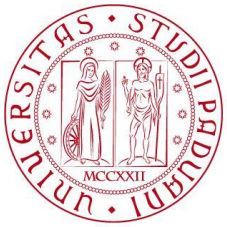


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

BASI DI DATI

PROGETTAZIONE CONCETTUALE, LOGICA E SQL

(Versione 30/04/2018)

A cura di:
Stefano Ivancich

INDICE

Modello a cascata per lo sviluppo di un software 1

1. Progettazione concettuale 3

1.1. Modello Entità-Associazione 3

1.2. Documentazione di schemi E-A 5

1.3. Pattern 6

1.4. Strategie di progetto 8

1.5. Metodologia generale 9

2. Progettazione logica 11

2.1. Ristrutturazione dello schema ER 12

2.1.1. Analisi delle ridondanze 12

2.1.2. Eliminazione delle generalizzazioni 13

2.1.3. Partizionamento/accorpamento di entità e associazioni 14

2.1.4. Scelta degli identificatori principali 14

2.2. Traduzione verso il modello relazionale 15

2.2.1. Modello logico relazionale 15

2.2.2. Passi per la traduzione 15

2.2.3. Traduzione entità forte 16

2.2.4. Traduzione associazioni binarie (x,n)-(y,m) 16

2.2.5. Traduzione associazioni binarie (x,1)-(y,m) 16

2.2.6. Traduzione associazioni binarie (x,1)-(y,1) 17

2.2.7. Traduzione associazioni ricorsive (x,n)-(y,m) 17

2.2.8. Traduzione associazioni ternarie (x,n)-(y,m)-(z,o) 17

2.2.9. Traduzione entità debole 18

3. Algebra relazionale 19

3.1. Ridenominazione, Selezione e Proiezione 19

3.2. Join 19

3.3. Aggregazione e Raggruppamento 20

3.4. Valori nulli 21

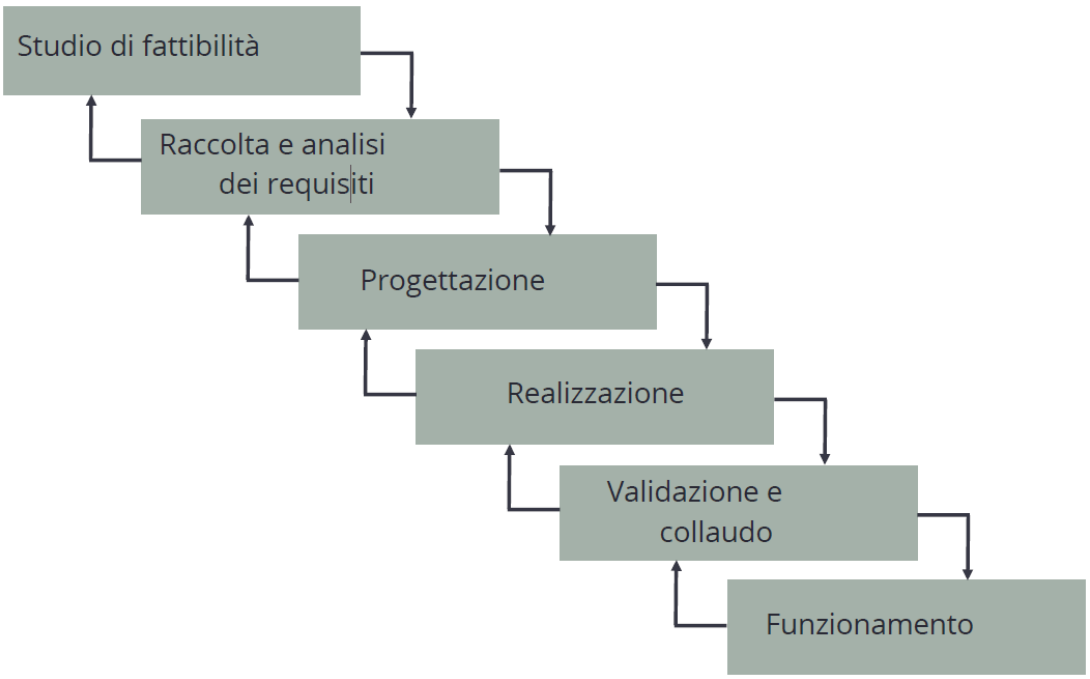
3.5. Equivalenza di espressioni algebriche 21

4. SQL 23
4.1. Definizione dei dati 23
4.2. Interrogazioni 26
4.3. Modifica dei dati 28
5. Normalizzazione 29
5.1. Anomalie e ridondanze 29
5.2. Dipendenze funzionali 29
5.3. Forma normale di Boyce e Codd 29
5.4. Decomposizioni 30
5.4.1. Proprietà 30
5.5. Terza forma normale 30
5.6. Teoria delle dipendenze e normalizzazione 31
5.6.1. Implicazione di dipendenze funzionali 31
5.6.2. Coperture di insiemi di dipendenze funzionali 31
5.6.3. Sintesi di schemi in terza forma normale 32
5.7. Progettazione di basi di dati e normalizzazione 32
5.7.1. Verifiche di normalizzazione su entità 32
5.7.2. Verifiche di normalizzazione su associazioni 32

Lo scopo di questo documento è quello di riassumere i testi "*Basi di dati - Paolo Atzeni 4 edizione McGraw-Hill 2014 Capitoli 1-9*" e "*Basi di dati Progettazione logica e SQL - Di Nunzio Esculapio 2017*" per poter essere utilizzato nella realizzazione di Basi di Dati come un manuale "pratico e veloce" da consultare. Non sono presenti esempi e spiegazioni dettagliate, per questi si rimanda ai testi citati.

Se trovi errori ti prego di segnalarli qui:
www.stefanoivancich.com
ivancich.stefano@gmail.com
Il documento verrà aggiornato entro 48h.

Modello a cascata per lo sviluppo di un software



Metodologia di **progettazione di una base di dati:**

- **Analisi dei requisiti:** l'obiettivo di questa fase è quello di raccogliere ed analizzare i requisiti relativi ai fabbisogni informativi degli utenti. I requisiti descrivono sia i dati sia le operazioni che si prevedono di eseguire su di essi.
- **Progettazione concettuale:** l'obiettivo è tradurre i requisiti in una descrizione formale e indipendente dalle scelte relative alla realizzazione fisica. Tale descrizione è chiamata schema concettuale.
- **Progettazione logica:** l'obiettivo è quello di tradurre lo schema concettuale nella struttura logica scelta per la realizzazione del DBMS. La traduzione ha l'obiettivo primario di pervenire a quella rappresentazione dei dati che, rispondenti ai requisiti, assicuri la maggiore efficienza rispetto alle esecuzioni delle operazioni.
- **Progettazione fisica:** è la fase di implementazione dello schema logico che dipende dal particolare DBMS.

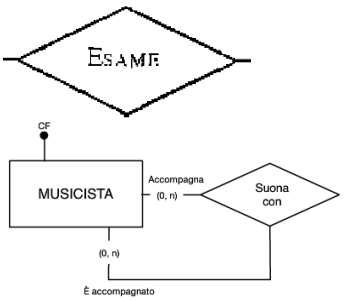
1. Progettazione concettuale

1.1. Modello Entità-Associazione

Entità: Rappresentano classi di oggetti
Es. Città, Dipartimento, Impiegato, …
Occorrenza di un'entità: è un oggetto della classe che lo rappresenta.
Es. Città: Roma, Milano, …

Associazioni(Relazioni): rappresentano legami logici tra entità.

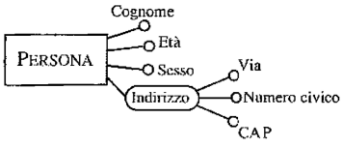
- **Occorrenza**: è un n-pla costituita da occorrenze di entità. *Es. (Mario, Venezia), (Paolo, Roma).*
- Il nome è preferibile utilizzare un sostantivo anzi che un verbo.
- L'insieme delle occorrenze è il prodotto cartesiano delle entità.
- Tra le occorrenze non ci possono essere n-ple ripetute.
- Possono esserci relazioni ricorsive. (*Es. Collega-Impiegato*)



E' preferibile dare un nome a ciascun ramo avendo così un verso di lettura.
Possono essere simmetriche o asimmetriche.

Attributi: descrivono le proprietà elementari delle entità ed associazioni.

- **Semplice**: ha un dominio di dati (ad esempio numeri interi o caratteri) e può assumere un unico valore.
- **Composto**: costituito da più attributi semplici raggruppati.

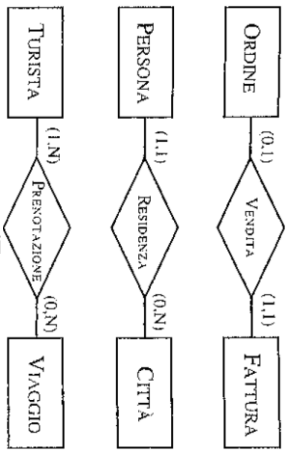


Cardinalità:

- Se (1,1) non si scrive.
- Se min=0: opzionale
- Se min=1: obbligatorio
- Se max=N: multivalore.

Cardinalità delle associazioni: descrivono il numero minimo e massimo di occorrenze di relazione a cui un'occorrenza dell'entità può partecipare. Quante volte un'occorrenza di una entità può essere legata a occorrenze di altre entità.

- Se Min:
  - 0: partecipazione opzionale
  - 1: partecipazione obbligatoria
- Se Max:
  - 1: una funzione che associa ad una occorrenza dell'entità una sola occorrenza (o nessuna) dell'altra entità.
  - Molti (N): associazione con un numero arbitrario di occorrenze dell'altra entità.



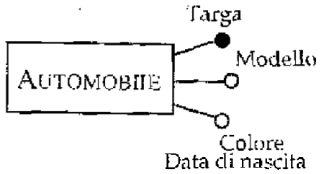
Associazioni binare:

- **Uno a uno:** max=1 per entrambe le entità.
- **Uno a molti:** max=1 e max=N.
- **Molti a molti:** max=N per entrambe le entità.

Associazioni 3-arie:
Scelta delle **cardinalità:** "quante coppie(n-uple) di elementi delle altre entità che possono avere questa associazione?".
Da 3-aria a binaria: Se la cardinalità massima di un'entità è 1.

Identificatori delle entità: permettono di identificare in maniera univoca le occorrenze delle identità.

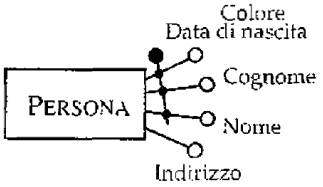
- Identificatore **interno**: uno o più attributi. (Entità forte)
- Identificatore **esterno**: utilizza altre entità. (Entità debole)



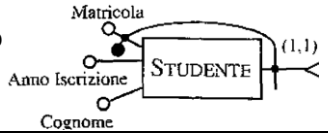
Un **identificatore** può coinvolgere più **attributi**, ognuno dei quali deve avere **cardinalità (1,1)**.

Un'**identificazione esterna** può coinvolgere più entità, ognuna delle quali deve avere cardinalità di **associazione (1,1)**.

Possono esserci più attributi identificativi indipendenti.



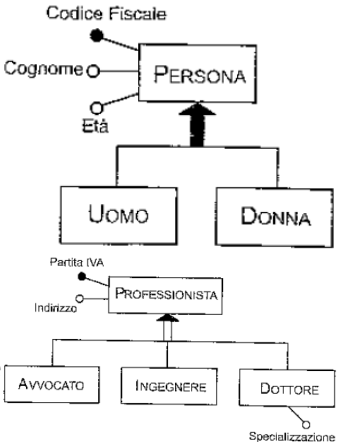
NB.: Cerca di creare più entità forti possibili, ma senza mettere un attributo fittizio "id" come identificatore.



