Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Fondamenti di Informatica

Laurea in
Ingegneria Civile e Ingegneria per l'ambiente e il territorio

Gli applicativi software

Stefano Cagnoni e Monica Mordonini

Programmi Applicativi

- Il calcolatore è diventato lo strumento principale per le normali operazioni di ufficio come:
  - Scrivere un documento.
  - Gestire la contabilità.
  - Mantenere un archivio dei clienti / fornitori.
- Per eseguire queste operazioni è molto diffuso l'utilizzo di tre tipi di programmi:
  - Editor di testi
  - Fogli elettronici
  - Sistemi per la Gestione di Basi di Dati

Editor di Testi

- Un editor di testo è un programma che permette di scrivere, modificare, eventualmente formattare, salvare su file e stampare un documento.
- Rispetto alla macchina da scrivere ha molti vantaggi:
  - Separazione tra la fase di stesura e di stampa.
  - Composizione di documenti copiando e incollando parti di testo.
- Gli editor più avanzati permettono di formattare un documento (cioè di impaginare il testo, di usare diversi tipi di caratteri, …) di includere immagini e disegni, di controllare l'ortografia, … .

Fogli Elettronici

- I programmi per la gestione di fogli elettronici hanno lo scopo di fornire all'utente strumenti semplici e potenti per l'analisi dei dati.
- Un foglio elettronico è costituito da una matrice bidimensionale.
- In ogni cella della matrice è possibile inserire:
  - Dei valori numeri e testuali.
  - Delle formule le cui variabili sono riferimenti a celle del foglio elettronico o di altri fogli elettronici.
- I fogli elettronici più avanzati permettono di costruire delle rappresentazioni grafiche dei valori nel foglio elettronico (ad esempio, istogrammi e diagrammi a torte).

Una Base di Dati

- Una base di dati è collezione di dati logicamente coerenti che hanno dei precisi significati.
- Un assortimento casuale di dati non può essere indicato come una base di dati.
- Una base di dati é disegnata, costruita ed utilizzata per mantenere dei dati condivisi da un certo numero di utenti e / o da un certo numero di applicazioni.
- Una base di dati rappresenta degli aspetti del mondo reale. I cambiamenti del mondo hanno riflesso sulla base di dati.

Sistemi per la Gestione di Basi di Dati

- I sistemi per la gestione di basi di dati (DBMS) sono programmi per il mantenimento di grosse quantità di dati strutturati.
- Un DBMS ha il compito:
  - Memorizzare i dati.
  - Aggiornare i dati garantendo la consistenza, la riservatezza e l'integrità.
  - Fornire un accesso ai dati attraverso interrogazioni ad alto livello.

Sistemi per la Gestione di Basi di Dati

- Un Database Management System (DBMS) è un sistema software che si interpone fra le applicazioni e la memoria di massa dove si trovano collezioni di dati, per consentirne la gestione, in modo indipendente dalle applicazioni stesse.
- I dati non appartengono ad una specifica applicazione, ma le diverse applicazioni vi accedono attraverso il DBMS.
- Le basi di dati gestite dai DBMS sono in generale strutture organizzate di dati secondo modelli ben definiti e specificati a priori:
  - **Grandi**: possono avere notevoli dimensioni (fino a migliaia di Gbyte) e devono quindi risiedere nella memoria secondaria.
  - **Condivise**: applicazioni ed utenti diversi devono potere accedere ai dati.
  - **Persistenti**: il tempo di vita dei dati va oltre la durata dell'esecuzione delle singole applicazioni.

Sistemi per la Gestione di Basi di Dati Vantaggi

- Tutte le azioni sui dati vengono svolte dal DBMS:
  - Si controlla la ridondanza sui dati
  - Si controlla l'inconsistenza dei dati
  - Si controlla l'integrità dei dati
  - Si permette la condivisione dei dati
  - Si gestisce l'accesso concorrente ai dati
- Inoltre un DBMS permette di:
  - Imporre degli standard sul formato dei dati
  - Gestire il back up e il ripristino dei dati
  - Restringere l'accesso ai dati
  - Fornire un insieme di interfacce utente
  - Aumentare la flessibilità del sistema
  - Ottimizzare l'uso delle risorse

Sistemi per la Gestione di Basi di Dati Svantaggi

- Quando un programma usa un insieme di dati:
  - Piccolo e ben definito
  - Non cambia col passare del tempo
  - Non è condiviso con altri programmi
  - Quando un programma richiede delle prestazioni in tempo reale.

