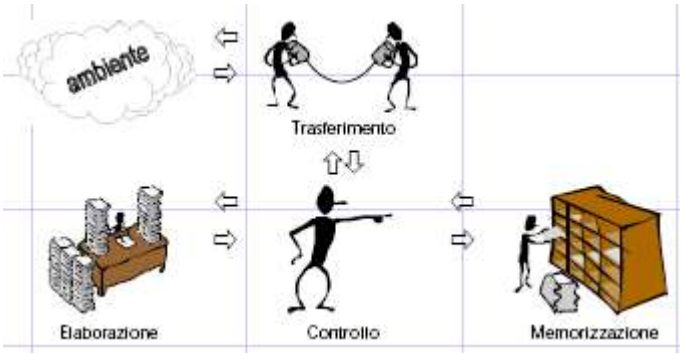
Sistema informativo e sistema informatico di un'organizzazione

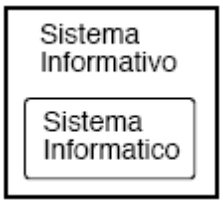
Un **sistema informativo** *(information system, IS)* è un insieme organizzato di procedure e risorse umane e materiali utilizzate per *la raccolta*, *l'archiviazione*, *l'elaborazione* e *la comunicazione* di informazioni necessarie ad un'organizzazione (azienda, comune, ferrovie, aeroporti, scuola etc.) per gestire sia le attività operative che quelle di governo quindi:

- *attività operative* (per il funzionamento normale, di servizio)
- *attività di gestione* (per la previsione dell'aumento o della diminuzione delle vendite in alcuni periodi dell'anno - strategie)
- *attività di programmazione, controllo e valutazione* (per garantire il prodotto al cliente)

Il sistema informativo è quindi formato da due parti strettamente legate tra loro, quella **tecnica** (Information Technology, IT) e quella **sociale** (personale, strategie, strutture organizzative, norme, etc.).

Le attività di un SI:
raccolta,
comunicazione,
archiviazione,
elaborazione

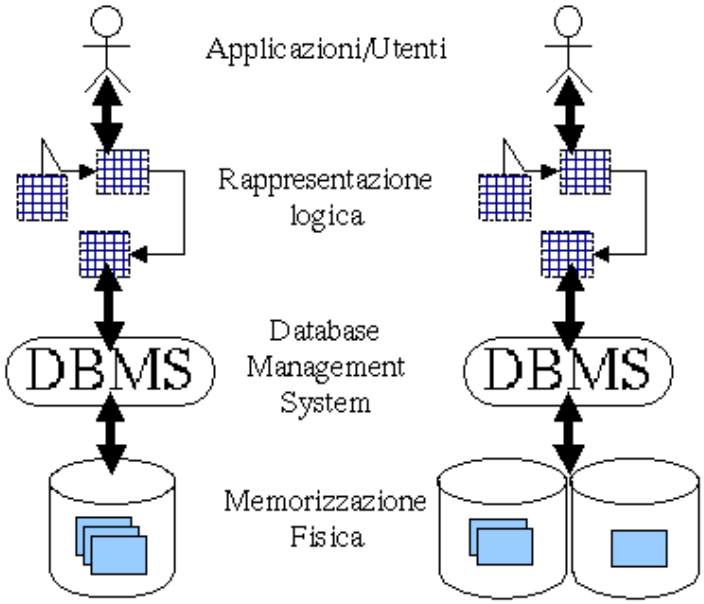




Un **sistema informatico** è un *sottoinsieme del sistema informativo dedicato alla gestione automatica di informazioni* e fa uso di risorse materiali di tipo informatico.

database e collegamento Web

***Def**: Un database è un insieme di dati strutturati in cui è possibile effettuare ricerche e modifiche.*
*Le operazioni fondamentali che si possono eseguire su un database sono: **inserimento, ricerca, aggiornamento** e **cancellazione**.*

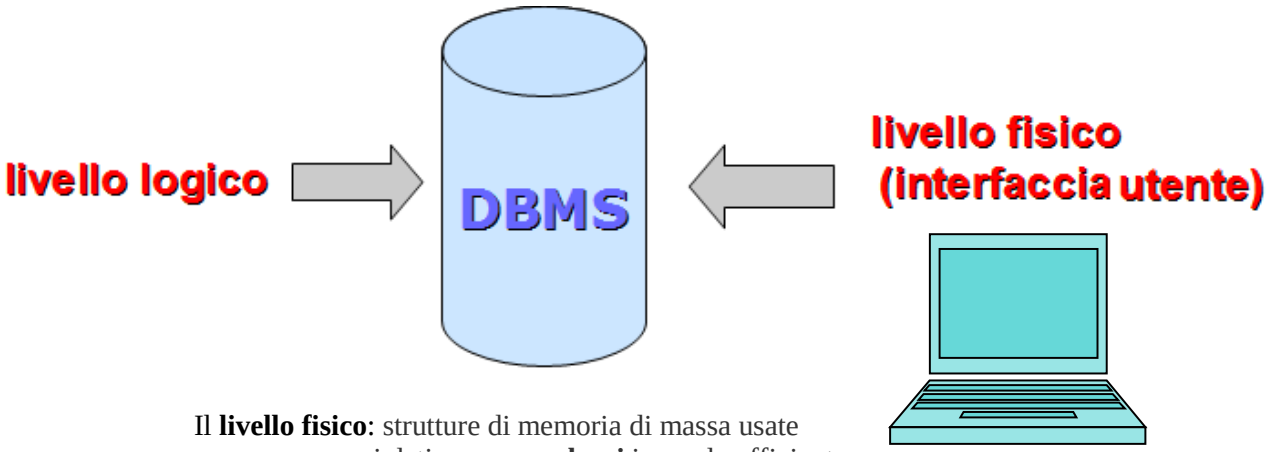


In un'architettura semplificata nella gestione di una base di dati (in inglese ***database)***, si distinguono in particolare due livelli (tradotta l'astrazione dello schema concettuale in quello relazionale):

- **Il livello logico**

- **Il livello fisico**

Il **Data Base Management System** (**DBMS**[1]) mette in relazione i due livelli, ne è il *collante* e su di esso si basano applicazioni come Microsoft Access. Esempi: Oracle, Microsoft SQL server, MySQL, PostgreSQL, InterBase.

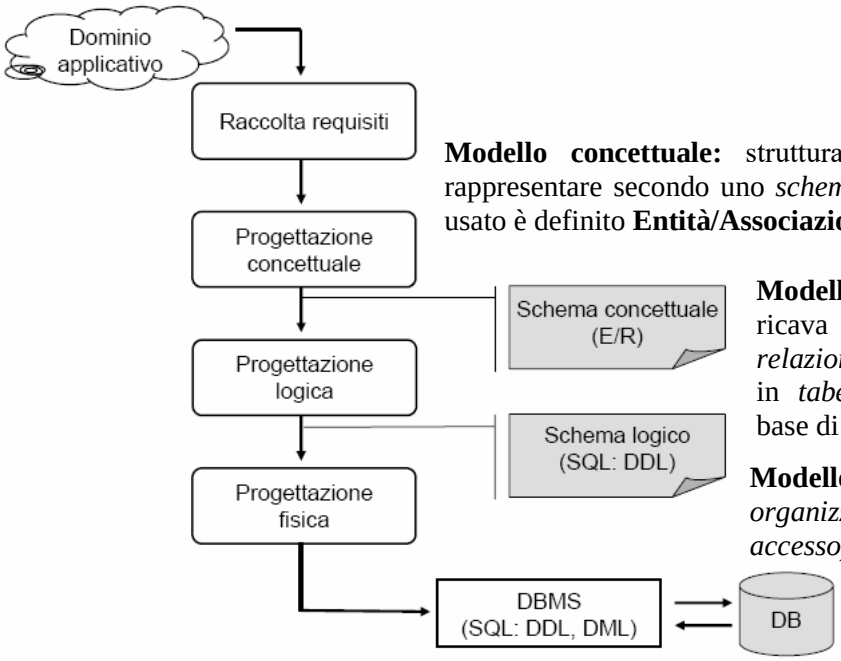


Il **livello fisico**: strutture di memoria di massa usate per conservare i dati e per **accedervi** in modo efficiente. Nascosto all'utente (*l'interfaccia utente* è ciò che l'utente visualizza a video)

I modelli dei dati e la progettazione di una base di dati

Un **modello dei dati** è uno strumento linguistico in grado di attribuire una "struttura" ai dati (imponendo vincoli e permettendo di manipolarli). I diversi modelli dei dati si possono dividere in tre grandi categorie, corrispondenti a tre livelli decrescenti di *astrazione con cui i dati vengono resi disponibili all'utente.*

Le **fasi di progettazione** di una **base di dati** sono **strutturate** infatti secondo i seguenti **modelli:**



Modello concettuale: strutturazione della "realtà" che si vuole rappresentare secondo uno *schema concettuale*. Il modello concettuale usato è definito **Entità/Associazioni (**entity-relationship**).**

Modello logico: dallo schema concettuale si ricava lo schema *logico*. Si usa il *modello relazionale* che traduce entità e associazioni in *tabelle* (**relazioni**) che costituiscono la base di dati.

Modello fisico: più che modello, *organizzazione fisica* dei dati *(o metodi di accesso)*

nb: DDL o linguaggio di definizione dei dati, DML o linguaggio di manipolazione dei dati (implementati con *SQL)*

---

[1]**DBMS**: *Data Base Management System*. Sistema centralizzato (programmi coordinati) o distribuito (rete) che permette di memorizzare, modificare ed estrarre ***informazioni*** da un database, permettendo l'indipendenza del SW dall'organizzazione fisica e logica delle strutture dati. Un DBMS:

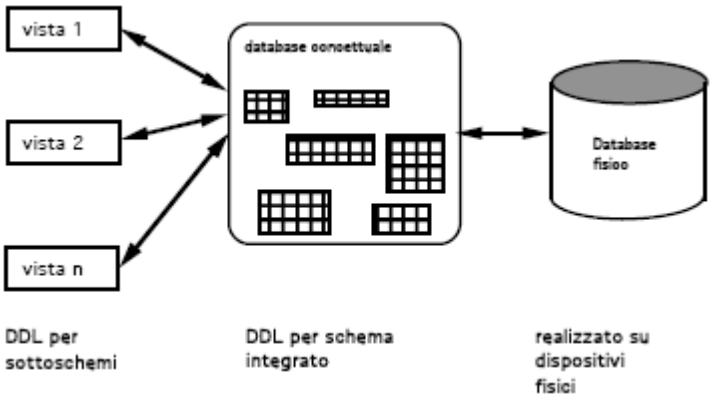
- Garantisce *l'integrità dei dati* (unica raccolta di dati anziché copie distinte scoordinate che potrebbero causare duplicazioni, ridondanze) assicurando **consistenza** cioè coerenza (specie sicurezza negli aggiornamenti) evitando contraddizioni tra i dati archiviati.
- Organizza le informazioni del database secondo la struttura di un database **gerarchico**, di un database **di rete** o di un database **relazionale** o ad **oggetti** gestendo grandi moli di dati in un **ambiente multiutente**, consentendo **elaborazione concorrente**
- Garantisce l'accesso **concorrente** alle informazioni aumentando la sicurezza - intesa come riservatezza - sia a livello logico (solo a persone **autorizzate** p.e. tramite una password) sia a livello fisico (si possono impostare *privilegi diversi*)
- Un DBMS oltre a stabilire schemi organizzativi e di controllo, rende le informazioni accessibili agli utenti, tramite query.

