FASI NEL PROGETTO DEI DATI

Il progetto della base di dati

si inserisce nel:

Ciclo di vita del sistema informativo

comprendente in generale le seguenti attività:

- **Raccolta ed analisi dei requisiti**
- **Progettazione (di schemi e applicazioni)**
- **Implementazione**
- **Validazione e collaudo**
- **Funzionamento**

Ci concentriamo sulla parte più tecnica, specifica di questo corso: la progettazione degli schemi

Prerequisiti della progettazione

- L'analisi dei requisiti è condotta dalla una specifica figura professionale (analista) tramite interviste dell'utente (che deve essere per quanto possibile parte attiva della definizione dei requisiti)
- Produce una descrizione che esprimiamo tramite un testo riassuntivo (tipicamente ambiguo).
- Nella realtà si aggiungono anche:
  - Descrizioni terminologiche, glossari, raccolte informali di definizioni
  - Descrizioni astratte dei programmi (dataflow diagrams) e del loro uso da parte degli utenti (use case diagrams).

Assunzione tecnologica

- Architettura client-server con un unico database server cui si collegano le varie applicazioni

(di fatto questa scelta non è vincolante e architetture pù complesse sono discusse nell'ambito del corso di Sistemi Informativi)

Fasi della progettazione

- **la progettazione concettuale**
- **la progettazione logica**
- **la progettazione fisica**

La progettazione concettuale

Ha per scopo tradurre il risultato dell'analisi dei requisiti in una DESCRIZIONE FORMALE che dovrà essere indipendente dal DBMS

La descrizione formale è espressa tramite uno SCHEMA CONCETTUALE, costruito utilizzando un MODELLO CONCETTUALE DEI DATI

La progettazione logica

Ha per scopo tradurre lo SCHEMA CONCETTUALE in uno SCHEMA LOGICO, scelto all'interno dei modelli logici dei dati:

- **Gerarchico**
- **Reticolare**
- **Relazionale**
- **Orientato ad oggetti**
- **XML**

Lo schema logico è dipendente dal DBMS ma non dallo specifico prodotto

La progettazione fisica

Ha per scopo produrre un PROGETTO FISICO della base dei dati, cioè un progetto che ottenga prestazioni ottimali tramite scelta e dimensionamento di strutture fisiche di accesso.

Il progetto fisico viene eseguito in modo differente su ciascun prodotto.

Dipendenze da MODELLO e DBMS

	Dipende dal MODELLO	Dipende dal DBMS
Progetto concettuale	NO	NO
Progetto logico	SI	NO
Progetto fisico	SI	SI

LE ASTRAZIONI NEI MODELLI CONCETTUALI PER BASI DI DATI

Ingredienti dei modelli concettuali

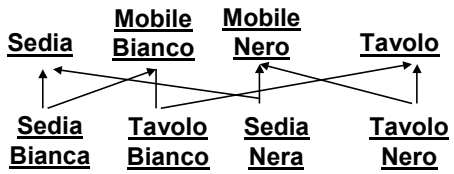
- ASTRAZIONI: capacità di evidenziare caratteristiche comuni ad insiemi di oggetti
- Tre ASTRAZIONI di base per la rappresentazione della conoscenza:
  - Classificazione
  - Aggregazione
  - Generalizzazione

Classificazione

Capacità di definire classi di oggetti o fatti del mondo reale

- LIBRO
- BICICLETTA
- PERSONA
- APPARTAMENTO

Per ogni classe esiste un implicito "test di appartenenza" che consente di dire se un oggetto o fatto del mondo reale è una istanza della classe.

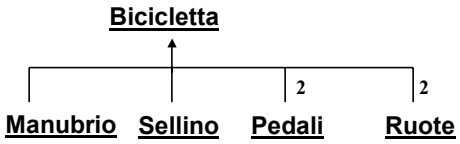


Aggregazione

Costruzione di una classe complessa aggregando classi più semplici (componenti)

- BICICLETTA
  - RUOTE
  - PEDALI
  - MANUBRIO

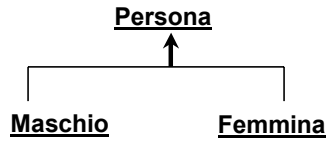
Per ogni componente si indica quante istanze sono presenti in una istanza della classe aggregata



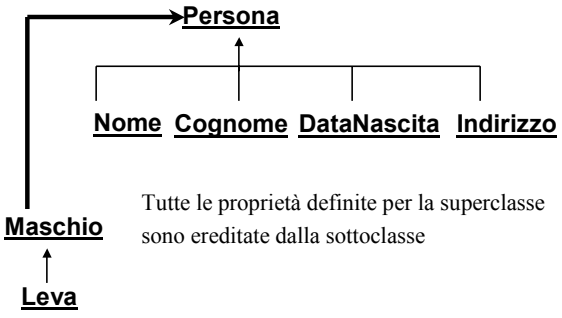
Generalizzazione

Stabilisce legami di sottoinsieme fra classi

- FEMMINA < PERSONA
- MASCHIO < PERSONA



Ereditarietà



Tutte le proprietà definite per la superclasse sono ereditate dalla sottoclasse

IL MODELLO ENTITA'-RELAZIONE (ENTITY-RELATIONSHIP)

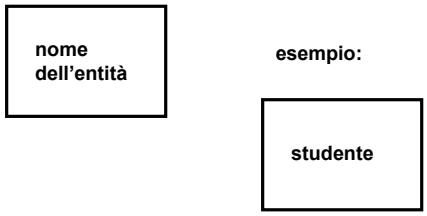
MODELLO ENTITA' RELAZIONE

- il modello Entity-Relationship (ER), (P.P.Chen 1976) si è affermato come standard industriale di buona parte delle metodologie e degli strumenti per il progetto concettuale di basi di dati.
- Attenzione: Relationship = Associazione (pero' poi si dice informalmente "relazione")

Entità

- Rappresenta una classe di oggetti (es., automobili, impiegati, studenti) o di fatti (es., conti correnti, corsi universitari)
- Devono essere oggetti rilevanti per la applicazione
- Ogni entità è caratterizzata da un nome

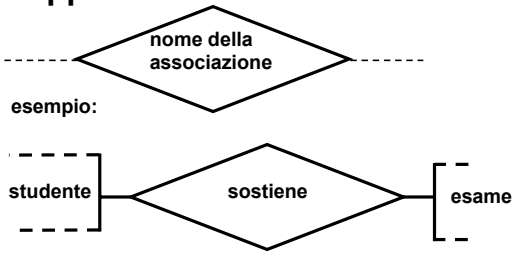
simbolo grafico per rappresentare entità



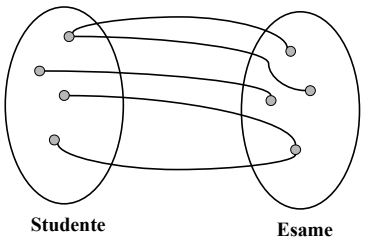
Relazione (o Associazione)

- Rappresenta una aggregazione di entità di interesse per l'applicazione
- Ogni istanza di una associazione è una ennupla tra istanze di entità (es., legame tra un automobile e il suo proprietario)
- Ogni associazione è caratterizzata da un nome

simbolo grafico per rappresentare associazioni



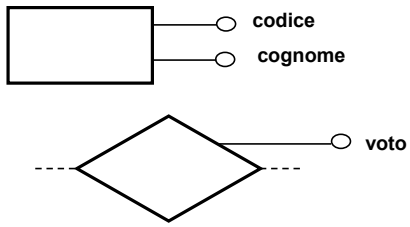
Rappresentazione grafica delle istanze



Attributi

- Rappresentano caratteristiche delle entità e delle associazioni di interesse per l'applicazione
- Ogni attributo è caratterizzato da un nome

simbolo grafico per rappresentare attributi



Linee guida per il progetto

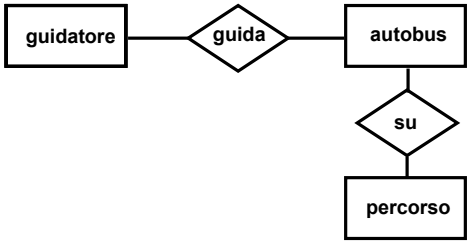
- Se il concetto è significativo per il contesto applicativo: entità
- Se il concetto è descrivibile tramite un dato elementare: attributo
- Se il concetto definisce un legame tra entità: associazione

Corrispondenza tra concetti e elementi del modello ER

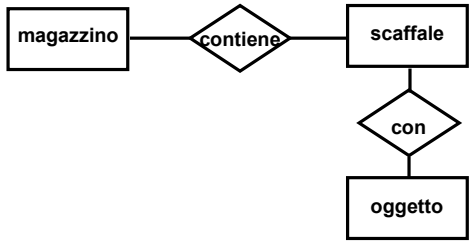
La corrispondenza tra oggetti e fatti del mondo reale e entità, associazioni e attributi non è assoluta ma dipende dal contesto:

Es.: l'auto BOF34675 ha colore rosso
il colore rosso ha lunghezza d'onda = ~700 nm

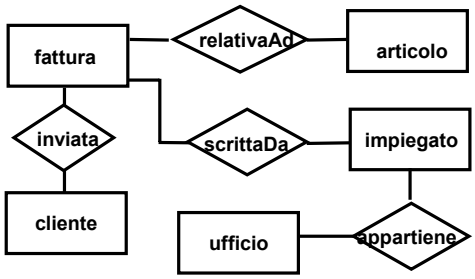
Esempio: gestione viaggi



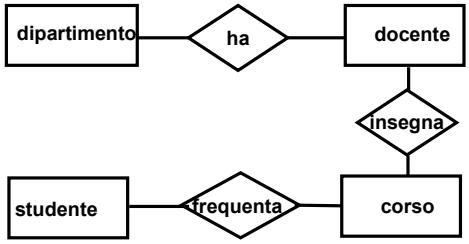
Esempio: gestione magazzino



Esempio: gestione fatture



Esempio: università



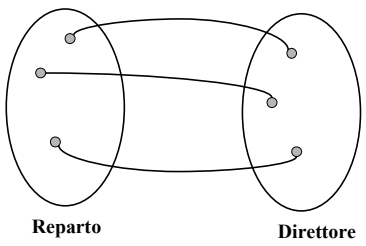
ASSOCIAZIONI NEL MODELLO ER

Cardinalità delle associazioni

- **Per cardinalità si intende un vincolo sul numero di istanze di associazione cui ciascuna istanza di entità deve partecipare.**
- **È una coppia (MIN-CARD, MAX-CARD)**
  - **MIN-CARD = 0 (opzionale)**
  - **= 1 (obbligatoria)**
  - **MAX-CARD = 1 (uno)**
  - **= N (molti)**
- **In base alla sola cardinalità massima si hanno associazioni uno-uno, uno-molti, molti-molti**