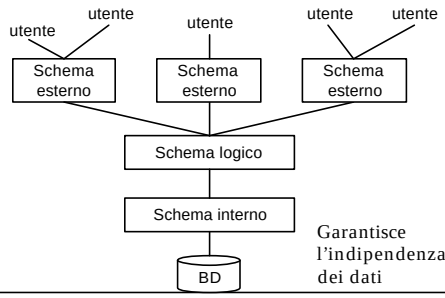
Sistemi per la Gestione di Basi di Dati: Architettura

- Ogni DBMS é basato su:
  - Un modello dei dati
  - Un linguaggio di manipolazione
- I DBMS sono classificati:
  - Per tipo di modello di dati
  - Per numero di utenti che può servire in parallelo
  - Per numero di nodi di calcolo coinvolti

Sistemi per la Gestione di Basi di Dati: Architettura

- Un DBMS dovrebbe fornire una visione astratta sulla base di dati:
  - livello (schema) fisico
  - livello (schema) logico
  - livello (schema) esterno
- Un DBMS dovrebbe fornire:
- Indipendenza fisica: il livello logico non dipende:
  - Dall'implementazione del DBMS
  - Dal calcolatore su cui è installato il DBMS
  - Dal modo in cui in cui i dati sono distribuiti su più macchine
- Indipendenza logica: il livello esterno non dipende dal livello logico

Architettura standard (ANSI/SPARC) a tre livelli per DBMS



Garantisce l'indipendenza dei dati

Indipendenza dei dati

Permette agli utenti di operare a livello astratto, indipendentemente dai dettagli realizzativi del DBMS

- Indipendenza fisica

Consente di mantenere inalterata la struttura logica dei dati al variare della realizzazione fisica del sistema. Consente di utilizzare basi di dati su piattaforme diverse, o la distribuzione di una base di dati su più macchine.

- Indipendenza logica

Rende indipendente lo schema esterno da quello logico, consentendo di inserire nuove viste senza alterarlo, o di alterarlo mantenendo inalterate le viste definite in precedenza

Utenti e progettisti

- Amministratore della base di dati

responsabile della progettazione controllo e manutenzione della base di dati

- Progettisti e programmatori

realizzano i programmi che accedono alla base di dati

- Utenti
  - utenti finali
    accedono alla base di dati frequentemente e attraverso procedure predefinite
  - utenti casuali
    interrogano o aggiornano la base di dati con procedure e modalità variabili, e non predefinite, utilizzando i linguaggi forniti dal DBMS

Sistemi per la Gestione di Basi di Dati

- Esistono diversi tipi di DBMS:
  - Gerarchico
  - Orientato agli oggetti
  - Relazionale
  - Reticolare
- I DBMS relazionali sono i più diffusi.
- I DBMS orientati agli oggetti sono i DBMS del futuro.

Sistemi per la Gestione di Basi di Dati-Il modello relazionale

- In una base di dati relazionale i dati sono organizzati in relazioni (dette anche tabelle).
- Ogni tabella corrisponde ad una relazione
- Una tabella ha un numero fisso di colonne (dette attributi) e un numero variabile di righe.
  - Ogni tabella ha un nome.
  - Ogni attributo ha un nome e un tipo.
  - Il nome della tabella e dei suoi attributi definiscono lo schema della tabella.
  - Ogni riga (detta record o tupla) rappresenta un'istanza della relazione
- Il linguaggio di interrogazione più comune per i DBMS relazionali è SQL (Structured Query Language).