Generalizzazioni/specializzazioni

Legame tra un'entità genitore (superclasse) e una o più entità figlie(sottoclassi). Cioè le figlie sono casi particolari(specializzazioni) del padre.

- Un'occorrenza dell'entità figlia è anche occorrenza dell'entità padre.
- Attributi, identificatori, associazioni e altre generalizzazioni del padre vengono ereditate dai figli.



Tipi di generalizzazione:

- **Totale**: ogni occorrenza del padre è in almeno un figlio. Cioè l'unione delle occorrenze dei figli fa il padre. (**Freccia piena**)
- **Parziale**: se ci sono occorrenze del padre che non sono nei figli. (**Freccia vuota**)
- **Esclusiva**(disgiunta): se l'intersezione di qualsiasi coppia delle occorrenze delle figlie è vuota. (Graficamente si aggiunge "d")
- **Sovrapposta**: almeno un'occorrenza viene condivisa tra le figlie. Possono essere trasformate in esclusive aggiungendo un'entità figlia che le comprende. (*Es. studente, lavoratore, aggiungo: StudLav*)

Sottoinsieme(IS-A): se il padre ha una sola figlia.

1.2. Documentazione di schemi E-A

Lo schema E-A avvolte non è sufficiente, o addirittura risulta impossibile rappresentare alcune proprietà dei dati. È meno adatto a rappresentare vincoli tra i dati (*Es. impiegato deve guadagnare meno del capo*).

Tipi di regole aziendali:

- **Descrizione** di un concetto: definizione di un'entità, un attributo o associazione.
- **Vincolo** di integrità: sia che sia espresso nel modello EA sia che non sia esprimibile da esso.

  (RV*N°*) < concetto > deve/non deve < espressione su concetti >
- **Derivazione:** concetto ottenuto tramite deduzione o calcolo aritmetico da altri concetti dello schema.

  (RD*N°*) < concetto > si ottiene < operazioni su concetti >

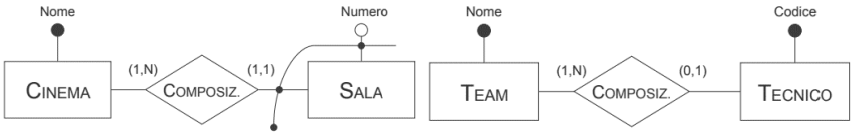
Entità	Descrizione	Attributi	Identificatore
Impiegato	Impiegato che lavora nell'azienda.	Codice, Cognome, Stipendio, Età	Codice
Progetto	Progetti aziendali sui quali lavorano gli impiegati.	Nome, Budget, Data consegna	Nome
Dipartimento	Dipartimenti delle sedi dell'azienda.	Telefono, Nome	Nome, Sede
Sede	Sede dell'azienda in una certa città.	Città, Indirizzo (Numero, Via e CAP)	Città

Relazione	Descrizione	Entità Coinvolte	Attributi
Direzione	Associa un dipartimento al suo direttore.	Impiegato (0,1), Dipartimento (1,1)	
Afferenza	Associa un impiegato al suo dipartimento.	Impiegato (0,1), Dipartimento (1,N)	Data afferenza
Partecipazione	Associa agli impiegati i progetti sui quali lavorano.	Impiegato (0,N), Progetto (1,N)	Data inizio
Composizione	Associa una sede ai dipartimenti di cui è composta.	Dipartimento (1,1), Sede (1,N)	

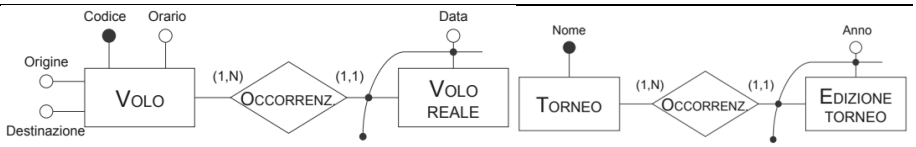
Regole di vincolo
(RV1) Il direttore di un dipartimento deve afferire a tale dipartimento.
(RV2) Un impiegato non deve avere uno stipendio maggiore del direttore del dipartimento al quale afferisce.
(RV3) Un dipartimento con sede a Roma deve essere diretto da un impiegato con più di dieci anni di anzianità.
(RV4) Un impiegato che non afferisce a nessun dipartimento non deve partecipare a nessun progetto.
Regole di derivazione
(RD1) Il budget di un progetto si ottiene moltiplicando per 3 la somma degli stipendi degli impiegati che vi partecipano.

1.3. Pattern

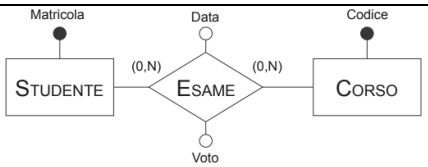
"Parte di"
Tipicamente uno a molti

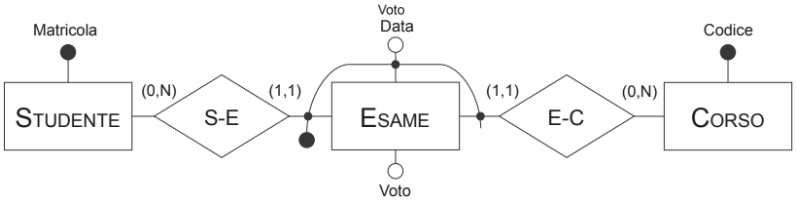


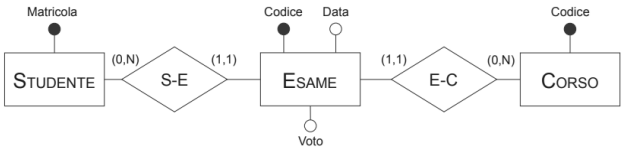
"Istanza di"



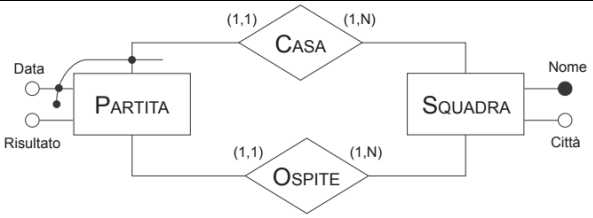
Concetto che lega concetti
Tipicamente molti a molti



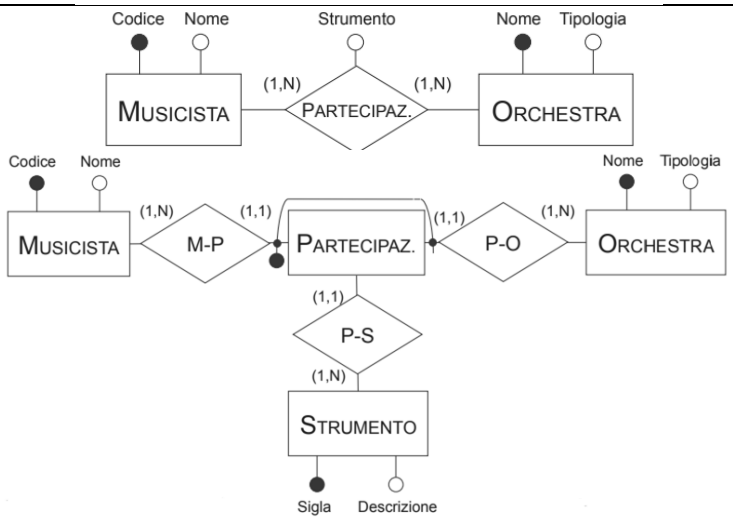




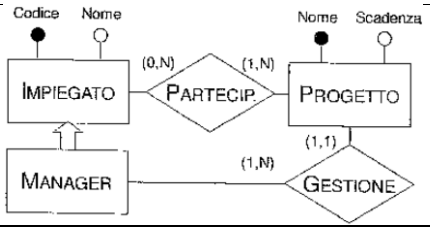
Reificazione di relazione ricorsiva.
Es. due squadre si incontrano più volte.



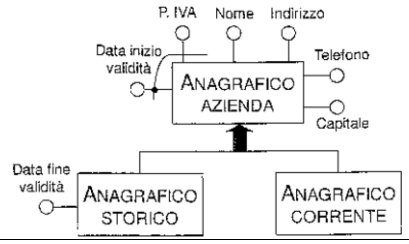
Relazione molti a molti con attributo.
Es. Se strumento è un concetto importante va reificato nella seconda figura.



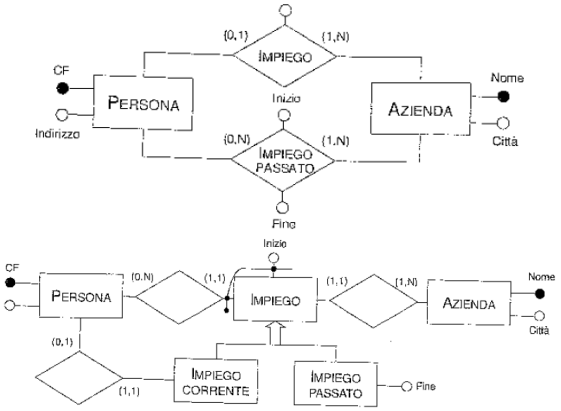
Caso particolare di entità



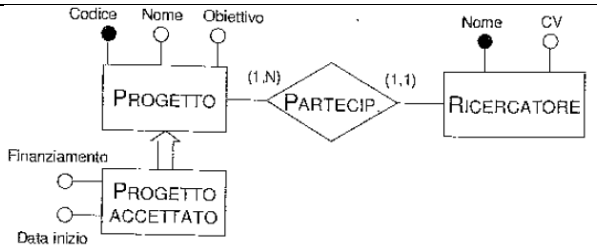
Storicizzazione di un'entità



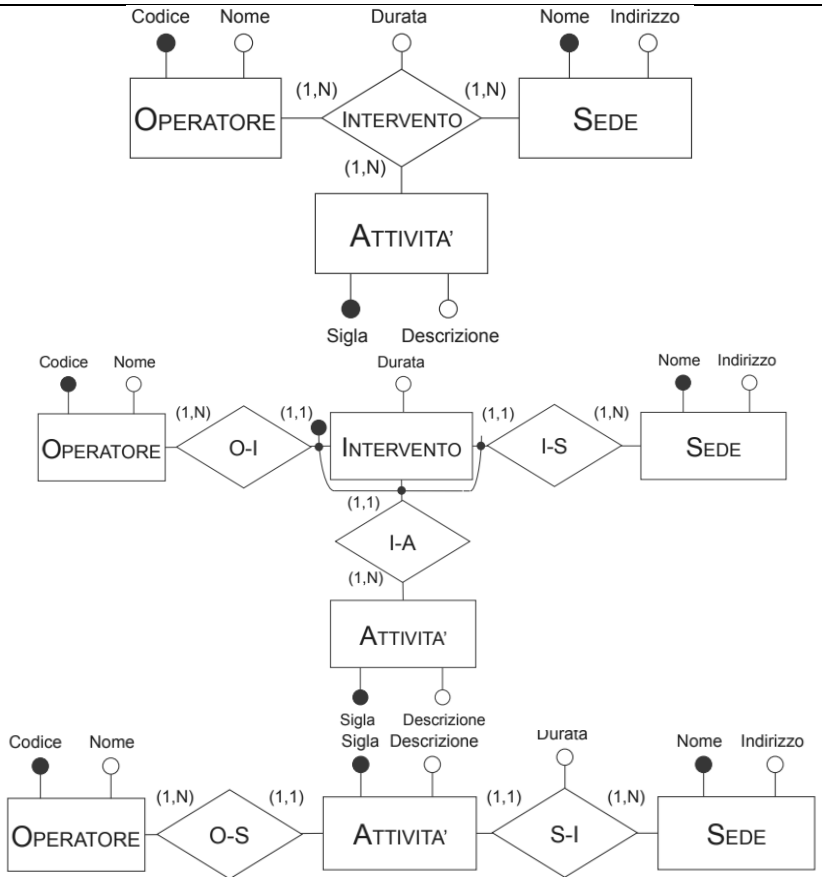
Storicizzazione di relazione



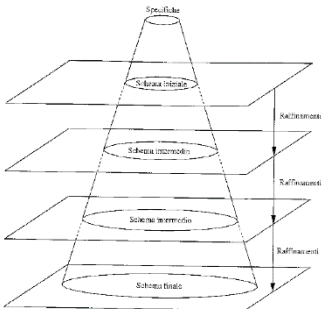
Evoluzione di un concetto



Relazione ternaria

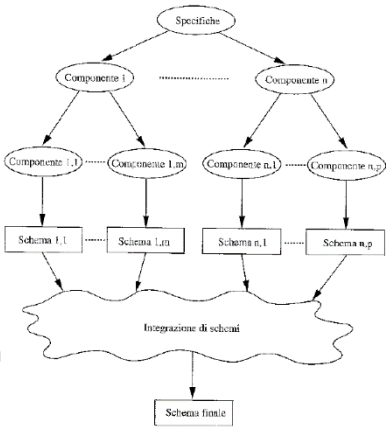


1.4. Strategie di progetto



Top-down:
Schema concettuale prodotto da una serie di raffinamenti a partire da uno schema che descrive concetti molto astratti. Utile quando si possiede sin dall'inizio una visione globale di tutte le componenti.
Da un livello all'altro vengono eseguite delle primitive elementari:

- Definizione degli attributi di un'entità o relazione.
- Reificazione di un attributo o entità.
- Decomposizione di una relazione in due distinte.
- Trasformazione di un'entità in una gerarchia di generalizzazione.



Bottom-up:
Le specifiche iniziali sono divise in componenti via via sempre più piccole finché non diventano elementari. Vengono rappresentati da semplici schemi concettuali che poi vengono fusi nello schema finale.
Primitive di trasformazione:

- Introduzione di una nuova entità o relazione.
- Individuazione nelle specifiche di un legame tra entità una generalizzazione.
- Aggregazione di una serie di attributi in una entità o in una relazione.

Inside-out:
Caso particolare di bottom-up, si individuano inizialmente solo alcuni concetti importanti e poi si procede "a macchia d'olio". Cioè si rappresentano prima i concetti in relazione con i concetti iniziali, per poi muoversi verso quelli più lontani attraverso una "navigazione" tra le specifiche.

Qualità di uno schema concettuale:

- **Correttezza**: deve utilizzare propriamente i costrutti del modello concettuale, possono esserci errori sintattici o semantici.
- **Completezza**: deve rappresentare tutti i dati, e tutte le operazioni possono essere eseguite.
- **Leggibilità**: scelta opportuna dei nomi, disporre i costrutti in griglia scegliendo quelli centrali quelli con più legami, tracciare solo linee perpendicolari, disporre le entità genitori sopra elle entità figlie.
- **Minimalità**: tutte le specifiche sui dati sono rappresentate una volta sola nello schema.

1.5. Metodologia generale

1. **Analisi dei requisiti:**
    - Fonti dei requisiti:
        - Utenti dell'applicazione.
        - Documentazione esistente: moduli, regolamenti interni, procedure aziendali, normative…
        - Realizzazioni preesistenti.
    - Specifiche sui dati (requisiti strutturali):
        - Scegliere il corretto livello di astrazione
        - Standardizzare la struttura delle frasi.
          (*Es. "per <dato> rappresentiamo <insieme di proprietà>")*
        - Evitare frasi contorte.
    - Costruire il glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Partecipante	Partecipante ai corsi. Può essere un dipendente o un professionista.	Studente	Corso, Datore
Docente	Docente dei corsi. Possono essere collaboratori esterni.	Insegnante	Corso
Corso	Corsi offerti. Possono avere varie edizioni.	Seminario	Docente, Partecipante
Datore	Datori di lavoro attuali e passati dei partecipanti ai corsi.	Posto	Partecipante

    - Analizzare i requisiti ed eliminare le ambiguità
        - Individuare sinonimi/omonimi e unificare i termini.
        - Rendere esplicito il riferimento tra termini.
    - Riscrivere le specifiche in gruppi di frasi relative agli stessi concetti

Frasi relative ai datori di lavoro
Relativamente ai datori *di lavoro presenti e passati dei partecipanti, rappresentiamo il nome, l'indirizzo e il numero di telefono.*

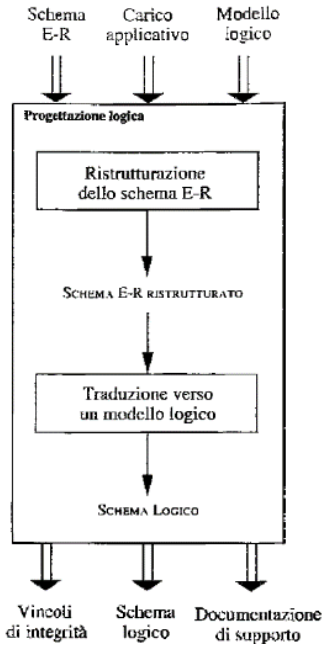
Frasi relative ai corsi
Per i corsi *(circa 200), rappresentiamo il titolo e il codice, le varie edizioni con date di inizio e fine e, per ogni edizione, rappresentiamo il numero di partecipanti e il giorno della settimana, le aule e le ore dove si sono tenute le lezioni.*

2. **Passo base:** Individuare i concetti più rilevanti e rappresentarli in uno schema scheletro.
    - Rappresentazione concettuale dei dati:
        - Proprietà significative/descrittive: entità
        - Struttura semplice, senza proprietà rilevanti: attributo
        - Concetto che associa due o più entità: relazione
        - Uno o più concetti sono casi particolari di un altro: generalizzazione.
3. **Passo di decomposizione** (solo se appropriato o necessario): decomposizione dei requisiti con riferimento ai concetti presenti nello schema scheletro.
4. **Passo iterativo** (per tutti i sotto schemi finché ogni specifica è stata rappresentata):
    - Raffinare i concetti presenti sulla base delle loro specifiche.
    - Aggiungere i nuovi concetti allo schema per descrivere specifiche non ancora descritte.
5. **Passo di integrazione** (solo se fatto passo 3): Integrare i vari sotto-schemi in uno schema generale facendo riferimento allo schema scheletro.
6. **Analisi di qualità:** Verificare correttezza, completezza, minimalità, leggibilità.

2. Progettazione logica

Due fasi:

- **Ristrutturazione dello schema ER**
  - Ingresso: schema concettuale ER, carico applicativo
  - Uscita: schema ER ristrutturato
- **Traduzione verso il modello logico**
  - Verifiche sulla qualità dello schema, ulteriori ottimizzazioni(normalizzazione).
  - Uscita: modello logico, vincoli di integrità e documentazione di supporto.



Analisi delle prestazioni dello schema ER

- Costo di un'operazione
- Occupazione di memoria

Si deve conoscere:

- Volume dei dati:
  - N° occorrenze di ogni entità e associazione dello schema
  - Dimensioni di ogni attributo
- Caratteristiche delle operazioni:
  - Tipo di operazione: interattiva (viene dato cosa cercare) o batch (per ogni…)
  - Frequenza
  - Dati coinvolti

Creare:

- Tabella dei volumi
- Tabella delle operazioni
- Per ogni operazione una tabella degli accessi

Tabella dei volumi

Concetto	Tipo	Volume
Sede	E	10
Dipartimento	E	80
Impiegato	E	2000
Progetto	E	500

Tabella delle operazioni

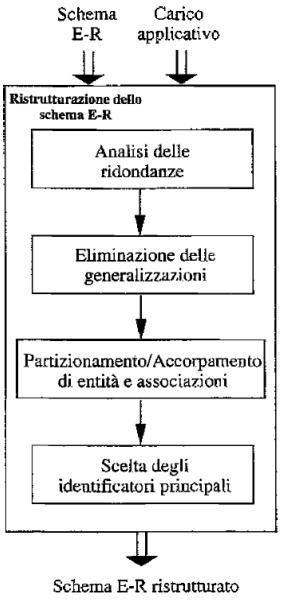
Operazione	Tipo	Frequenza
Op. 1	I	50 al giorno
Op. 2	I	100 al giorno
Op. 3	I	10 al giorno
Op. 4	B	2 a settimana

Tabella degli accessi (Op.2)

Concetto	Costrutto	Accessi	Tipo
Impiegato	Entità	1	L
Afferenza	Relazione	1	L
Dipartimento	Entità	1	L
Partecipazione	Relazione	3	L

2.1. Ristrutturazione dello schema ER

- **Analisi delle ridondanze**: per mantenerle o eliminarle.
- **Eliminazione delle generalizzazioni:** sostituite da altri costrutti.
- **Partizionamento/accorpamento** di entità e associazioni
- Scelta degli **identificatori principali**



2.1.1. Analisi delle ridondanze

Una ridondanza è un dato che può essere derivato da altri dati.

- Attributi derivabili da altri attributi della stessa entità
- Attributi derivabili da attributi di altre entità (o associazioni)
- Attributi derivabili da conteggio di occorrenze
- Associazioni derivabili dalla composizione di altre associazioni in presenza di cicli.

La decisione di mantenere o eliminare una ridondanza va presa confrontando il costo di esecuzione delle operazioni che coinvolgono il dato ridondante e la relativa occupazione in memoria.
Di solito costo lettura=1, scrittura=2.

Tavole degli accessi in presenza di ridondanza

Operazione 1			
Concetto	**Costr.**	**Acc.**	**Tipo**
Persona	E	1	S
Residenza	R	1	S
Città	E	1	L
Città	E	1	S

Operazione 2			
Concetto	**Costr.**	**Acc.**	**Tipo**
Città	E	1	L

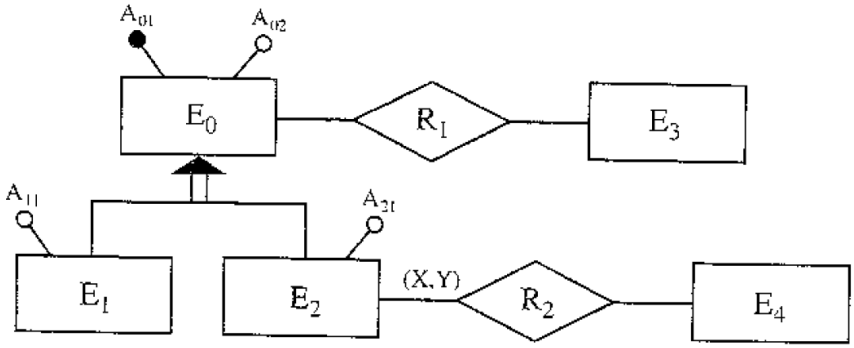
Tavole degli accessi in assenza di ridondanza

Operazione 1			
Concetto	**Costr.**	**Acc.**	**Tipo**
Persona	E	1	S
Residenza	R	1	S

Operazione 2			
Concetto	**Costr.**	**Acc.**	**Tipo**
Città	E	1	L
Residenza	R	5000	L

2.1.2. Eliminazione delle generalizzazioni

I DMBS tradizionali non supportano la rappresentazione delle generalizzazioni, quindi vanno trasformate in altri costrutti, in 3 modi possibili:

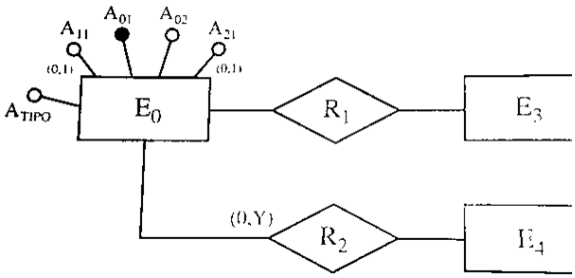


1) Accorpamento delle figlie nel genitore

Attributi e partecipazioni ad associazioni vengono aggiunte all'entità genitore, a cui viene aggiunto un ulteriore attributo che serve a distinguere il "tipo" di una occorrenza, cioè a quale figlia apparteneva, o nel caso di generalizzazione parziale a nessuna di esse.