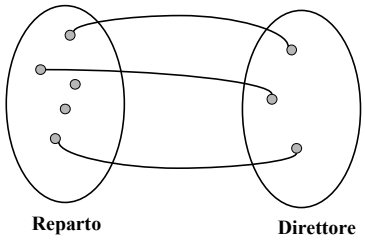
Associazione 1:1

- **un reparto deve essere diretto da uno e un solo direttore (1,1)**
  un direttore deve dirigere uno ed un solo reparto (1,1)



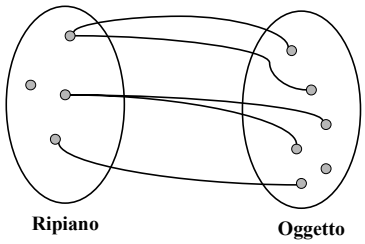
Associazione 1:1 con opzionalità

- un reparto puo' essere diretto da uno e un solo direttore (0,1)
  un direttore deve dirigere uno ed un solo reparto (1,1)



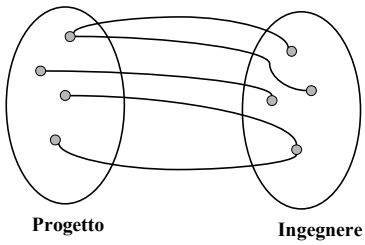
Associazione 1:N

- un reparto può contenere molti oggetti (0,n)
  un oggetto può essere contenuto al più su un ripiano (O,1)



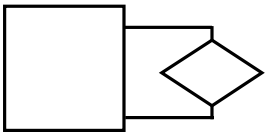
Associazione N:M

- un progetto puo' essere fatto da molti ingegneri (0,n),
  un ingegnere deve partecipare ad uno o più progetti (1,m)

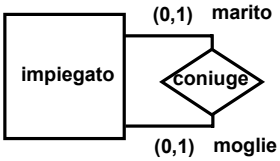


Auto-associazioni

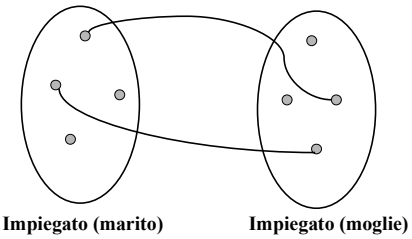
associazioni aventi come partecipanti istanze provenienti dalla stessa entità (chiamate anche "ad anello"):



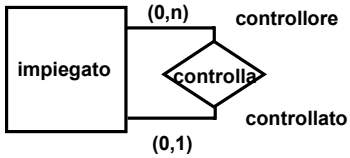
Auto-associazioni 1:1



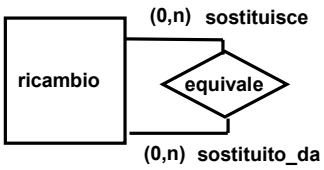
può essere riportato il "ruolo" sul ramo



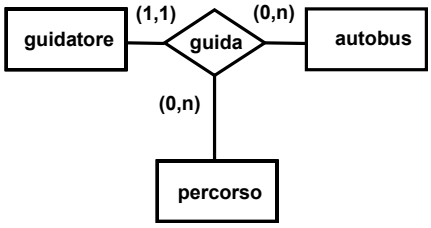
Auto-associazioni 1:N

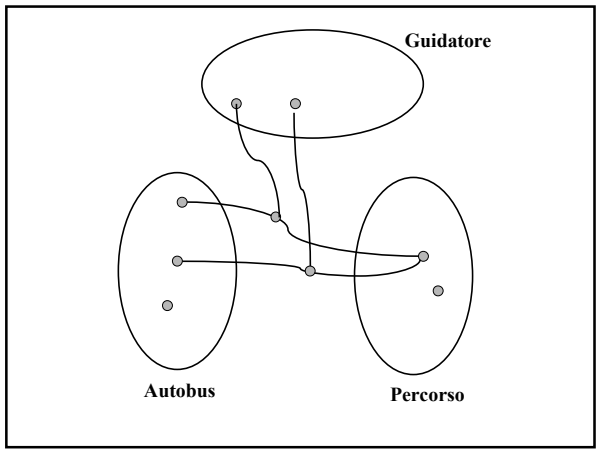


Auto-associazioni N:M

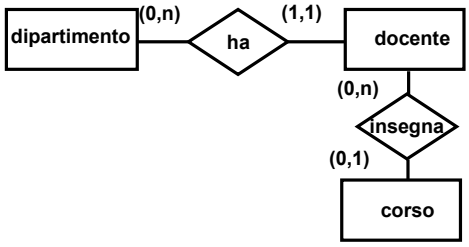


Associazioni ternarie

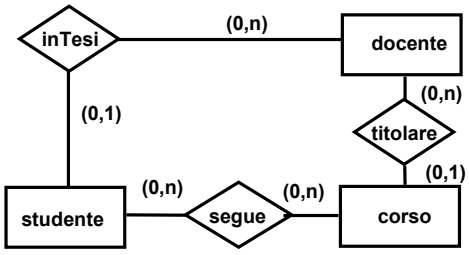




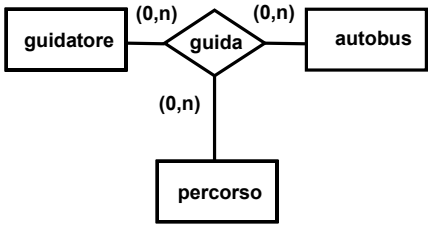
Esempio:università



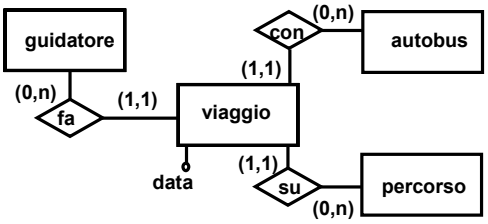
Esempio: università



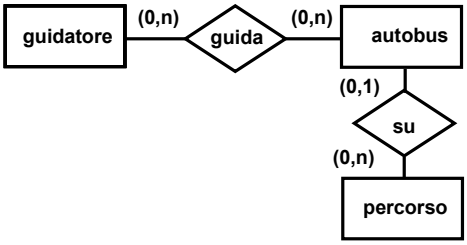
Esempio: gestione autobus



Esempio: gestione autobus



Esempio: gestione autobus



ATTRIBUTI E IDENTIFICATORI NEL MODELLO ER

Cardinalità degli attributi

- una prima classificazione:

attributo scalare (semplice, ad un sol valore)

es.: matricola, cognome, voto

attributo multiplo (sono ammessi n valori)

(1,n)

es.: qualifica, titolo, specialità

il simbolo (n,m) esprime la cardinalità dell'attributo.

Attributi composti

attributo composto

(1,1)

es.: data (gg,mm,aaaa), indirizzo (via, numero civico, città, provincia, cap)

attributo multiplo composto

(1,n)

es.: telefono (stato, città, numero)

Opzionalità

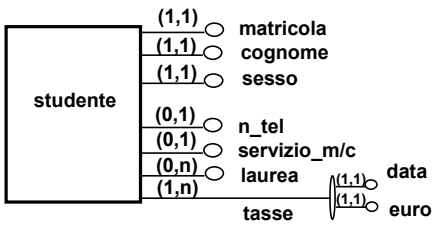
Attributo opzionale (è ammessa la "non esistenza del valore")

(0,n) (0,1)

es.: tel., qualifica, targa

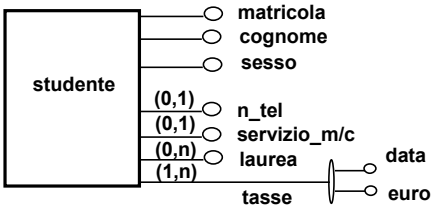
Esempio

Esempio:



Esempio con default

Esempio:

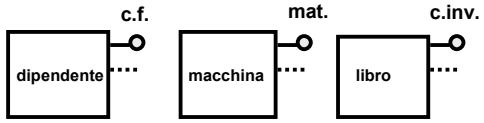


Identificatore

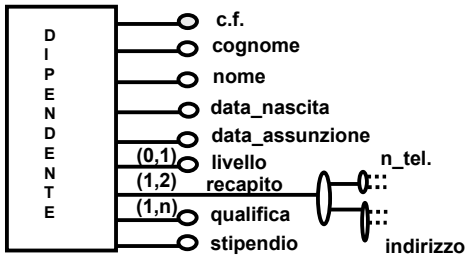
Un identificatore caratterizza in modo univoco ciascuna singola istanza di entità

simbolo

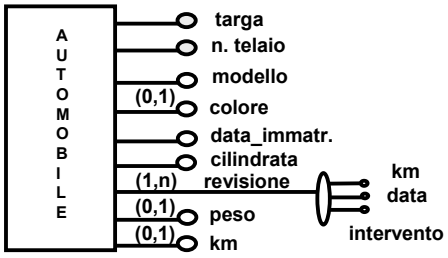
non è modificabile (in generale…)



esempio

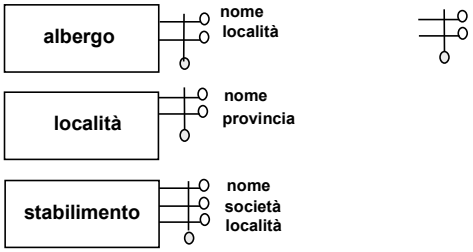


esempio

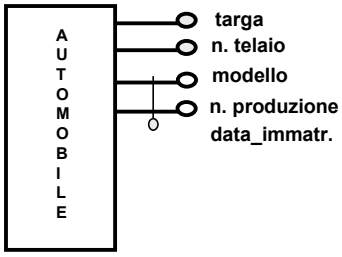


Identificatori composti

L'identificatore di un'entità può essere composto



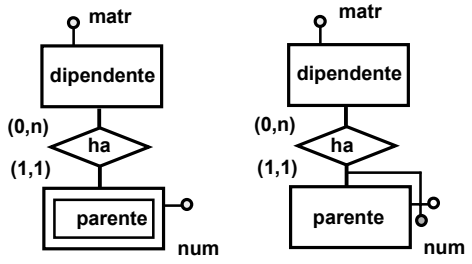
esempio



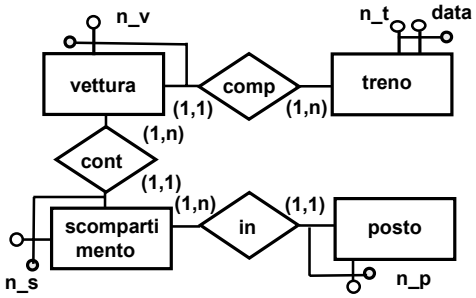
Le entità deboli

- **Le entità deboli possono esistere se e solo se sono presenti entità "forti" da cui queste dipendono**
  - **In caso di eliminazione dell'istanza "forte" di riferimento le istanze deboli collegate devono essere eliminate**

