si dice che la struttura della relazione è **posizionale**.

Definizione (Attributo). A ciascun dominio della relazione si associa un nome, chiamata **attributo**.

Definizione (Tupla). Una **tupla** su un insieme di *attributi* chiamato X , è una funzione t che associa a ciascun *attributo* un valore del suo *dominio*.

Una **relazione su X** è un insieme di *tuple* su X .

3.2 Tabelle

Una **tabella** rappresenta una relazione se:

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

In una *tabella* l'ordinamento tra le righe e le colonne è irrilevante. In ogni tabella è possibile distinguere due parti:

- Lo **schema** è rappresentato dalle intestazioni della tabella, che sono invarianti nel tempo e descrivono la struttura della tabella (**aspetto intensionale**).
- L'**istanza** sono i valori attuali presenti nella tabella, che possono cambiare nel tempo (**aspetto estensionale**).

Definizione (Tipo Ennupla). Un **tipo ennupla** chiamato T è un insieme finito di coppie (*Attributo*, *Tipo Elementare*).

Definizione (Schema di Relazione). Se T è un *tipo ennupla*, allora $R(T)$ è lo **schema della relazione** R .

Definizione (Schema di Database). Lo **schema di un database** è un insieme di *schemi di relazioni* $R_i(T_i)$.

Definizione (Istanza di Relazione). Un'**istanza di relazione**, anche chiamata **relazione** su uno schema $R(T)$ è l'insieme r di tuple di tipo T .

Definizione (Istanza di Database). Un'**istanza di database** su uno schema $R = \{R_1(T_1), \dots, R_n(T_n)\}$ è l'insieme delle *relazioni* $r = \{r_1, \dots, r_n\}$, dove r_i è un'*istanza di relazione* su $R_i(T_i)$.

Valore Nullo Si indica con NULL, e indica l'assenza di un valore del dominio.

3.3 Vincoli di Integrità

Un **vincolo d'integrità** è una proprietà che deve essere soddisfatta da ogni singola *istanza* della relazione, in modo tale che rappresenti informazioni corrette per l'applicazione.

Il vincolo viene espresso mediante un predicato, che associa ad ogni *istanza* il valore **vero** o **falso**.

I vincoli si possono classificare in:

- Vincoli **intrarelazionali**: ovvero quelli che devono essere rispettati da valori della relazioni presa in considerazione, e possono essere:
 - Vincoli sul **dominio**, che coinvolgono un solo *attributo*.
 - Vincoli di **ennupla**, che esprimono condizioni sui valori di ogni ennupla, indipendentemente dalle altre ennuple.
- Vincoli **interrelazionali**: sono quei vincoli che devono essere rispettati da valori presenti in relazioni diverse.

3.4 Chiavi

Definizione (Superchiave). Un insieme K di attributi viene chiamato **superchiave** per una relazione r , se r non contiene due ennuple distinte t_1, t_2 tali che $t_1[K] = t_2[K]$.

Definizione (Chiave). K invece viene definito **chiave** per r se è una **superchiave minimale** per r , ovvero non deve contenere altre *superchiavi*.

Nota Bene Dato che una *relazione* non può contenere ennuple con valori uguali, allora per ogni *relazione* esiste sempre una **superchiave** rappresentata dall'insieme di tutti gli attributi su cui è definita.

Definizione (Chiave Primaria). Una **chiave primaria** è una *chiave* sulla quale non sono ammessi valori nulli.