❑ **Progettazione logica:** si sviluppa secondo il *modello dei dati* che in fase di progettazione è stato scelto per rappresentare una realtà in base alle caratteristiche che si vogliono evidenziare. L'unità logica di *database relazionali* è la tabella (**relazione** identificata da un nome, costituita da un insieme di *tuple* (le righe della tabella) ciascuna composta da diversi *campi* (le colonne) che conservano il valore degli attributi della relazione.

❑ **Progettazione fisica:** dallo schema logico si passa allo *schema fisico* che porterà all'implementazione finale del database.

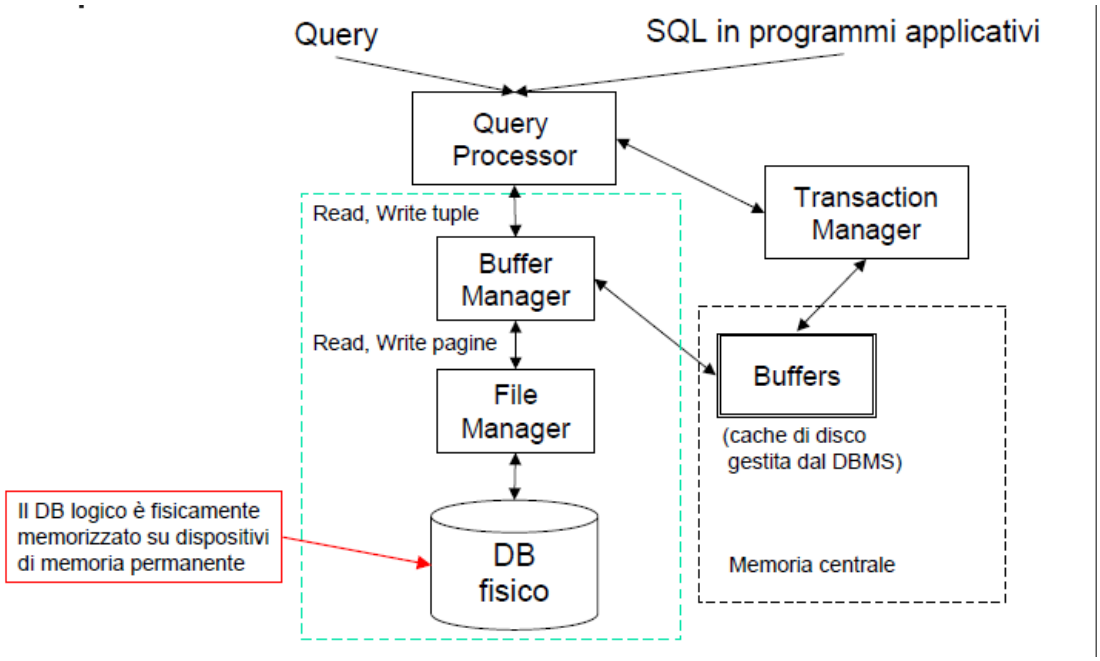
Il DBMS mantiene un modello astratto dei dati, l'utente **vede** quindi i dati in base a ciò che, per esso, rappresentano.

Esisteranno dunque *diversi livelli di astrazione*, sia per l'istanza che per lo schema.



(**l**inguaggio di **d**efinizione dei **d**ati che permette di agire sullo schema del database, creando, modificando o eliminando oggetti)

Tra tali livelli di astrazione, il **livello esterno**, o **vista**: quello più vicino all'utente 'finale' del database, che corrisponde al **modo di vedere i dati** da parte dell'utente stesso.



Livello fisico o **interno**: il più vicino all'elaboratore, che corrisponde al modo secondo il quale i dati sono effettivamente memorizzati. È rappresentato dalle strutture di memoria di massa usate per conservare i dati e per accedervi in modo rapido ed efficiente. È necessario distinguere tra:

- I **dati** veri e propri
- Le **strutture** che li contengono e che ci permettono di **accedere** ai medesimi

Le **strutture fisiche di accesso** descrivono il modo in cui vengono organizzati i dati per garantire operazioni di ricerca e di modifica efficienti da parte dei programmi applicativi. In genere, ciascun DBMS ha a disposizione un numero abbastanza limitato di tipi di strutture di accesso; ad esempio, nei sistemi relazionali sono disponibili **semplici indici**, che vengono definiti dal progettista tramite istruzioni *DDL* (con possibilità di eliminare tale indicizzazione in modalità diversa a seconda del DBMS).

Il livello fisico è, per l'utente, del tutto trasparente: egli, infatti, non si preoccupa affatto di come i dati vengano registrati sui supporti, tale funzione è compito esclusivo del DBMS. L'utente si occuperà principalmente ***del cosa*** vi è registrato: quali sono i dati e in quale relazione si trovano tra loro.

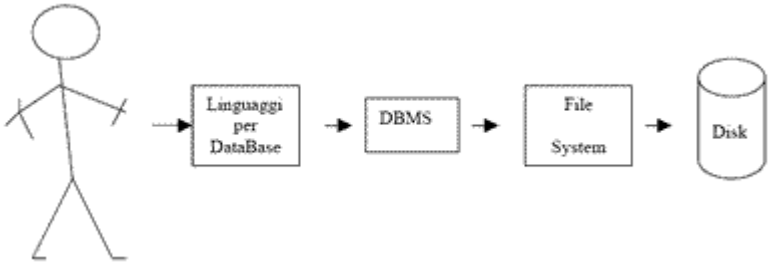
Essendo la gestione dei file realizzata dal sistema, l'utente non deve interessarsi della forma assunta dai dati in memoria di massa e di come accedervi; non gli resta dunque che occuparsi del valore informativo dei dati.

Differenze tra DBMS e file system

Il file system è un nucleo, costituito da programmi, presente in ogni sistema operativo. La sua funzione è quella di gestire le varie operazioni sui file; questa gestione non è visibile all'utente, infatti il file system opera direttamente al servizio di altri programmi o utility.

Il DBMS è per l'appunto un software che poggiando sul sistema operativo utilizza il file system di quest'ultimo, consentendo **indipendenza del SW** dall'organizzazione logica e fisica.

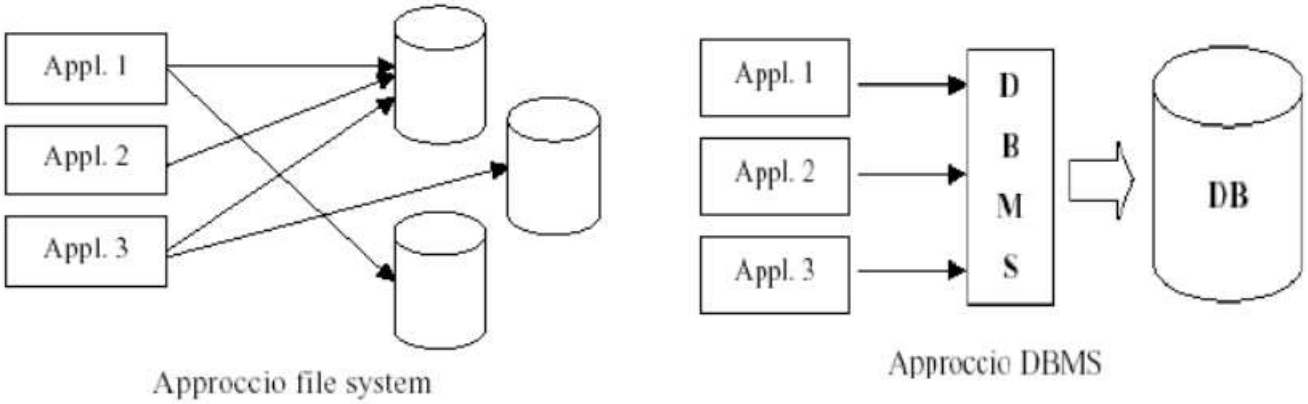
In questo caso dunque, la maggiore 'distanza' del DBMS dall'hardware, rispetto al file system, permette un grado di iterazione maggiore. Questo significa che l'utente (programmatore, amministratore di sistema, ecc.) non dovrà più avere a che fare con record e file, bensì con entità astratte che rappresentano la realtà.



Una suddivisione semplificata (quindi parziale), utile a comprendere per linee generali il comportamento di un DBMS, potrebbe essere questa:

1. Gestore delle interrogazioni
2. Gestore dei ***metodi di accesso*** (ad esempio ***accessi sequenziali indicizzati*** ***I**ndexed **S**equential **A**ccess **M**ethod o ISAM*)
3. Gestore del buffer (***Buffer manager***)

*Per un confronto: la tradizionale gestione mediante archivi **vs.**[2] l'attuale gestione con DBMS*



Prima dello sviluppo dei DBMS l'approccio che veniva applicato al problema dell'archiviazione prevedeva l'uso diretto delle strutture del file system. In tale soluzione, le applicazioni accedevano direttamente agli archivi, quindi il programmatore doveva conoscere la struttura interna degli archivi e le relazioni tra i dati e doveva evitare la duplicazione degli stessi. Inoltre la non volatilità dei dati e la gestione degli accessi contemporanei di più applicazioni agli archivi veniva relegata a strati software non specializzati per tali compiti, quali il sistema operativo.

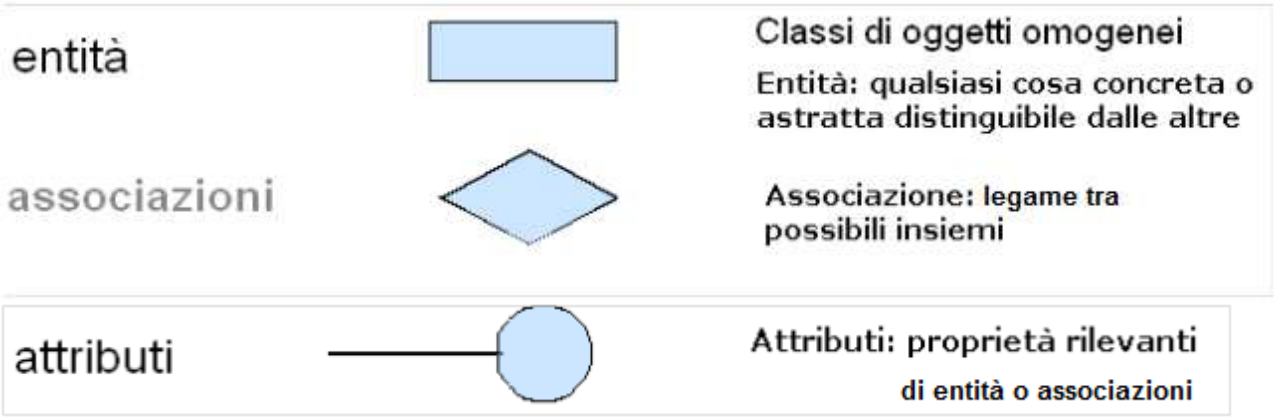
La caratteristica saliente che differenzia un sistema per la gestione di database è la presenza di un componente specializzato a tale ruolo. In tale soluzione, le applicazioni rivolgono al DBMS le proprie **richieste di accesso alla base di dati**. Si ottiene così un triplice scopo: da una parte le funzionalità di gestione del database sono raggruppate in un unico insieme, dall'altra le applicazioni risultano alleggerite e quindi più veloci da realizzare e, soprattutto, **nessuna potrà effettuare operazioni scorrette sul database**.

