Le basi di dati e il linguaggio SQL

Autori:
dott.BOLLA Luciano
dott.sa BROCCHIERO Gabriella
dott.PALLADINO Marco

INDICE

Modulo: Le basi di dati e il linguaggio SQL

Il modulo è costituito da 3 unità didattiche:

- Modello dei dati e progetto software
- Le basi di dati
- Il linguaggio SQL

- **Premessa**

Le seguenti unità didattiche verranno proposte agli ad allievi con prerequisiti che possono essere in sintesi riassunti nel progamma ECDL (escluso MS Access)

Le attività verranno svolte sia in classe che in laboratorio. In alcuni momenti la lezione sarà frontale con la partecipazione attiva della classe che verrà coinvolta nelle esercitazioni pratiche e nella progettazione dei DB.

- **Prerequisiti**
  - Nozioni sui file, record, campo, chiave
  - Conoscenze di base sull'organizzazione logica degli archivi
  - Metodologia top-down per l'analisi di un problema

- **Obiettivi specifici**
  - Saper utilizzare le tecniche di definizione del modello di dati
  - Documentare l'analisi di un problema in modo efficace
  - Comprendere i limiti dell'organizzazione non integrata degli archivi
  - Conoscere i concetti e le tecniche per la progettazione delle basi di dati
  - Applicare correttamente i principi del modello relazionale
  - Rappresentare le operazioni relazionali
  - Codificare e validare le operazioni in linguaggio SQL
  - Utilizzare in laboratorio l'applcativo MS-Access e il VBA

AL TERMINE DELLO SVOLGIMENTO DEL MODULO L'ALLIEVO DOVRÀ POSSEDERE LE SEGUENTI CONOSCENZE:

- Modellizzazione dei dati: entità, attributi, associazioni, chiave
- Schema E\R
- Regole di derivazione del modello logico
- Modello relazionale
- Operazioni relazionali
- Normalizzazione delle relazioni
- Integrità referenziale
- Comandi per le interrogazioni
- Funzioni di aggregazione
- Ordinamenti e raggruppamenti
- Interrogazioni nidificate

AL TERMINE DELLO SVOLGIMENTO DEL MODULO L'ALLIEVO DOVRÀ AVER ACQUISITO LE SEGUENTI ABILITÀ:

- Determinare entità, attributi, associazioni, chiavi relativi alla realtà da rappresentare
- Disegnare il modello E/R e esplicitare regole di lettura
- Applicare le regole di derivazione del modello logico
- Definire relazioni normalizzate
- Utilizzare gli operatori di selezione, proiezione e congiunzione
- Applicare le regole per l'integrità
- Utilizzare i comandi e le funzioni del linguaggio SQL
- Codificare le operazioni relazionali di selezione, proiezione e congiunzione
- Costruire interrogazioni complesse attraverso strutture nidificate

- **Mezzi**: Libro di testo, appunti vari scaricati da internet. Videoproiettore per esempi in diretta e per la costruzione del DB stesso.
- **Spazi**: Aula e Laboratorio
  L'attività svolta in laboratorio si avvale del "cooperative learning"
  I ragazzi durante il triennio dovrebbero migliorare il metodo di studio e dovrebbero dimostrare di saper lavorare in gruppo utilizzando questa tecnica.
  L'obiettivo principale del "cooperative learning" è l'apprendimento cognitivo.
  Il metodo permette un'interdipendenza positiva: il successo del singolo dipende dal successo di tutti gli altri; inoltre responsabilizza gli alunni e permette di sviluppare un'interazione costruttiva faccia a faccia e un'abilità sociale da spendere nel mondo del lavoro.
  Lavorando insieme si mettono a disposizione di tutti le competenze individuali: la conoscenza è del singolo ma si può costruire attraverso il gruppo. Ogni elemento del gruppo deve sentire la responsabilità di fare ciò che si è stabilito (assegnazione di un ruolo e di un compito preciso). Il gruppo si propone un obiettivo e ciascuno deve fare la propria parte; è il gruppo stesso, attraverso l'esperienza che scoprirà le strategie più efficaci per raggiungerlo.

- **Tempi**: 40 ore (tempi non rigidi suddivisi in ore di laboratorio e ore di teoria). Eventuale recupero

- **Verifica e valutazione**

La fase di valutazione dell'apprendimento è strettamente legata alla fase di verifica.
Con la valutazione sarà possibile conoscere il grado di apprendimento degli allievi e le capacità organizzative (analisi e sviluppo del progetto)

La valutazione sommativa verrà fatta al termine di ogni unità didattica e permetterà di valutare complessivamente l'iter formativo sia in riferimento alle competenze disciplinari acquisite che alle capacità di analisi ed elaborazione conquistate.

Il lavoro di laboratorio verrà condotto a gruppi formati da circa 4-5 allievi. Sarà richiesta una relazione scritta che documenti in modo efficace il progetto sviluppato e che descriva le strategie risolutive adottate.
Il progetto verrà realizzato a livello fisico con l'applicativo MS-Access. [I1][I2][I3][I4][ES1][ES2][ES3] .
La valutazione sarà fatta sia a livello collettivo, cioè sul progetto, sia a livello individuale attraverso un colloquio sempre incentrato sul lavoro svolto e sulle conoscenze individuali.

MODELLO DEI DATI E PROGETTO SOFTWARE [I1][ES1][I2][I3]

Con il termine base di dati si indicano in informatica gli archivi di dati, organizzati in modo integrato attraverso tecniche di modellizzazione dei dati e gestite sulle memorie del computer attraverso appositi software, con l'obiettivo di raggiungere una grande efficienza nel trattamento e nel ritrovamento dei dati , superando anche i limiti presenti nelle organizzazioni tradizionali degli archivi. Scopo delle basi di dati è:

- organizzare i dati in modo integrato per superare la distanza tra dato e informazione
- i dati registrati nelle memorie vengono convertiti in informazioni utilizzabili
- gli stessi dati sono disponibili per utenti diversi con applicazioni diverse

La **base di dati** diventa un ***modello*** del mondo reale. Il **contenuto** della base di dati rappresenta lo **stato** della realtà modellata.
I cambiamenti nella base di dati rappresentano gli *eventi* che accadono nella realtà modellata.
Il termine **Database** indica la basi di dati.
La sigla **DBMS** (*Data Base Management System*) indica il software utilizzato per la gestione di un database.

OBIETTIVI DELLA BASE DI DATI

- **consistenza** i dati in essa contenuti devono essere significativi ed essere effettivamente utilizzabili nelle applicazioni dell'azienda.
- **sicurezza** impedire che il data base venga danneggiato da interventi accidentali o non autorizzati;
- **integrità** garantire che le operazioni effettuate sul data base da utenti autorizzati non provochino una perdita di consistenza ai dati.

DATABASE DISTRIBUITI

Il termine *base di dati* fornisce un'idea di integrazione degli archivi di dati. Questo significa anche *dati accentrati*? non necessariamente...

Essi possono essere *distribuiti* sulle memorie di massa di computer diversi, facenti parte di una rete aziendale, i cui nodi possono essere anche fisicamente lontani.