Gli attributi delle figlie possono assumere valori nulli, in quanto non significativi per alcune occorrenze del genitore, quindi la loro cardinalità sarà (0,1).

Conveniente quando: le operazioni non fanno molta distinzione tra le occorrenze e gli attributi del genitore e figlie.

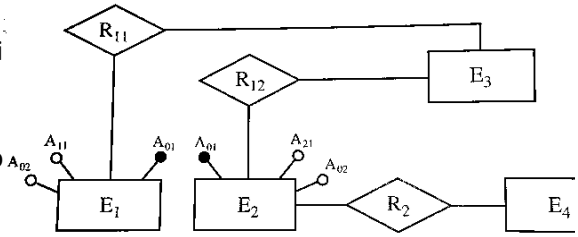


2) Accorpamento del genitore nelle figlie

Gli attributi del genitore, l'identificatore e le relazioni vengono aggiunti alle figlie.

Possibile solo se la generalizzazione è totale.

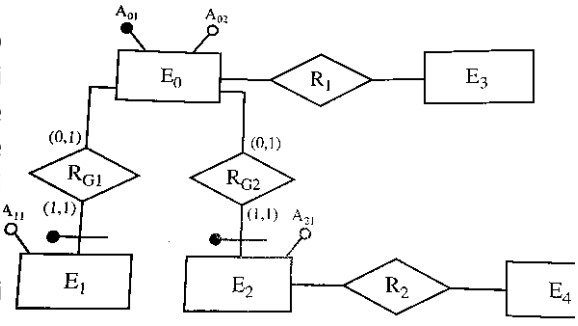
Conveniente quando: ci sono operazioni che si riferiscono solo a occorrenze di una figlia o dell'altra.



3) Sostituzione della generalizzazione con associazioni

La generalizzazione si trasforma in associazioni uno a uno che legano il genitore con le figlie, vanno aggiunti dei vincoli: ogni occorrenza del genitore non può partecipare contemporaneamente alle associazioni sostitutive delle figlie, inoltre, se la generalizzazione è totale, ogni occorrenza del genitore deve partecipare ad un occorrenza di una figlia.

Conveniente quando: la generalizzazione non è totale e ci sono operazioni che si riferiscono solo ai figli o al genitore.



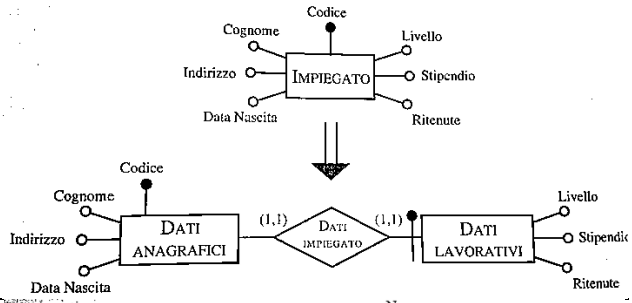
Per le generalizzazioni su più livelli si parte analizzando dal fondo della gerarchia.

2.1.3. Partizionamento/accorpamento di entità e associazioni

Entità e associazioni possono essere partizionati o accorpati per una migliore efficienza delle operazioni. Gli accessi si riducono separando attributi di uno stesso concetto che vengono acceduti da operazioni diverse e raggruppando attributi di concetti diversi che vengono acceduti dalle medesime operazioni.

Partizionamento di entità

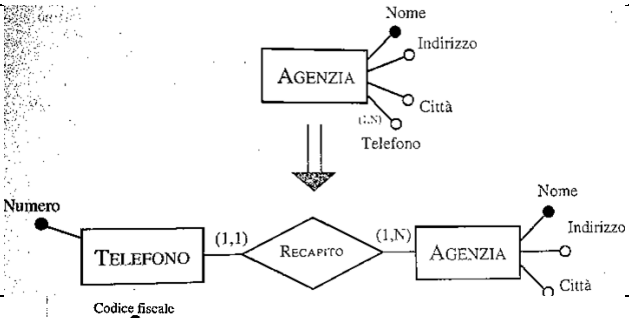
Entità divisa in due perché ci sono operazioni che richiedono solo informazioni riguardanti un aspetto e altre un altro.



Eliminazione di attributi multivalore

Il modello relazionale non permette di rappresentare attributi multivalore.

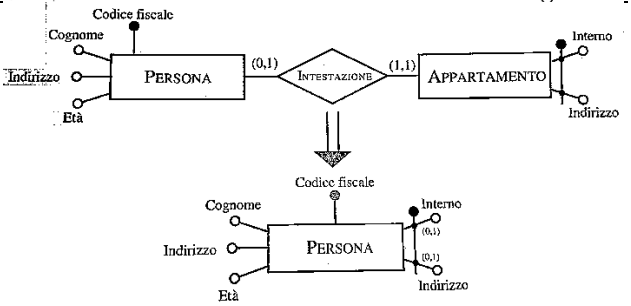
Entità partizionate in due: una con stesso nome e attributi eccetto quello multivalore, e una con il nome dell'attributo multivalore e con un attributo "numero". Se attributo opzionale, cardinalità minima dell'entità originale =0.



Accorpamento di entità

Operazioni su un'entità che spesso richiedono attributi di un'altra.

Tipicamente su associazioni uno a uno.



Altri tipi di partizionamento/accorpamento

Si può decomporre un'associazione tra due entità in due o più associazioni tra le medesime entità, per separare occorrenze dell'associazione originale accedute sempre separatamente.

2.1.4. Scelta degli identificatori principali

I DMBS richiedono di specificare una chiave primaria sulla quale vengono costruite automaticamente delle strutture ausiliarie dette indici. Se ci sono entità con più identificatori bisogna decidere quale tra questi verrà utilizzato come chiave primaria. **Criteri di decisione**:

- Attributi con valori nulli non possono essere identificatori principali.
- Un identificatore composto da uno o pochi attributi è preferibile a quelli composti da più attributi.
- Un identificatore interno è preferibile ad uno esterno.
- Un identificatore che viene utilizzato da molte operazioni è preferibile rispetto agli altri.

Se nessuno dei candidati soddisfa i requisiti, si può introdurre un ulteriore attributo all'entità che conterrà valori speciali (codici) generati appositamente per identificare le occorrenze dell'entità.

2.2. Traduzione verso il modello relazionale

2.2.1. Modello logico relazionale

Schema di relazione(relazione): $\boldsymbol{R(X)}\ con\ X = \{A_1, \dots, A_j, \dots, A_m\} = insieme\ di\ attributi$

Istanza di uno schema: $\boldsymbol{r(X)} = \{t_1, \dots, t_i, \dots, t_n\} = particolare\ insieme\ di\ tuple(insieme\ di\ righe)$

$$con\ t_i = \langle v_{i1}, \dots, v_{ij}, \dots, v_{im} \rangle = vettore\ di\ valori\ con\ v_{ij} \in dom(A_j)$$

Database: insieme delle relazioni R(X).

Istanza di database: insieme delle istanze di relazione r(X).

L'ordine degli attributi una volta stabilito va mantenuto. Ordine delle tuple è ininfluente.

R	A_{11}	...	A_j	...	A_m
t_1	v_{11}	...	v_{1j}	...	v_{1m}
...	...	...	...	...	...
t_i	v_{i1}	...	v_{ij}	...	v_{im}
...	...	...	...	...	...
t_n	v_{n1}	...	v_{nj}	...	v_{nm}

Si possono avere valori non noti, non disponibili attualmente o non pertinenti nel caso di attributi opzionali. Il valore NULL rappresenta queste situazioni.

Vincoli del modello relazionale

Vincoli intra-relazionali:

Vengono verificati sulla singola istanza di uno schema di relazione fino ad arrivare al singolo valore di una tupla.

- **Vincolo di univocità**: una relazione R(X) non ha 2 righe uguali.
- **Vincolo di dominio:** $v_{ij} \in dom(A_j)$ con un possibile limite inferiore e superiore.
- **Vincolo di chiave:**
  Superchiave: sottoinsieme di attributi in cui non ci sono tuple uguali.
  Chiave: è una superchiave minimale, ovvero non si possono togliere attributi senza perdere il vincolo di univocità.
  Vincolo di integrità: sulla chiave primaria non devono esserci valori nulli.

Vincoli inter-relazionali:

- **Vincolo di integrità referenziale:** tra due relazioni per mantenere la consistenza tra i dati contenuti nelle tuple.

Chiave esterna: è un insieme di attributi di una relazione che:

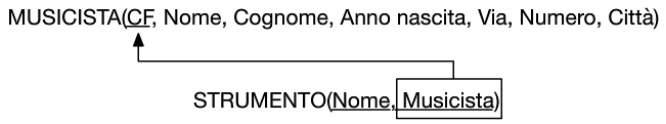
- La cardinalità deve essere uguale a quella di una chiave della tabella riferita e gli attributi devono avere lo stesso dominio.
- I valori degli attributi corrispondenti devono essere uguali oppure NULL.

2.2.2. Passi per la traduzione

1) Tradurre le entità forti che partecipano con cardinalità massima n a tutte le associazioni collegate ad esse.
2) Tradurre le entità forti che partecipano con cardinalità (1,1) ad almeno una delle associazioni collegate.
3) Tradurre le entità forti che partecipano con cardinalità (0,1) ad almeno una delle associazioni collegate.
4) Tradurre le entità deboli.

2.2.3. Traduzione entità forte

- **Nome** della relazione: nome dell'entità
- **Attributi**: attributi dell'entità.
  Attributo composto viene separato nelle sue componenti.
  Attributo opzionale: avrà dei valori NULL nelle istanze.
  Attributo multivalore: nuovo schema
  - Nome: nome attributo
  - Attributi: nome dell'attributo e chiave primaria dell'entità.
  - Chiave primaria: composizione di tutti gli attributi o il nome dell'attributo multivalore.
  - Vincolo di integrità referenziale: tra chiave primaria schema principale e attributo chiave primaria dell'entità sulla relazione.



- **Chiave primaria:** è l'identificatore.
  Se ci sono più attributi identificatori: la chiave primaria viene scelta arbitrariamente tenendo conto del vincolo di integrità (valori NULL non ammessi).

2.2.4. Traduzione associazioni binarie (x,n)-(y,m)

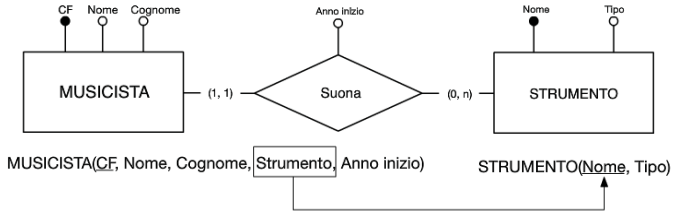
- Trasformare le due entità in schemi relazionali.
- Aggiungere un nuovo schema:
  - Attributi: chiavi primari delle due entità e attributi dell'associazione.
  - Chiave primaria: composizione delle 2 chiavi primarie
- Vincoli di integrità referenziale: tra attributi della chiave composta e le chiavi primarie dei due schemi.