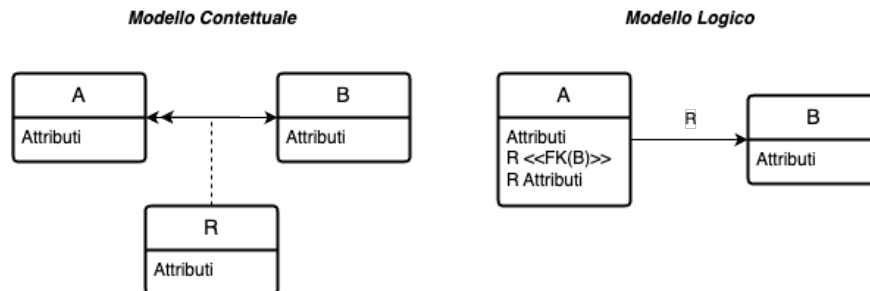
Nel modello relazionale per correlare due relazioni diversi si usano i valori delle *chiavi primarie*.

Definizione (Integrità Referenziale). Un vincolo di **integrità referenziale**, anche chiamato *foreign key*, fra alcuni attributi X di una relazione R_1 e un'altra relazione R_2 impone ai valori di X di comparire come valori della *chiave primaria* di R_2 .

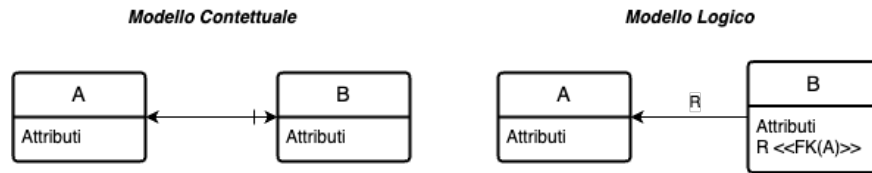
Violazioni Se per esempio si provasse ad eliminare una ennupla dalla tabella che viene riferita si verificherebbe un rifiuto dell'operazione, nel caso in cui la *chiave primaria* di quella ennupla viene riferita da altre ennuple di altre tabelle. Quindi in questo caso occorre procedere con un'*eliminazione a cascata*, ovvero si impostano prima a NULL i valori degli attributi delle ennuple delle tabelle che contengono riferimenti a quella *chiave primaria*; successivamente si può procedere con l'eliminazione della ennupla che adesso non verrà più riferita.

3.5 Rappresentazione delle Associazioni

Uno a Molti Le associazioni *uno a molti* si rappresentano aggiungendo, agli attributi della relazione rispetto alla quale l'associazione è univoca, una *chiave esterna* che si riferisce all'altra relazione. Nel caso in cui l'associazione ha degli attributi si aggiungono anch'essi alla relazione.

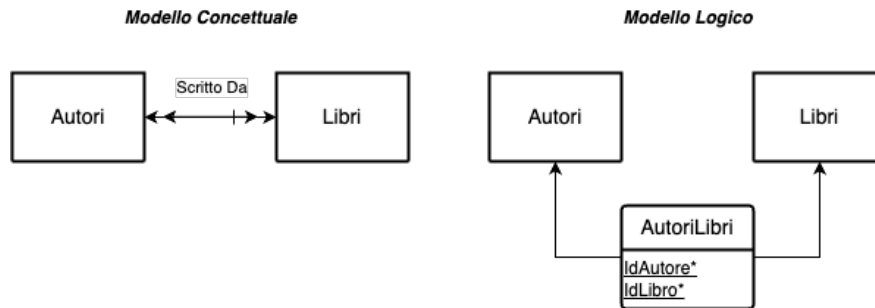


Uno a Uno In questo caso si aggiunge la *chiave esterna* scegliendo arbitrariamente una delle due relazioni ma, in caso in cui esiste un vincolo di totalità, si preferisce la relazione rispetto alla quale l'associazione è *totale*.