Vantaggi:

- gestire archivi di dimensioni limitate laddove vengono creati,
- facilitare il lavoro di manutenzione,
- maggior controllo sulla sicurezza,
- garantire comunque la disponibilità dei dati aggiornati a tutti gli utenti del sistema informativo aziendale, qualunque sia la loro posizione geografica o il computer da essi usato per l'attività di elaborazione.

Si ha quindi un'integrazione di tipo *logico* degli archivi, anche se i dati *fisicamente* possono risiedere su sistemi distanti.

ORGANIZZAZIONE DEI DATI: L'APPROCCIO TRADIZIONALE (NON INTEGRATO)

Esempio (da P. Chen, 1989)
In un'azienda si utilizzano in uffici diversi due diversi archivi A e B

1. Informazioni sui dipendenti che lavorano nei reparti di un'azienda

Matricola	Nome	Età	Rep#	Budget

2. Informazioni sui dipendenti per i diversi progetti

Dip#	NomeDip	Prog#	Nome	Tempo

Ognuno dei due modelli soddisfa le esigenze poste. Il problema nasce quando vogliamo rispondere alla domanda:

"QUALI SONO I REPARTI CHE HANNO DIPENDENTI CHE LAVORANO AL PROGETTO X?"

Difetti della gestione non integrata

- Sinonimi: gli stessi dati hanno nomi differenti (*Matr* in A e *Dip#* in B)
- Lo stesso nome per dati differenti: *Nome* in A per dipendente e *Nome* in B per progetto
- Ridondanza dei dati: *Budget* del reparto ripetuto per ogni dipendente; *Proj#* e *Nome* ripetuti per ogni dipendente
- Rischio di anomalie negli aggiornamenti: una modifica effettuata sui valori dei dati in un archivio, ma non nell'altro, provoca inconsistenza dei dati.

Difetti dell'organizzazione non integrata degli archivi

- **ridondanza** dei dati, cioè gli stessi dati compaiono in maniera duplicata (inefficienza);
- ⬇
- la ridondanza può portare all'**incongruenza**, nel caso in cui un dato venga aggiornato in un archivio e non in un altro, oppure siano presenti valori diversi per lo stesso dato;
- ⬇
- l'incongruenza porta all'**inconsistenza** dei dati, cioè i dati aziendali non sono più affidabili, perché non si sa in modo certo quale dei diversi valori sia quello corretto.

Tutto ciò deriva dal fatto che i dati sono organizzati in archivi diversi, in modo non integrato tra loro.

Caratteristiche vantaggiose delle basi di dati:

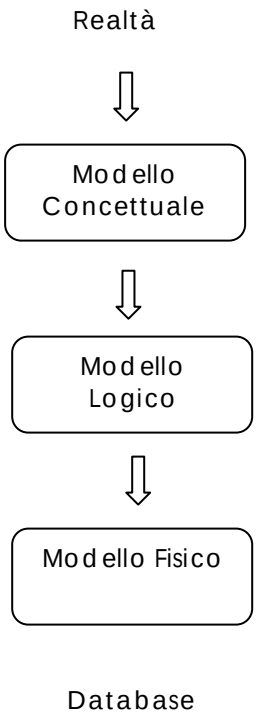
- indipendenza delle applicazioni dalla struttura fisica dei dati e dalla struttura logica dei dati
- utilizzo da parte di più utenti con applicazioni diverse
- eliminazione della ridondanzaed eliminazione della inconsistenza
- facilità di accesso
- integrità dei dati
- sicurezza dei dati

OPERAZIONI SUI DATI ORGANIZZATI

- la **manipolazione** dei dati, cioè la possibilità di inserire, modificare e cancellare i dati
- l'**interrogazione**, cioè la possibilità di ritrovare i dati, richiesti da un'applicazione, in modo semplice e veloce.

LA COSTRUZIONE DELLA BASE DI DATI [I1][ES1]

Progettazione	***modello concettuale***
Realizzazione	***modello logico***
Implementazione	***modello fisico***



- il **livello concettuale** rappresenta la realtà dei dati e le relazioni tra essi attraverso uno schema.
- il **livello logico** rappresenta il modo attraverso il quale i dati sono organizzati negli archivi elettronici: descrive quindi la composizione ed il formato dei dati nel loro aspetto di struttura logica di dati. Il livello logico viene derivato dal livello concettuale applicando alcune regole molto semplici.
- il **livello fisico** rappresenta l'effettiva installazione degli archivi elettronici: esso indica l'ubicazione dei dati nelle memorie di massa (dischi). Il livello fisico è quindi l'implementazione del livello logico sui supporti per la registrazione fisica dei dati: partizioni, puntatori, blocchi fisici, cluster, indici.

IL LIVELLO CONCETTUALE: IL MODELLO E/R [I1][I2][I3][E1][E2][E3]

Il modello entità\associazioni (in inglese **E/R = Entity/Relationship**), introdotto nel 1976 da **Peter P. Chen**, è uno strumento per analizzare le caratteristiche di una realtà in modo indipendente dagli eventi che in essa accadono, cioè per costruire un modello concettuale dei dati indipendente dalle applicazioni.

Gli elementi di un modello E/R sono:

- ENTITÀ
- ASSOCIAZIONI
- ATTRIBUTI

ENTITÀ E ISTANZE

L'**entità** è un oggetto (concreto o astratto) che ha un significato anche quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare.

Esempi di entità sono: una persona, un modello di automobile, un movimento contabile, una prova sostenuta da uno studente.

Per esempio gli studenti sono classificabili nel tipo entità *Studente*, i diversi modelli di automobile sono classificabili nel tipo entità *Automobile*.

Ciascun studente rappresenta un'istanza dell'entità *Studente*.

ASSOCIAZIONE (O RELATIONSHIP)

L'associazione (in inglese relationship) è un legame che stabilisce un'interazione tra le entità.

Per esempio tra l'entità *Persona* e l'entità *Automobile* esiste un'associazione che può essere descritta nel linguaggio naturale secondo due *versi*: una persona possiede una o più automobili e un'automobile è posseduta da una persona.

Quindi si può dire che tra l'entità *Persona* e l'entità *Automobile* esiste l'associazione *Possiede*; tra l'entità *Automobile* e l'entità *Persona* esiste l'associazione *Posseduta da*.

L'esempio mostra che solitamente i *sostantivi* che compaiono nelle frasi del linguaggio naturale corrispondono alle entità, mentre i *verbi* corrispondono alle associazioni.

ATTRIBUTI

Le proprietà delle entità e delle associazioni vengono descritte attraverso gli attributi.

Esempi di attributi per l'entità *Automobile* sono: *Modello, Produttore, Cilindrata, PrezzoListino.*

Caratteristiche di ogni attributo sono:

- Il ***formato*** di un attributo indica il tipo di valori che assume; i tre formati base sono: carattere, numerico, data/ora.
- La ***dimensione*** indica la quantità massima di caratteri o cifre inseribili.
- L'***opzionalità*** indica la possibilità di non essere sempre valorizzato: l'attributo è obbligatorio se deve avere valore non nullo (per esempio il nome di una persona in un'anagrafica), facoltativo se sono accettabili valori nulli (per esempio il titolo di studio di una persona).