---

[2] Da *guida alla lettura di appunti sui Database e DBMS*

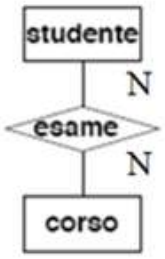
MODELLO CONCETTUALE

Il **modello E-R** consiste **in una descrizione a diagrammi** i cui **concetti base** sono**:**

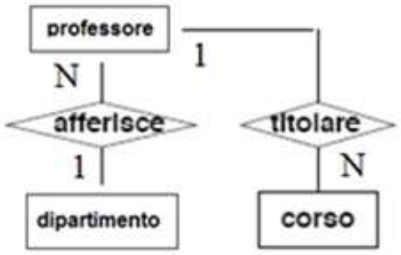


La rilevanza degli attributi si intende per gli scopi informativi dell'organizzazione che ha commissionato il progetto e, quindi, dipende dal contesto (detto universo del discorso o *minimondo*)

- **Contesto 1 – Vista 1**
  - Gli studenti sostengono esami per i vari corsi



- **Contesto 2 – Vista 2**
  - I professori afferiscono ai dipartimenti e hanno titolarità di corsi (unico dipartimento e più corsi)



Si tradurrà[3]

Modello logico relazionale (*intensionale*)

```
STUDENTE(Matricola,Cognome,Nome,DataNascita)
ESAME(Matricola,CodCorso,Voto,DataEsame)
CORSO(CodCorso,NomeCorso,CodProf)
PROFESSORE(CodProf,Cognome,Nome,CodDip)
DIPARTIMENTO(CodDip,Denominazione,Indirizzo)
```

[3] Con opportune **regole di trasformazione**

Conoscenza astratta

Di seguito gli elementi che rappresentano l'aspetto della *conoscenza astratta* nel **modello Entità/Associazioni**: ***classificazione*** e ***aggregazione***

Classificazione (meccanismo di astrazione):

→ *astrarre* dalle differenze fra le singole istanze (di entità o associazioni) per evidenziare ciò che le rende **omogenee in un certo contesto**.

Esempio: istanze diverse (cioè elementi dell'insieme diversi) di entità come il Prof. Rossi, il Prof. Verdi vengono classificate come "**Docente**" per mettere in evidenza che di essi interessano i valori di proprietà tipiche dei docenti come: *il nome, la data di nascita, la materia insegnata, etc.*

Le istanze vengono raggruppate (***classificate***) in *classi* denominate appunto ENTITÀ per le istanze di entità, ASSOCIAZIONE per le istanze di associazione.

La **classe** è quindi un insieme di *istanze* (oggetti) considerate dello stesso ***tipo*** in un certo *contesto*.

Esempio: C*ittadino, Docente, Studente;*

La classificazione nei **sistemi informativi** è **rigida**.

Aggregazione: meccanismo che permette di definire il ***tipo*** (la struttura) delle istanze delle classi come ***aggregazione*** di *proprietà comuni*.

Esempio*:* Docente *(nome, cognome, materia insegnata,…).*

Per chiarire il concetto di ***aggregazione***: *nome, cognome, materia insegnata, qualifica* sono i nomi delle proprietà che, aggregate, costituiscono il **tipo** delle istanze della classe Docente.

Due istanze della classe possono essere: Giovanni, Rossi, Informatica, Prof di Ruolo;
Dario, Verdi, Matematica, Prof di Ruolo ;
per le quali si registrano i valori.

La ***classificazione*** introduce dei ***vincoli di integrità***. Per **VINCOLO di INTEGRITÀ** si intende una regola che si esprime sullo schema (concettuale o logico) ma che specifica una condizione che deve valere **per ogni istanza** dello schema. Tali vincoli servono per mantenere la ***consistenza*** della base di dati. Infatti, non tutte le istanze della base di dati sono lecite**.** I vincoli di integrità nel modello E-R:

- gli elementi di una classe sono dello stesso tipo ma identificabili → **unicità** rispetto ad una **chiave**: un insieme di attributi non nullo – detto *chiave primaria* – deve permettere di individuare univocamente ogni istanza,
- vincoli di **cardinalità** sul numero di elementi (valori massimo e minimo) che possono essere coinvolti nella partecipazione all'associazione o più sinteticamente vincoli di **cardinalità sulle associazioni** che esprimono il numero di massimo di entità a cui un'altra entità può essere associata tramite legami dello stesso tipo
- vincoli di **cardinalità sugli attributi**: gli attributi possono assumere solo un certo range di ***valori*** → **restrizione sul dominio** (che dipende dal **tipo** o è ulteriormente limitato: si pensi all'età che non può essere < 1 anno)
- vincoli di **partecipazione** (o vincoli di *cardinalità minima*): specificano se l'esistenza di una entità dipende dal suo essere correlata ad un'altra entità mediante una associazione. Esprime l'obbligatorietà od opzionalità
  - o *Partecipazione Totale* se i requisiti asseriscono che ogni elemento dell'entità deve partecipare all'associazione
  - o *Partecipazione Parziale* o *Opzionale* se i requisiti asseriscono che elementi dell'entità possono non partecipare all'associazione
- Altri vincoli (esterni) detti ***non esprimibili*** (nel DEA) quali le ***regole di derivazione*** o *business rules:* regole o criteri per definire e controllare la struttura, il funzionamento e la strategia di un'organizzazione. Non sarà sufficiente un DDL ma richiedono elementi di programmazione

I vincoli di integrità nel modello logico: le associazioni, nel loro costituire legami tra istanze, potranno essere controllate nel trasferimento degli effetti delle modifiche → **integrità referenziale** (regole di controllo: un campo "puntatore" - detto *chiave esterna* - deve far riferimento ad un'istanza esistente)

***Classificazione* Contesto:**

Entità, associazioni e proprietà **non sono fatti assoluti** ma **dipendono dal contesto**:

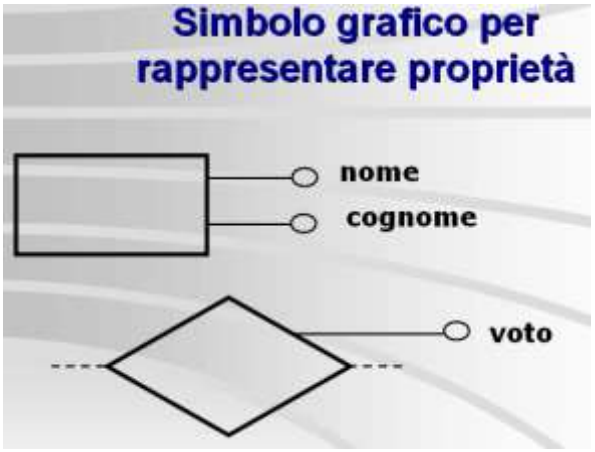
Esempio: l'auto CF859ZA ha colore rosso;
il colore rosso ha lunghezza d'onda = ~700 nm;
Il Prof. Verdi insegna Matematica;

Linee guida per il progetto concettuale

- 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 (*non esiste senza dipenderne*) : **associazione**

***Classificazione* Simboli grafici:**

- ❑ Il modello E-R usa simboli grafici per favorire l'immediatezza della comprensione (come altri metodi e modelli tipici dell'ingegneria, es.: mappe topografiche, schemi elettrici, meccanici ecc.);
- ❑ gli schemi E-R sono schemi essenzialmente grafici con aggiunte di frasi di specifica e di vincolo.



Non esiste uno standard per rappresentare tale schema: vengono accettati diversi tipi di schemi

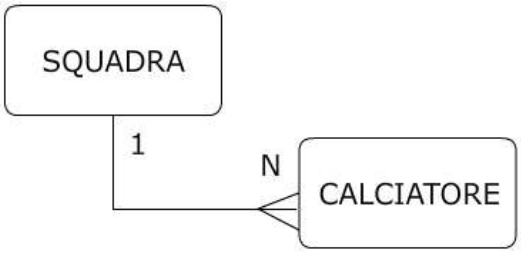
Dal modello per la rappresentazione concettuale dei dati ad un alto livello di astrazione, formalizzato dal prof. Peter Chen nel 1976, semplificato nella proposta sopra descritta, si passa a rappresentazioni più adatte[4] se si usano applicativi (*SW per il design di DB*) che subiscono anche l'influenza dell'uso del linguaggio UML.(*Per confronto e motivazione all'uso)*

Ad esempio - con riferimento alle proposte risolutive di temi di maturità - viene usata la **Crow's Foot Notation** per illustrare lo schema E/R.

In tal caso, volendo considerare un "*minimondo*" che illustri il legame tra una squadra di calcio ed i molti calciatori che la compongono si potrà usare il grafico a lato,

arricchendo poi con frasi di s*pecifica* e di *vincolo*

TERMINOLOGIA: Il **grado** di un'associazione è il numero di entità *associate* alla stessa.



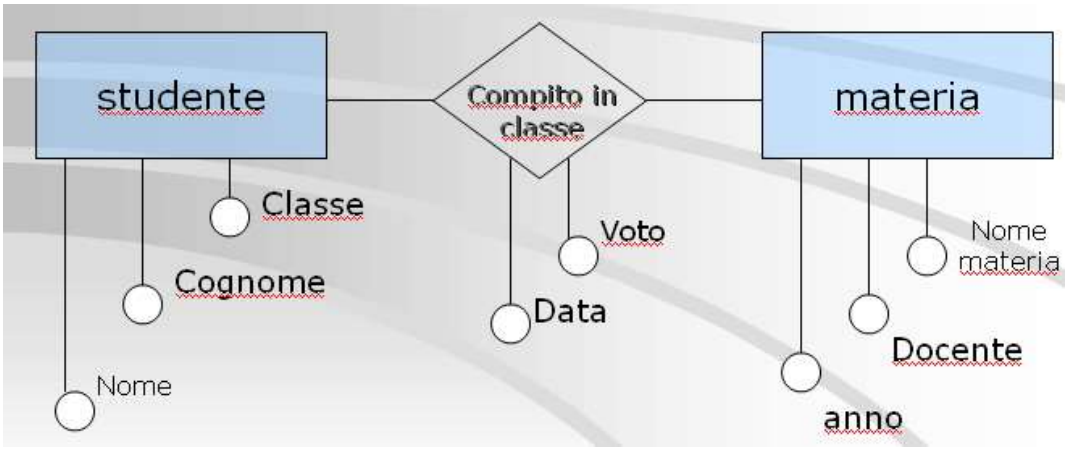
[4] **Corso Progettazione Data Base ITA 1 - presentazione del corso e strumenti** (videolezione di f. camuso)

Conoscenza concreta

Costrutti fondamentali del modello E-R: **proprietà o attributi** cioè fatti che descrivono le caratteristiche delle istanze di entità e le caratteristiche delle istanze di associazione.
Le *proprietà* assumono **valori**.
Le *proprietà* rappresentano l'aspetto della *conoscenza concreta* nel modello Entità/Associazioni.