Sistemi per la Gestione di Basi di Dati-Il modello relazionale

- Le righe di una tabella sono definite dall'insieme dei valori corrispondenti ai campi della tabella stessa.
- I valori che possono essere assegnati a ciascun campo sono il dominio di quel campo.
- Il dominio di un campo è l'insieme di tutti e soli i valori che possono

essere assunti da un determinato attributo di una relazione.
(Ad esempio,il dominio dei codici fiscali è formato da stringhe di 16 caratteri composte seguendo una precisa regola).

- Informazioni contenute in tabelle distinte possono essere associate semplicemente per mezzo della condivisione di campi (ossia di domini) tra tali tabelle.

Sistemi per la Gestione di Basi di Dati-Il modello relazionale:Esempio

Consideriamo il seguente schema di basi di dati:

Studenti (`Matricola, Cognome, Nome, DataNascita`)

Corsi (`Codice, Titolo, Docente`)

Esami (`Studente, Voto, Corso`)

Studenti contiene dati su un insieme di studenti

Corsi contiene dati su un insieme di corsi

Esami contiene dati su un insieme di esami e **fa riferimento** alle altre due attraverso i numeri di matricola e il nome del corso.

Quindi `Matricola` e `Studente`, come anche `Corso` e `Titolo`, sono definiti sullo stesso dominio e possono assumere gli stessi valori.

studenti

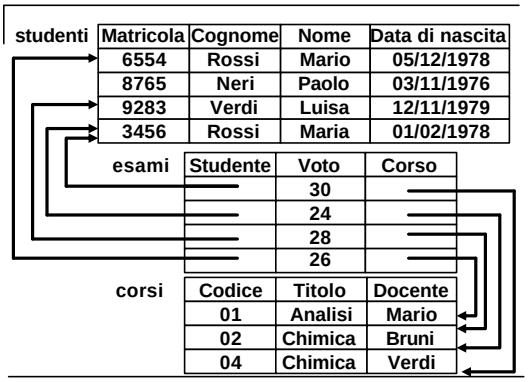
Matricola	Cognome	Nome	Data di nascita
6554	Rossi	Mario	05/12/1978
8765	Neri	Paolo	03/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	01/02/1978

esami

Studente	Voto	Corso
3456	30	04
3456	24	02
9283	28	01
6554	26	01

corsi

Codice	Titolo	Docente
01	Analisi	Mario
02	Chimica	Bruni
04	Chimica	Verdi



Sistemi per la Gestione di Basi di Dati-Il modello relazionale: Le chiavi

- Una tabella (relazione) non dovrebbe contenere due righe identiche per
  - Velocizzare ricerca
  - Mantenimento di update del database
- E' sempre possibile scegliere un sottoinsieme di campi di una tabella in maniera da identificare univocamente ciascuna riga della tabella.
- **Chiave Primaria** (Primary Key, PK) di una tabella: il minimo sottoinsieme di campi che permette di identificare univocamente le righe della tabella.
- Esempio:
  - Il numero di matricola di uno studente ne
  - Il numero di matricola di uno studente e il nome della facolta' a cui e' iscritto in un DB inter-universitario

Sistemi per la Gestione di Basi di Dati-Il modello relazionale: Gli operatori

- Gli operatori relazioni rappresentano la base teorica per i linguaggi di interrogazioni delle basi di dati relazionali.
- Gli operatori relazionali permettono l'esecuzione di ricerche considerando le tabelle come insiemi, senza operare record per record.
- Gli operatori prendono in input tabelle e generano in output nuove tabelle.
- Esistono tre operatori relazionali fondamentali:
  - **SELECT**
  - **PROJECT**
  - **JOIN**

Sistemi per la Gestione di Basi di Dati-Il modello relazionale: Select

- L'operatore SELECT costruisce una nuova relazione:
  - Estraendo un sottoinsieme orizzontale delle righe di una relazione specificata in input.
  - Le righe selezionate sono quelle che soddisfano una condizione espressa sui valori degli attributi della relazione in input.