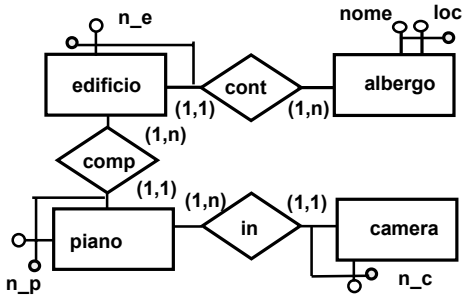
Simboli usati



esempio: posti treno



Esempio: gestione camere di un albergo



Esempi: commento

- le entità con identificatore esterno sono deboli poiché a tutti i livelli la cancellazione di una entità provoca la cancellazione delle entità deboli collegate

  (eliminazione di vettura, chiusura di scompartimento, inagibilità edificio o piano, ecc.)

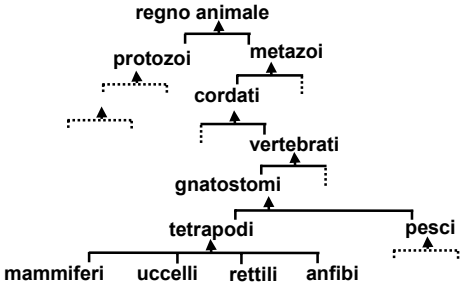
GERARCHIE DI GENERALIZZAZIONE

Gerarchie di generalizzazione

- Una gerarchia di generalizzazione è un legame logico tra un'entità padre E ed alcune entità figlie E_1 E_2 .. E_n dove:
  - E è la generalizzazione di E_1 E_2 .. E_n
  - E_1 E_2 .. E_n sono specializzazioni di E

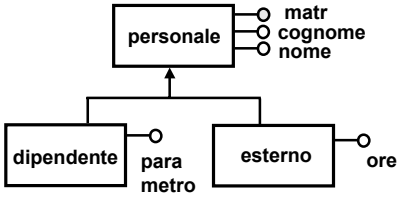
  tale per cui:
  - ogni istanza di E_k è anche istanza di E
  - una istanza di E può essere una istanza di E_k
- Le entità figlio ereditano le proprietà (attributi, relazioni, identificatori) dell'entità padre.

Una delle gerarchie più note



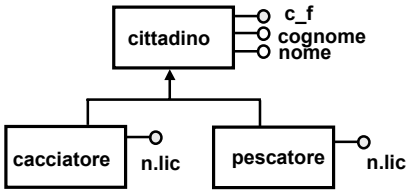
Esempio di gerarchia

un'azienda si avvale dell'opera di professionisti esterni, quindi il suo personale si suddivide in esterni e dipendenti:



Esempio di gerarchia

un comune gestisce l'anagrafe ed i servizi per i suoi cittadini alcuni di questi richiedono la licenza di caccia o pesca :



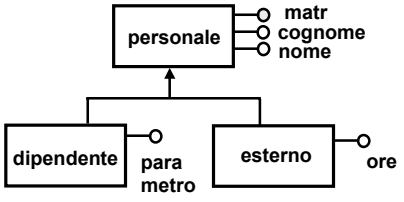
Proprieta' delle gerarchie

- ***t*** **sta per totale: ogni istanza dell'entità padre deve far parte di una delle entità figlie**
  - **nell'esempio il personale si divide (completamente) in esterni e dipendenti**
- ***p*** **sta per parziale: le istanze dell'entità padre possono far parte di una delle entità figlie**
  - **nell'esempio i cacciatori e pescatori sono un sottoinsieme dei cittadini**

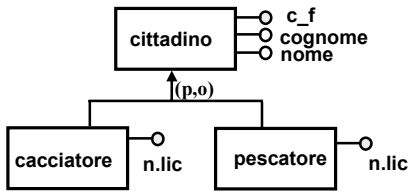
Proprieta' delle gerarchie

- ***e*** **sta per esclusiva: ogni istanza dell'entità padre non può far parte di più di una delle entità figlie**
  - **nell'esempio si esclude che una istanza di personale possa appartenere ad entrambe le sottoclassi**
- ***o*** **sta per overlapping: ogni istanza dell'entità padre può far parte di più entità figlie**
  - **nell'esempio un cittadino può essere al tempo stesso cacciatore e pescatore**

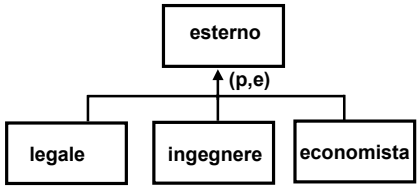
Default: (t,e)



Indicazione della proprietà

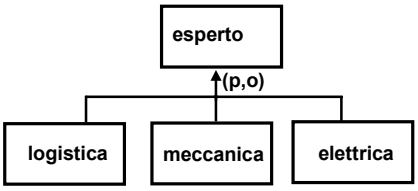


Un'ulteriore specializzazione



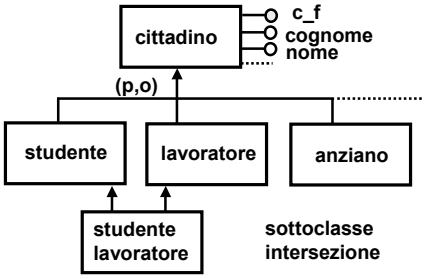
***p*: possono esistere esterni generici che non sono né legali, né ingegneri, né economisti ma non interessa stabilire una sottoclasse ad hoc**

Un'ulteriore specializzazione

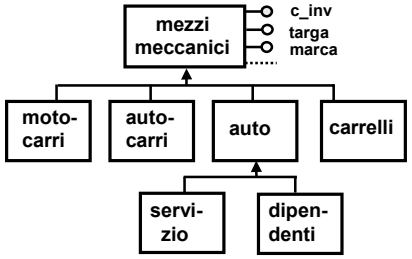


***p* : possono esistere esperti sia meccanici, sia elettrici, sia della logistica**
O: le tre qualifiche non si escludono

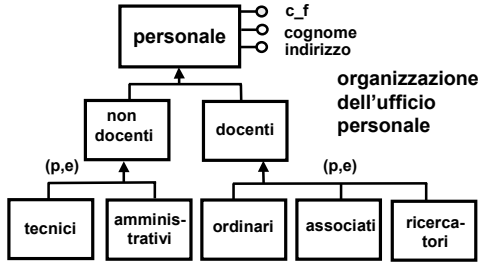
Esempio: un comune



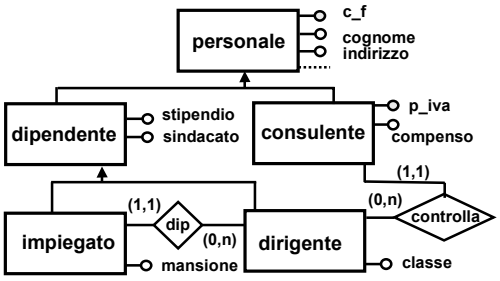
Esempio: parco mezzi meccanici



Esempio: università



Esempio: personale d'azienda



STRATEGIE DI PROGETTO

Strategie di progetto

- **Lo sviluppo dello schema si può eseguire seguendo due strategie fondamentali:**
  - **Top-Down**
  - **Bottom-Up**
- **Per quanto riguarda le strategie bottom-up, vedremo i casi:**
  - **"A macchia d'olio"**
  - **Mista**

Progetto top-down e bottom-up

- Top-down: si procede per raffinamenti a partire da una descrizione che comprende TUTTA la realtà di interesse
- Bottom-up: si disegnano separatamente aspettti della realtà e poi li si integra costruendo un unico schema

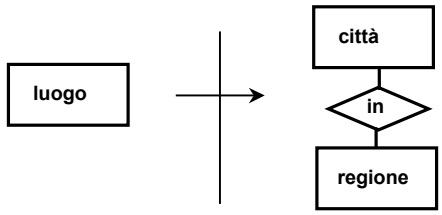
Confronto

- Top down: il progetto è più ordinato e razionale, ma più difficile (il progettista deve possedere una "visione d'assieme")
- Bottom-up: si possono prendere decisioni differenti nell'affrontare i sotto-problemi, che si tradurranno in "conflitti" (modelli diversi della stessa realtà).