2.2.5. Traduzione associazioni binarie (x,1)-(y,m)

Cioè quando una entità ha cardinalità massima 1. (da non confondere con 4,4 che è un x,n)

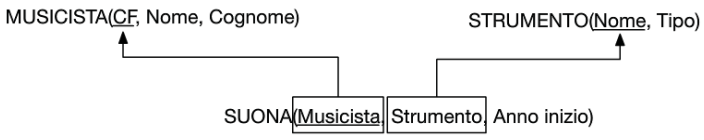
Nel caso partecipazione (1,1):

- Trasformare entità con cardinalità massima m in schema relazionale.
- Entità (1,1) trasformata anch'essa ma con aggiunta di:
  - Attributi: chiave primaria dell'altro e attributi dell'associazione.
  - Chiave primaria: chiave primaria (1,1)
- Vincolo di integrità referenziale: tra chiave primaria (y,m) e attributo chiave primaria aggiunta a (1,1)



Nel caso partecipazione (0,1):

- Se ci sono pochi valori nulli: allora la soluzione (1,1) va bene.
- Altrimenti:
  - Trasformare le due entità in schemi relazionali.
  - Aggiungere un nuovo schema:
    - Attributi: chiavi primarie delle due entità e attributi dell'associazione.
    - Chiave primaria: chiave primaria (1,1)
  - Vincoli di integrità referenziale: tra attributi delle chiavi primarie (di cui una non lo è) e le chiavi primarie dei due schemi.

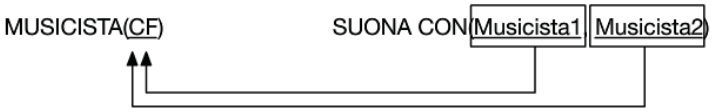


2.2.6. Traduzione associazioni binarie (x,1)-(y,1)

Come traduzione (x,1)-(y,m).

2.2.7. Traduzione associazioni ricorsive (x,n)-(y,m)

- Trasformare entità forte in schema relazionale.
- Aggiungere un nuovo schema per l'associazione:
  - Attributi: chiave primaria entità forte ripetuta 2 volte, e attributi dell'associazione.
  - Chiave primaria: composizione delle 2 chiavi primarie.
- Vincoli di integrità referenziale: tra attributi della composta e le chiave primaria entità forte.



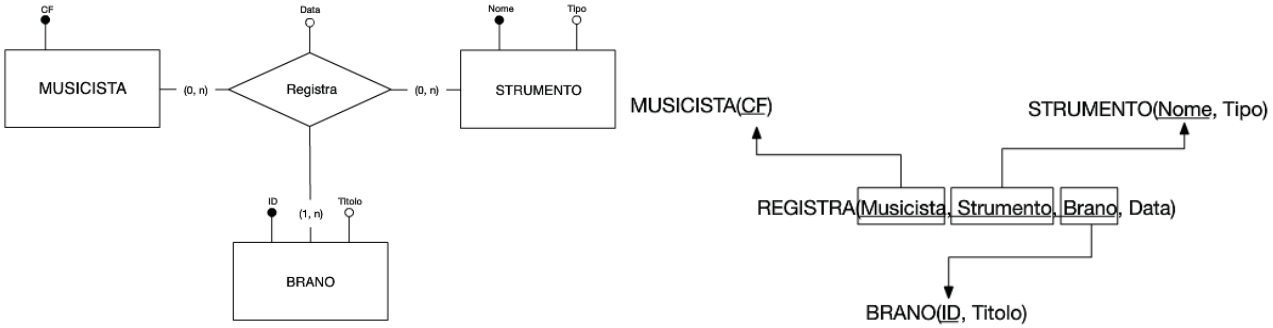
Se uno dei rami ha max=1:

- Trasformare entità forte in schema relazionale aggiungendo una copia della chiave primaria e attributi dell'associazione.
- Vincolo di integrità referenziale: tra attributo aggiunto e chiave primaria schema.

2.2.8. Traduzione associazioni ternarie (x,n)-(y,m)-(z,o)

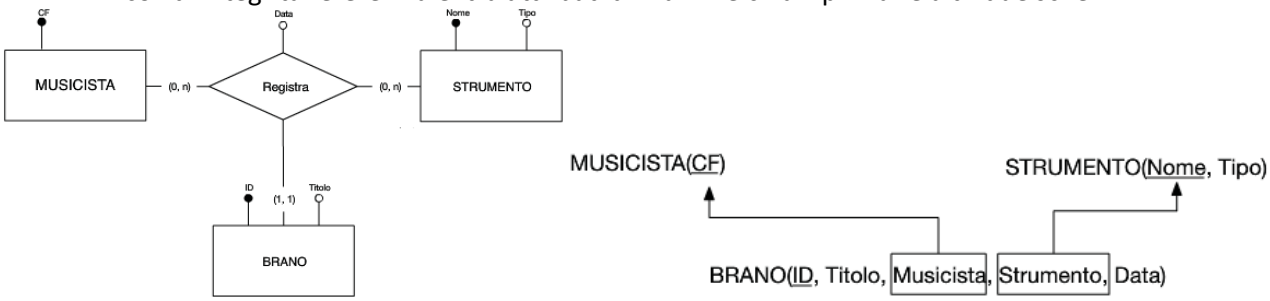
Se max=n per tutte:

- Trasformare le tre entità in schemi relazionali.
- Aggiungere un nuovo schema:
  - Attributi: chiavi primarie delle tre entità e attributi dell'associazione.
  - Chiave primaria: composizione delle 3 chiavi primarie.
- Vincoli di integrità referenziale: tra attributi della chiave composta e le chiavi primarie dei tre schemi.



Se una sola entità max=1:

- Trasformare le 2 entità con cardinalità massima m in schema relazionale.
- Entità max=1 trasformata anch'essa ma con aggiunta di:
  - Attributi della chiave primaria degli altri due.
  - Attributi dell'associazione.
- Vincoli di integrità referenziale: tra attributi di max=1 e chiavi primarie altri due schemi.

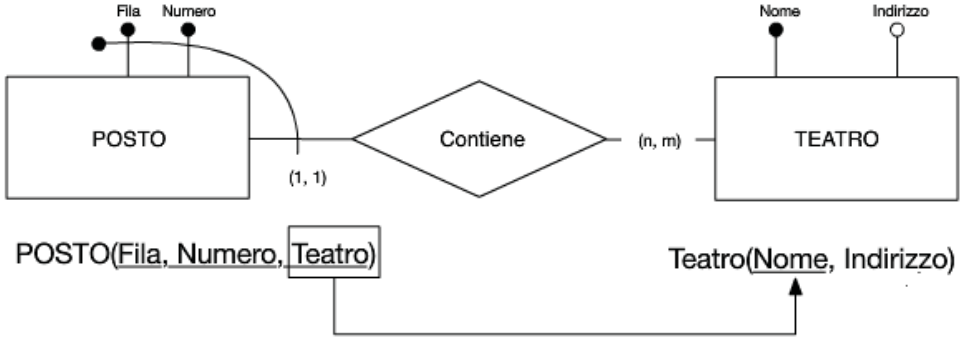


Se più di un'entità ha cardinalità massima 1 si sceglie in quale accorpare le chiavi delle altre due e gli attributi dell'associazione in base alla frequenza di operazioni che vengono fatte.

2.2.9. Traduzione entità debole

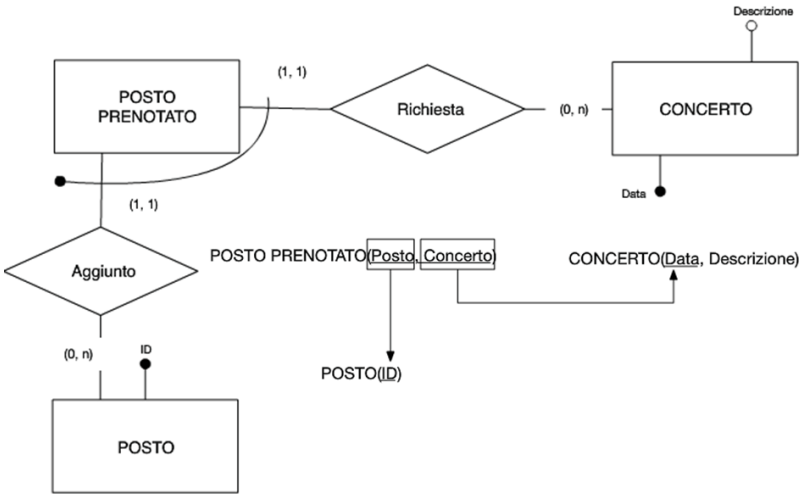
In associazione binaria:

- Trasformare l'entità forte in schema relazionale
- Trasformare l'entità debole aggiungendo:
  - Attributi: chiave primaria dell'entità forte e attributi dell'associazione.
  - Chiave primaria: composizione di chiave debole e forte.
- Vincolo di integrità referenziale: tra chiave primaria e attributo chiave primaria aggiunta.



Se è identificata da più entità o partecipa ad associazione n-aria: basta aggiungere attributi che identificano la chiave primaria nello schema dell'entità debole.

Se è totalmente dipendente da un'entità forte: la chiave primaria è la composizione delle chiavi delle entità forti da cui dipende.



Entità deboli in cascata: partire a tradurre dalle entità forti.

3. Algebra relazionale

Linguaggio procedurale costituito da un insieme di operatori, definiti su relazioni e che restituiscono relazioni come risultati.
Si possono costruire espressioni che coinvolgono più operatori per formulare interrogazioni complesse.

Unione: $r_1(X) \cup r_2(X) = \{t | t \in r_1(X)\ OR\ t \in r_2(X)\}$
Intersezione: $r_1(X) \cap r_2(X) = \{t | t \in r_1(X)\ AND\ t \in r_2(X)\}$
Differenza: $r_1(X) \backslash r_2(X) = \{t | t \in r_1(X)\ AND\ t \notin r_2(X)\}$

3.1. Ridenominazione, Selezione e Proiezione

Ridenominazione: operatore che cambia nome agli attributi e allo schema.

$$\boldsymbol{\rho_{B_1 \dots B_k \leftarrow A_1 \dots A_k}(r(X))}\ oppure\ \boldsymbol{\rho_{nuovoNomeSchema(B_1 \dots B_k)}(r(A_1 \dots A_k))}$$

Selezione: $\boldsymbol{\sigma_F(r(X))} = \{t | t \in r(X), F(t) = vero\}$ il risultato contiene le tuple che soddisfano la condizione F. È un taglio orizzontale, ovvero la tabella risultante avrà lo stesso numero di colonne (stesso grado), ma eventualmente meno righe (cardinalità inferiore).
F: è una formula proposizionale ottenuta combinando i connettivi logici ∧ (and),∨ (or), ¬(not) e condizioni atomiche AθB o Aθc:

- Con θ operatore di confronto $=, \neq, <, >, \leq, \geq$
- A e B attributi in X
- c costante compatibile col dominio di A

Proiezione: $\boldsymbol{\pi_Y(r(X))} = \{t[Y] | t \in r(X)\}$ è un taglio verticale, ovvero si prendono tutte le tuple appartenenti all'insieme di attributi Y sottoinsieme di X. Cioè si prendono solo alcune colonne della tabella.
Può succedere che il numero di righe sia inferiore alla tabella di partenza perché alcune di esse private anche di un solo valore possono essere uguali e quindi collassano in una unica.

Proiezione generalizzata: estensione della proiezione che permette di aggiungere alla lista di attributi alcune colonne contenenti funzioni sugli attributi il cui risultato viene calcolato per ogni tupla. L'attributo aggiunto non ha un nome. Es. $\rho_{Data,Tatro,Ricavo}\left(\boldsymbol{\pi_{Data,Teatro,prezzo*biglietti}(r(X))}\right)$

3.2. Join

Prodotto cartesiano: $r_1(X) \times r_2(Y) = \{ts | t \in r_1(X)\ AND\ s \in r_2(Y)\}$ con ts concatenazione delle tuple. X e Y disgiunti. Combina ogni elemento di una relazione con tutti quelli dell'altra.