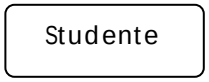
Valore Nullo Il valore nullo , in inglese ***Null***, (da non confondere con la stringa di caratteri blank o con un numero di valore zero) rappresenta un'informazione mancante, inapplicabile o sconosciuta.

Dominio I diversi valori assunti dagli attributi determinano le diverse istanze dell'entità. L'insieme dei possibili valori assunti da un attributo si chiama dominio dell'attributo.

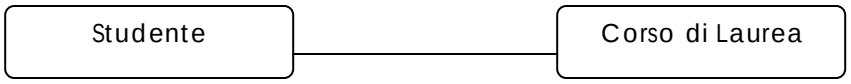
Chiave Si indica con il termine chiave o chiave primaria (*primary key*) l'insieme di uno o più attributi che consentono di distinguere un'istanza dall'altra: esempi di chiavi sono il codice di un prodotto o la matricola di un dipendente.

RAPPRESENTAZIONE GRAFICA: SCHEMA E/R

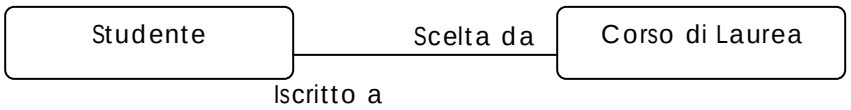
Entità: un rettangolo contenente all'interno il nome dell'entità.



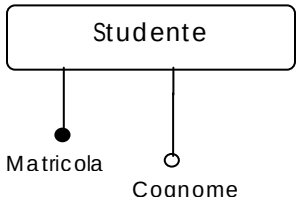
Associazione: linea, la quale unisce le due entità interessate.



Versi dell'associazione: la descrizione compare a fianco della linea e dell'entità di partenza del verso.



RAPPRESENTAZIONE GRAFICA DI ATTRIBUTI E CHIAVE



Attributi: linea che parte dall'entità e termina con il nome ed un piccolo cerchio.
Chiave: sottolineatura del nome oppure colore nel cerchietto dell'attributo.

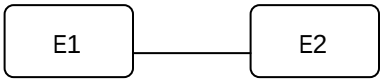
LE ASSOCIAZIONI TRA ENTITÀ: GRADO

Il **grado** del verso dell'associazione è la caratteristica che indica quante istanze dell'entità di arrivo si associano all'istanza dell'entità di partenza.

Il grado può essere:
- **a uno**
- **a molti**

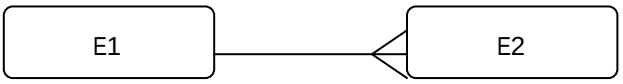
Associazione 1:1 (uno a uno) o biunivoca

Ogni istanza della prima entità si deve associare ad una sola istanza della seconda entità e viceversa. Il simbolismo che indica il grado a uno nell'associazione tra le entità è la linea stessa.



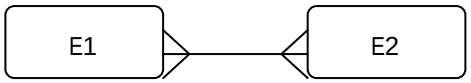
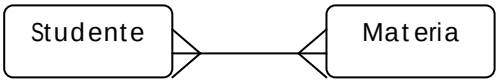
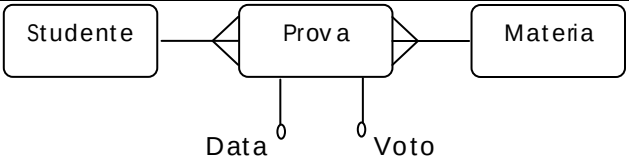
Associazione 1:N (uno a molti) o semplice

Ogni istanza della prima entità si può associare a una o più istanze della seconda entità, mentre ogni istanza della seconda entità si deve associare ad una sola istanza della prima.



Il grado a molti si rappresenta con l'aggiunta di altre due linee in prossimità dell'entità di arrivo.

Associazione N:N (molti a molti) o complessa

<table>
<tr><td>Ogni istanza della prima entità si può associare a una o più istanze della seconda entità e viceversa.</td><td> 
</td></tr>
<tr><td>Associazioni molti a molti
L'associazione molti a molti può essere facilmente scomposta in due associazioni uno a molti, anche per consentire di rappresentare gli attributi dell'associazione.
Ogni Studente può essere valutato in una o più Materie. Ogni Materia può essere oggetto di valutazione per uno o più Studenti.</td><td> 
</td></tr>
<tr><td>Ogni Studente può essere sottoposto a una o più Prove
Ogni Prova deve essere riferita a un solo Studente.
Ogni Materia può essere controllato con una o più Prove
Ogni Prova deve essere riferita ad una sola Materia.</td><td> 

L'associazione molti a molti è stata trasformata in due associazioni uno a molti, con l'aggiunta di una terza entità.</td></tr>
</table>

IL LIVELLO LOGICO: MODELLO RELAZIONALE [I1][ES1][ES2][E1][E2]

Il **modello relazionale** rappresenta il database come un insieme di tabelle. Si basa su alcuni concetti tipicamente matematici di relazione tra insiemi di oggetti.

Importanza all'uso rigoroso del linguaggio matematico, con due obiettivi importanti:

1. utilizzare un linguaggio conosciuto a livello universale, quale è il linguaggio matematico,
2. eliminare i problemi di ambiguità nella terminologia e nella simbologia.

LA VISIONE RELAZIONALE DEI DATI: LA RELAZIONE COME TABELLA

Tabella Acquisti

NumFattura	Fornitore	CodProd	Reparto	Quantità

1. Ogni riga rappresenta una n-pla (riga) della relazione(tabella) R
2. L'ordine delle righe non è importante
3. Tutte le righe sono diverse tra loro
4. Ogni colonna è rappresentata con un titolo

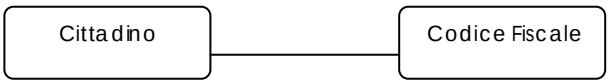
DAL MODELLO CONCETTUALE AL MODELLO LOGICO: REGOLE DI DERIVAZIONE

- OGNI *ENTITÀ* DIVENTA UNA TABELLA
- OGNI *ATTRIBUTO* DI UN'ENTITÀ DIVENTA UN CAMPO DELLA RELAZIONE (NOME DI UNA COLONNA DELLA TABELLA)
- OGNI COLONNA DELLA RELAZIONE EREDITA LE CARATTERISTICHE DELL'ATTRIBUTO DELL'ENTITÀ DA CUI DERIVA
- L'IDENTIFICATORE UNIVOCO DI UN'ENTITÀ DIVENTA LA *CHIAVE PRIMARIA* DELLA RELAZIONE DERIVATA
- ***Associazioni***
- L'ASSOCIAZIONE *UNO A UNO* DIVENTA UN'UNICA RELAZIONE CHE CONTIENE GLI ATTRIBUTI DELLA PRIMA E DELLA SECONDA ENTITÀ
- L'IDENTIFICATORE UNIVOCO DELL'ENTITÀ DI PARTENZA NELL'ASSOCIAZIONE *UNO A MOLTI* DIVENTA **CHIAVE ESTERNA** (*FOREIGN KEY*) DELL'ENTITÀ DI ARRIVO ASSOCIATA, CIOÈ I SUOI ATTRIBUTI - IDENTIFICATORI UNIVOCI - DIVENTANO COLONNE DELLA SECONDA RELAZIONE
- L'ASSOCIAZIONE *MOLTI A MOLTI* DIVENTA UNA NUOVA RELAZIONE (IN AGGIUNTA ALLE RELAZIONI DERIVATE DALLE ENTITÀ) COMPOSTA DAGLI IDENTIFICATORI UNIVOCI DELLE DUE ENTITÀ E DAGLI EVENTUALI ATTRIBUTI DELL'ASSOCIAZIONE.