Strategia top-down "pura"

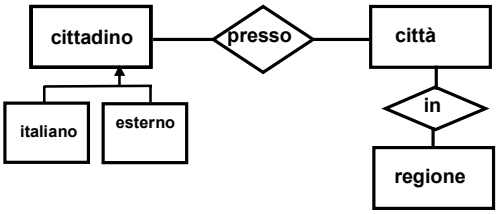
- **A partire dalle specifiche si costruisce uno schema iniziale**
- **Dallo schema iniziale si arriva per raffinamenti successivi a schemi intermedi e poi allo schema finale**
- **I raffinamenti prevedono l'uso di trasformazioni elementari (primitive) che operano sul singolo concetto per descriverlo con maggior dettaglio**

Trasformazione elementare top-down

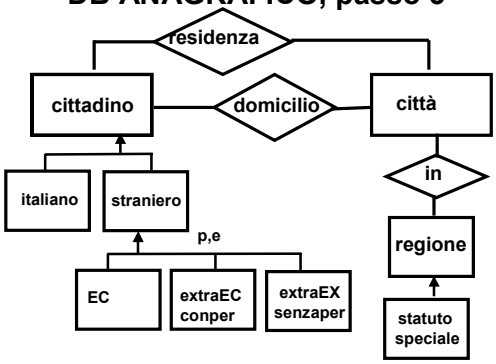


DB ANAGRAFICO, passo 1

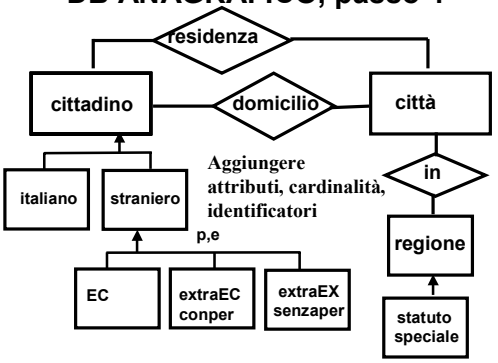
DB ANAGRAFICO, passo 2



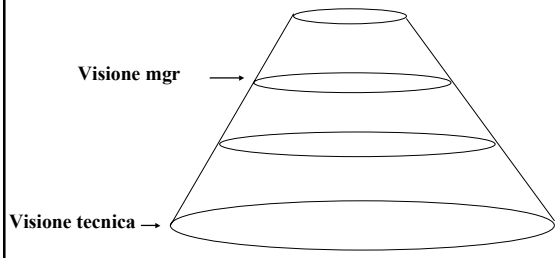
DB ANAGRAFICO, passo 3



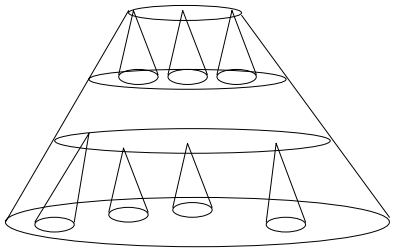
DB ANAGRAFICO, passo 4



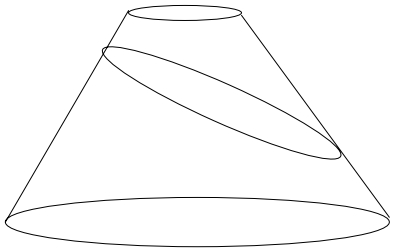
"Piani" del progetto



Progetto “equilibrato”



Progetto “squilibrato”



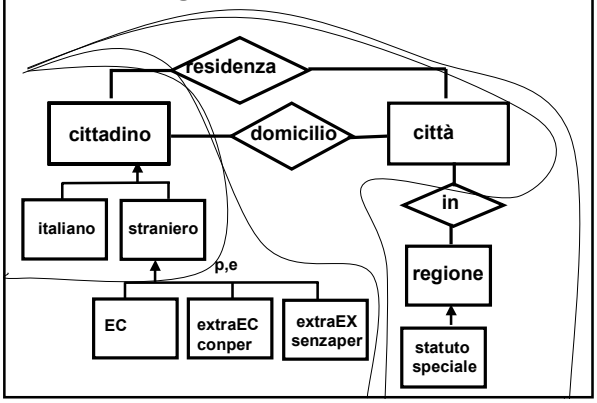
Valutazione di una strategia top down

- vantaggi:
  - il progettista descrive inizialmente lo schema trascurando i dettagli
  - precisa lo schema gradualmente
- problema:
  - non va bene per applicazioni complesse perché è difficile avere una visione globale precisa iniziale di tutte le componenti del sistema

Strategia “a macchia d’olio”

- Le specifiche nascono progressivamente, affrontando i requisiti fino al massimo dettaglio e “avanzando” per sottoproblemi
- La tecnica è adatta a tradurre pian piano una descrizione testuale in un diagramma

Strategia a macchia d’olio



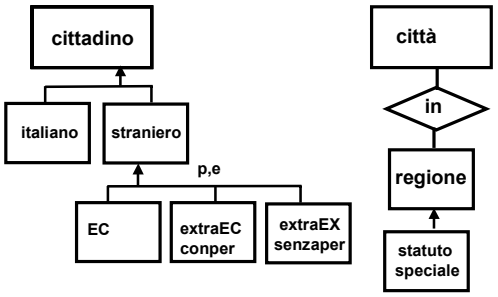
Valutazione della strategia “a macchia d’olio”

- La tecnica è adatta a tradurre pian piano una descrizione testuale in un diagramma
- Pur essendo bottom-up, il progettista analizza le specifiche in modo “stratificato” e le aggiunge progressivamente a un unico schema, perciò i conflitti sono meno probabili

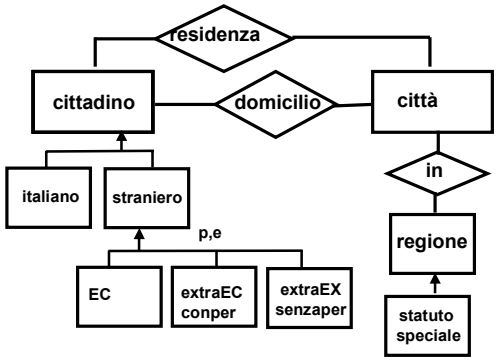
Strategia Mista

- Adatta di fronte a progetti ampi
- Si suddividono le specifiche in parti (ad esempio: le funzioni amministrazione, personale, marketing, vendita, produzione di una azienda)
- Si realizzano top-down le varie parti
- Si realizza (bottom-up) l'integrazione delle varie parti sviluppate

Sviluppo di due sottoschemi



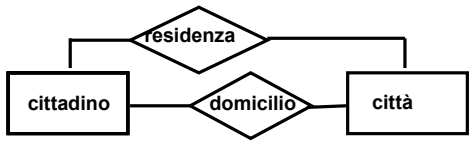
Integrazione



Valutazione della strategia mista

- **vantaggi:**
  - **diversi progettisti elaborano gli schemi parziali, il singolo progettista ha una visione più precisa del proprio settore**
- **problema:**
  - **conflitti e difficoltà di integrazione**
- **soluzione possibile:**
  - **sviluppare un piccolo schema dei soli concetti principali (schema scheletro) in modo top-down e attenersi alle scelte presenti nello schema scheletro in tutti gli altri schemi.**

Schema Scheletro



Sintesi

- Un progettista esperto procede (inconsciamente) sia in modo top-down che in modo bottom-up
- Per affrontare gli esercizi, la tecnica a macchia d'olio viene usata spesso
- In ogni caso, è possibile (e molto conveniente) DOCUMENTARE un progetto in modo top-down a posteriori

QUALITA' DI UNO SCHEMA CONCETTUALE

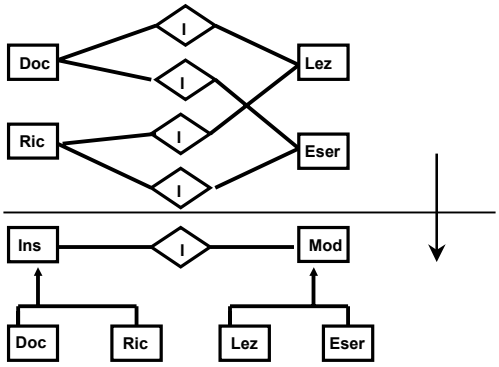
Qualità di uno schema concettuale

- Completezza
- Correttezza
- Leggibilità
- Minimalità
- Auto-Esplicatività

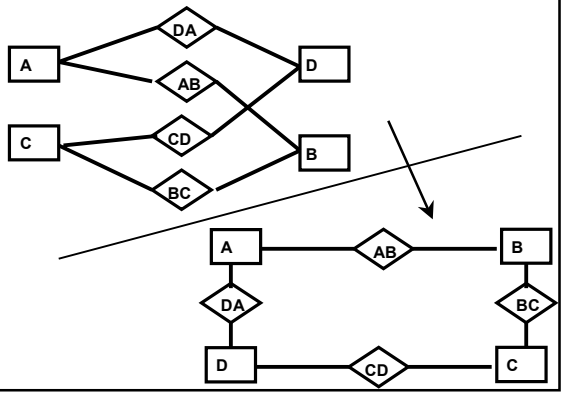
Completezza e correttezza

- Rappresentare in modo completo e corretto i requisiti
- Sono proprietà ovvie ma sulle quali c'è poco da aggiungere
  - Per la completezza: assicurarsi che i dati consentano di eseguire tutte le applicazioni
  - Per la correttezza: assicurarsi che sia possibile popolare la base di dati anche con informazione parzialmente incompleta durante fasi iniziali della sua evoluzione

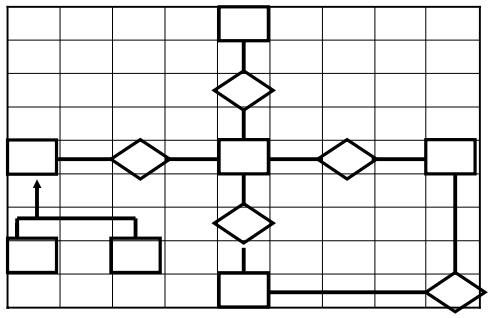
Leggibilità concettuale



Leggibilità grafica



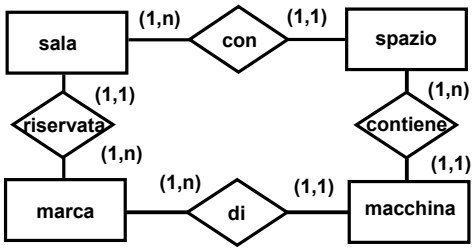
Disegnare in una griglia



Ridondanze negli schemi

- Una società gestisce delle sale di esposizione
- le sale di esposizione sono riservate a marche di macchine
- le sale comprendono spazi di esposizione
- gli spazi contengono macchine
- le macchine appartengono ad una certa marca

Ridondanze negli schemi

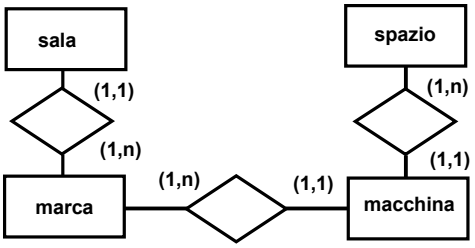


potrebbe esserci una ridondanza !