Join naturale: $\boldsymbol{r_1(X) \bowtie r_2(Y)} = \{t | t \in r_1(X \cup Y), t[X] \in r_1(X)\ AND\ t[Y] \in r_2(Y)\}$
Correla dati di due tabelle in base a valori uguali su attributi che hanno lo stesso nome.
Il grado (numero di attributi) è pari all'unione di X e Y
La cardinalità:

Musicista	CF	Nome	Cognome
t_1	cf123	Mario	Rossi
t_2	cf456	Alberto	Bianchi
t_3	cf678	Alessandro	Verdi

Suona	CF	Strumento
t_1	cf123	Piano
t_2	cf123	Violino
t_3	cf456	Sax
t_4	cf678	Violino

	CF	Nome	Cognome	Strumento
t_1	cf123	Mario	Rossi	Piano
t_2	cf123	Mario	Rossi	Violino
t_3	cf456	Alberto	Bianchi	Sax
t_3	cf678	Alessandro	Verdi	Verdi

- Se fatto su attributi che non sono superchiave possono non esserci valori uguali sulle tuple e la relazione è vuota.
- Se eseguito su una superchiave di R1(X), cardinalità è ≤|R2(Y)|
- Se X e Y sono disgiunti equivale ad un prodotto cartesiano.
- Se X=Y è equivalente ad un'intersezione.

È commutativo e associativo (si può scambiare l'ordine dei membri).

Theta-Join: $\boldsymbol{r_1(X) \bowtie_F r_2(Y)} = \sigma_F\big(r_1(X) \times r_2(Y)\big)$
Join condizionale: theta-join in cui F può essere qualsiasi predicato logico.
Equi-join: theta-join in cui F può essere solo di uguaglianza.
Join-naturale: equi-join in cui le condizioni di uguaglianza sono specificate su tutti gli attributi con lo stesso nome.

Join completo: quando tutte le tuple di entrambe le tabelle partecipano al risultato del join.
Le tuple assenti vengono chiamate "pendenti" (dangling). Sono la conseguenza di un join fatto su attributi che non sono chiave o sui quali non è presente un vincolo di integrità referenziale.

Join esterno:

- **Left Join:** $\boldsymbol{r_1(X) \rtimes r_2(Y)}$ si tengono tutte le tuple pendenti della tabella a sinistra.
- **Right Join:** $\boldsymbol{r_1(X) \ltimes r_2(Y)}$ si tengono tutte le tuple pendenti della tabella a destra.
- **Outer Join:** $\boldsymbol{r_1(X) \rtimes\ltimes r_2(Y)}$ si tengono tutte le tuple pendenti.

3.3. Aggregazione e Raggruppamento

Operazioni di aggregazione: SUM, MAX, MIN, AVARAGE, COUNT (conta il numero delle tuple nella relazione).
$\boldsymbol{\mathcal{F}_{f_1(A_1),\dots}\big(r(X)\big)}$ risultato è una relazione che ha numero di attributi pari al numero di funzioni calcolate e cardinalità 1.

Raggruppamento: permette di calcolare le funzioni di aggregazione su gruppi di tuple definite su un insieme di attributi della relazione invece che su tutta.

$$\boldsymbol{_{A_i,..}\mathcal{F}_{f_1(A_1),\dots}\big(r(X)\big)}$$

A sinistra ci sono gli attributi che indicano alle funzioni di aggregazione di calcolare il risultato all'interno di ciascun gruppo.
La relazione calcolata ha grado pari al numero di funzioni calcolate + numero attributi di raggruppamento e cardinalità pari al numero di gruppi creati (cioè il numero di elementi distinti sugli attributi).

Es. $_{(Cognome,Strumento)}\mathcal{F}_{COUNT(*)}(Artirsta)$

ARTISTA	CF	Nome	Cognome	Strumento
t_1	cf123	Mario	Rossi	Piano
t_2	cf120	Mario	Rossi	Violino
t_3	cf456	Alberto	Bianchi	NULL
t_4	cf678	Alessandro	Verdi	Sax
t_5	cf234	Stefano	Ferrari	NULL
t_6	cf678	Giorgio	Romano	Piano
t_7	cf567	Emanuele	Greco	Chitarra
t_8	cf013	Andrea	Bianchi	NULL
t_9	cf313	Ivano	Rossi	Piano

	Cognome	Strumento	
t_1	Rossi	Piano	2
t_2	Rossi	Violino	1
t_3	Bianchi	NULL	2
t_4	Verdi	Sax	1
t_5	Ferrari	NULL	1
t_6	Romano	Piano	1
t_7	Greco	Chitarra	1

3.4. Valori nulli

Logica a tre valori: T, F, U (sconosciuto, valore NULL, viene trattato come falso).

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

NOT	
T	F
F	T
U	U

A IS NULL: restituisce vero se A è null.

Selezione con valori nulli: utilizza la logica a tre valori e scarta le tuple che restituiscono U.
Proiezione con valori nulli: non cambia.
Operazioni insiemistiche con valori nulli: si comportano come si i valori NULL fossero confrontabili.
Operazioni di Join con valori nulli: utilizza la logica a tre valori
Funzioni di aggregazione con valori nulli:

- SUM, MAX, MIN, AVARAGE: si comportano come se NULL non fosse presente, il risultato viene calcolato sul sottoinsieme di valori che non sono nulli.
- COUNT: se si fa su un insieme di attributi non superchiave, rimuove i duplicati prima di calcolare.

Funzione di raggruppamento:

- Se gli attributi sono superchiave: si avranno gruppi di cardinalità 1.
- Se non sono superchiave e ammettono valori nulli: si generano tuple distinte e conta quante tuple ci sono in ogni raggruppamento.

Il raggruppamento su un solo attributo che ha dei valori nulli: ha una riga in più "NULL" con il numero di NULL presenti.

3.5. Equivalenza di espressioni algebriche

- Atomizzazione delle selezioni: $\boldsymbol{\sigma_{F_1 \wedge F_2}(E) \equiv \sigma_{F_1}\left(\sigma_{F_2}(E)\right)}$
- Idempotenza delle proiezioni: $\boldsymbol{\pi_X(E) \equiv \pi_X\left(\pi_{XY}(E)\right)}$ con E definita su attributi che contiene Y e X
- Anticipazione della selezione rispetto al join (pushing selection down):
  $\boldsymbol{\sigma_F(E_1 \bowtie E_2) \equiv E_1 \bowtie \sigma_F(E_2)}$ con F solo su attributi di E2
- Anticipazione della proiezione rispetto al join (pushing projections down):
  $Se\ Y_2 \subseteq X_2\ e\ Y_2 \supseteq (X_1 \cap X_2)\ cio\grave{e}\ gli\ attributi\ X_2 - Y_2\ non\ sono\ coinvolti\ nel\ join$
  $\boldsymbol{\pi_{X_1Y_2}(E_1 \bowtie E_2) \equiv E_1 \bowtie \pi_{Y_2}(E_2)}$
  $\boldsymbol{\pi_Y(E_1 \bowtie_F E_2) \equiv \pi_Y\left(\pi_{Y_1}(E_1) \bowtie_F \pi_{Y_2}(E_2)\right)}$
  In sostanza si possono cancellare subito da ogni relazione gli attributi che non compaiono nel risultato finale e non sono coinvolti nel join.
- Inglobamento di una selezione in un prodotto cartesiano a formare un join:
  $\boldsymbol{\sigma_F(E_1 \bowtie E_2) \equiv E_1 \bowtie_F E_2}\ con\ X_1 \cap X_2 = 0$
- Distributività della selezione rispetto l'unione: $\boldsymbol{\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)}$
- Distributività della selezione rispetto alla differenza: $\boldsymbol{\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)}$
- Distributività della proiezione rispetto all'unione: $\boldsymbol{\pi_X(E_1 \cup E_2) \equiv \pi_X(E_1) \cup \pi_X(E_2)}$
- $\boldsymbol{\sigma_{F_1 \vee F_2}(R) \equiv \sigma_{F_1}(R) \cup \sigma_{F_2}(R)}$
- $\boldsymbol{\sigma_{F_1 \wedge F_2}(R) \equiv \sigma_{F_1}(R) \cap \sigma_{F_2}(R) \equiv \sigma_{F_1}(R) \bowtie_F \sigma_{F_2}(R)}$
- $\boldsymbol{\sigma_{F_1 \wedge \neg F_2}(R) \equiv \sigma_{F_1}(R) - \sigma_{F_2}(R)}$
- $\boldsymbol{E \bowtie (E_1 \cup E_2) \equiv (E \bowtie E_1) \cup (E \bowtie E_2)}$

4. SQL

Un DBMS fornisce diversi linguaggi per implementare una base di dati:

- Linguaggio di definizione dei dati (DDL, Data Definition Language): usato per specificare lo schema concettuale, che consente di descrivere le entità, le associazioni e loro vincoli.
- Linguaggio di definizione della memorizzazione (SDL, Storage Definition Language): usato per specificare lo schema interno, che consente di de-scrivere la struttura di memorizzazione fisica della base di dati.
- Linguaggio di definizione delle viste (VDL, View Definition Language): usato per specificare schemi esterni o viste d'utente, che consentono di specificare la parte della base di dati di interesse per un gruppo di utenti.
- Linguaggio di manipolazione dei dati (DML, Data Manipulation Language): usato per la manipolazione dei dati, ad esempio per l'inserimento, la cancellazione, la modifica ed il reperimento dei dati.

SQL è DDL e DML.

Leggenda
In corsivo parametri.
<,> isolare un termine della sintassi
[,] termini all'interno sono opzionali (0 volte o 1)
{,} termini possono comparire un numero arbitrario di volte (0 o n)
| deve essere scelto uno solo tra i termini

4.1. Definizione dei dati

Domini elementari

`CHARACTER` [`VARYING`] [(*lunghezza*)] [`CHARACTER SET` *NomeFamigliaCaratteri*]
Possono essere scritti anche come: `VARCHAR`(n) e `CHAR`(n) con n lunghezza massima.
`TEXT`: stringa di lunghezza arbitraria.
Se lunghezza non specificata rappresenta un singolo carattere.
Se su `CHAR` viene inserito una stringa più corta vengono aggiunti degli spazi in coda.

Tipi numerici esatti

`NUMERIC` [(*Precisione* [, *Scala*])]
`DECIMAL` [(*Precisione* [, *Scala*])]
`INTEGER` (4 byte $[-2^{31},2^{31}-1]$)
`SMALLINT` (2 byte $[-2^{15},2^{15}-1]$)
`BIGINT` (8 byte $[-2^{63},2^{63}-1]$)
Precisione: indica il numero di cifre significative.
Scala: indica il numero di cifre a destra della virgola. Se non specificata vale 0.

Tipi numerici approssimati (Virgola mobile. Mantissa*10^esponente.)
`float` [(*Precisione*)]
`reala`
`double precision`
Precisone: n° di cifre della mantissa

Istanti temporali
`DATE`
`TIME` [(*Precisione*)] [`WITH TIME ZONE`]
`TIMESTAMP` [(*Precisione*)] [`WITH TIME ZONE`]

`DATE` ammette i campi `YEAR`, `MONTH` e `DAY`.

TIME ammetti i campi HOUR, MINUTE e SECOND.
TIMESTAMP gli ammette tutti.

Precisione: n° cifre decimali per rappresentare le frazioni di secondo.
Se WITH TIME ZONE è specificata, allora si può accedere a due campi TIMEZONE_HOUR e TIMEZONE_MINUTE che rappresentano la differenza di fuso orario tra l'ora locale e l'ora universale.