Esempi di proprietà - proprietà di **istanze** di ***associazione***.
"Lo studente Rossi Fabio merita votazione 8 nello svolgimento del compito di Chimica"

Esempi di proprietà - proprietà di **istanze** di ***entità*** (rappresentazione grafica seguente):



nb: la semantica del modello E/R richiede che una associazione, pur con attributi, non abbia identificatori

La struttura concreta, a *livello logico,* può essere implementata con **tabelle** composte da *record* e *campi*. A *livello fisico*, diversi DBMS archiviano in unico file o in file distinti (consentendo interazione con la directory che li contiene)

Un **database** è una ***raccolta di dati*** (*strutturati e logicamente connessi*)

Un database è composto da *tabelle*. Ogni tabella è composta da *record* e *campi*. I database possono essere composti da più tabelle. Tali **tabelle** possono essere messe in ***relazione*** tra loro tramite connessioni logiche (presenza in più tabelle della stessa informazione). Da questa definizione deriva il nome di Database Relazionale (RDBMS, Relational Database Management System) assegnato alla tipologia di prodotti che esamineremo. Attenzione, spesso si indica con lo stesso nome (Database) sia il "motore" cioè il software che mi permette di gestire le informazioni, sia le informazioni stesse. Questo, in generale, non è corretto.

Facciamo un esempio di unica tabella: una **rubrica** telefonica che raccoglie in un **Elenco** le *proprietà* considerate significative

Nome	Indirizzo	Città	CAP	PR	Tel
Centro Internazionale Reiki	Via Lonate 6	Turbino	20029	MI	0331891111
Ritorante San Pietro	Via Alzaia Naviglio Grande 18	Robecchetto con Induco	20020	MI	0331875402
Silvio Crispiatico	Via Lonate 6	Turbino	20029	MI	0331891111

I *campi* sono "Nome", "Indirizzo", "Città", "CAP", "PR", "Tel" (ogni colonna) mentre i *record* sono gli utenti della rubrica cioè un insieme di campi (ogni riga o *t_pla*).

Nella **progettazione** di un database – passati dallo schema concettuale a quello logico – si parla di *definizione delle tabelle* che fanno parte del database. Per ogni tabella si *definiscono i campi* che rappresentano la **struttura** della tabella: i *nomi* ed il *tipo* dei valori di tali campi.

In un approccio ingenuo, si impostano i "*legami*" tra le tabelle che permettono di **normalizzare** (spezzare la *tabella grassa* in più tabelle *magre*) evitando **ridondanze,** raggiungendo un adeguato grado di **efficienza** e si prevederà un controllo su errori (anomalie di inserimento, cancellazione, aggiornamento) impostando l'**integrità referenziale**[5].

> *" insieme di regole usate per assicurare che le relazioni tra i record delle tabelle correlate siano valide e che non vengano eliminati o modificati per errore i dati correlati."*

[5] Possibile se il campo della tabella primaria è una chiave primaria e i campi correlati contengono lo stesso tipo di dati.

Si distinguono infatti tre tipi di **anomalia**:

1. *Anomalia di inserimento*. Se nell'inserire un nuovo record in una tabella si è costretti a inserire informazioni già presenti nel DB.

2. *Anomalia di cancellazione*. Se nel cancellare un record si è costretti a cancellare informazioni che possono essere ancora utili nel DB.

3. *Anomalia di aggiornamento*. Se per aggiornare un record si è costretti ad aggiornarne molti altri.

Un errore di progettazione molto comune che dà quasi sempre luogo ad anomalie è quello di voler realizzare il Data-Base con un'unica grande tabella che contenga tutte le informazioni possibili. In gergo una tale tabella è detta ***tabella grassa*** (in inglese: *fat table*). Si deve, invece porre attenzione ad **ottimizzare** la **struttura logica** trovando il giusto compromesso tra **normalizzazione** (spezzando in più tabelle relazionate con codici di riferimento) e **denormalizzazione** (per aumentare le performance nel recupero dell'informazione originale si creano attributi che contengono dati ricavati da altre relazioni)

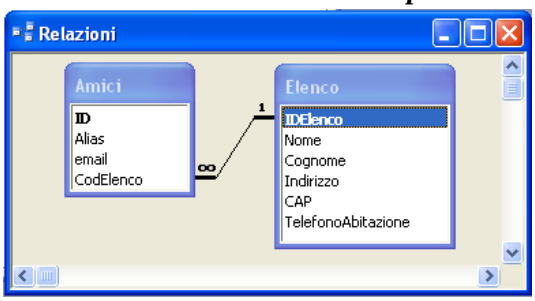
Vediamo come si costruisce una rubrica telefonica con uno dei tanti **database tipo desktop** presenti in commercio di tipo *Relational DataBase Management System* (Access della suite Office oppure Base della suite OpenOffice)

Diamo un nome significativo al nostro **database** (es. **rubrica**). In questo file verranno raccolti tutti i dati che andremo ad inserire. Creiamo con procedura guidata la **tabelle** di nome **Elenco** e **Amici**: la prima, la principale, memorizzerà le informazioni tipiche di una rubrica telefonica come nell'esempio sopra illustrato, la seconda memorizzerà *alias* ed email dei nostri amici e si imposterà un campo che costituisca *riferimento* all'altra tabella per recuperare dei nostri amici le informazioni relative al "Nome", "Indirizzo", "Città", "CAP", "PR", "Tel" senza ridondanze e possibili errori di digitazione: prima si imposta la struttura (nome e tipo dei campi), si impostano le ***relazioni*** che connettono logicamente le tabelle e solo in seguito si popolano le tabelle.

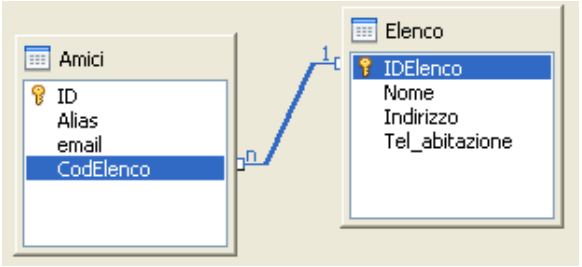
È obbligatorio creare, per ogni tabella, la "*chiave primaria*": per il massimo di efficienza un identificativo (ID) numerico (un contatore che si auto-incrementa).

Chiave primaria

Derivando dallo schema E-R, ogni tabella deve contenere un campo (o un insieme di campi non nullo) che consenta di *identificare ogni dato* (t_pla) in *modo univoco*. Questo insieme viene definito ***chiave primaria***. Se nell'esempio si fissa che non possano esistere due o più nomi con lo stesso **valore** allora si potrebbe impostare questo campo come chiave primaria.



Access o Base semplificano l'impostazione delle *relazioni* tra tabelle mediante interfaccia visuale



Suggerimento nella scelta della chiave primaria: *unico campo di tipo contatore auto-incrementante*

Normalizzazione: le prime tre forme normali

Definizioni: dove con *attributo* si intende *valore di campo*

- Prima forma normale (1FN): una tabella si dice in 1FN se non presenta *attributi* multipli.
- Seconda forma normale(2FN): una tabella si dice in 2FN se è in 1FN e inoltre non vi sono *attributi* che dipendono parzialmente dalla chiave. (Se la tabella ha già una chiave su un solo *attributo atomico* allora è già in 2FN).
- Terza forma normale(3FN): una tabella si dice in 3FN se è in 2FN e inoltre non vi sono *attributi* che dipendano transitivamente dalla chiave (cioè non esistono dipendenze tra le colonne di una tabella se non quelle basate sulla chiave primaria).

La prima forma normale (1NF)

La **prima forma normale** definita per un database esprime un concetto semplice ma fondamentale: **ogni riga di ciascuna tabella deve poter essere identificata in modo univoco tramite un gruppo di dati in essa contenuti**. In altre parole, in una tabella del tipo:

Nome	Età	Professione
Alberto	**30**	**Impiegato**
Gianni	24	Studente
Alberto	**30**	**Impiegato**
Giulia	50	Insegnante

non è possibile distinguere il dato inserito nella prima riga da quello inserito nella terza: le due righe sono infatti identiche. L'Alberto della prima riga, di 30 anni impiegato, non è infatti distinguibile dall'Alberto, 30 anni, impiegato, della terza riga.

Il problema potrebbe essere risolto inserendo un altro campo nella tabella, con valore diverso per ogni riga, ad esempio il codice fiscale. A questo punto il database sarebbe in **prima forma normale**.

Il campo o l'insieme di campi diversi per ciascuna riga e sufficienti ad identificarla sono detti **chiave primaria** della tabella (in questo caso il codice fiscale).

Codice Fiscale	Nome	Età	Professione
LBRRSS79Y12T344A	Alberto	30	Impiegato
GNNBNCT84A11L611B	Gianni	24	Studente
LBRMNN79E64A112A	Alberto	30	Impiegato
GLSTMT59U66P109B	Giulia	50	Insegnante

Il miglioramento delle prestazioni in questo caso è piuttosto evidente: la presenza di righe uguali all'interno di una tabella è infatti un evidente sintomo di *inefficienza*, che viene così eliminato.

Proprio per questo tutti i principali software di gestione database (come ad esempio SQL Server e Oracle) richiedono all'utente di definire una chiave primaria per ogni tabella creata.

Si noti che la presenza di una chiave primaria è requisito indispensabile ma non sufficiente affinché una base dati sia in prima forma normale; infatti è altresì necessario che non vi siano *gruppi di attributi* che si ripetono (ossia **ciascun attributo deve essere definito su un dominio con valori atomici**).

La tabella vista poco sopra è in 1NF; per chiarezza facciamo un esempio di una tabella che, **seppur munita di una chiave primaria**, **non può essere considerata in forma normale**:

Codice Fiscale	Nome	Dettagli
LBRRSS79Y12T344A	Alberto	età: 30; professione: Impiegato
GNNBNCT84A11L611B	Gianni	età: 24; professione: Studente

La tabella qui sopra NON è in 1NF in quanto, pur avendo una chiave primaria, presenta un **campo (dettagli) che non contiene dati in forma atomica.**

La seconda forma normale (2NF)

Un ulteriore miglioramento è possibile passando alla **seconda forma normale**. Perché una base dati possa essere in 2NF è necessario che:

- si trovi già in 1NF;
- tutti i campi non chiave **dipendano** dall'intera chiave primaria (e non solo da una parte di essa).

Per fare un esempio si supponga di avere a che fare con il database di una scuola con una **chiave primaria composta** dai campi "Codice Matricola" e "Codice Esame":

Codice Matricola	Codice Esame	Nome Matricola	Voto Esame
1234	**M01**	Rossi Alberto	6
1234	**L02**	Rossi Alberto	7
1235	**L02**	Verdi Mario	8

Il database qui sopra si trova in 1NF ma non in 2NF in quanto il **campo "Nome Matricola"** non **dipende** dall'intera chiave ma solo **da una parte di essa ("Codice Matricola").**

Per rendere il nostro database 2NF dovremo scomporlo in due tabelle:

Codice Matricola	Codice Esame	Voto Esame
1234	**M01**	6
1234	**L02**	7
1235	**L02**	8

e

Codice Matricola	Nome Matricola
1234	Rossi Alberto
1235	Verdi Mario

Nella prima tabella il campo "Voto" dipende correttamente dalla chiave primaria composta da "Codice Matricola" e "Codice Esame", nella seconda tabella il campo "Nome Matricola" dipende correttamente dalla sola chiave primaria presente ("Codice Matricola").

Ora il nostro database è normalizzato in seconda forma normale.

La terza forma normale

La **terza forma normale**, che definisce in modo ancora più puntuale ed efficiente la struttura delle tabelle di un database, si basa sul concetto di **dipendenza funzionale**.

> ***Dipendenza funzionale (FD):*** *un particolare vincolo di integrità che esprime legami funzionali tra gli attributi; una proprietà semantica (cioè che dipende dai fatti) per formalizzare la nozione di schema senza anomalie*.