Ridondanze negli schemi

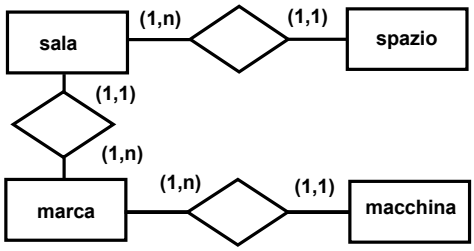
- il ciclo è ridondante se la sistemazione delle macchine negli spazi viene effettuata nel rispetto del vincolo che **una sala sia assegnata per intero ad una sola marca**
- proviamo ad eliminare le 4 associazioni a turno e verificare il rispetto delle specifiche

Eliminazione di: comprendere



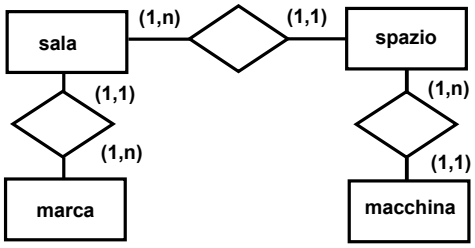
dato uno spazio non è possibile risalire alla sala che lo comprende

Eliminazione di: contenere



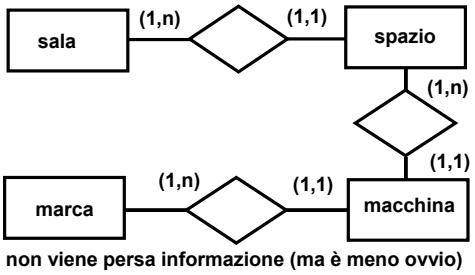
data una macchina non è possibile risalire allo spazio che la contiene

Eliminazione di: appartenere



non viene persa informazione

Eliminazione di: riservare



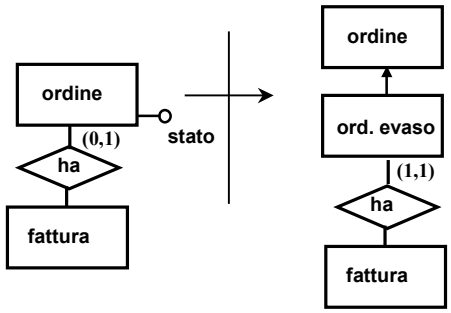
non viene persa informazione (ma è meno ovvio)

Discussione

- Ragionevole la prima eliminazione
- Piuttosto irragionevole la seconda eliminazione
- E in ogni caso: occorre conoscere bene il significato delle associazioni!

Auto-esplicatività

- Fare in modo che lo schema rappresenti esplicitamente il massimo di conoscenza sulla realtà



A questo punto....

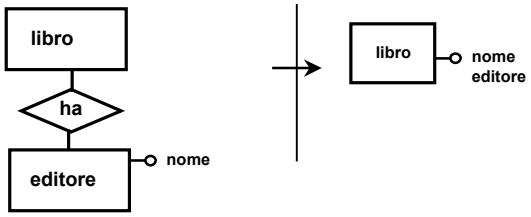
- Il progetto è stato condotto (o documentato) top-down
- Il progetto risponde ai requisiti di qualità

Ultimo passo: post-processing

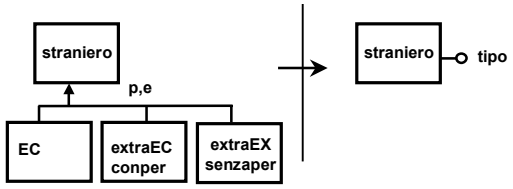
Post-processing

- Verificare che:
  - Tutte le entità abbiano un identificatore
  - Tutte le associazioni abbiano cardinalità ben definite
  - Le entità siano significative (consentano di rappresentare più di un attributo o siano collegate ad altre entità tramite associazioni)
  - Le generalizzazioni siano utili (consentano di ereditare proprietà)

Esempio di entità inutile



Esempio di gerarchia inutile



ESERCIZIO DI PROGETTAZIONE CONCETTUALE ...

Progetto logico

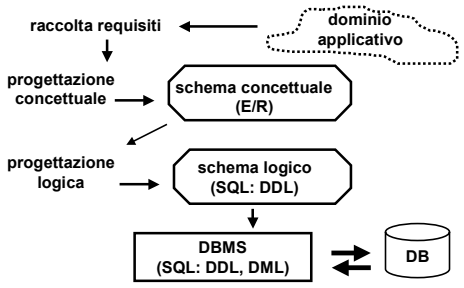
Progetto logico

- **Lo schema E/R descrive un dominio applicativo a un dato livello di astrazione**
- **Lo schema E/R serve per:**
  - **fornire una descrizione sintetica e visiva**
  - **rappresentare buona parte della semantica dell'applicazione**
  - **scambiare informazioni sull'attività progettuale tra i membri del team di progetto e mantenere una documentazione**

Progetto logico

- **Non esistono DBMS in grado di operare direttamente sui concetti di schemi E/R**
  - **è quindi necessario tradurli in altri schemi di dati (logico relazionale in queste lezioni)**
  - **questa traduzione può essere eseguita in modo semi-automatico**
  - **le scelte alternative devono tenere conto dell'efficienza dello schema logico risultante e delle operazioni da effettuare (derivanti da flussi e processi)**

processo di design



scelte alternative

si possono individuare alcune linee guida:

- **considerare le proprietà logiche comunque primarie rispetto ai motivi di efficienza**
- **tenere sulla stessa entità informazioni che verranno di frequente consultate insieme**
- **tenere su entità separate informazioni che verranno consultate separatamente**
- **limitare l'incidenza di valori nulli per attributi opzionali**

fasi del progetto

il progetto produce trasformazioni e traduzioni dello schema E/R con le seguenti fasi:

1. **eliminazione delle gerarchie isa**
2. **selezione delle chiavi primarie, eliminazione delle identificazioni esterne**
3. **normalizzazione degli attributi composti o multipli**
4. **traduzione di entità e associazioni in schemi di relazioni**
5. **verifica di normalizzazione**

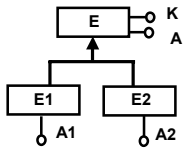
eliminazione delle gerarchie

il modello relazionale non rappresenta le gerarchie, le gerarchie sono sostituite da entità e associazioni:

1) mantenimento delle entità con associazioni

2) collasso verso l'alto

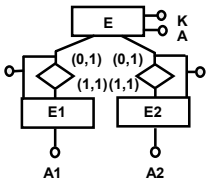
3) collasso verso il basso



l'applicabilità e la convenienza delle soluzioni dipendono dalle proprietà di copertura e dalle operazioni previste

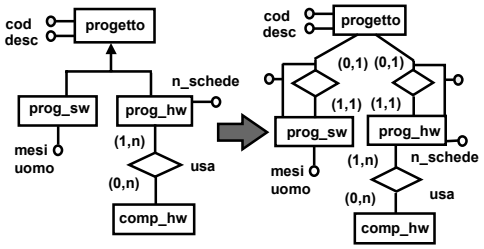
mantenimento delle entità

- **tutte le entità vengono mantenute**
- **le entità figlie sono in associazione con l'entità padre**
- **le entità figlie sono identificate esternamente tramite l'associazione**



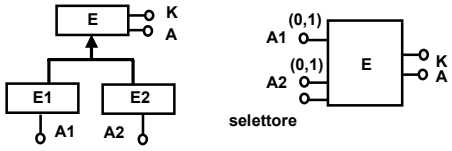
questa soluzione è sempre possibile, indipendentemente dalla copertura

mantenimento entità - es.:



eliminazione delle gerarchie

- Il collasso verso l'alto riunisce tutte le entità figlie nell'entità padre

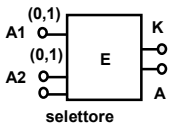


selettore è un attributo che specifica se una istanza di E appartiene a una delle sottoentità

isa: collasso verso l'alto

- il collasso verso l'alto favorisce operazioni che consultano insieme gli attributi dell'entità padre e quelli di una entità figlia:
  - in questo caso si accede a una sola entità, anziché a due attraverso una associazione
- gli attributi obbligatori per le entità figlie divengono opzionali per il padre
  - si avrà una certa percentuale di valori nulli

isa: collasso verso l'alto

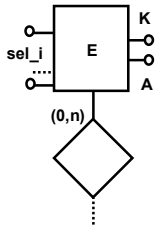


Copertura dell'ISA
totale esclusiva:
selettore ha N valori,
quante sono le sottoentità
parziale esclusiva:
selettore ha N+1 valori; il valore in più serve per le istanze che non appartengono ad alcuna sottoentità

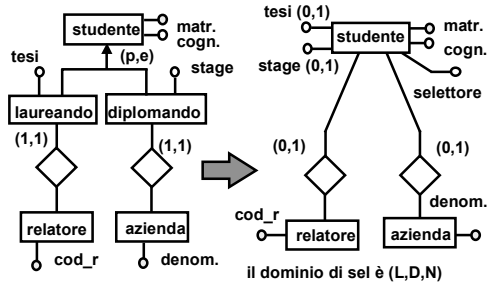
isa: collasso verso l'alto

Copertura dell'ISA:

overlapping: occorrono tanti selettori booleani quante sono le sottoentità, sel_i è "vero" per ogni istanza di E che appartiene a E_i se la copertura è parziale i selettori possono essere tutti "falsi"
le associazioni connesse alle sottoentità si trasportano su E, le eventuali cardinalità minime diventano 0



isa: collasso verso l'alto



il dominio di sel è (L,D,N)

isa: collasso verso il basso

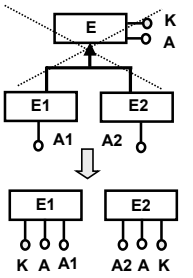
Collasso verso il basso:

- si elimina l'entità padre trasferendone gli attributi su tutte le entità figlie
  - una associazione del padre è replicata, tante volte quante sono le entità figlie
  - la soluzione è interessante in presenza di molti attributi di specializzazione (con il collasso verso l'alto si avrebbe un eccesso di valori nulli)
  - favorisce le operazioni in cui si accede separatamente alle entità figlie

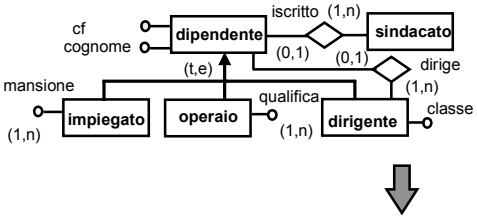
isa: collasso verso il basso

limiti di applicabilità:

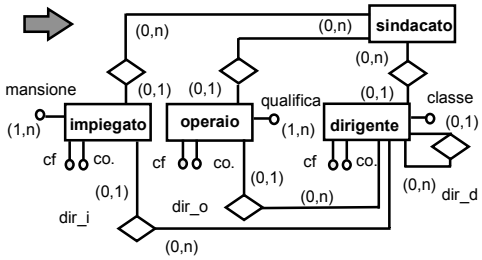
- **se la copertura è parziale non si può fare:**
  dove mettere gli E che non sono né E1, né E2 ?
- **se la copertura è overlapping introduce ridondanza: per una istanza presente sia in E1 che in E2 si rappresentano due volte gli attributi di E**



collasso verso il basso: es.



collasso verso il basso: es.