REGOLE DI DERIVAZIONE: ESEMPIO 1

Cittadini e codici fiscali

(*associazione uno a uno*)



Anagrafe

CodiceFiscale	Cognome	Nome	DataNascita

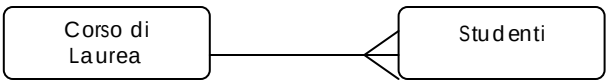
Anagrafe (CodiceFiscale, Cognome, Nome, DataNascita)

In generale dall'associazione *uno ad uno* del modello concettuale viene derivata un'unica tabella che contiene gli attributi della prima e della seconda entità.

REGOLE DI DERIVAZIONE: ESEMPIO 2

Corso di Laurea e Studenti

(*associazione uno a molti*)



Ogni Corso di Laurea *può essere* scelto da *uno o più* Studenti.

Ogni Studente *deve essere* iscritto a *un solo* Corso di Laurea.

Corso di Laurea

Cod_Corso	Descrizione	Facoltà

Studente

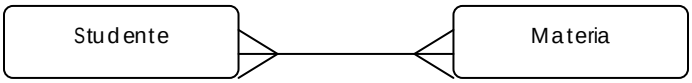
Matricola	Cognome	Nome	Data_N	*Cod_Corso*

Corso di Laurea (Cod_Corso, Descrizione, Facoltà)

Studente (Matricola, Cognome, Nome, Data_N, *CodFac*)

In generale dal modello concettuale vengono derivate le tabelle che rappresentano le entità; l'associazione *uno a molti* viene tradotta aggiungendo agli attributi dell'entità a molti la chiave dell'entità a uno (**chiave esterna**).

REGOLE DI DERIVAZIONE: ESEMPIO 3



Studente e Materia

(*associazione molti a molti*)

Ogni Studente *può essere* valutato in *una o più* Materie.

Ogni Materia *può essere* oggetto di verifica per *uno o più* Studenti.

Studente		
Matricola	Cognome	Nome

Materia

Cod_materia	Descrizione

Prova

Matricola	*Cod_Materia*	Data	Voto

Studente (Matricola, Cognome, Nome, Indirizzo)
Materia(Cod_Materia, Descrizione)
Prova (*Matricola, CodCorso*, Data, Voto)

In generale, vengono derivate le relazioni corrispondenti alle entità; l'associazione *molti e molti* viene tradotta con una terza relazione contenente le chiavi delle due entità e gli eventuali attributi dell'associazione.

ESEMPI di Verifiche

VERIFICA DI INFORMATICA CLASSE

Cognome e Nome:_____

1) Come viene rappresentata l'associazione molti a molti nel modello logico?
   a) L'identificatore univoco dell'entità di arrivo diventa chiave esterna dell'entità di partenza associata;
   b) Viene creata una nuova entità che contiene solo gli attributi che non sono chiavi ed eventualmente gli attributi dell'associazione
   c) L'identificatore univoco dell'entità di partenza diventa chiave esterna dell'entità di arrivo associata;
   d) Viene creata una nuova entità che contiene gli attributi dell'associazione e le chiavi primarie della prima e della seconda entità;

2) Quale delle seguenti affermazioni esprime meglio il significato di ridondanza dei dati:
   a) Garantire che le operazioni effettuare sul database non provochino una perdita di consistenza ai dati;
   b) Impedire che il database venga danneggiato da interventi accidentali o non autorizzati;
   c) I dati contenuti nel database devono essere effettivamente utilizzabili nelle applicazioni dell'azienda;
   d) Gli stessi dati compaiono più volte in archivi diversi, cioè il database è costituito da archivi non integrati di dati.

3) Quale delle seguenti frasi esprime meglio il significato di DBMS (Data Base Management System)
   a) È l'insieme degli archivi che formano una base di dati
   b) È l'insieme delle operazioni che si possono eseguire su una base di dati
   c) È il software che consente di costruire e gestire una base di dati
   d) È la persona che si occupa della gestione di una base di dati

4) Quale fra le seguenti frasi esprime meglio il significato di integrità:
   a) Garantire che le operazioni effettuate sul database da utenti autorizzati non provochino una perdita di consistenza ai dati;
   b) Impedire che il database venga danneggiato da interventi accidentali o non autorizzati;
   c) I dati contenuti nel database devono essere effettivamente utilizzabili nelle applicazioni dell'azienda;
   d) Gli stessi dati non compaiono più volte in archivi diversi, cioè il database è costituito da archivi integrati di dati.

5) In MS-Access le sottomaschere sono particolarmente utili per visualizzare i dati delle tabelle tra le quali esiste:
   a) Una relazione molti a molti
   b) Una relazione uno a molti
   c) Un campo con lo stesso nome
   d) Una relazione uno a uno

6) La regola di integrità sull'entità:
   a) Consente di avere valori nulli per la chiave
   b) Non consente di avere valori nulli per la chiave
   c) Consente di eliminare un record nella tabella primaria, se a questa non corrispondono righe nella tabella correlata
   d) Non consente di eliminare un record nella tabella primaria, se a questa corrispondono righe nella tabella correlata

7) Per ciascuno degli oggetti elencati a sinistra riferiti ad una tabella con i dati sulle PROVE degli studenti, specifica se si tratta di una tabella (T), di un campo (C), di una chiave primaria (K), di una chiave esterna (E), indicando a destra la lettera corretta:

Oggetto	**Tipo (T, C, K, E)**
Voto	
Identificativo della prova	
Data della prova	
Materia della prova	
Matricola dello studente	
Prove	

8) Spiegare il significato di *integrità referenziale* (max 8 righe)

9) Quali sono i limiti principali dell'organizzazione dei dati in archivi non integrati? (max 8 righe)

10) Perché in una relazione è presente una chiave primaria? (max 5 righe)

Valutazione:

7 punti	Domande con risposta a scelta multipla	42
12 punti	Domanda n° 7	12
22 punti	Domanda n° 8	22
14 punti	Domanda n° 9	14
10 punti	Domanda n° 10	10
	Totale	100