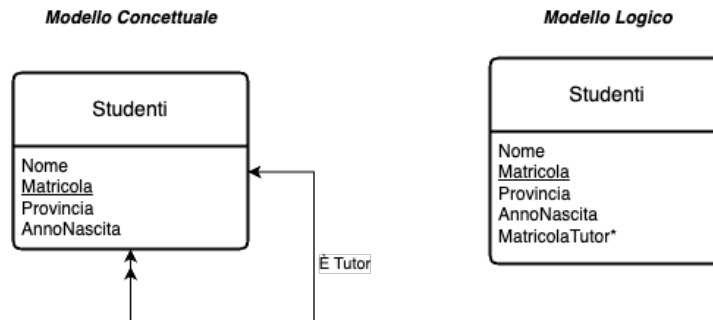


Vincoli sulle Cardinalità La direzione dell'associazione rappresentata dalla *chiave esterna* è chiamata la *diretta* dell'associazione. Per imporre un vincolo di *univocità* della diretta occorre definire un vincolo di chiave sulla *chiave esterna*; mentre per descrivere un vincolo di *totalità* della diretta si impone un vincolo NOT NULL sempre sulla *chiave esterna*.

Molti a Molti Un'associazione *molti a molti* si traduce aggiungendo tra le due relazioni, una terza che ha come attributi (e come *chiave primaria*) le *chiavi primarie* delle due relazioni.

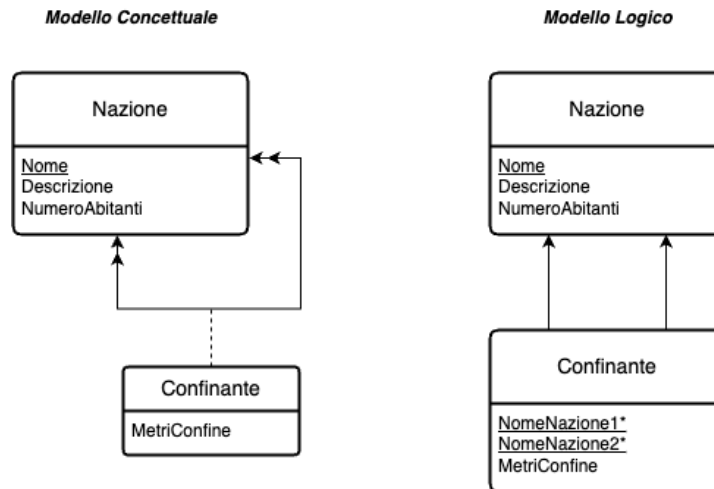


Ricorsione In questo caso la *chiave esterna* si aggiunge alla stessa e sola relazione.



Ricorsione Molti a Molti Anche qui si costruisce una seconda relazione che ha come *chiave primaria* le due *chiavi primarie* delle due istanze

coinvolte.



3.6 Rappresentazione delle Gerarchie

Le *gerarchie* non possono essere rappresentate direttamente, quindi vanno eliminate sostituendole con altre classi e relazioni.

Ci sono tre metodi:

- **Relazione Unica:** se A_0 è la classe genitore di A_1 e A_2 , queste vengono eliminate e accorpate ad A_0 . Ad A_0 viene aggiunto un attributo chiamato **discriminatore**, che indica per ogni istanza da quale classe figlia deriva. Ovviamente anche gli attributi delle classi figlie vengono assorbiti dal genitore e assumono valore NULL sulle istanze di una classe figlia che non possedeva quegli attributi. Per quanto riguarda le relazioni, invece, anche queste vengono assorbite dalla classe genitore ma avranno comunque cardinalità minima uguale a 0, anche qui per le istanze di una classe figlia che non aveva quella relazione.
- **Partizionamento Orizzontale:** la classe genitore A_0 viene eliminata e le classi figlie A_1 e A_2 ereditano gli attributi e le relazioni del genitore. L'unico caso in cui non si può adoperare questo metodo è quando è presente un *vincolo referenziale* verso la classe genitore, in quanto non è possibile spezzare il vincolo in più relazioni diverse.
- **Partizionamento Verticale:** la gerarchia si trasforma in tante associazioni *uno a uno* che legano ogni classe figlia con la classe genitore. In questo caso vanno aggiunti dei vincoli, ovvero ogni istanza di A_0 non può partecipare a tutte le associazioni, ma ad al più una; e nel caso in cui la gerarchia è totale deve essere esattamente 1.