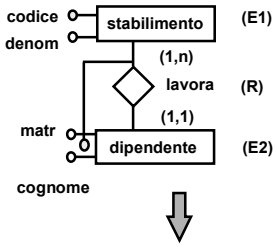


Scelta della chiave primaria

- **È necessario che tra i diversi identificatori di una entità venga designata una chiave primaria:**
  per la chiave primaria occorrerà, infatti, che il DBMS sia provvisto di strumenti per garantire l'unicità dei valori
- **criteri euristici di scelta:**
  - **primo: scegliere la chiave che è usata più frequentemente per accedere all'entità**
  - **secondo: si preferiscono chiavi semplici a chiavi composte, interne anziché esterne**

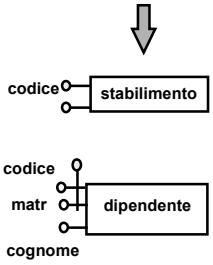
identificatori esterni

- **una componente di identificazione esterna di una entità E2 da una entità E1 attraverso una associazione R comporta il trasporto della chiave primaria di E1 su E2**



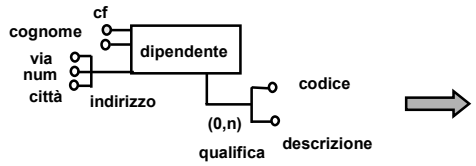
identificatori esterni

- **in questo modo l'associazione è rappresentata attraverso la chiave, e può essere eliminata**
- **la chiave trasportata è chiave esterna**
- **in presenza di più identificazioni in cascata, è necessario iniziare la propagazione dall'entità che non ha identificazioni esterne**



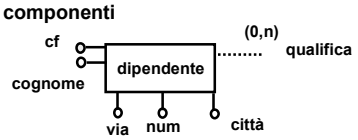
attributi composti/ripetuti

le relazioni non possono contenere attributi composti o attributi ripetuti, ma solamente attributi "atomici"



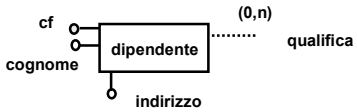
attributi composti

- **Due possibili soluzioni**
  - **eliminare l'attributo composto e considerare i suoi componenti come attributi semplici**
    - **in questo modo si perde la visione unitaria ma si mantiene l'articolazione dei componenti**



attributi composti

- **eliminare i componenti e considerare l'attributo come semplice**
  - **in questo modo lo schema risulta semplificato, perdendo parte della struttura**

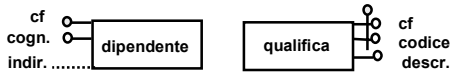


attributi ripetuti

la definizione di relazione impone che, se una entità E ha un attributo A ripetuto, si crei una nuova entità che contenga l'attributo e sia collegata a E:

Caso a) - un valore può comparire una volta sola nella ripetizione:

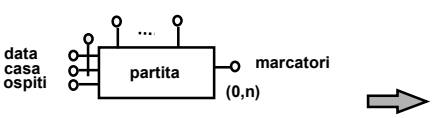
la nuova entità EA ha l'identificatore composto dall'identificatore di E più l'attributo A



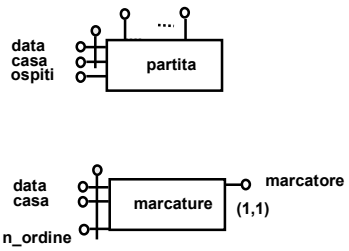
attributi ripetuti

Caso b) - un valore può comparire più volte nella ripetizione:

la nuova entità EA ha l'identificatore composto dall'identificatore di E più un valore identificante sintetico (ad esempio, un numero d'ordine)



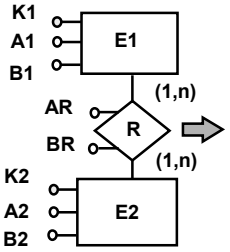
attributi ripetuti



Traduzione standard

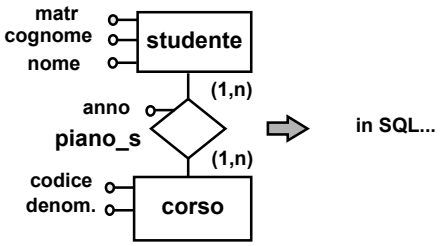
- ogni entità è tradotta con una relazione con gli stessi attributi
  - la chiave è l'identificatore dell'entità stessa (già visto)
- ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega (già visto)
  - la chiave è composta dalle chiavi delle entità collegate

traduzione standard



E1 (<u>K1</u>, A1, B1,...)
E2 (<u>K2</u>, A2, B2,...)
R (<u>K1,K2</u>, AR, BR,...)

traduzione standard: es.



traduzione standard: es.

```
CREATE TABLE STUDENTE   (MATR... NOT NULL,
…, NOME... ,    PRIMARY KEY (MATR));

CREATE TABLE CORSO   (CODICE... NOT NULL,
DENOM       ... ,    PRIMARY KEY (CODICE));

CREATE TABLE PIANO_ST (MATR... NOT NULL,
CODICE... NOT NULL, ANNO…
PRIMARY KEY (MATR, CODICE),
FOREIGN KEY (MATR) REFERENCES STUDENTE
FOREIGN KEY (CODICE) REFERENCES CORSO);
```

altre traduzioni

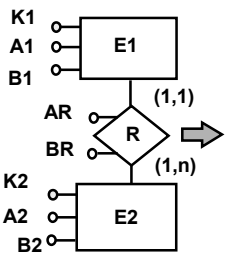
- La traduzione standard è sempre possibile ed è l'unica possibilità per le associazioni N a M
- Altre forme di traduzione delle associazioni sono possibili per altri casi di cardinalità (1 a 1, 1 a N)
- Le altre forme di traduzione fondono in una stessa relazione entità e associazioni

altre traduzioni

- Le altre forme di traduzione:
  - danno origine a un minor numero di relazioni e generano quindi uno schema più semplice
  - richiedono un minor numero di join per la navigazione attraverso un'associazione, ovvero per accedere alle istanze di entità connesse tramite l'associazione
  - penalizzano le operazioni che consultano soltanto gli attributi di una entità che è stata fusa

Associazione binaria 1 a N

- traduzione standard:



E1 (K1, A1, B1)

E2 (K2, A2, B2)

R (K1,K2, AR, BR)

associazione binaria 1 a N

- Se E1 partecipa con cardinalità (1,1) può essere fusa con l'associazione, ottenendo una soluzione a due relazioni:

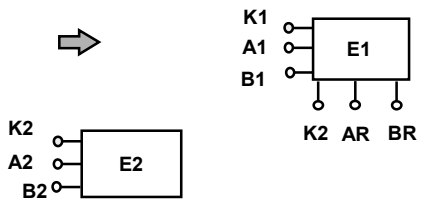
  E1(K1, A1, B1,K2, AR, BR)

  E2(K2, A2, B2)

- Se E1 partecipa con cardinalità (0,1) la soluzione a due relazioni ha valori nulli in K2, AR, BR per le istanze di E1 che non partecipano all'associazione

Associazione binaria 1 a N

- equivale a:



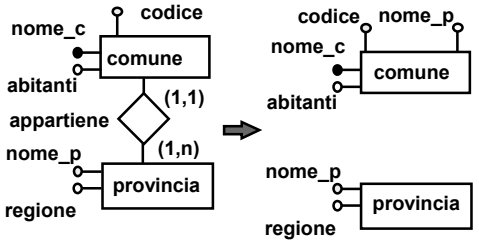
associazione binaria 1 a N

- Attenzione : in questo caso, poiché la partecipazione di E1 è 0,1 o 1,1, si nota facilmente che ad un dato valore di K1 corrisponde uno e un sol valore di K2 (non è vero il contrario), quindi si può dire che K1 implica K2 o, anche, che esiste una dipendenza funzionale da K1 a K2
- nella soluzione a 3 relazioni la chiave della relazione che traduce l'associazione è riducibile a K1:

  E1(K1,A1,B1) , E2(K2,A2,B2)

  R(K1,K2,AR,BR)

ass. binaria 1 a N es.



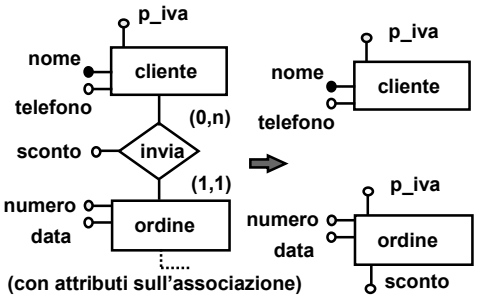
(senza attributi sull'associazione)

ass. binaria 1 a N es.

```
CREATE TABLE PROVINCIA
 (NOME_P    ... NOT NULL,
  REGIONE ...   PRIMARY KEY (NOME_P));

CREATE TABLE COMUNE
 (CODICE  ... NOT NULL, NOME_C   ...
  ABITANTI ...,  NOME_P   ... NOT NULL
  PRIMARY KEY (CODICE)
  FOREIGN KEY NOME_P
   REFERENCES PROVINCIA);
```

ass. binaria 1 a N es.