Intervalli temporali
INTERVAL *PrimaUnitàDiTempo* [TO *UltimaUnitàDiTempo*]
PrimaUnitàDiTempo e *UltimaUnitàDiTempo* definiscono le unità di misura da utilizzare, anche con una precisione.
Es. INTERVAL YEAR(5) TO MONTH rappresenta intervalli fino a 99.999 anni e 11 mesi.

BOOLEAN: logica a tre valori TRUE, FALSE e NULL.

BLOB e CLOB
Rappresentano oggetti di grandi dimensioni binari (BLOB) o caratteri (CLOB), ma non possono essere utilizzati come criterio per la selezione.

Definizione di domini
CREATE DOMAIN *NomeDominio* AS *TipoDiDato* [*ValoreDiDefault*] [*Vincolo*]
Es. CREATE DOMAIN gender AS CHAR(1) CHECK (VALUE='M' OR VALUE='F')
(Non supportato in MySql)

Definizione di schema
CREATE SCHEMA [*NomeSchema*] [[AUTHORIZATION] *Autorizzazione*] { *DefElementoSchema* }
Autorizzazione: nome dell'utente proprietario dello schema, se omesso si assume che sia chi lancia il comando.
NomeSchema: può essere omesso, si assume come nome il nome del proprietario.

Definizione di database
CREATE DATABASE *NomeDatabase*

Definizione di tabelle
CREATE TABLE *NomeTabella* (*NomeAttributo Dominio* [*ValoreDiDefault*] [*Vincoli*]
{ , *NomeAttributo Dominio* [*ValoreDiDefault*] [*Vincoli*] }
AltriVincoli
)

Specifica valori di default
DEFAULT < *GenericoValore* | USER | NULL >

Vincoli intrarelazionali
NOT NULL
Es. Cognome VARCHAR(20) NOT NULL

CHECK
Attributo può assumere solo un numero finito di valori.
Es. Sesso CHAR(1) CHECK (Sesso='M' OR Sesso='F')

UNIQUE
Impone che un insieme di attributi siano (super)chiave, cioè non abbiano righe uguali, a eccezione del valore nullo.
Su un solo attributo: `Cognome VARCHAR(20) UNIQUE`
Su un insieme di attributi: `UNIQUE (Cognome, Nome)`

PRIMARY KEY
Specifica la chiave primaria, gli attributi che fanno parte della chiave primaria non possono assumere valori nulli.
Su un solo attributo: `Cognome VARCHAR(20) PRIMARY KEY`
Su un insieme di attributi: `PRIMARY KEY (Cognome, Nome)`

Vincoli interrelazionali

Vincolo integrità referenziale

FOREIGN KEY
Crea un legame tra i valori di un attributo della tabella in cui è definito(interna) e i valori di un attributo di un'altra tabella (esterna).
Il vincolo impone che per ogni riga della tabella interna, il valore dell'attributo specificato, se diverso da NULL, sia presente nelle righe della tabella esterna. L'attributo della tabella esterna deve essere UNIQUE o PRIMARY KEY.

Su un solo attributo: *NomeAttrInt* … REFERENCES *NomeTabEsterna*(*NomeAttrEst*)
Su un insieme di attributi: FOREIGN KEY (*Attr*$_1$, …) REFERENCES *NomeTabEsterna*(*Attr*$_1$, …)

Quando viene rilevata una violazione del vincolo sulla tabella esterna sono offerte diverse alternative (per la tabella interna il comando viene semplicemente rifiutato):
La tabella esterna viene considerata la principale (master) alle cui variazioni la tabella interna (slave) deve adeguarsi.
Per le operazioni di modifica (ON UPDATE):

- CASCADE: nuovo valore dell'attributo della tabella esterna viene riportato su tutte le tabelle interne.
- SET NULL: viene assegnato il valore nullo su tutte le tabelle interne al posto del valore modificato.
- SET DEFAULT: viene assegnato il valore di default su tutte le tabelle interne al posto del valore modificato.
- NO ACTION: l'azione di modifica non viene consentita.

Per le operazioni di cancellazione (ON DELETE):

- CASCADE: tutte le righe delle tabelle interne referenti vengono cancellate.
- SET NULL: viene assegnato il valore nullo su tutte le tabelle interne al posto del valore cancellato.
- SET DEFAULT: viene assegnato il valore di default su tutte le tabelle interne al posto del valore cancellato.
- NO ACTION: l'azione di cancellazione non viene consentita.

ON < DELETE | UPDATE > < CASCADE | SET NULL | SET DEFAULT | NO ACTION >

Modifica degli schemi

```
ALTER DOMAIN NomeDominio < SET DEFAULT ValoreDefault |
                           DROP DEFAULT |
                           ADD CONSTRAINT DefVincolo |
                           DROP CONSTRAIN NomeVincolo >

ALTER TABLE NomeTabella <
   ALTER COLUMN NomeAttributo < SET DEFAULT NuovoDefault | DROP DEFAULT > |
   ADD CONSTRAINT DefVincolo |
   DROP CONSTRAINT NomeVincolo |
   ADD COLUMN DefAttributo |
   DROP COLUMN NomeAttributo >
```

Quando si definisce un vincolo questi deve essere soddisfatto già dai dati presenti.

`DROP` < `SCHEMA` | `DOMAIN` | `TABLE` | `VIEW` | `ASSERTION` > *NomeElemento* [`RESTRICT` | `CASCADE`]

`RESTRICT`: è l'opzione di default e specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti, cioè uno schema non viene rimosso se contiene tabelle o altri oggetti.
`CASCADE`: tutti gli oggetti specificati vengono rimossi, attivando una reazione a catena se necessario.

Una base di dati contiene due tipi di tabelle: quelle che contengono i dati e quelle che contengono i metadati (dati che descrivono i dati). L'insieme di metadati viene chiamato catalogo.

4.2. Interrogazioni

In SQL si specifica l'obiettivo dell'interrogazione e non il modo in cui ottenerlo (dichiaratività).
L'interrogazione SQL per essere eseguita viene passata all'ottimizzatore di interrogazioni (query optimizer) che la analizza e la traduce in una equivalente nel linguaggio procedurale interno al sistema.

```
SELECT ListaAttributi
FROM ListaTabelle
[ WHERE Condizione ]
[ GROUP BY ListaAttrRaggrup ]
[ HAVING CondizioneDiRaggrup ]
[ ORDER BY ListaAttr ]

SELECT  AttrEspr [ [AS] Alias ] {, AttrEspr [ [AS] Alias ] }
FROM  Tabella [ [AS] Alias ] {, Tabella [ [AS] Alias ] }
[ WHERE Condizione ]
```

Operazione di selezione: `SELECT * FROM` *Tabella* `WHERE` *CondizioneDiSelezione*
Operazione di proiezione: `SELECT DISTINCT` *ListaAttributiDiProiez* `FROM` *Tabella*
Operazione di ridenominazione: *NomeTabella/Attrib* `AS` *NuovoNome*

Sul prodotto cartesiano delle tabelle elencate nel `FROM`, vengono applicate le condizioni di `WHERE`.

Carattere speciale `*`, che seleziona tutti gli attributi delle tabelle del `FROM`.
Possono apparire espressioni sul valore degli attributi: Es. `SELECT StipendioAnnuale/12 FROM …`

Operatore punto identifica le tabelle da cui vengono estratti gli attributi, necessario se ci sono attributi di tabelle diverse con gli stessi nomi. Es. `Professore.nome, Scuola.nome`

WHERE ammette un'espressione booleana combinando predicati AND, OR, NOT. Ciascun predicato usa gli operatori =, <>, <, >, <= e >=.
Un predicato può restituire true, false o unknown se il valore è null.

Operatore LIKE: confronto tra stringhe, _ (rimpiazza 1 carattere), % (rimpiazza 0 o più caratteri).

IS NULL: restituisce true se il valore è nullo. *Attributo* IS [NOT] NULL

Interpretazione formale delle interrogazioni SQL: Pag 113

Duplicati: in algebra una tabella viene vista come un insieme di elementi (tuple) diversi tra loro, in SQL si possono avere righe uguali.
SELECT DINSTINCT: elimina i duplicati dai risultati. SELECT ALL invece li mantiene.

Join

SELECT *AttrEspr* [[AS] *Alias*] {, *AttrEspr* [[AS] *Alias*] }
FROM *Tabella* [[AS] *Alias*] { [*TipoJoin*] JOIN *Tabella* [[AS] *Alias*] ON *CondizioneDiJoin*}
[WHERE *AltraCondizione*]

Tipo join: INNER (di default, rappresenta il **theta join**), RIGHT OUTER, LEFT OUTER, FULL OUTER, NATURAL.
Non è consigliato utilizzare il join naturale in quanto il suo comportamento varia profondamente in base allo schema delle tabelle.

Operatori aggregati

COUNT (< * | [DISTINCT | ALL] *ListaAttributi* >)
* restituisce il numero di righe, DISTINCT il numero di diversi valori degli attributi in *ListaAttributi*; ALL il numero di righe diverse da NULL. ALL è di default.

< SUM | MAX | MIN | AVG > ([DISTINCT | ALL] *AttrEspr*)
Si applicano sulle righe che soddisfano WHERE.
SUM: restituisce la somma dei valori posseduti dall'espressione.
MAX e MIN: restituiscono il valore min e max
AVG: restituisce la media.
DISTINCT elimina le righe doppie, ALL le mantiene.
Non tengono conto delle righe con attributo nullo.

Es. SELECT SUM(Stipendio) FROM Impiegato WHERE Dipart = 'Amministrazione'

Interrogazioni con raggruppamento
GROUP BY specifica come dividere le tabelle in sottoinsiemi. Ammette come argomento un insieme di attributi e raggrupperà le righe che possiedono gli stessi valori per questo insieme di attributi.

Predicati sui gruppi
HAVING ogni sottoinsieme costruito da GROUP BY fa parte del risultato solo se il predicato di HAVING viene soddisfatto.
Se non è presente il GROUP BY le righe vengono viste come un raggruppamento unico.
Solo predicati in cui compaiono operatori aggregati devono essere argomento di HAVING.

Interrogazioni di tipo insiemistico
InterrogSQL < UNION | INTERSECT | EXCEPT > [ALL] *AltraInterrog*
Eliminano di default i duplicati, se si vogliono conservare su usa ALL.
Es. SELECT Nome FROM Impiegato UNION SELECT Cognome FROM Impiegato

Interrogazioni nidificate
Si può confrontare un valore (ottenuto come risultato di una espressione valutata sulla singola riga) con il risultato dell'esecuzione di un'interrogazione SQL.
Si possono estendere, con ALL o ANY, gli operatori di confronto (=, <>, <, >, <= e >=).
ANY specifiche che la riga soddisfa la condizione se risulta vero il confronto tra il valore dell'attributo per la riga e almeno uno degli elementi dell'interrogazione.
ALL specifica che la riga soddisfa la condizione solo se tutti gli elementi restituiti dall'interrogazione nidificata rendono vero il confronto.
Per rappresentare l'appartenenza/esclusione rispetto ad un insieme si possono usare in (=ANY) e NOT IN (<>ALL).

Interrogazioni nidificate complesse
Passaggio di binding: quando una variabile definita nella query più esterna viene utilizzata nella query interna. Essa è usabile solo nell'abito in cui è definita e in sue query nidificate.
EXISTS ammette come parametro un'interrogazione nidificata e restituisce vero solo se l'interrogazione fornisce un risultato non vuoto.

Ordinamento
ORDER BY *AttrDiOridinam* [ASC | DESC] { , *AttrDiOridinam* [ASC | DESC] }
Le righe vengono ordinate in base al primo attributo nell'elenco, per righe che hanno lo stesso valore si guarda il secondo e così via.

4.3. Modifica dei dati

Inserimento
INSERT INTO *NomeTabella* [*ListaAttributi*] < VALUES (*ListaValori*) | *SelectSQL* >
Se non vengono specificati i valori di alcuni attributi vengono assegnati valori di default o nulli.