LE BASI DI DATI

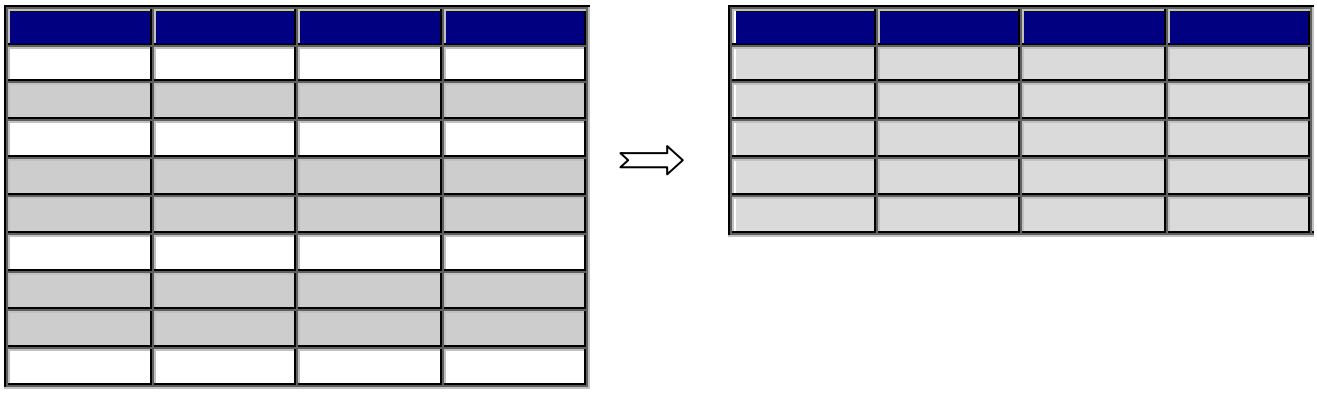
GLI OPERATORI RELAZIONALI [I1][I2][ES1][E1]

Gli operatori relazionali agiscono su una o più relazioni per ottenere una nuova relazione (servono a realizzare le *interrogazioni* sul database).

Ci sono tre operazioni fondamentali:

- SELEZIONE
- PROIEZIONE
- CONGIUNZIONE

L'OPERAZIONE DI SELEZIONE

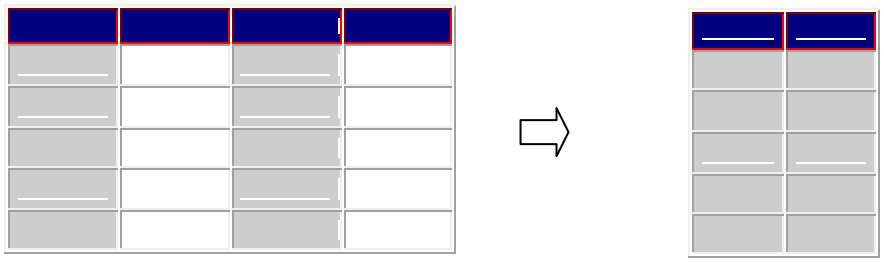


La **selezione** (*select*) genera una **nuova tabella** costituita **solo dalle righe della relazione di partenza che soddisfano a una determinata condizione**; vengono cioè selezionate le righe con i valori degli attributi corrispondenti alla condizione prefissata.

La tabella risultante ha:

- **grado** uguale, cioè lo stesso numero di colonne della tabella di partenza
- **cardinalità** minore o uguale (di solito minore), cioè un minor numero di righe (solo quelle che soddisfano alla condizione).

L'OPERAZIONE DI PROIEZIONE

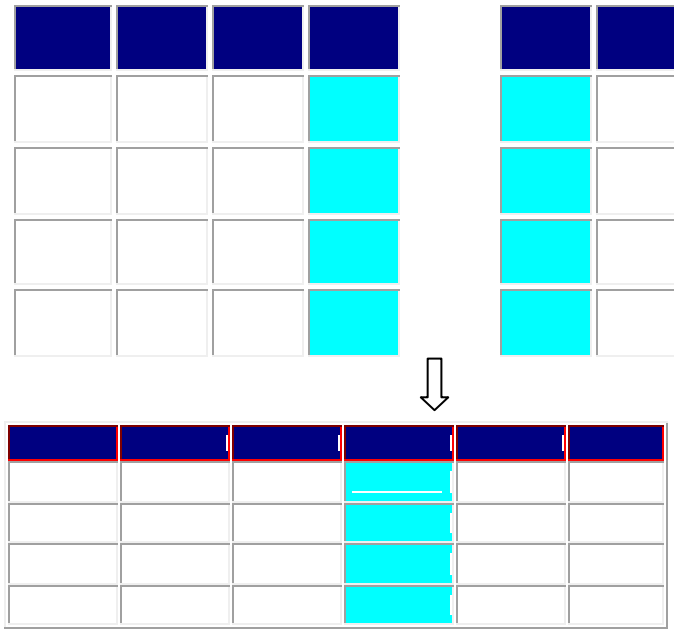


La **proiezione** (*project*) genera una nuova relazione estraendo dalla tabella iniziale due o più colonne corrispondenti agli attributi prefissati.

La tabella risultante ha:

- **grado** minore, cioè un numero di colonne minore o uguale alla relazione di partenza;
- **cardinalità** minore o uguale a quella di partenza (perché le righe uguali vengono ridotte a una).

L'OPERAZIONE DI CONGIUNZIONE



La **congiunzione** (*join*) serve a combinare due tabelle aventi uno o più attributi in comune, generando una nuova tabella che contiene le righe della prima e della seconda tabella, che possono essere combinate secondo i valori uguali dell'attributo comune.

La tabella risultante ha:

- **grado** (numero di colonne) uguale alla somma dei gradi delle tabelle di partenza meno uno, perché l'attributo comune compare una sola volta.

- **cardinalità** (numero di righe) non prevedibile a priori, in quanto si ottengono solo le righe che possono essere combinate.

TIPI DI CONGIUNZIONE (JOIN)

- **equi-join**: corrispondenza di valori uguali per attributi comuni nelle due tabelle.
- **join esterno (*outer join*)**: restituisce le righe dell'una e dell'altra tabella anche se non sono presenti valori uguali per l'attributo comune; di tutte queste vengono combinate solo le righe per le quali il valore dell'attributo comune nella prima tabella trova un valore uguale nella colonna dell'attributo comune nella seconda tabella.
- **left join**: elenca comunque tutte le righe della prima tabella congiungendo alle righe della seconda solo quelle per le quali si trovano valori corrispondenti per l'attributo comune.
- **right join**: restituisce comunque tutte le righe della seconda tabella e di queste congiunge con le righe della prima tabella solo quelle per le quali si possono trovare valori corrispondenti per l'attributo comune.
- **self-join**: vengono combinate righe di una tabella con le righe della stessa tabella quando sono presenti valori corrispondenti per attributi, cioè due attributi con lo stesso dominio.

LA NORMALIZZAZIONE DELLE RELAZIONI [I1][E1]

Processo formalizzato con il quale le tabelle vengono trasformate in altre tabelle in modo tale che ogni tabella corrisponda a un singolo oggetto della realtà rappresentata.

Le regole della normalizzazione sono definite per evitare inconsistenza dei dati e anomalie nelle operazioni di aggiornamento.

Livelli crescenti di normalizzazione corrispondono a diverse forme normali delle relazioni.

Garantire che la scomposizione di una tabella in altre tabelle di forma normale superiore non provochi perdita di dati.