Impiegati

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
9553	Milano	Milano	44
5698	Neri	Napoli	64

$SEL_{Stipendio > 50}$ **(Impiegati)**

Impiegati (che guadagnano piu' di 50)

Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
5698	Neri	Napoli	64

Sistemi per la Gestione di Basi di Dati-Il modello relazionale: Project

- L'operatore PROJECT costruisce una nuova relazione:
  - Estraendo un sottoinsieme verticale degli attributi di una relazione specificata in input.
  - Rimuovendo inoltre eventuali n-uple duplicate

- visualizzare matricola e cognome di tutti gli impiegati

Matricola	Cognome
7309	Neri
5998	Neri
9553	Rossi
5698	Rossi

PROJ $_{\text{Matricola, Cognome}}$ (Impiegati)

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

PROJ $_{\text{Cognome, Filiale}}$ (Impiegati)

Sistemi per la Gestione di Basi di Dati-Il modello relazionale: Join

- L'operatore JOIN genera una tabella unione di due tabelle sulla base di un attributo (dominio) comune alle due tabelle originali.
- La nuova tabella è formata da righe che sono la concatenazione delle righe della prima tabella con le righe della seconda tabella che hanno lo stesso valore per l'attributo comune.

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
A	Mori
B	Bruni

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

- ogni ennupla contribuisce al risultato:
  - join **completo**

Sistemi per la gestione di dati e Transazioni

- Un DBMS deve garantire una gestione dei dati:
  - efficiente
  - concorrente
  - affidabile
  - integra
  - sicura (protetta)
- Ciascuno degli aspetti precedenti è supportato dal DBMS da specifiche componenti, che complessivamente rappresentano l'architettura del sistema

Transazioni

- Per mantenere le informazioni consistenti è necessario controllare opportunamente le sequenze di accessi e aggiornamenti ai dati
- Gli utenti interagiscono con la base di dati attraverso programmi applicativi ai quali viene dato il nome di **transazioni**
- Una transazione si può interpretare come un insieme parzialmente ordinato di operazioni di lettura e scrittura; essa costituisce l'effetto dell'esecuzione di programmi che effettuano le funzioni desiderate dagli utenti

Transazioni- Definizione

- Un'unità elementare di lavoro svolta da un programma applicativo, cui si vogliono associare particolari caratteristiche di correttezza, robustezza, isolamento.
- Sistema transazionale: mette a disposizione un meccanismo per la definizione e l'esecuzione di transazioni

Transazioni

- Ogni transazione è eseguita o completamente (cioè effettua il commit), oppure per nulla (cioè effettua l'abort) se si verifica un qualche errore (hardware o software) durante l'esecuzione
  - Necessità di garantire che le transazioni eseguite concorrentemente si comportino come se fossero state eseguite in sequenza (controllo della concorrenza)
  - Necessità di tecniche per ripristinare uno stato corretto della base di dati a fronte di malfunzionamenti di sistema (tecniche di ripristino - recovery-)

Transazioni – Concorrenza: esempio

- base di dati che organizza le informazioni sui conti dei clienti di una banca
- il sig. Rossi e' titolare di due conti: un conto corrente (intestato anche alla sig.ra Rossi) e un libretto di risparmio, i cui saldi sono Lit. 100.000 e Lit. 1.000.000
- con la transazione T1 il sig. Rossi trasferisce Lit. 150.000 dal libretto di risparmio al conto corrente
- contemporaneamente con la transazione T2 la sig.ra Rossi deposita Lit.500.000 sul conto corrente

Transazioni – Concorrenza: esempio

T1	T2
Read(Lr)	
Lr = Lr -- 150000	
	Read(Cc)
Write(Lr)	
	Cc = Cc + 500000
Read(Cc)	
	Write(Cc)
	Commit
Cc = Cc + 150000	
Write(Cc)	
Commit	

Transazioni – Concorrenza: esempio

- La somma depositata da T2 è persa (lost update) e non si ottiene l'effetto voluto sulla base di dati
  - l'esecuzione concorrente di più transazioni genera un'alternanza di computazioni da parte delle varie transazioni, detta interleaving
  - l'interleaving tra le transazioni T1 e T2 nell'esempio produce uno stato della base di dati scorretto
  - si sarebbe ottenuto uno stato corretto se ciascuna transazione fosse stata eseguita da sola o se le due transazioni fossero state eseguite l'una dopo l'altra, consecutivamente