Nota Bene In quest'ultimo metodo la *chiave primaria* della classe genitore è sia *chiave esterna* che *chiave primaria* per le figlie.

3.7 Rappresentazione Campi Multivalore

Per la gestione dei *campi multivalore* viene creata una nuova relazione che ha come attributi il nome del campo e una *chiave esterna* che si riferisce alla relazione in cui si trovava. La *chiave primaria* di questa nuova relazione è l'intera ennupla.

4 Algebra Relazionale

Definizione (Algebra Relazionale). Con **algebra relazionale** si intende un insieme di *operatori* su relazioni che danno come risultato altre relazioni. Non viene usato come linguaggio di interrogazione dei DBMS, ma come rappresentazione interna delle interrogazioni.

4.1 Operatori Insiemistici

Le relazioni vengono viste come degli *insiemi*. Gli operatori di *unione*, *differenza* ed *intersezione* sono applicabili solo a relazioni definite sugli stessi *attributi*.

- **Unione**: l'unione di due relazioni R e S definite sullo stesso insieme di attributi X è indicata con $R \cup S$ ed è una relazione su X che contiene le tuple che appartengono a R o a S senza ripetizioni.
- **Differenza**: la differenza è indicata con $R - S$ ed è una relazione sempre su X che contiene tutte le tuple che appartengono ad R ma non a S .
- **Ridenominazione**: è un operatore *monadico*, ovvero su una sola relazione, che permette di modificare lo schema, ridenominando il nome di uno o più attributi, ma lasciando inalterate le istanze.

$$\rho \text{ Nuovo Nome Attributo} \leftarrow \text{Vecchio Nome Attributo} (\text{Relazione})$$

- **Proiezione**: è un altro operatore *monadico* che produce un risultato su un sottoinsieme degli attributi della relazione e contiene le ennuple della relazione ristrette solo agli attributi del sottoinsieme.

$$\pi \text{ Lista Attributi} (\text{Relazione})$$

La cardinalità di una *proiezione* è basata sul sottoinsieme X degli attributi:

- Se X è una *superchiave* di R , allora $\pi_X(R)$ contiene esattamente tante ennuple quante ne contiene R .
- Altrimenti se X non è *superchiave*, potrebbero essere presenti dei valori che su quegli attributi sono ripetuti e che quindi vengono rappresentati una sola volta.
- **Selezione o Restrizione:** è sempre un operatore *monadico* che produce una relazione con lo stesso schema dell'operando, che contiene un sottoinsieme delle sue ennuple che soddisfano una data condizione.

$$\sigma_{Cond}(R)$$

Con $Cond$ generata dalla seguente grammatica:

```
Cond ::= Exp Theta Exp | Cond And Cond | Cond Or Cond | Not Cond
Exp ::= Attributo | Costante | Exp Op Exp
Theta ::= = | < | > | != | <= | >=
Op ::= + | - | * | StringConcat
```

Le condizioni atomiche si riferiscono solo ai valori non nulli, per riferirsi anche ai valori nulli si usano forme di condizioni come IS NULL e IS NOT NULL.

- **Prodotto:** operatore fra più relazioni che devono avere fra loro tutti i nomi degli attributi distinti, e che esegue il prodotto cartesiano fra gli insiemi di tuple.
- **Intersezione:** l'intersezione è indicata con $R \cap S$ ed è una relazione definita sullo stesso insieme di attributi X che contiene le tuple che appartengono sia ad R che ad S . L'*intersezione* è chiamato operatore *derivato*, dato che è possibile derivarlo usando altri operatori:

$$R \cap S = \{x \mid x \in R \wedge \exists y \in S \text{ t.c. } x = y\}$$

allora se per esempio R ed S sono definiti con $X = \{A, B\}$.

$$\pi_{A,B}(\sigma_{A=S.A \text{ AND } B=S.B}(R \times \rho_S(S)))$$

Partendo dall'interno, si ridenominano tutti gli attributi di S ponendogli davanti il prefisso S , successivamente viene fatto il *prodotto* con R e tramite la *selezione* selezioniamo solo le tuple che hanno gli attributi semanticamente uguali con gli stessi valori. Infine si fa una *proiezione* per eliminare gli attributi ridondanti.