DEFINIZIONI

- chiave (*chiave primaria*): l'insieme di uno o più attributi che identificano in modo univoco una n-tupla (riga della tabella);
- attributo non-chiave: un campo che non fa parte della chiave primaria;
- dipendenza funzionale tra attributi: il valore di un attributo A1 determina un singolo valore dell'attributo A2

  $A1 \longrightarrow A2$

- dipendenza transitiva: un attributo A2 dipende da A1 e l'attributo A3 dipende da A2; allora A3 dipende transitivamente da A1

  se $A1 \longrightarrow A2$ e $A2 \longrightarrow A3$ ....

  allora $A1 \longrightarrow A3$ in modo transitivo

PRIMA FORMA NORMALE (1FN)

Relazione con i requisiti fondamentali del modello relazionale:

- tutte le righe della tabella contengono lo stesso numero di colonne;
- gli attributi rappresentano informazioni elementari**;**
- i valori che compaiono in una colonna appartengono allo stesso dominio;
- ogni riga è diversa da tutte le altre;
- l'ordine con il quale le righe compaiono nella tabella è irrilevante.

In particolare gli attributi devono essere informazioni non ulteriormente scomponibili, cioè non devono avere sottoattributi, né essere gruppi di attributi ripetuti.

ESEMPIO DI 1FN

Dipendente

Matricola	Nome	DataNascita	Assunzione Stipendi	Figlio1 Figlio2 Figlio3

è una relazione **non normalizzata** (ci sono domini non semplici e attributi ripetuti).

Relazioni ottenute con la normalizzazione:

Dipendente

Matricola	Nome	DataNascita

Assunzione

Matricola	DataAssunto	Qualifica

Stipendio

Matricola	DataPagam	Importo

Figli

Matricola	NomeFiglio	DataNFiglio

SECONDA FORMA NORMALE (2FN)

È una relazione

- in prima forma normale
- tutti i suoi attributi non-chiave dipendono dall'intera chiave, cioè non possiede attributi che dipendono soltanto da una parte della chiave.

La seconda forma normale elimina la dipendenza parziale degli attributi dalla chiave.

La relazione
R1(A1,A2,A3,A4,A5)
con (A1, A2) $\longrightarrow$ A3
A1 $\longrightarrow$ A4
A2 $\longrightarrow$ A5
non è in 2FN, e può essere normalizzata in 2FN con le relazioni:
R21(A1, A2, A3)
R22(A1, A4)
R23(A2, A5)

ESEMPIO DI 2FN [Kent, 1983]

Inventario

CodiceProd	**Magazzino**	Quantità	IndirizzoMagaz

Chiave composta: CodiceProd + Magazzino
IndirizzoMagaz dipende da Magazzino *(si ha dipendenza parziale dalla chiave)*

Problemi:

- indirizzo del magazzino ripetuto per ogni prodotto (ridondanza)
- se cambia l'indirizzo, occorre modificare molte righe
- con errori nell'aggiornamento si avrebbero indirizzi diversi per lo stesso magazzino (inconsistenza)
- se non ci sono prodotti in un magazzino, non si può conoscere il suo indirizzo.

SOLUZIONE: RELAZIONI IN 2FN

Inventario

CodiceProd	**Magazzino**	**Quantità**

Magazzino

Magazzino	**IndirizzoMagaz**

TERZA FORMA NORMALE (3FN)

È una relazione:

- in seconda forma normale
- tutti gli attributi non-chiave dipendono direttamente dalla chiave, cioè non possiede attributi non-chiave che dipendono da altri attributi non-chiave.

La terza forma normale elimina la dipendenza **transitiva** degli attributi dalla chiave.
La relazione
R2(A1,A2,A3,A4)
con A2 $\longrightarrow$ A4
non è in 3FN, e può essere normalizzata in 3FN con le relazioni:
R31(A1, A2, A3)
R32(A2, A4)

ESEMPIO DI 3FN [Kent, 1983]
Impiegato

NomeImp	Reparto	TelefReparto

Chiave: NomeImp
TelefReparto dipende da Reparto, oltre che da NomeImp *(si ha dipendenza transitiva dalla chiave).*
Problemi:

- telefono del Reparto ripetuto per ogni Impiegato di quel Reparto
- se il telefono cambia, occorre modificare molte righe
- con errori di aggiornamento, si avrebbero telefoni differenti
- se un Reparto non ha impiegati, non si può conoscere il suo telefono.

SOLUZIONE: RELAZIONI IN 3FN
Impiegato

NomeImp	**Reparto**

Reparto

Reparto	**TelefReparto**

RIASSUMENDO...

con la normalizzazione:

- la relazione iniziale viene scomposta in più relazioni
- complessivamente forniscono le stesse informazioni di partenza
- mantengono le dipendenze tra gli attributi
- in ciascuna di esse ogni attributo dipende direttamente dalla chiave
- vengono evitati problemi di ridondanza e di inconsistenza dei dati
- non ci deve essere perdita complessiva delle informazioni

I dati possono essere ritrovati attraverso operazioni di *congiunzione* tra le relazioni sugli attributi comuni.

Prima forma normale: possiede i requisiti fondamentali del modello relazionale, in particolare ogni attributo è elementare, non ci sono righe uguali, non ci sono attributi di gruppo o ripetuti.

Seconda forma normale: è in prima forma normale e non ci sono attributi non-chiave che dipendono *parzialmente* dalla chiave.

Terza forma normale: è in seconda forma normale e non ci sono attributi non-chiave che dipendono *transitivamente* dalla chiave.

INTEGRITÀ REFERENZIALE

Insieme di regole del modello relazionale che garantiscono l'integrità dei dati:

- rendere valide le associazioni tra le tabelle
- eliminare gli errori di inserimento, cancellazione o modifica di dati collegati tra loro.

Integrità referenziale: per ogni valore non nullo della chiave esterna, deve esistere un valore corrispondente della chiave primaria nella tabella associata.
Relazione R1 con chiave K1,
Relazione R2 chiave esterna KE2 associata a K1:

- ogni valore di KE2 deve avere un valore uguale di K1 in una delle righe della tabella R1

oppure

- il valore di KE2 deve essere nullo.

R1 R2

K1		

K2			***KE2***

REGOLE PRATICHE PER L'INTEGRITÀ REFERENZIALE

⇨ non è possibile immettere un valore nella chiave esterna della tabella associata, se tale valore non esiste tra le chiavi della tabella primaria.

Esempio: un ordine non può essere assegnato ad un cliente che non esiste nella tabella dei clienti.

⇨ non è possibile eliminare una riga dalla tabella primaria, se esistono righe legate ad essa attraverso la chiave esterna nella tabella correlata.

Esempio: non è possibile eliminare un cliente dalla tabella dei clienti se ci sono ordini assegnati a quel cliente nella tabella degli ordini.

⇨ non si può modificare il valore alla chiave nella tabella primaria, se ad essa corrispondono righe nella tabella correlata.

Esempio: non è possibile modificare il valore alla chiave di un cliente se si sono ordini per quel cliente già registrati nella tabella degli ordini.