Transazioni – Concorrenza: lock

- Idea base:
  - ritardare l'esecuzione di operazioni in conflitto imponendo che le transazioni pongano dei blocchi (lock) sui dati per poter effettuare operazioni di lettura e scrittura
  - due operazioni si dicono in conflitto se operano sullo stesso dato e almeno una delle due e' un'operazione di scrittura
  - una transazione può accedere ad un dato solo se ha un lock su quel dato
- esistono vari protocolli di locking
  - il più noto: two-phase locking

Transazioni Locking a due fasi

- Il controllore della concorrenza e' la parte di software che gestisce le risorse (I dati) e le transazioni serializzando il loro accesso alle risorse che creano conflitti
- Nel locking a due fasi na transazione dopo aver rilasciato un lock non può acquisirne degli altri:
  - prima fase: la transazione acquisisce tutti i lock che le servono (fase crescente)
  - seconda fase (calante) i lock acquisiti vengono rilasciati

Transazioni Locking a due fasi

- Stato di un oggetto:
  - Libero
  - r-locked (bloccato da un lettore)
  - w-locked (bloccato da uno scrittore)
- Ogni read di un oggetto deve essere preceduto da r_lock e seguito da unlock
- Ogni write di un oggetto deve essere preceduto da w_lock e seguito da unlock

Transazioni Locking a due fasi : Tabella dei conflitti

	libera	r-locked	w-locked
r-lock	si	ok	no
wlock	si	no	no

Si: blocco della risorsa il programma procede
no: il programma va in attesa che la risorsa venga sbloccata
ok : il contatore dei lettori viene incrementato per ogni lettore e decrementato dopo operazione unlock (una risorsa può essere letta da più lettori, ma non modificata mentre si legge)

Transazioni: Problema del deadlock

- T1
  - w-lock(d1)
  - w-lock(d2)
  - lettura e scrittura di d1 e d2
  - unlock(d1)
  - unlock(d2)
- T2
  - w-lock(d2)
  - w-lock(d1)
  - lettura e scrittura di d1 e d2
  - unlock(d2)
  - unlock(d1)

Transazione Insorgenza del deadlock

- T1 esegue w-lock(d1)
- T2 esegue w-lock(d2)
- T1 attende una risorsa controllata da T2
- T2 attende una risorsa controllata da T1

Tecniche risolutive del deadlock

- Time-out
  - un'attesa eccessiva è interpretata come deadlock
- Prevenzione
  - evitare alcune condizioni di attesa
- Determinazione
  - ricerca dei cicli sul grafo di attesa

Tecnica più usata: time-out

Dati e Gestione del ripristino dopo un malfunzionamento

- Tre principali tipi di malfunzionamenti:
  - **malfunzionamenti del disco**: le informazioni residenti su disco vengono perse (rottura della testina, errori durante il trasferimento dei dati)
  - **malfunzionamenti di alimentazione** : le informazioni memorizzate in memoria centrale e nei registri vengono perse
  - **errori nel software**: si possono generare risultati scorretti e il sistema potrebbe essere in uno stato inconsistente (errori logici ed errori di sistema)
- Il sottosistema di recovery (ripristino) ***deve*** identificare i malfunzionamenti e ripristinare la base di dati allo stato (consistente) precedente il malfunzionamento

Ripristino: classificazione delle memorie

- Ai fini del ripristino, le memorie vengono classificate come segue:
  - **Memoria volatile**
    - le informazioni contenute vengono perse in caso di cadute di sistema
    - esempi: memoria principale e cache
  - **Memoria non volatile**:
    - le informazioni contenute sopravvivono a cadute di sistema, possono però essere perse a causa di altri malfunzionamenti
    - esempi: disco e nastri magnetici
  - **Memoria stabile**:
    - le informazioni contenute non possono essere perse (astrazione)
    - se ne implementano approssimazioni, duplicando le informazioni in diverse memorie non volatili con
    - probabilità di fallimento indipendenti