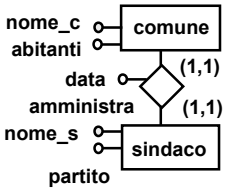
ass. binaria 1 a N es.

traduzione con due relazioni:

CREATE TABLE CLIENTE (P_IVA….. NOT NULL,
NOME …,TELEFONO …, PRIMARY KEY (P_IVA));

CREATE TABLE ORDINE (NUMERO ... NOT NULL,
DATA ... P_IVA ... NOT NULL, SCONTO ...,
PRIMARY KEY (NUMERO)
FOREIGN KEY P_IVA REFERENCES CLIENTE);

Associazione binaria 1 a 1



- traduzione con una relazione:

E12 (K1, A1, B1, K2, A2, B2, AR, BR)

associazione binaria 1 a 1

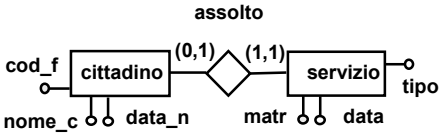
CREATE TABLE AMMINISTRAZIONE
(NOME_C ... NOT NULL, ABITANTI ...,
NOME_S ... NOT NULL UNIQUE,
INDIRIZZO ..., DATA
PRIMARY KEY (NOME_C));

se le cardinalità minime sono entrambe 1 la chiave può essere indifferentemente K1 o K2 si sceglierà quella più significativa

associazione binaria 1 a 1

- se la cardinalità di E2 è 0,1 e quella di E1 è 1,1 allora la chiave sarà K2; E2 è l'entità con maggior numero di istanze alcune della quali non si associano, ci saranno quindi valori nulli in corrispondenza di K1, K1 in questo caso non potrebbe essere scelta
- se la cardinalità è 0,1 da entrambe le parti allora le relazioni saranno due per l'impossibilità di assegnare la chiave all'unica relazione a causa della presenza di valori nulli sia su K1 che su K2

associazione binaria 1 a 1



CREATE TABLE CITTADINO
(COD_F ... NOT NULL, NOME_C ... NOT NULL,
INDIRIZZO ..., DATA_N …., MATR …., DATA….,
TIPO ….., PRIMARY KEY (COD_F));

associazione binaria 1 a 1

- **Traduzione con due relazioni**
  - **l'associazione può essere compattata con l'entità che partecipa obbligatoriamente (una delle due se la partecipazione è obbligatoria per entrambe) la discussione sulla chiave è analoga al caso di traduzione con una relazione**

E1 (K1, A1, B1,...)

E2 (K2, A2, B2,... K1, AR, BR)

associazione binaria 1 a 1

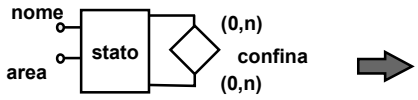
- **Traduzione con tre relazioni**
  - **la chiave della relazione che traduce l'associazione può essere indifferentemente K1 o K2, non ci sono problemi di valori nulli**

E1 (K1, A1, B1,...)
E2 (K2, A2, B2,...)
R (K1, K2, AR, BR,...)

Auto associazione N a M

viene tradotta con:

- **una relazione per l'entità ed**
- **una per l'associazione,**
  - **quest'ultima contiene due volte la chiave dell'entità, è necessario però modificare i nomi degli attributi, per non avere omonimia**



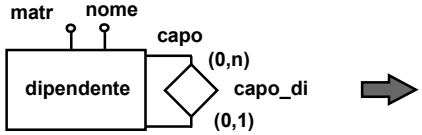
auto associazione N a M

```
CREATE TABLE STATO
 (NOME ... NOT NULL,  AREA …
  PRIMARY KEY (NOME));

CREATE TABLE CONFINA
  STATO_A ... NOT NULL, STATO_B ... NOT NULL,
  PRIMARY KEY (STATO_A, STATO_B)
  FOREIGN KEY (STATO_A)
  REFERENCES STATO
  FOREIGN KEY (STATO_B)
  REFERENCES STATO);
```

auto associazione 1 a N

- **è traducibile con una sola relazione che contiene due volte l'attributo chiave: una volta come chiave ed una come riferimento all'istanza connessa, con nome diverso per specificare il ruolo**

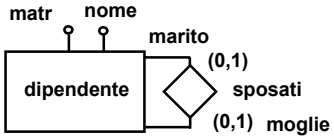


auto associazione 1 a N

```
CREATE TABLE DIPENDENTE
 (MATR ... NOT NULL, NOME ..., CAPO ...
  PRIMARY KEY (MATR)
  FOREIGN KEY (CAPO)
   REFERENCES DIPENDENTE);
```

- **nel caso di associazione 1 ad 1 il concetto di ruolo assume maggiore importanza:**

auto associazione 1 a 1



- **su entrambi i rami è bene specificare il ruolo: conviene la soluzione con due relazioni per evitare ridondanze, vincoli ed eccesso di valori nulli.**

auto associazione 1 a 1

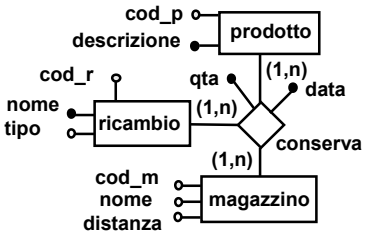
```
CREATE TABLE DIPENDENTE (MATR ... NOT
NULL, NOME ...,  PRIMARY KEY (MATR)

CREATE TABLE SPOSATI
 (MOGLIE ... NOT NULL, MARITO ... NOT NULL
  PRIMARY KEY (MOGLIE)
  FOREIGN KEY (MOGLIE)
    REFERENCES DIPENDENTE
  FOREIGN KEY (MARITO)
    REFERENCES DIPENDENTE);
```

Associazione n-aria

- **segue la traduzione standard**
- **talvolta, nella relazione che traduce l'associazione, la chiave ottenuta componendo le chiavi di tutte le entità partecipanti è una superchiave, cioè una chiave composta il cui set di componenti non è minimale (la chiave vera è un sottoinsieme)**
- **Esempio: prodotti-ricambi-magazzini**

associazione n-aria



associazione n-aria

```
CREATE TABLE PRODOTTO (COD_P... NOT NULL,
  DESCRIZIONE... ,  PRIMARY KEY (PRD));
CREATE TABLE RICAMBIO (COD_R ... NOT NULL,
  NOME…, TIPO…,  PRIMARY KEY (COD_R));
CREATE TABLE MAGAZZINO (COD_M…. NOT
  NULL, NOME ..., DISTANZA…, PRIMARY KEY
  (COD_M));
```

associazione n-aria

l'associazione diventa:

```
CREATE TABLE CONSERVA (COD_P ... NOT NULL,
  COD_R... NOT NULL, COD_M... NOT NULL,
  DATA... NOT NULL,  QTA ...
  PRIMARY KEY (COD_P, COD_R, COD_M)
  FOREIGN KEY (COD_P)
       REFERENCES PRODOTTO
  FOREIGN KEY (COD_M)
       REFERENCES MAGAZZINO
  FOREIGN KEY (COD_R)
       REFERENCES RICAMBIO);
```

ipotizziamo che COD_M sia ridondante ⇨

associazione n-aria

- un ricambio esiste in un solo magazzino, quindi COD_R è associato ad un solo COD_M, cioè determina COD_M, allora la presenza di COD_M nella chiave è ridondante:

```
CREATE TABLE CONSERVA (COD_P ... NOT NULL,
   COD_R... NOT NULL, COD_M... NOT NULL,
   DATA... ,  QTA ...
   PRIMARY KEY (COD_P, COD_R)
   FOREIGN KEY (COD_P) REFERENCES PRODOTTO
   FOREIGN KEY (COD_R) REFERENCES RICAMBIO);
```

- COD_M non e' piu' parte della chiave

commento

- nel caso precedente la dipendenza tra magazzino e ricambio non era stata espressa sulla associazione n-aria; abbiamo ipotizzato di scoprirla nella fase di progetto logico
- se il progetto concettuale è ben fatto casi del genere non sono frequenti
- il ricontrollo delle chiavi delle relazioni è quindi importante e se ne occupa la teoria della normalizzazione

LA NORMALIZZAZIONE DELLE RELAZIONI

Nelle lezioni precedenti

- Abbiamo visto la conversione degli schemi E/R in schemi logici relazionali
- questa attività, che va sotto il nome di progetto logico, prevede una serie di fasi che applicano regole di trasformazione e traduzione

In questa lezione

- continueremo a risolvere i problemi legati alla costruzione di schemi relazionali, vedremo, in particolare, come produrre schemi relazionali esenti da anomalie e non suscettibili di perdita di informazioni nelle operazioni di join
- riprenderemo il concetto di dipendenza funzionale
- introdurremo il concetto di forma normale

impiegato	stipendio	progetto	budget	funzione
Rossi	2	biella	300	tecnico
Verdi	3	valvola	500	progettista
Verdi	3	albero	1500	progettista
Neri	7	albero	1500	direttore
Neri	7	valvola	500	consulente
Neri	7	biella	300	consulente
Mori	6	biella	300	direttore
Mori	6	albero	1500	progettista
Bianchi	6	albero	1500	progettista
Bianchi	6	biella	300	progettista

ridondanze e anomalie

1) ridondanza :

- **si ripete più volte la notizia che un impiegato percepisce un certo stipendio**
- **si ripete più volte che un progetto ha un certo budget**
- **i valori di progetto e di impiegato si ripetono e quindi non possono singolarmente essere presi come chiave**
- **la chiave è (progetto, impiegato) : non si hanno ripetizioni**

ridondanze e anomalie

2) aggiornamento :

- **poiché si ripete più volte la notizia che un impiegato percepisce un certo stipendio, se lo stipendio viene aggiornato questo deve essere fatto su tutte le tuple che riguardano un certo impiegato**
- **poiché si ripete più volte che un progetto ha un certo budget, se il budget viene aggiornato lo si deve fare su tutte le tuple che riguardano un certo progetto**

ridondanze e anomalie

3) cancellazione :

- **supponendo che un impiegato lasci l'azienda o non partecipi a progetti rischiamo di perdere i dati sui progetti se era l'ultimo impiegato del progetto**
- **analogamente per i dati degli impiegati se un progetto viene eliminato**
- **se la chiave è (progetto, impiegato) in entrambi i casi di eliminazione si potrebbero avere valori nulli nella chiave**

ridondanze e anomalie

4) inserimento :

- **se la chiave è (progetto, impiegato) non è possibile inserire i dati di un impiegato se non è stato assegnato ad almeno un progetto, analogamente per un nuovo progetto a cui non è stato ancora assegnato un impiegato**
- **accettare un inserimento di (progetto) o, (impiegato) vuol dire che si inseriscono valori nulli (incompatibili con la chiave)**