Un database è in 3NF se:

- è già in 2NF (e quindi, necessariamente, anche 1NF);
- tutti gli attributi non chiave dipendono direttamente dalla chiave (quindi non ci sono attributi "non chiave" che dipendono da altri attributi "non chiave").

Per fare un esempio torniamo all'ipotetico database della palestra; supponiamo di avere una base dati che associ il codice fiscale dell'iscritto al corso frequentato ed all'insegnante di riferimento. Si supponga che il nostro DB abbia un'unica chiave primaria ("Codice Fiscale") e sia così strutturato:

Codice Fiscale	Codice Corso	Insegnante
LBRRSS79Y12T344A	BB01	Marco
GNNBNCT84A11L611B	BB01	Marco
LBRMNN79E64A112A	BB01	Marco
GLSTMT59U66P109B	AE02	Federica

Il nostro database non è certamente 3NF in quanto il **campo "insegnante"** non **dipende** dalla chiave primaria ma **dal campo "Codice Corso" (che non è chiave).**

Per normalizzare il nostro DB in 3NF dovremo scomporlo in due tabelle:

Codice Fiscale	Codice Corso
LBRRSS79Y12T344A	BB01
GNNBNCT84A11L611B	BB01
LBRMNN79E64A112A	BB01
GLSTMT59U66P109B	AE02

e

Codice Corso	Insegnante
BB01	Marco
AE02	Federica

Il nostro database è ora in terza forma normale.

Esercitiamoci: progetto mediateca

Interessanti video lezioni: **Corso progettazione data base ITA** **di fcamuso**

Prima forma normale

Seconda e terza forme normali

Denormalizzazione

La **denormalizzazione** è il processo che permette di ottimizzare le **performance in lettura** di un database aggiungendo ***ridondanza*** nei dati o raggruppandoli. Tale operazione viene effettuata spesso per risolvere l'inefficienza dei database relazionali, visto che, se normalizzati, possono soffrire di pesantezza nel caricamento dei dati.

La **denormalizzazione** intenzionale dei dati può essere motivata dal rilevamento di problemi di *prestazioni* oppure nel caso in cui si desideri semplificare la creazione di *report ad-hoc*.

I problemi di *prestazioni* derivano dalle query per le quali, in produzione, è richiesto un numero troppo elevato di **join di tabelle** con un accesso intensivo al disco (ad esempio in contesti come il **datawarehouse**).

Lo scopo della creazione di *report ad-hoc* è consentire anche agli utenti finali di eseguire query non strutturate, in quanto è possibile che utenti finali poco esperti non siano certi delle procedure necessarie per ottenere informazioni da più tabelle correlate.

Le tecniche di **denormalizzazione** comprendono la *duplicazione* dei dati, la possibilità di disporre di dati di riepilogo, la suddivisoine di tabelle in *partizioni orizzontali o verticali* e la creazione di visualizzazioni denormalizzate per semplificare la creazione di report, ovvero un'alternativa intelligente che lascia invariato il database normalizzato.

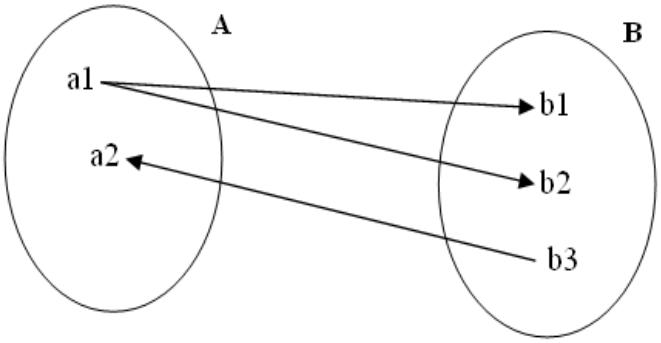
Inoltre, una **progettazione normalizzata** permette di memorizzare differenti, ma relazionati, "pezzi" di informazione in tabelle separate logicamente (**relazioni** appunto). Se queste relazioni vengono fisicamente memorizzate su file system separati, il completamento di una query che recuperi le informazioni tra le diverse relazioni (con una operazione di join) potrebbe essere lento.

In generale, se un numero significativo di query richiede il join di più di cinque o sei tabelle, è consigliabile considerare la denormalizzazione.

Si preferisce, come metodo di ottimizzazione, **memorizzare le informazioni in maniera ridondante** e si affida al DBMS la responsabilità di mantenere consistenti queste copie ridondanti, come fanno **Microsoft SQL Server** (con le *indexed views*) o **Oracle DB** (con le *materialized views*), ma i dati vengono memorizzati sullo stesso file system.

Vedremo in seguito altre tecniche per migliorare l'efficienza.

Cardinalità delle associazioni e tipi

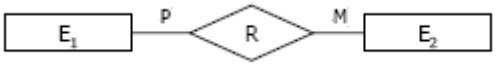


Un'associazione binaria deve essere ***classificata*** in uno dei seguenti quattro casi (tipi):

- **uno a uno** indicato con **1:1**
- **uno a molti** sia del tipo indicato con **1:N**
  sia del tipo indicato con **N:1**
- **molti a molti** indicato con **N:M** o **N:N**

Tali **tipi** di associazioni mettono in evidenza la **molteplicità** dell'associazione o *numero delle istanze di partenza* coinvolte nell'associazione (**uno** o **molti**) e della sua inversa *che indica quante istanze dell'entità di arrivo si associano all'istanza di partenza* (**a uno** oppure **a molti**)

Vincolo di cardinalità
Per ogni partecipazione di un'entità ad una associazione si può specificare il numero minimo e massimo di volte che *possono/devono* coinvolgere ogni un'istanza dell'entità (0 indica *opzionalità*). Nel modello E/R si evidenziano solo le cardinalità massime:



Ogni istanza di E_1 partecipa all'associazione con almeno *m* ed al più ad *M* istanze di E2
Ogni istanza di E2 partecipa all'associazione con almeno *p* ed al più ad *P* istanze di E1

Esistono **regole di trasformazione** da schema concettuale (*entità/associazioni*) a logico (*relazioni*).

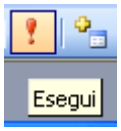
LE QUERY

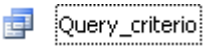
Una Query[6] è una visualizzazione dei dati contenuti su una o più tabelle, filtrati e/o aggregati secondo vari criteri. La traduzione letterale sarebbe "*interrogazione*", infatti la query è il risultato di una *domanda* posta al database. Esistono due tipi di query:

- **dettaglio** : vengono visualizzati tutti i campi di tutti i record
- **riepilogo**: consente di effettuare calcoli sui campi numerici (somma, media, minimo, massimo) oppure di scegliere raggruppamenti di dati

Access o Base semplificano la costruzione delle query (Ricerche) mediante la creazione guidata e rendono possibile creare anche Formulari e Rapporti.

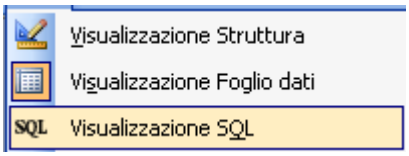
Per **eseguire Query**

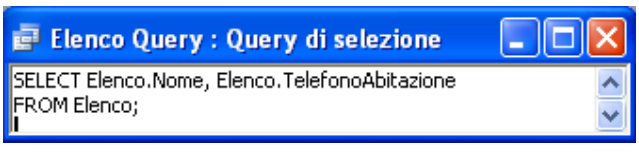
In *modalità struttura*, uso dell'icona 

Oppure, dopo averla salvata con nome, doppio click sull'icona 

Query in visualizzazione struttura e SQL

1. Con il tasto destro del mouse sulla query





scegliere dal menu "Visualizzazione SQL".

[6] Il linguaggio SQL permette due tipi di query: interrogazione **statica** (tipo *compile and store*) e **dinamica** (tipo *compile and go*) costruita in modo interattivo (visuale) come stringa ed eseguita "on –fly".

2. Modifichiamo la selezione da

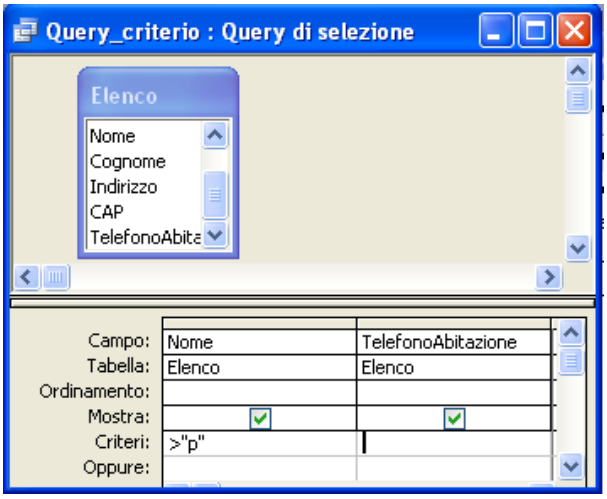
   SELECT Elenco.Nome, Elenco,TelefonoAbitazione

   FROM Elenco;

   in

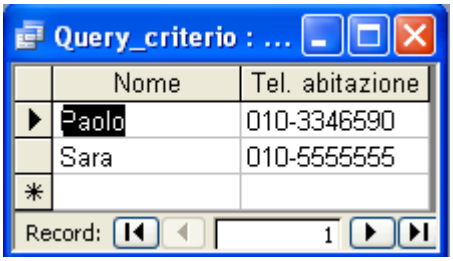
   SELECT Elenco.Nome, Elenco.TelefonoAbitazione

   FROM Elenco **WHERE** Nome > "p";



nb: risultato analogo se in "Visualizzazione Struttura" si imposta il criterio come in figura:

3. Rieseguiamo la query chiudendo la finestra salvando le modifiche e selezionando dal menù principale "Query" –"Esegui". Notiamo che il risultato è differente perché in questo caso non visualizza i nomi prima della lettera "p".

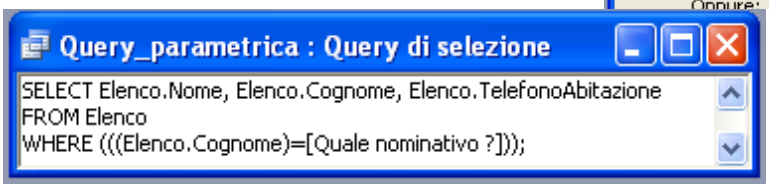


In questo modo è possibile estrarre un sottoinsieme di dati secondo un preciso **criterio** deciso da noi: si può fare su un campo o su un insieme di campi legando le singole condizione con operatori logici "AND","OR","NOT".

SQL Statement	Syntax
AND / OR	**SELECT** column_name(s) **FROM** table_name **WHERE** condition **AND|OR** condition

Query parametrica

Si imposta come criterio, tra parentesi quadre, la domanda che permette all'utente di selezionare il valore nella condizione di ricerca



SQL – Quick Reference: **operazioni di amministrazione e *transazioni***; query complesse

Usando **Access *tipo desktop***: primo esercizio guidato, passi nel design, query complesse

Migliorare l'efficienza: indici[7] e filtri

Indici

Una volta definita la struttura della tabella, ogni volta che si aggiunge una riga, il motore di Db *accoda* sul disco le informazioni all'archivio aperto. Nella consultazione, invece, l'ordine alfabetico è molto comodo.

Quindi potrebbe essere utile creare un *indice* sul campo (**chiave di ricerca**) in modo da poter fornire le informazioni nell'ordine più logico.