Progettare una base di dati

Progettazione di una base di dati

Ciclo di vita di un sistema informativo

- **Studio di fattibilità** definisce le varie alternative possibili e i relativi costi e le priorità di realizzazione.
- **Raccolta e analisi dei requisiti** individua proprietà e funzionalità del sistema tramite interazione con gli utenti e definizione informale dei dati e delle operazioni.
- **Progettazione** divisa in *progettazione dei dati* e *progettazione delle applicazioni.* Individua struttura e organizzazione dei dati e caratteristiche degli applicativi
- **Implementazione** realizza la base di dati e il codice dei programmi conformemente alle specifiche
- **Validazione e collaudo** verifica il corretto funzionamento del sistema informativo
- **Funzionamento** il sistema informativo diviene operativo

Metodologia di progettazione per una base di dati

Una metodologia di progettazione di una base di dati consiste in:

- **decomposizione** del progetto in più passi
- **strategie** realizzative e **criteri** di scelta di alternative
- **modelli di riferimento** per descrivere dati di ingresso e uscita delle varie fasi

e ha le proprietà:

- **generalità** permette l'utilizzo del sistema indipendentemente dal problema affrontato e degli applicativi
- **qualità del prodotto** (correttezza, completezza, efficienza)
- **facilità d'uso** di strategie e modelli di riferimento

Fasi della progettazione

- **Progettazione concettuale**

rappresenta le specifiche informali in modo formale e completo, ma indipendente dalla rappresentazione usata nei DBMS. Produce lo *schema concettuale* e fa riferimento ad un *modello concettuale* dei dati. Rappresenta il contenuto informativo e non la codifica.

Diagrammi Entita'-Relazioni

- **Progettazione logica**

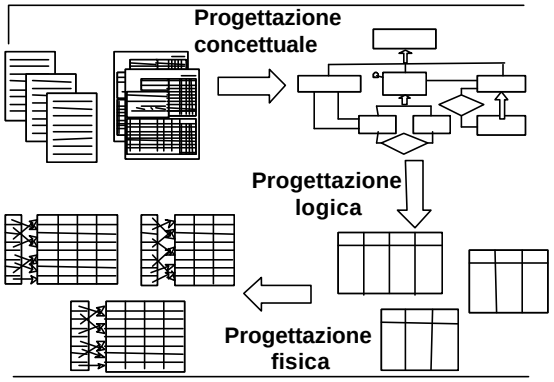
traduce lo schema concettuale nello *schema logico* basato su un *modello logico* (es. modello relazionale) ancora indipendente dalla realizzazione fisica della base di dati.

Modello Relazionale

- **Progettazione fisica**

completa lo schema logico con la specifica dei parametri fisici di memorizzazione dei dati. Produce uno schema fisico e fa riferimento ad un modello fisico dei dati,

Dipendente dal DBMS scelto.



Modello Entità-Relazione (E-R)

E' un modello concettuale di dati e fornisce una serie di strutture (*costrutti*) per descrivere il problema di interesse in modo chiaro e semplice. I costrutti vengono utilizzati per definire schemi che descrivono *struttura* e *occorrenze* dei dati.

Entità

Rappresentano classi di oggetti (cose, fatti ecc.) con proprietà comuni ed esistenza autonoma rispetto all'applicazione. Viene rappresentata graficamente da un rettangolo.

Es. Città, Dipartimento, Impiegato, Acquisto, Vendita

Un'*occorrenza* di una entità è un oggetto della classe relativa.

NB Un'occorrenza di una entità è l'oggetto in sé, non un insieme di valori che lo rappresentano. E' un concetto. Quindi nel modello ER una entità può esistere indipendentemente dalle sue proprietà.