Verifiche assegnate [ES1]

VERIFICA DI INFORMATICA

Nome: Classe: Data:

Costruire il modello dei dati per organizzare le informazioni relative ai dipendenti e alle aziende dove lavorano.

Tra gli attributi dei dipendenti, oltre ai dati anagrafici, sono contenuti anche il reddito annuo lordo, il numero dei famigliari a carico. Negli attributi dell'azienda, oltre alla denominazione, occorre inserire anche l'indirizzo e il settore di attività.

1) Individuare (le entità), gli attributi e le associazioni, motivando le scelte effettuate. Disegnare il modello E/R con i versi delle associazioni, verificare lo schema con le regole di lettura.
2) Definire il modello logico con le tabelle.
3) Rappresentare le interrogazioni con un linguaggio di pseudocodifica, individuando le operazioni relazionali:
   a) Elenco dei dipendenti che risiedono a Torino o Genova.
   b) Denominazione e indirizzo dell'azienda avente un codice prefissato.
   c) Cognome e nome dei dipendenti che hanno un reddito annuo lordo compreso tra L1 e L2 (dove L1<L2).
   d) Elenco dei dipendenti che hanno più di quattro famigliari a carico.
   e) Cognome e nome dei dipendenti che operano in un settore dato in input.
   f) Data in input una professione restituire il cognome, il nome dei dipendenti e la denominazione della ditta presso la quale lavorano.
   g) Lista delle differenti professioni presenti in un'azienda di cui si conosce il codice (facoltativa).

GRIGLIA DI VALUTAZIONE

Quesito	1	2	3.a	3.b	3.c	3.d	3.e	3.f	3.g	P. totale
Punteggio	10	16	10	8	12	10	14	10	10	100
Punti										/100

SIMULAZIONE 3ª PROVA ESAME di STATO 5ª
Cognome e Nome:_____

1) Come viene rappresentata l'associazione uno a molti nel modello logico?
   a) l'identificatore univoco dell'entità di arrivo diventa chiave esterna dell'entità di partenza associata;
   b) viene creata una nuova entità che contiene le chiavi della prima e della seconda entità;
   c) l'identificatore univoco dell'entità di partenza diventa chiave esterna dell'entità di arrivo associata;
   d) viene creata una nuova entità che contiene solo gli attributi che non sono chiavi;

2) Quale fra le seguenti frasi esprime meglio il significato di integrità:
   a) garantire che le operazioni effettuate sul database da utenti autorizzati non provochino una perdita di consistenza ai dati;
   b) impedire che il database venga danneggiato da interventi accidentali o non autorizzati;
   c) i dati contenuti nel database devono essere effettivamente utilizzabili nelle applicazioni dell'azienda;
   d) gli stessi dati non compaiono più volte in archivi diversi, cioè il database è costituito da archivi integrati di dati.

3) L'operazione di _____ serve a combinare due relazioni aventi uno o più attributi comuni:
   a) Unione
   b) Selezione
   c) Congiunzione
   d) Proiezione

4) Supponendo di avere le seguenti tabelle Artisti(**codiceartista**, cognome, none, nazionalità) e Opere(**codiceopera**, titolo, anno, tipo, ***codiceartista***) quale di queste interrogazioni è corretta?
   a) Congiunzione di (Proiezione di (Selezione di Artisti per cognome="Y" and nome="X") su codiceartista) e di Opere su codiceartista) su titolo, anno.
   b) Proiezione di (Congiunzione di (Selezione di Artisti per cognome="Y" and nome="X") su codiceartista) e di Opere su codiceartista) su titolo, anno.
   c) Congiunzione di (Proiezione di (Selezione di Artisti per cognome="Y" , nome="X") su codiceartista) e di Opere su codiceartista) su titolo, anno.
   d) Selezione di (Congiunzione di (Proiezione di Artisti per cognome="Y" and nome="X") su codiceartista) e di Opere su codiceartista) su titolo and anno.

5) Operatori relazionali: che cosa si ottiene con l'operazione di selezione?(max 10 righe)

6) Come possono essere classificate le associazioni tra entità? (max 10 righe)

IL LINGUAGGIO SQL [I1][I2][I3][I4][I5][E1]

I LINGUAGGI PER DATABASE

- **DDL** (*Data Definition Language*): linguaggio per la descrizione dei dati, delle tabelle e delle viste (strumento con il quale si crea la struttura fisica del data base, facendo riferimento allo schema logico).
- **DML** (*Data Manipulation Language*): linguaggio per il trattamento (o manipolazione) dei dati contenuti nel data base (inserimenti, modifiche o cancellazioni).
- **Query Language**: linguaggio per le interrogazioni alla base di dati (ritrovamento dei dati sulla base dei criteri di ricerca richiesti dall'utente).

Nei moderni prodotti software DBMS si ha più frequentemente un **linguaggio per la basi di dati** che comprende i comandi di tipo DDL + DML + Query Language.

CARATTERISTICHE DEI LINGUAGGI PER DATABASE

I linguaggi per basi di dati relazionali possiedono i comandi per:

- definizione del data base
- manipolazione dei dati
- associazione tra tabelle diverse
- interrogazioni degli utenti

Ci sono alcune caratteristiche comuni ai diversi linguaggi:

- si basano sulla visione tabellare dei dati
- non richiedono la specificazione dei percorsi per ritrovare i dati
- operano su gruppi di righe o sull'intera tabella, anziché su una riga per volta
- usano interfacce per l'utente (a menu o grafiche).

IL LINGUAGGIO SQL (*STRUCTURED QUERY LANGUAGE*):

- è di fatto lo standard tra i linguaggi per la gestione di data base relazionali.
- prima versione IBM alla fine degli anni '70 per un prototipo di ricerca (System R)
- negli anni '80 linguaggio per DBMS della IBM (DB2 e SQL/DS)
- standard ANSI (American National Standards Institute) nel 1986
- standard ISO (International Standards Organization) nel 1987
- aggiornamenti degli standard nel 1992 da ANSI (ANSI X3.135) e ISO (ISO 9075).

IDENTIFICATORI, DATI, COSTANTI E OPERATORI

- **Identificatori**: nomi di tabelle e di colonne
  Per identificare il nome di una colonna: NomeTabella.NomeColonna (separati dal punto).
- **Tipi standard** per gli attributi:
  CHARACTER, DATE, INTEGER, SMALLINT, FLOAT, ecc. *N.B. Ci possono essere differenze della dichiarazione dei dati in diverse versioni del linguaggio SQL nei prodotti DBMS (per esempio in Access per Windows).*

♦ Il valore **Null** nelle colonne della tabella indica un valore non disponibile o non definito.
♦ Le **costanti** stringa sono delimitate dai caratteri ' (apice).
♦ Si possono usare gli **operatori** NOT, AND e OR nella scrittura delle condizioni.

LA DEFINIZIONE DELLE TABELLE (DDL= *Data Definition Language*)

Il linguaggio SQL possiede i comandi per creare, modificare ed eliminare le tabelle dal database relazionale:

- **CREATE TABLE** seguito dal nome della tabella e dall'elenco degli attributi;
  per ogni attributo occorre specificare il nome e il tipo di dato.
- **ALTER TABLE** per aggiungere una nuova colonna (**ADD**) a quelle già esistenti
  per togliere una colonna (**DROP**).
- **DROP** seguito dal nome della tabella, per eliminare una tabella.