Un *indice* è, in generale, un *ordinamento* creato su uno o più *campi* in modo che il reperimento delle informazioni contenute nell'indice stesso sia molto rapido.

Un indice è un **puntatore** numerico creato allo scopo di migliorare e potenziare i criteri di ricerca, senza basarsi sulla sola chiave primaria.

Tale **chiave di ricerca** è utile se di uso frequente permettendo un accesso non sequenziale alle informazioni ma mediante un "puntatore" con incremento della velocità di reperimento.

Si ricordi che **troppi indici rallentano** il sistema. Ogni operazione di modifica dei dati, comporta infatti, se il campo è indicizzato, anche l'aggiornamento degli indici stessi. Inoltre **gli indici occupano memoria e spazio su disco**.

L'**efficacia** di un indice è **inversamente proporzionale alla lunghezza del campo**. Più il campo è piccolo, più l'indice è efficiente.

Filtri nell'ottimizzare le interrogazioni

Per ottimizzare le interrogazioni in termini di efficienza (velocità nel recupero delle informazioni) si seguono le tre seguenti strategie:

- anticipare la selezione delle informazioni, riducendo il numero di tuple che costituiscono un risultato intermedio
- eliminare il maggior numero di colonne inutili (quelle che non possono contribuire alla formazione del risultato) con opportuno *filtraggio*
- sostituire se possibile ad operazioni di giunzione (join[8]) delle *semi-join* "tagliando" gli attributi della relazione le cui colonne non fanno parte del risultato intermedi. Infatti, l'operazione di *giunzione* che permette di selezionare dati prelevandoli fra più tabelle, evidentemente correlate tra loro (sulla base di un qualche criterio tra attributi appartenenti alle due tabelle) è quella più critica dal punto di vista dell'efficienza (il tempo di esecuzione potrebbe risultare inaccettabile in molte applicazioni).

Query di estrazione da tabelle relazionate

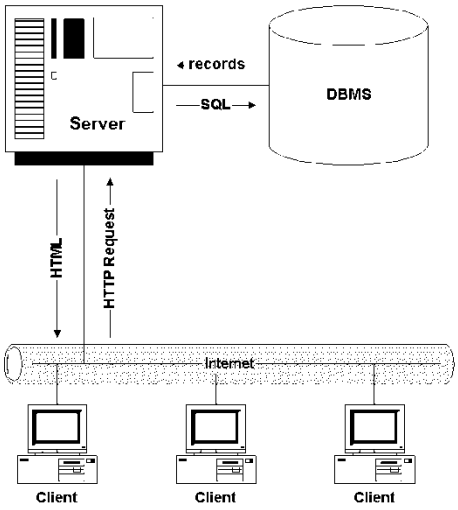
Eseguiamo una query di estrazione dati da due tabelle relazionate tra loro:

SELECT Elenco.Nome, Elenco.Cognome, Amici.email
FROM Elenco INNER JOIN Amici ON Elenco.IDElenco=Amici.CodElenco;

equivalente a:

SELECT Elenco.Nome, Elenco.Cognome, Amici.email
FROM Elenco, Amici WHERE Elenco.IDElenco=Amici.CodElenco;

[7] Ottimizzare le prestazioni delle query (SQL ServerCompact)
[8] References con equivalenza filtrando con uso di clausola WHERE ed esempi.



In architettura **C/S** la possibilità di ***consultare***, in base alle proprie esigenze, i contenuti di *data-base multimediali distribuiti* su Internet.

DBMS Access - web-hosting **Somee**

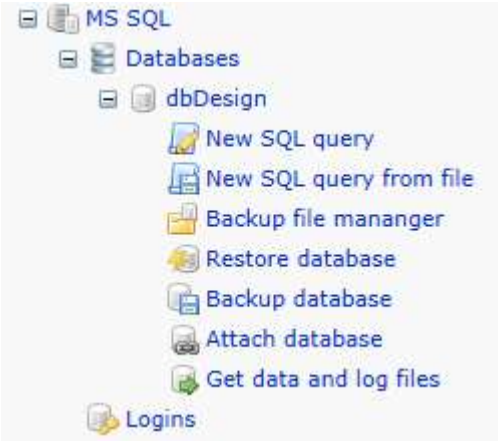
- MS Access 2003
- MS Access 2007

DBMS **MS SQL Server**: web-hosting **Somee**

Somee, made in USA, offre un hosting Microsoft (free) dotato sia di tecnologia ASP che della più recente Asp.NET. Ha un'ottima velocità di connessione e regala ben 150Mb di spazio gratuito e una banda di utilizzo mensile di 3Gb. Inserisce un **banner** come controprestazione nella parte superiore del sito. Consente **unico DB MS SQL**, potendo archiviare **più tabelle**:

- Free ASP.Net web hosting
- 150MB storage, 5GB transfer
- ASP ASP.Net 1.1/2.0/3.5/4.0/4.5
- 15MB MSSQL 2008R2/2012/2014
- Free third level domain
- FTP access

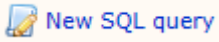
Account utilizzabile id: infcol5ai psw Colinf_16



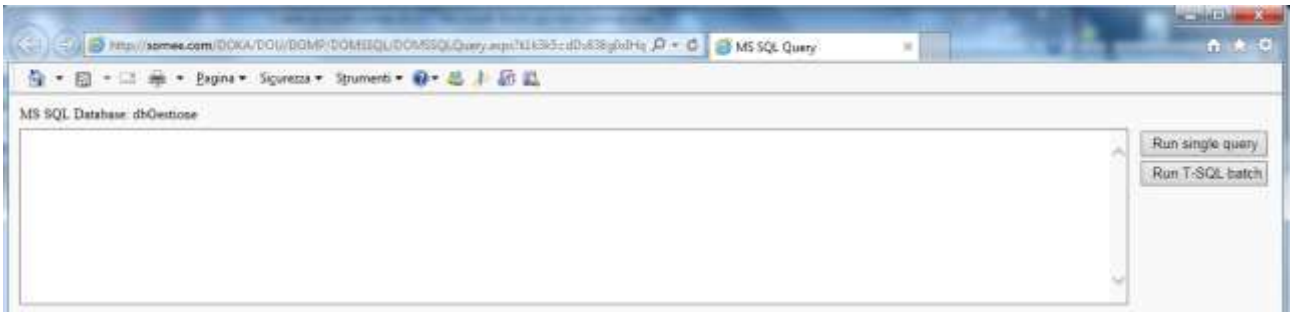
FTP address to backups: **dbDesign.backup.somee.com**

FTP username: **infcol5ai**

FTP credentials (Username and password) are the same as your Username and password.

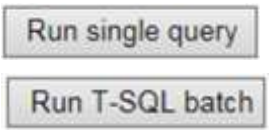
Selezionando opzione 

si possono **inserire le *query*** desiderate nella "SQL[9] Box" (area di testo)



[9]Si utilizza la sintassi del linguaggio SQL (Structured Query Language) vedi http://www.w3schools.com/sql/default.asp

Per eseguire ogni query sono previsti due pulsanti:



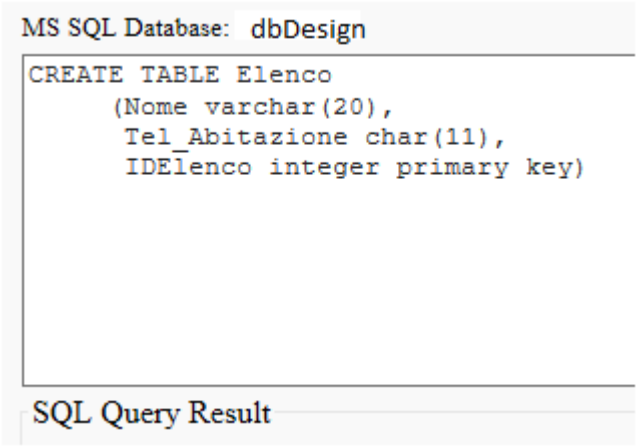
- esegue senza visualizzare informazione sul successo e **visualizzando tabella dinamica** creata al volo (da usare per estrazione dati)
- esegue e **visualizza informazione sul successo** senza visualizzare tabella dinamica creata al volo

Supponiamo di voler creare le **tabelle** di una **rubrica telefonica**.

- Creiamo la **struttura** delle tabelle inserendo i nomi dei campi che necessitano ed il loro tipo: usiamo la sintassi "**Create table**" editando il comando voluto nel SQL Box; premiamo uno dei pulsanti per eseguire tale comando (nel caso di errori sintattici compariranno avvisi)

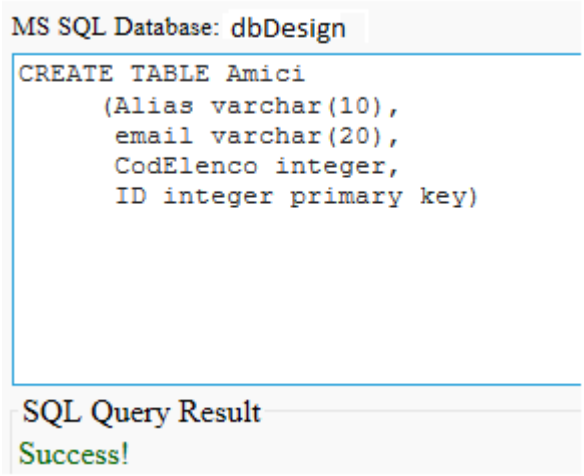
CREATE TABLE Elenco (Nome varchar(20), Tel_Abitazione char(11), IDElenco integer primary key)	**CREATE TABLE** Amici (Alias varchar(10), email varchar(20), CodElenco integer, ID integer primary key)

Dopo qualche minuto



Usando "Run single query"

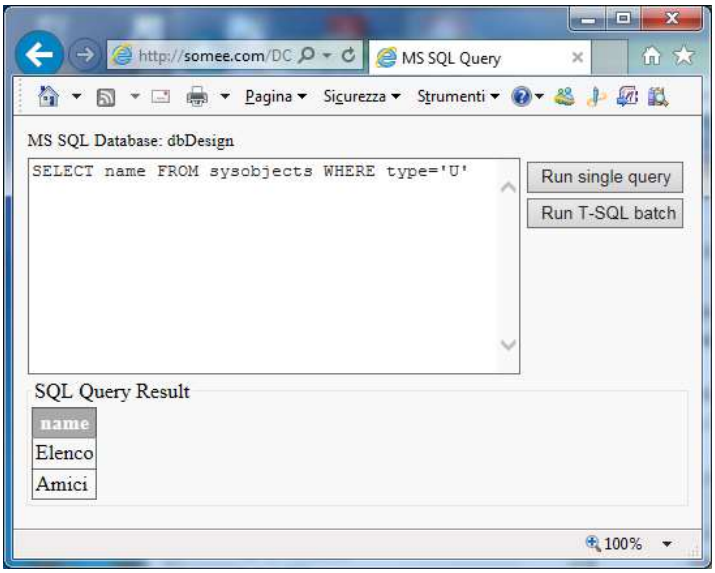
Cancellato e sostituito il comando:



Usando "Run T-SQL batch"

Per verificare che le due tabelle sono state create, si può **vedere la lista delle tabelle**

SELECT name FROM sysobjects WHERE type='U'



- A questo punto dobbiamo **popolare** le tabelle inserendo i nomi dei campi che necessitano ed il loro valore: editiamo direttamente il comando "**Insert into**" SQL

INSERT INTO Elenco (Nome, Tel_Abitazione,IDElenco) **VALUES** ('Paola Notini', '010-3346590', 1) …..	**INSERT INTO** Amici (Alias,email,CodElenco,ID) **VALUES** ('Pupo','pupo@yahoo.it',1,1) …..