Es: *l'impiegato esiste indipendetemente dal fatto di avere -nome e cognome,…; nel modello relazionale non possiamo rappresentare un oggetto senza conoscerne le sue proprietà che formano la sua tupla*

Modello E-R

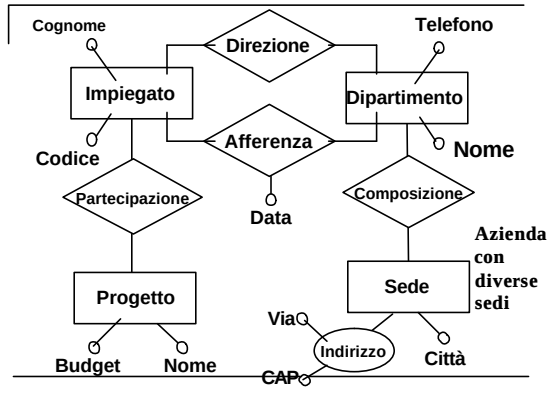
Relazioni (associazioni):

- rappresentano relazioni logiche tra entità, significative per l'applicazione di interesse.
  - Es. Residenza è una relazione fra Impiegato e Città
- Marini-Firenze è un'occorrenza della relazione.
- Ogni relazione ha un nome e viene rappresentata graficamente mediante un rombo.
- L'insieme delle occorrenze di una relazione è una relazione matematica e non ammette duplicazioni.
- Esistono anche relazioni *ricorsive*.
  - *Es.* Collega mette in relazione due occorrenze della stessa entità Impiegato
- Relazioni possono sussistere anche fra più di 2 entità.
  - Es. Fornitura è una relazione fra Prodotto Fornitore e Dipartimento.

Modello E-R

Attributi

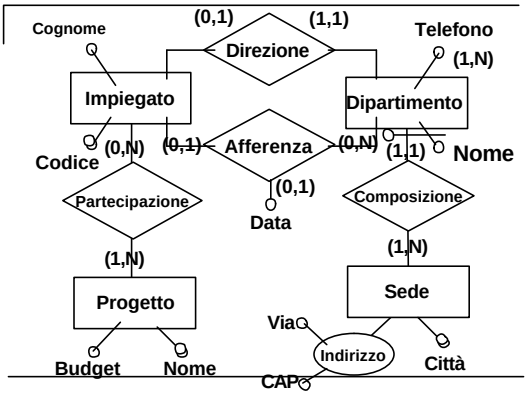
- descrivono le proprietà elementari di entità o relazioni.
  - Es: cognome, stipendio possono essere attributi dell'entità impiegato
- Associano un valore appartenente ad un dominio ad ogni occorrenza di entità o relazione.
- I domini non si riportano di solito nello schema grafico, ma solo nella documentazione. Rappresentano l'insieme dei valori ammissibili per l'attributo.
  - Es: cognome, attirbuto di impiegato, può avere come dominio l'insieme delle stringhe di 20 caratteri
- Se più attributi sono logicamente affini fra loro possono essere combinati insieme in un *attributo composto*.
  - Es. Indirizzo può essere composto da Via, N. Civico e CAP.
- Graficamente si rappresentano:



Modello E-R

Identificatori delle entità

- Si specificano per descrivere i concetti (attributi e/o entità) che permettono di identificare univocamente le occorrenze di un'entità.
- Se un certo numero di attributi sono sufficienti ad identificare univocamente una entità si parla di *identificatore interno* o *chiave*.
  - Es: un identificatore interno per l'entita' Automobile è l'attributo targa
- Se c'è bisogno anche di altre entità per identificare univocamente un'entità si parla di *identificatore esterno*.
  - Es. Studente non è identificato univocamente da Matricola se nella base di dati si contempla la presenza di più Università.



Osservazioni

- Il nome di una città identifica una sede dell'azienda
  - Non c'è più di una sede in ogni città
- Un dipartimento viene identificato dal nome e dalla sede in cui fa parte
  - Una sede ha diversi dipartimenti, ma un dipartimento ha una sola sede
- Un impiegato (identificato da un codice) può afferire ad un solo dipartimento (anche nessuno se appena assunto)
  - Viceversa ogni dipartimento può avere più impiegati ma un solo direttore
- Il nome individua i progetti
  - A cui possono lavorare più impiegati (almeno uno)
  - Un impiegato può a lavorare a più progetti ma anche a nessuno