4.1.1 Join

Il **join** o **giunzione** permette di correlare dati in relazioni diverse. Anch'esso è un operatore derivato in quanto si può ottenere per composizione degli operatori visti precedentemente. Si indica con il simbolo \bowtie .

Il **join naturale** produce un'unione degli attributi dei due operandi, combinando le ennuple degli operandi che hanno valori uguali sugli attributi in comune. Ad esempio $R_1 \bowtie R_2$ è una relazione su $X_1 \cup X_2$ definita come segue:

$$R_1 \bowtie R_2 = \{t \text{ su } X_1 \cup X_2 \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \\ \text{con } t[X_1] = t_1 \text{ e } t[X_2] = t_2\}$$

Un *join* viene chiamato **completo** se ogni ennupla delle due relazioni contribuisce al risultato. Al contrario si parla di **join non completo**. Generalmente possiamo definire un limite inferiore e superiore alla cardinalità di una join:

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

Nel caso il join è **completo** possiamo restringere il limite inferiore dicendo che $|R_1 \bowtie R_2| \geq \max\{|R_1|, |R_2|\}$.

Nel caso in cui il join coinvolge come attributo comune una *chiave* di R_2 , allora:

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

Se oltre a coinvolgere una *chiave* di R_2 , vi è un vincolo di *integrità referenziale* tra un attributo di R_1 e la chiave di R_2 allora la cardinalità della join è esattamente $|R_1|$.

Un **join esterno** estende con valori nulli le ennuple che verrebbero tagliate fuori da un **join naturale**. Esiste in tre versioni:

- **Join esterno sinistro** $R \overset{\leftarrow}{\bowtie} S$: mantiene tutte le ennuple del primo operando, estendendole con valori nulli se necessario.
- **Join esterno destro** $R \overset{\rightarrow}{\bowtie} S$: mantiene tutte le ennuple del secondo operando, estendendole con valori nulli se necessario.
- **Join esterno completo** $R \overset{\leftrightarrow}{\bowtie} S$: mantiene tutte le ennuple di entrambi gli operandi, estendendole con valori nulli se necessario.

Nota Bene Il *prodotto cartesiano* si può vedere come un *join naturale* su relazioni che non hanno attributi in comune.

Dato che il *prodotto cartesiano* non ha senso se non lo facciamo seguire da una selezione (che permette di specificare la condizione sui campi che semanticamente rappresentano lo stesso attributo), si usa l'operazione di **theta-join** definito come segue:

$$R_1 \bowtie_{Condizione} R_2$$

che è equivalente a:

$$\sigma_{Condizione} (R_1 \bowtie R_2)$$

N.B.: dato che R_1 ed R_2 non hanno attributi comuni, in questo caso $\bowtie = \times$.

Se nel *theta-join* l'operatore di confronto nella *Condizione* è sempre l'uguaglianza, allora si parla di **equi-join**.

Self Join Supponiamo di avere questa relazione Genitori:

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

e di voler ottenere una relazione Nonno-Nipoti. Per far ciò occorre riutilizzare la stessa tabella ma facendo $Genitori \bowtie Genitori$ si otterrebbe di nuovo la tabella Genitori. In questo caso occorre ridenominare la tabella due volte:

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

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

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

$$\rho_{Nipote \leftarrow Figlio}(Genitore)$$

Infine, effettuando una *join* seguita da una *proiezione* otteniamo:

Nonno	Nipote
Giorgio	Anna
Silvia	Anna
Enzo	Anna

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