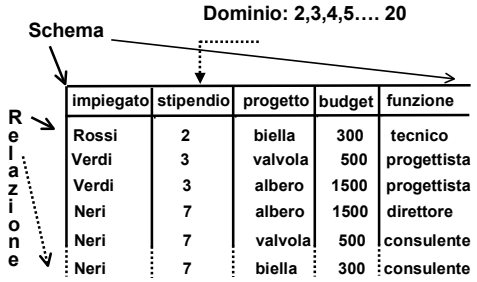
ridondanze e anomalie

- **casi così eclatanti non succedono se si è seguita la prassi corretta di progettazione: prima lo schema E/R e poi la traduzione in schema relazionale**
- **può però succedere che carenze di specifiche o errori di schematizzazione possano portare a relazioni con anomalie**
- **i casi sono invece più frequenti quando si esaminano vecchi DB scarsamente documentati o, addirittura, si cerca di intuire la natura dei dati da documenti che sintetizzano le informazioni su moduli cartacei**

dipendenze funzionali

- **La dipendenza funzionale è un vincolo di integrità per il modello relazionale**
- **dall'osservazione della relazione ricaviamo che:**
  - **ogni volta che in una tupla compare un certo impiegato lo stipendio è sempre lo stesso**
  - **possiamo dire che il valore dell'impiegato determina il valore dello stipendio, cioè:**
    - **esiste una funzione che associa ad ogni valore nel dominio impiegato uno ed un solo valore nel dominio stipendio**
  - **analogamente per un valore di progetto**

ricordiamo che:



impiegato	stipendio	progetto	budget	funzione
Rossi	2	biella	300	tecnico
Verdi	3	valvola	500	progettista
Verdi	3	albero	1500	progettista
Neri	7	albero	1500	direttore
Neri	7	valvola	500	consulente
Neri	7	biella	300	consulente

dipendenze funzionali

- La dipendenza funzionale si può definire formalmente :
  - data una relazione R definita su uno schema S(X) e due sottoinsiemi di attributi Y e Z non vuoti di X, esiste una dipendenza funzionale Y→ Z , se, per ogni coppia di tuple t1 e t2 aventi lo stesso valore di Y risulta che hanno lo stesso valore di Z
- dall'osservazione della relazione ricaviamo che:
  - impiegato → stipendio e progetto → budget

dipendenze funzionali

- Attenzione : se prendiamo la chiave K della relazione R si verifica facilmente che esiste una dipendenza funzionale tra K ed ogni attributo dello schema
- infatti per definizione di chiave esiste un solo valore di K in R e quindi la dipendenza di cui sopra è banalmente soddisfatta
- nell'esempio:

  impiegato, progetto →
  stipendio, budget, funzione

dipendenze funzionali

Però:

- impiegato, progetto → funzione
  è una dipendenza completa,
- mentre
  impiegato, progetto → stipendio e
  impiegato, progetto → budget
  sono in realtà
  impiegato→ stipendio e progetto → budget
  queste sono dipendenze parziali che causano anomalie

dipendenze funzionali

- Le ridondanze e le anomalie sono causate da dipendenze X → Y che permettono ripetizioni all'interno della relazione (impiegato,stipendio e progetto, budget si ripetono nella relazione), in altre parole :
- Le ridondanze e le anomalie sono causate da dipendenze X → Y tali che X non contiene la chiave della relazione
- Una relazione R è in forma normale (Boyce e Codd) se, per ogni dipendenza X → Y in R, X contiene una chiave K di R (X è superchiave)

dipendenze funzionali

- Una relazione non in forma normale è possibile che venga decomposta in due o più relazioni in forma normale
- la decomposizione si può attuare effettuando proiezioni in modo tale da ottenere che ciascuna dipendenza funzionale corrisponda ad una relazione separata
- nell'esempio :

  FUNZIONI per impiegato, progetto → funzione

  IMPIEGATI per impiegato→ stipendio

  PROGETTI per progetto → budget

impiegato	stipendio
Rossi	2
Verdi	3
Neri	7
Mori	6
Bianchi	6

progetto	budget
biella	300
valvola	500
albero	1500

impiegato	progetto	funzione
Rossi	biella	tecnico
Verdi	valvola	progettista
Verdi	albero	progettista
Neri	albero	direttore
Neri	valvola	consulente
Neri	biella	consulente
Mori	biella	direttore
Mori	albero	progettista
Bianchi	albero	progettista
Bianchi	biella	direttore

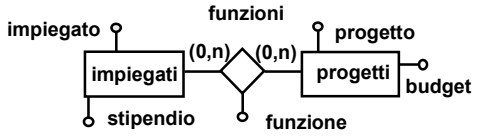
dipendenze funzionali

- FUNZIONI, IMPIEGATI e PROGETTI sono normalizzate perché soddisfano la definizione di forma normale
- la relazione non decomposta può essere ricostruita con il join:

```
SELECT *
FROM IMPIEGATI I, PROGETTI P, FUNZIONI F
WHERE  I.IMPIEGATO  = F.IMPIEGATO
AND      F.PROGETTO = P.PROGETTO
```

dipendenze funzionali

- Quando la relazione originale è ricostruibile con il join la decomposizione è corretta e si dice essere senza perdita
- notare che lo schema corretto corrisponde alla traduzione di:

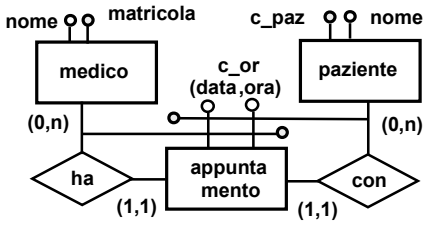


dipendenze funzionali

- Schemi E/R corretti producono in generale buoni schemi relazionali senza problemi di anomalie e ridondanze e corrispondono a decomposizioni senza perdita
- Schemi E/R dove non tutti i vincoli sono espressi nello schema e/o in presenza di associazioni n_arie possono però venire tradotti (non intenzionalmente) in schemi relazionali non ben normalizzati
- È quindi importante ricontrollare la normalizzazione: operazione questa non sempre facile o, possibile per carenza di specifice

dipendenze funzionali

ad esempio, lo schema già visto in precedenza:



dipendenze funzionali

Per mezzo dell'identificazione esterna individua correttamente le due dipendenze:

matricola, data, ora → c_paz
c_paz, data, ora → matricola

e si traduce nello schema relazionale:

medico (matricola, nome)
paziente (c_paz, cognome)
appuntamento (matricola, data, ora, c_paz)

oppure

appuntamento (c_paz , data, ora, matricola)

dipendenze funzionali

- Non sempre le decomposizioni producono effetti desiderabili
- decomposizioni errate possono generare relazioni che, ricongiunte con il join, producono relazioni con dati incerti; si ha quindi una perdita di informazione
- consideriamo un esempio di relazione:

SEDI (impiegato, progetto, sede)
con le dipendenze:

impiegato, progetto → sede
impiegato → sede e progetto → sede

dipendenze funzionali

chiave di SEDI :
impiegato, progetto

Vincolo:
gli impiegati hanno come sede la sede dei loro progetti

impiegato	progetto	sede
Rossi	biella	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	cinghia	milano
Neri	valvola	torino

decomponendo secondo le due dipendenze:

dipendenze funzionali

impiegati

impiegato	sede
Rossi	milano
Verdi	torino
Bianchi	milano
Neri	torino

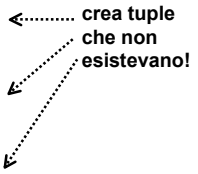
progetti

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

il join sull'attributo comune:
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE
FROM IMPIEGATI I, PROGETTI P
WHERE I.SEDE = P.SEDE

dipendenze funzionali

impiegato	progetto	sede
Rossi	biella	milano
Rossi	cinghia	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	biella	milano
Bianchi	cinghia	milano
Neri	valvola	torino
Neri	albero	torino



dipendenze funzionali

- Inoltre , anche se sono corrette, le due relazioni non rispettano il vincolo che la sede di un impiegato è la sede dei suoi progetti, un progetto potrebbe cambiare sede indipendentemente dagli impiegati
- Regola:
una buona decomposizione deve prevedere la ricostruzione della relazione di partenza con operazioni di join su chiavi
- Osservazione :
i join su attributi che si corrispondono n a m sono rischiosi

dipendenze funzionali

progetti

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

impiegati

impiegato	progetto
Rossi	biella
Verdi	valvola
Verdi	albero
Bianchi	cinghia
Neri	valvola

il join sull'attributo comune:
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE
FROM IMPIEGATI I, PROGETTI P
WHERE I.PROGETTO = P.PROGETTO

dipendenze funzionali

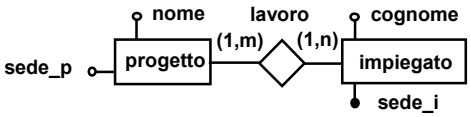
la decomposizione è corretta però abbiamo perso nello schema precedente la dipendenza impiegato → sede

impiegato	progetto	sede
Rossi	biella	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	cinghia	milano
Neri	valvola	torino

problema: che ne è di Verdi se il progetto Albero va a Roma?

dipendenze funzionali

Se fossimo partiti dallo schema E/R?



avremmo comunque dovuto dichiarare, a parte dallo schema E/R, il vincolo che la sede di un impiegato deve essere uguale alla sede dei progetti in cui lavora, e lo schema relazionale: ⇨

progetti

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

impiegati

impiegato	sede
Rossi	milano
Verdi	torino
Bianchi	milano
Neri	torino

lavoro

impiegato	progetto
Rossi	biella
Verdi	valvola
Verdi	albero
Bianchi	cinghia
Neri	valvola

questa soluzione consente anche (0,n) nell'associazione

dipendenze funzionali

il join adesso sarà:

```
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE
FROM IMPIEGATI I, PROGETTI P, LAVORO L
WHERE I.IMPIEGATO = L.IMPIEGATO
AND L.PROGETTO = P.PROGETTO
```

che ottiene la relazione richiesta senza perdita perché lavora su chiavi

però non c'è garanzia sull'uguaglianza di sede

conclusioni generali

- Progettare i dati è difficile
  - il lavoro di gruppo è importantissimo per evitare differenze di percezione e di visione dei problemi
  - DFD, schemi E/R, dipendenze funzionali sono utilissimi per descrivere e capire i problemi
    - tanta più conoscenza si riesce a descrivere negli schemi, tanta meno verrà espressa con vincoli meno leggibili, o dispersa in programmi di difficile lettura e aggiornamento

conclusioni generali

- è bene però anche non eccedere con schemi particolarmente complicati contenenti un eccesso di concetti fittizi e di collegamento che rendono difficile la lettura e la soluzione innaturale
- la documentazione di progetto è pertanto fondamentale