Cancellazione
DELETE FROM *NomeTabella* [WHERE *Condizione*]

DROP TABLE *NomeTabella* [CASCADE] [RESTRICT]
CASCADE elimina anche tutte le viste e tabelle che gli fanno riferimento.
RESTRICT: il comando fallisce se ci sono righe all'interno della tabella.

Modifica
UPDATE *NomeTabella*
SET *Attributo* = < *Espressione* | *SelectSQL* | NULL | DEFAULT >
{ , *Attributo* = < *Espressione* | *SelectSQL* | NULL | DEFAULT > }
[WHERE *Condizione*]

5. Normalizzazione

Le forme normali certificano la qualità di uno schema relazionale. Quando una relazione non soddisfa una forma normale, allora presenta delle ridondanze, ma è possibile applicare un procedimento di normalizzazione.

5.1. Anomalie e ridondanze

Ridondanza: se un impiegato partecipasse a 20 progetti lo stipendio verrebbe ripetuto 20 volte

Anomalia di aggiornamento: se lo stipendio di un impiegato cambia bisogna andare a modificarlo in tutte le tuple.

Anomalia di cancellazione: se un impiegato interrompe la partecipazione a tutti i progetti senza lasciare l'azienda, le tuple vengono eliminate senza lasciare traccia del suo nome.

Anomalia di inserimento: se arriva un nuovo impiegato, non è possibile inserire le sue informazioni finché non partecipa ad un progetto.

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20 000	Marte	2000	tecnico
Verdi	35 000	Giove	15 000	progettista
Verdi	35 000	Venere	15 000	progettista
Neri	55 000	Venere	15 000	direttore
Neri	55 000	Giove	15 000	consulente
Neri	55 000	Marte	2000	consulente
Mori	48 000	Marte	2000	direttore
Mori	48 000	Venere	15 000	progettista
Bianchi	48 000	Venere	15 000	progettista
Bianchi	48 000	Giove	15 000	direttore

5.2. Dipendenze funzionali

Dipendenza funzionale $Y \rightarrow Z$: è un particolare vincolo di integrità che descrive legami di tipo funzionale tra gli attributi di una relazione.

Data una relazione r su uno schema R(X) e due sottoinsiemi di attributi non vuoti Y e Z di X, esiste su r una dipendenza funzionale tra Y e Z, se per ogni coppia di tuple t_1 t_2 di r aventi gli stessi valori sugli attributi di Y, risulta che hanno gli stessi valori anche su Z.

Z composto dagli attributi $A_1,\dots,A_k$. Una relazione soddisfa Y→Z se e solo se soddisfa tutte le k dipendenze $Y\rightarrow A_1,\dots,Y\rightarrow A_k$.

Dipendenza Y→A **non banale**: se A non compare tra gli attributi di Y.

Esiste una dipendenza funzionale tra la chiave di uno schema r e ogni altro suo attributo.

Il vincolo di dipendenza funzionale generalizza il vincolo di chiave. Cioè Y→Z su R(X) degenera nel vincolo di chiave se Y∪Z=X. In tal caso Y è superchiave di R(X).

5.3. Forma normale di Boyce e Codd

Una relazione r è in forma di Boyce e Codd se per ogni dipendenza funzionale (non banale) X→A definita su di essa, X contiene una chiave K di r, cioè X è superchiave per r.

Anomalie e ridondanze non si presentano in questo tipo di forma normale.

5.4. Decomposizioni

Se una relazione rappresenta più concetti indipendenti va decomposta in relazioni più piccole, una per ogni concetto.
A ogni dipendenza corrisponde una relazione, la cui chiave sarà il primo membro di ciascuna dipendenza, così da soddisfare la forma di Boyce e Codd.

5.4.1. Proprietà

5.4.1.1. Decomposizione senza perdita

Data una relazione r su un insieme di attributi X, $se\ X_1 \cup X_2 = X\ con\ X_1, X_2 \subseteq X$, allora il join delle due relazioni ottenute per proiezione da r su X_1 e X_2 è una relazione che contiene tutte le tuple di r, più eventualmente altre dette "spurie".

r si **decompone** senza perdita su X_1 e X_2: se il join delle due proiezioni è uguale ad r stessa, cioè non contiene spurie. Ovvero quando: $X_0\ soddisfa\ X_0 \rightarrow X_1\ oppure\ X_0 \rightarrow X_2\ con\ X_0 = X_1 \cap X_2$.

Quindi, quando decomponiamo una relazione in due parti, se l'insieme degli attributi comuni è chiave per almeno una delle due relazioni, allora siamo certi che tutte le istanze della relazione si decompongono senza perdita.

Garantisce che le informazioni nella relazione originaria siano ricostruibili con precisione.

5.4.1.2. Conservazione delle dipendenze

In ogni decomposizione, ciascuna delle dipendenze funzionali dello schema originario dovrebbe coinvolgere attributi che compaiono tutti insieme in uno degli schemi decomposti.
In questo modo, è possibile garantire, sullo schema decomposto. il soddisfacimento degli stessi vincoli il cui soddisfacimento è garantito dallo schema originario.

Garantisce che le relazioni decomposte hanno la stessa capacità della relazione originaria di rappresentare i vincoli di integrità.

5.5. Terza forma normale

Esistono schemi che violano la forma di Boyce e Codd per i quali non esiste alcuna decomposizione che conservi le dipendenze. Quindi si ricorre ad una forma meno restrittiva, che però ha il vantaggio di essere sempre ottenibile.

r è in terza forma normale se, per ogni sua dipendenza funzionale X→A, almeno una delle seguenti condizioni è vera:

- X contiene una chiave K di r
- A appartiene ad almeno una chiave di r

Se una relazione ha solo una chiave, allora le due forme normali coincidono.

Altre forme normali:

- **Prima forma normale:** gli attributi delle relazioni sono definiti su valori atomici e non su valori complessi come insiemi o relazioni. Ovvero:
  - Tutte le righe della tabella contengono lo stesso numero di colonne;
  - Gli attributi rappresentano informazioni elementari;
  - I valori che compaiono in una colonna sono dello stesso tipo, cioè appartengono allo stesso dominio;
  - Ogni riga è diversa da tutte le altre, cioè non ce ne possono essere due con gli stessi valori nelle colonne;
  - L'ordine con il quale le righe compaiono nella tabella è irrilevante

La prima forma normale è già parte integrante della definizione formale di relazione nel modello relazionale.

- **Seconda forma normale:** variante debole della terza. Una relazione è in secondo forma se su di essa non sono definite dipendenze parziali, cioè dipendenze fra un sottoinsieme proprio della chiave e altri attributi.
  Ovvero quando è in prima forma normale e tutti i suoi attributi non-chiave dipendono dall'intera chiave, cioè non possiede attributi che dipendono soltanto da una parte della chiave. La seconda forma normale elimina la dipendenza parziale degli attributi dalla chiave e riguarda il caso di relazioni con chiavi composte, cioè formate da più attributi.

5.6. Teoria delle dipendenze e normalizzazione

5.6.1. Implicazione di dipendenze funzionali

Un insieme di dipendenze funzionali F **implica** un'altra dipendenza f se ogni relazione che soddisfa tutte le dipendenze in F soddisfa anche f.

Dati uno schema di relazione R(U), un insieme di dipendenze funzionali F definita sugli attributi U, X⊆U.
La **chiusura di X rispetto F** è l'insieme di attributi che dipendono funzionalmente da X:

$$\boldsymbol{X_F^+ = \{A | A \in U \; e \; F \; implica \; X \rightarrow A\}}$$

Per vedere se X→A è implicata da F, basta vedere se A appartiene ad X_F^+.
Calcolo di X_F^+:

Input: insieme X di attributi e F di dipendenze **Output:** insieme di attributi X_P(ovvero X_F^+)
1) Inizializzare X_P=X 2) Se esiste una dipendenza Y→A di F con Y⊆X_P e A∉X_P allora si aggiunge A ad X_P. 3) Ripetere 2 finché non ci sono attributi che possano essere aggiunti a X_P.

Un insieme di attributi K è chiave per uno schema di relazione R(U) su cui è definito un insieme di dipendenze F, se F implica K→U.

5.6.2. Coperture di insiemi di dipendenze funzionali

Due insiemi di dipendenze funzionali **F_1 e F_2 sono equivalenti** se F_1 implica ciascuna dipendenza in F_2 e viceversa. Si dice che uno è **copertura** dell'altro.

F **non ridondante**: se non esiste nessuna dipendenza f∈F tale che F-{f} implica F.
Si esaminano ripetutamente le dipendenze di F, eliminando quelle implicate da altre.

F **ridotto**: se è non ridondante e non esiste nessun insieme F' equivalente a F ottenuto eliminando attributi dai primi membri di una o più dipendenze di F.

- Si sostituisce l'insieme dato con quello equivalente che ha tutti i secondi membri costituiti da singoli attributi.
- Si eliminano le dipendenze ridondanti. (Algoritmo calcolo X_F^+)
- Per ogni dipendenza si verifica se esistono attributi eliminabili al primo membro.
  Per ogni dipendenza con a sinistra più di un attributo: per ogni combinazione possibile di attributi del termine a SX: si calcola la chiusura, ovvero dalla combinazione si aggiungono gli attributi a DX delle dipendenze.
  Se l'insieme risultante contiene la parte destra, togli a sinistra gli attributi "in più" cioè che non sono quelli della combinazione scelta.
  Es. Dipendenze: E→N, EN→L.
  EN→L: E^+={E}, conoscendo E→N diventa E^+={E,N}, conoscendo EN→L diventa E^+={E,N,L}, quindi L appartiene alla copertura di E quindi posso eliminare N da EN→L che diventa E→L.

5.6.3. Sintesi di schemi in terza forma normale

Uno schema di relazione R(U) con l'insieme di dipendenze F è in terza forma normale se, per ogni dipendenza X→A∈F è verificata almeno una delle condizioni:

- X contiene una chiave K di r: cioè $X_F^+ = U$
- A è contenuto in almeno una chiave di r: esiste un insieme di attributi K⊆U tale che $X_F^+ = U \; e \; (K - A)_F^+ \subset U$

Algoritmo di decomposizione (dati R(U) e F):

- Calcolare la copertura ridotta G di F.
- G viene partizionato in sottoinsiemi $G_1,\ldots,G_k$ tali che a ogni insieme appartengono dipendenze che hanno primi membri con la stessa chiusura (cioè X→A e Y→B appartengono alla stessa partizione se e solo se $X_G^+ = Y_G^+$)
- Costruire un sottoinsieme $\mathcal{U}$ di U, per ciascuna partizione di dipendenze, con tutti gli attributi coinvolti nella partizione.
- Se un elemento di $\mathcal{U}$ è propriamente contenuto in un altro, allora esso viene eliminato da $\mathcal{U}$.
- Costruire uno schema di basi di dati con uno schema di relazione $R_i(U_i)$ per ciascun elemento $U_i \in \mathcal{U}$ con associate le dipendenze in G i cui attributi sono tuti contenuti in U_i.
- Se nessuno degli U_i costituisce una chiave per la relazione originaria R(U), allora viene acalcolata una chiave K di R(U) e viene aggiunto allo schema generato al passo precedente uno schema di relazione sugli attributi K, senza dipendenze.

5.7. Progettazione di basi di dati e normalizzazione

5.7.1. Verifiche di normalizzazione su entità

È sufficiente considerare le dipendenze funzionali che sussistono fra gli attributi dell'entità e verificare che ciascuna di esse abbia come primo membro l'identificatore (o lo contenga).

5.7.2. Verifiche di normalizzazione su associazioni

È necessario individuare le dipendenze funzionali fra le entità coinvolte.
Le associazioni binarie sono già in terza forma normale (quindi anche di Boyce e Codd).