MS SQL Database: dbDesign

```
INSERT INTO Elenco
(Nome, Tel_Abitazione,IDElenco)
VALUES ('Paola Notini', '010-3346590', 1)
```

SQL Query Result

Success!

MS SQL Database: dbDesign

```
INSERT INTO Amici
(Alias,email,CodElenco,ID)
VALUES ('Pupo','pupo@yahoo.it',1,1)
```

SQL Query Result

Success!

- Se vogliamo aggiornare con la modifica di un campo possiamo usare "**Update**"
- Se vogliamo aggiornare con cancellazione possiamo usare "**Delete**"

- Per compiere delle ricerche (**query di estrazione dati**) possiamo usare "**Select**" e pressione del pulsante

Run single query

Select * From Amici

MS SQL Database: dbDesign

```
Select * From Amici
```

SQL Query Result

Alias	email	CodElenco	ID
Pupo	pupo@yahoo.it	1	1

Select * From Elenco

MS SQL Database: dbDesign

```
Select * From Elenco
```

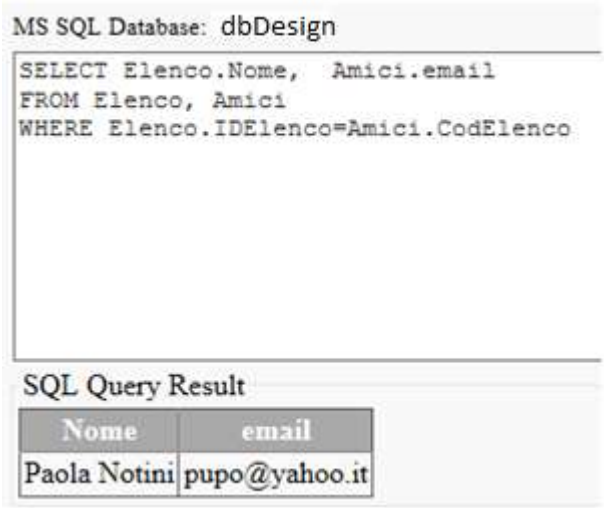
SQL Query Result

Nome	Tel_Abitazione	IDElenco
Paola Notini	010-3346590	1

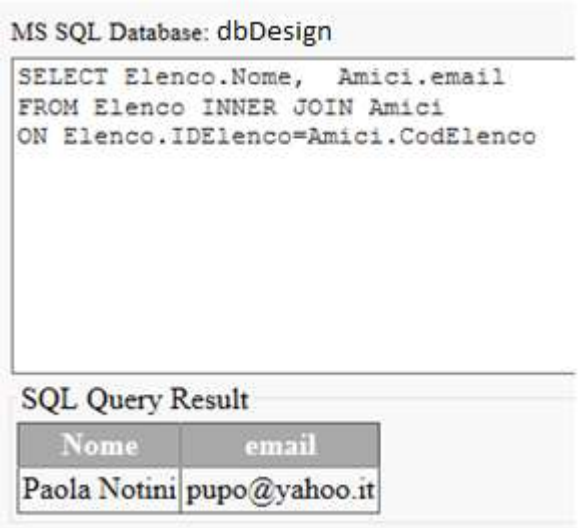
Equivalenza:

SELECT Elenco.Nome, Amici.email **FROM Elenco, Amici** **WHERE** Elenco.IDElenco=Amici.CodElenco	**SELECT** Elenco.Nome, Amici.email **FROM** Elenco **INNER JOIN** Amici **ON** Elenco.IDElenco=Amici.CodElenco

SELECT Elenco.Nome, Amici.email
FROM **Elenco, Amici**
WHERE Elenco.IDElenco=Amici.CodElenco



SELECT Elenco.Nome, Amici.email
FROM Elenco **INNER JOIN** Amici
ON Elenco.IDElenco=Amici.CodElenco



Nb: *Per maggior dettaglio, volendo creare un account e* ***gestire*** *DB – web hosting free somee*

Date le prime nozioni sui database possiamo ora utilizzare i DB per **ricerche** effettuate nel **web** e più in generale per **gestire DB remoti**.

Questa attività è sempre più utilizzata nel mondo Internet perché la maggior parte delle informazioni visualizzate nelle pagine WEB vengono raccolte nei database. Ecco i passi da effettuare per il *collegamento ad un database* con tecnologia ASP-ADO. Ricordiamo che il collegamento e le elaborazioni sui database non possono essere effettuate con linguaggio di marcatura HTML. Per gestire i database con **tecnologia ASP** si utilizzano **strumenti ADO** (ActiveX Data Objects): un'architettura che fornisce oggetti di **alto livello** per **l'accesso universale ai dati** e in particolare contiene **oggetti per la comunicazione con i database** infatti ADO[10] consente di scrivere un'applicazione per la gestione e l'accesso ai dati contenuti in un server di database tramite un provider[11] OLE DB (interfaccia a basso livello che fornisce un modello di accesso ai dati universale che ne consente la gestione indipendentemente dal formato e dal metodo di memorizzazione; permette l'accesso non solo ai database relazionali ma a qualsiasi fonte dati : database **locali** o **remoti**, non relazionali, sistemi di file, posta elettronica, testo, grafica e oggetti multimediali, aziendali, personalizzati ...).

[10] Ideato come interfaccia dati client/server. Una limitazione di ADO consiste nell'impossibilità di creare fonti di dati ODBC pur se il provider predefinito è Microsoft OLE DB per ODBC (componente Microsoft Windows ed in particolare di Windows Open Services Architecture, che consente l'accesso a tutti i tipi di database relazionali).
[11] Il termine provider indica in generale un elemento fornitore dal quale si ottengono servizi o dati.

Poiché tuttavia ciascun provider dispone di caratteristiche uniche, le modalità di interazione tra l'applicazione e ADO variano leggermente a seconda del provider stesso. Per una descrizione più completa dell'architettura ADO si consulti: Il Modello ad Oggetti di ADO (documento compresso scaricabile Download il file Zip)

Progettare pagine per consentire le tipiche operazioni su un DB (hosting somee**):**

Un menù (*cercando di ovviare al fastidioso inserimento di banner)*

homepage.htm

```
1  <html>
2  <head><title>Homepage</title>
3      <style>
4      body{background-color: yellow; color:black}
5      table {background-color: red}
6      th {background-color: lightblue ; color:black}
7      a:link, a:visited { text-decoration: none}
8      img {float: left; padding-left: 5%}
9      div {width: 70%; float:right;}
10     .spazio {padding-top: 50%}
11 </style>
12 </head>
13 <body>
14 <center><h1>Gestione  database</h1></center>
15 <img src="DB.png" alt="Database" />
16 <div>
17 <h2><a href="insdati.htm">Inserimento</a></h2>
18 <h2><a href="tabella_stile.html">Visualizzazione</a></h2>
19 <h2><a href="find.htm">Ricerca</a></h2>
20 </div>
21 <div class= "spazio">  </div> <!-- per ottimizzare posizione banner -->
22 </body>
23 </html>
```

e **pagine di interfaccia**:

- per estrarre il contenuto di una data tabella
- inserire nuove tuple
- *etc*….

semplici *form* per lanciare l'esecuzione della opportune **pagina ASP** nell'interazione con DB remoto

Codice pagina html di ***interfaccia*:**

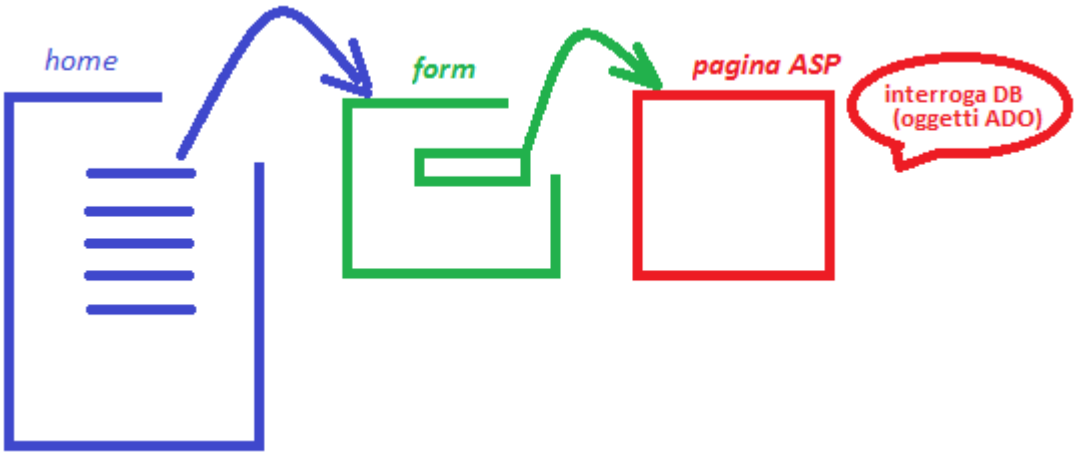
tabella_stile.html

```
1  <HTML>
2  <HEAD><TITLE>Visualizzazione tabella</TITLE></HEAD>
3  <style>body{background-color: yellow; color:black}</style>
4  <BODY>
5  <H2>Visualizzazione tabella</H2>
6  <FORM METHOD="GET" ACTION ="http://nomeID.somee.com/tabella_stile.asp">
7
8  <P><INPUT TYPE = "SUBMIT" VALUE = "Invio"></P>
9
10 </FORM>
11 </BODY>
12 </HTML>
```

pagina ASP: tabella_stile.asp

Codice pagina ASP: tabella_stile.asp

```
<%
    Option Explicit
    Dim oConn, Rs, sSQL

    rem creazione lato Server di oggetti ADO essenziali
    Set oConn = Server.CreateObject("ADODB.Connection")
    Set Rs = Server.CreateObject("ADODB.RecordSet")

    rem impostazione di Stringa di connessione per DBMS SQL Server – DSN less
    oConn.open ("Driver={SQL Server}; Server=nomeID.mssql.somee.com; Database= nomeID;
                Uid= nomeID _SQLLogin_2; Pwd=fornita_da_somee;")

    rem salvataggio come stringa del comando SQL
    sSQL ="SELECT * FROM nomeTabella"

    rem esecuzione della query a connessione apeta
    Rs.Open sSQL,oConn

   rem uso di oggetto ASP e del metodo per scrivere sulla finestra del browser
   Response.write("<table style='background-color: yellow'> <tr><th>Nominativo</th> <th>e-mail</th></tr>")

   While not Rs.eof
        Response.write("<tr><td>" & RS ("Alias") &  "</td>" & "<td>" & RS ("email") & "</td>" & "</tr>")
        RS.movenext
    Wend

    Response.Write("</table>")

    rem chiusura della connessione e deallocazione degli oggetto ADO
    oConn.close
    Set Rs=Nothing
    Set oConn = Nothing
%>
```



Glossario

DB: un insieme di dati logicamente connessi, **strutturati** in accordo ad un preciso **schema** (*modello dei dati concettuale* - E/R - o *logico* - rappresentazione tabellare - con espliciti vincoli), memorizzati su supporto permanente in cui è possibile effettuare operazioni (ricerche e modifiche) caratterizzate dalle proprietà **ACID** (**Atomicity**, **Consistency**, **Isolation** e **Durability**).

Sottolineando il diverso approccio rispetto al gestire file indipendenti (**archivi**), i dati sono usabili per diverse applicazioni, anche in *concorrenza tra loro* (per questo le operazioni sui dati devono apparire **I**solate), utenti diversi possono interessarsi ad un sottoinsieme dei dati presenti (vista o "*view*") ed esiste un'*integrazione* che rende minima la ridondanza (evita incontrollata duplicazione). Esistono, infine, meccanismi di *sicurezza* (**C**onsistenza cioè integrità dei dati) e *ripristino* (**D**urevolezza cioè persistenza)

Le operazioni fondamentali che si possono eseguire su un database (**ricerca, inserimento, aggiornamento** e **cancellazione**) sono ***procedure indivisibili*** (**A**tomicità).

Transazione: sequenza di azioni di una procedura che si propone di modificare od osservare la base di dati con la proprietà di **atomicità**: o vengono eseguite del tutto come un'unica azione elementare e indivisibile, o non vengono eseguite affatto ed eventuali effetti parziali vengono annullati.

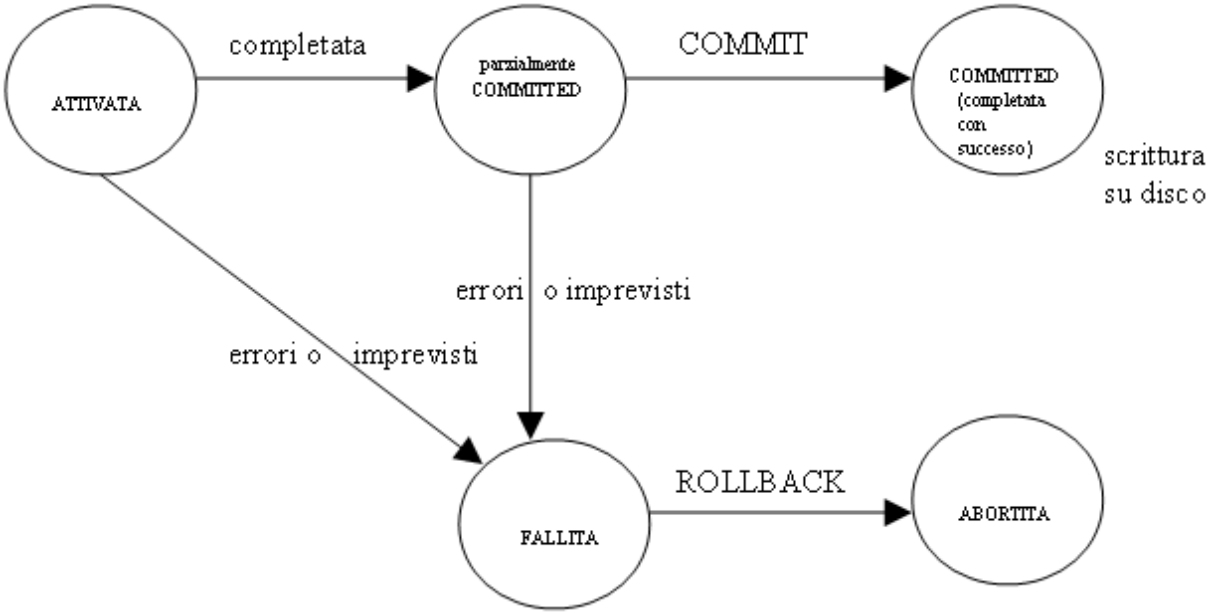
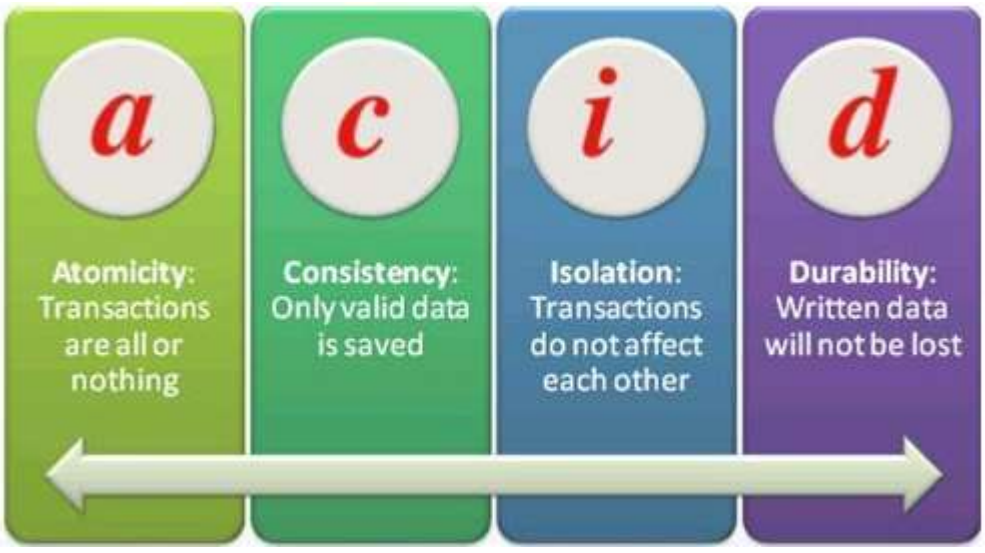


Diagramma degli stati di avanzamento di una transazione

ACID: deriva dall'acronimo inglese **Atomicity**, **Consistency**, **Isolation** e **Durability** (**A**tomicità, **C**oerenza o consistenza, **I**solamento e **D**urabilità) cioè le proprietà dei meccanismi che implementano transazioni che operino in modo corretto sui dati.

- **atomicità**: la transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere o totale o nulla, non sono ammesse esecuzioni parziali;
- **coerenza** o consistenza: quando inizia una transazione il database si trova in uno stato coerente e quando la transazione termina il database deve essere in uno stato coerente, ovvero non deve violare eventuali vincoli di integrità, quindi non devono verificarsi contraddizioni (*inconsistency*) tra i dati archiviati nel DB;
- **isolamento**: ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione;
- **durabilità**: detta anche **persistenza**, si riferisce al fatto che una volta che una transazione abbia richiesto un *commit work*, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log dove sono annotate tutte le operazioni sul DB.



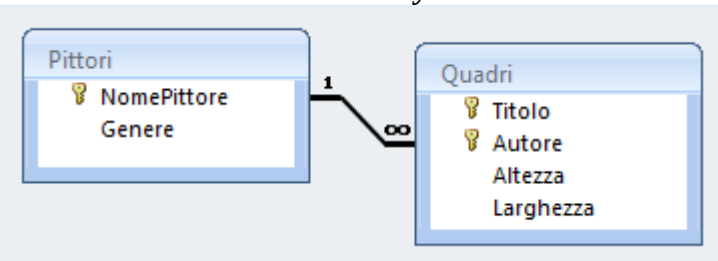
DBMS Access versione 2003 Esempio Access 2003

DSN-less connection

Stringa di connessione

"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & Server.mappath ("\directory\nomeDB.mdb")

DB Mostra.mdb in sottodirectory Access



Esposizione nella mostra:

Pittori

NomePittore	Genere	Aggiungi nuovo campo
Bianchi	ritrattista	

Titolo	Altezza	Larghezza	Aggiungi nuov
Andrea	0,6	0,4	
	0	0	

NomePittore	Genere
Hu	paeseggista
Neri	paesaggista
Petra	naif
Poli	ritrattista
Poliani	ritrattista
Politi	ritrattista
Rossi	paesaggista

Titolo	Altezza	Larghezza	Aggiungi nuov
Tramonto	1,2	2,4	
	0	0	

NomePittore	Genere
Verdi	impressionista

Titolo	Altezza	Larghezza	Aggiungi nuov
Tramonto	1,2	2,4	
Egitto	0,6	0,6	
	0	0	

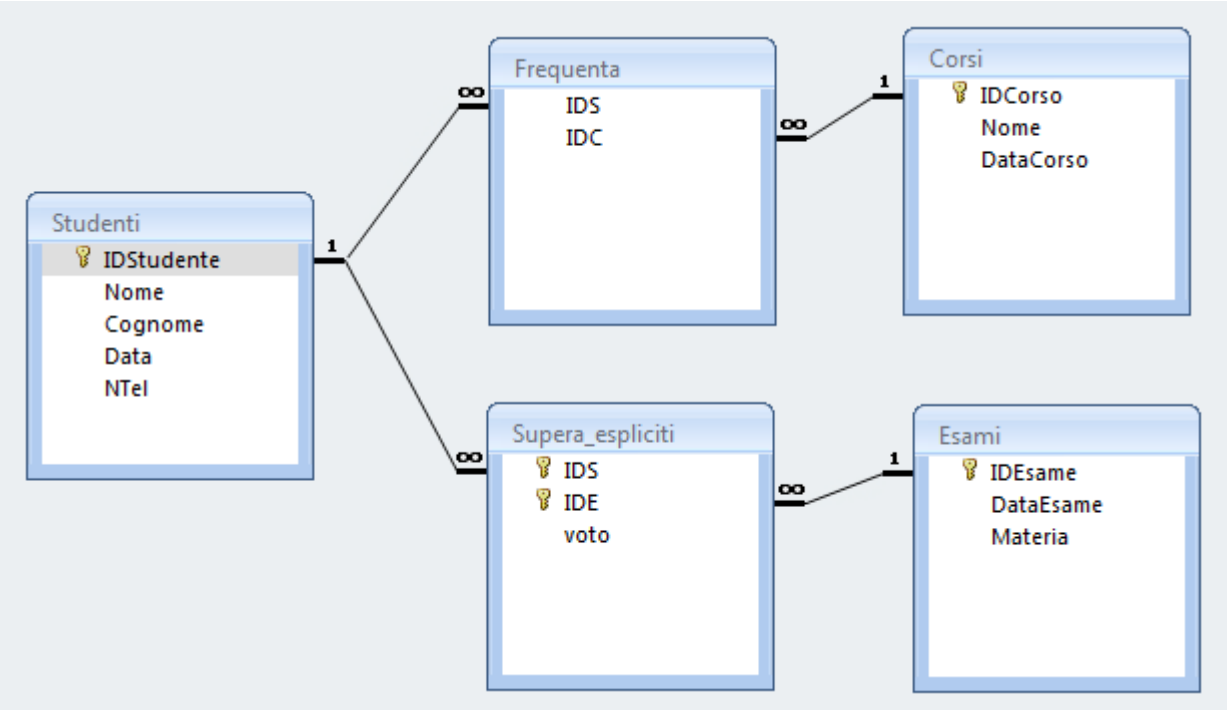
Correttamente visualizzati (nella ***gestione a remoto***):

DBMS Access versione 2007 e superiori Esempio Access 2007

Stringa di connessione per Access 2007 e superiori (.accdb)

"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" & Server.mappath ("\directory\nomeDB.accdb")

DB Ingegneria.accdb in sottodirectory Access



*Corretti inserimenti con gestione della data in "**formato italiano**"*

Cognome	Telefono	Data
Mancuso	3408355667	13/04/1997
Galletti	3408644556	25/02/1997
Oliva	3405433218	30/03/1997
Azzolini	3407812334	22/08/1997

Corso	Data
Informatica	01/01/2015
Geometria	05/05/2013
Matematica	08/08/2014
Analisi I	22/11/2015
Analisi II	28/11/2015