Nota: nei prodotti DBMS moderni queste operazioni vengono eseguite usando l'interfaccia utente (a menu o grafica).

I COMANDI PER LA MANIPOLAZIONE DEI DATI (DML)

Il linguaggio SQL possiede i comandi per inserire, modificare ed eliminare le righe di una tabella, cioè le funzioni di linguaggio DML (*Data Manipulation Language*):

- **INSERT:** inserire nuovi dati nelle righe della tabella
- **UPDATE:** aggiornare i valori nella tabella
- **DELETE:** cancellare righe della tabella.

Nota: nei prodotti DBMS moderni queste operazioni vengono eseguite in modo usando l'interfaccia utente (a menu o grafica).

Il comando SELECT

È il comando principale di SQL che realizza le funzioni di linguaggio per le interrogazioni (*Query Language*) :

- attivare le interrogazioni sulle relazioni
- implementare le operazioni relazionali per ottenere nuove tabelle.

Struttura generale del comando *SELECT :*

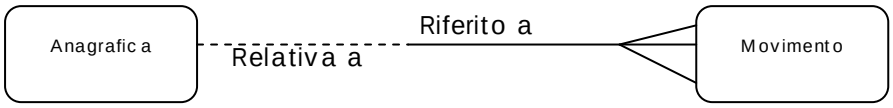
```
SELECT .....
FROM .....
WHERE .....
```

- dopo *SELECT:* nomi delle colonne da elencare *(per indicare tutti gli attributi si scrive l'asterisco * accanto a Select)*
- dopo *FROM* : il nome o i nomi delle tabelle
- dopo *WHERE*: la condizione da controllare sui valori delle righe (anche più condizioni combinate con gli operatori AND, OR e NOT).

Con SELECT DISTINCT... le righe duplicate nella tabella risultante vengono ridotte a una.

MODELLO DI DATABASE PER GLI ESEMPI SUCCESSIVI

Viene descritto ora il database che verrà utilizzato per gli esercizi successivi.
E' un database generico con anagrafica e movimenti di uso molto frequente: per esempio, clienti e fatture, studenti ed esami sostenuti, contribuenti e versamenti di imposta, prodotti e movimenti di magazzino, conti e movimenti contabili, ecc.)
Due entità : Anagrafica e Movimento
Associazione : uno a molti



Ogni *Anagrafica* può essere relativa a uno o più *Movimenti*.
Ogni *Movimento* deve essere riferito a una sola *Anagrafica*.

Tabelle derivate:

Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, *Codice*)
Codice: chiave della tabella Anagrafica
Numero: chiave della tabella Movimento
Codice : chiave esterna della tabella Movimento

LE OPERAZIONI RELAZIONALI NEL LINGUAGGIO SQL [I1][ES1][ES2][ES3][F1]

- **SELEZIONE**

```
SELECT *
FROM NomeTabella
WHERE Condizione
```

Esempio:
Date le tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, Codice)
si vuole ottenere: *l'elenco delle anagrafiche con Indirizzo = 'Milano'*
L'operazione relazionale che consente di ottenere i dati richiesti è:
Selezione di Anagrafica per Indirizzo = 'Milano'

```
SELECT *
FROM Anagrafica
WHERE indirizzo = 'Milano'
```

- **PROIEZIONE**

```
SELECT Colonna1, Colonna2, ...
FROM NomeTabella
```

Esempio:
Date le tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, Codice)
si vuole ottenere:
a) *Elenco dei codici e dei nomi delle anagrafiche*
Operazione relazionale: proiezione di Anagrafica su Codice e Nome

```
SELECT Codice, Nome
FROM Anagrafica
```

b) *Elenco dei movimenti con data e importo*
Operazione relazionale: proiezione di Movimento su Data e Importo

```
SELECT Data, Importo
FROM Movimento
```

- **CONGIUNZIONE**

```
SELECT *
FROM Tabella1, Tabella2
WHERE Tabella1.Attrib1 = Tabella2.Attrib2
```

Esempio:
Date le tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, Codice)
si vuole ottenere:
Elenco dei movimenti con i dati anagrafici ad essi relativi
Operazione relazionale: congiunzione di Anagrafica su Codice e di Movimento su Codice

```
SELECT *
FROM Anagrafica, Movimento
WHERE Anagrafica.Codice= Movimento.Codice
```

LE OPERAZIONI RELAZIONALI NEL LINGUAGGIO SQL: uso di più operatori

```
SELECT Colonna1, Colonna2, ...
FROM Tabella1, Tabella2
WHERE Tabella1.Attrib1 = Tabella2.Attrib2
AND Condizione
```

Esempio:
Date le tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, Codice)
si vuole ottenere:
Elenco dei movimenti con Nome e Importo riferiti alle anagrafiche aventi un indirizzo prefissato

Operazioni relazionali:
1. Selezione di Anagrafica per Indirizzo = prefissato
2. Congiunzione della tabella ottenuta su Codice e di Movimento su Codice
3. Proiezione della tabella ottenuta su Nome e Importo

```
SELECT Nome, Importo
FROM Anagrafica, Movimento
WHERE Anagrafica.Codice= Movimento.Codice
AND Indirizzo = [Quale Indirizzo]
```

Esercizio: Con riferimento al modello di database con Anagrafica e Movimento, risolvere la seguente interrogazione, specificando le operazioni relazionali e la codifica nel linguaggio SQL.

Tabelle derivate: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, *Codice*)

1. Elenco dei movimenti relativi a un codice prefissato.
Operazione relazionale: Selezione di Movimento per Codice = [prefissato]

```
SELECT *
FROM Movimento
WHERE Codice = [prefissato]
```

2. Elenco dei movimenti con codice, data, importo
Operazione relazionale: Proiezione di Movimento su Codice, Data, Importo

```
SELECT Codice, Data, Importo
FROM Movimento
```

3. Elenco con Nome anagrafico e Numero di registrazione dei movimenti
Operazioni relazionali:
Congiunzione di Movimento su Codice e di Anagrafica su Codice
Proiezione della tabella ottenuta su Nome, Numero

```
SELECT Nome, Numero
FROM Movimento, Anagrafica
WHERE Movimento.Codice = Anagrafica.Codice
```

4. Data dei movimenti con Indirizzo dell'anagrafica.
Operazioni relazionali:
Congiunzione di Movimento su Codice e di Anagrafica su Codice
Proiezione della tabella ottenuta su Data, Indirizzo

```
SELECT Data, Indirizzo
FROM Movimento, Anagrafica
WHERE Movimento.Codice = Anagrafica.Codice
```

5. Numero di registrazione dei movimenti riferiti all'anagrafica avente un Nome prefissato.
Operazioni relazionali:
Selezione di Anagrafica per Nome = [prefissato]
Congiunzione della tabella ottenuta su Codice e di Movimento su Codice
Proiezione della tabella ottenuta su Numero

```
SELECT Numero
FROM Movimento, Anagrafica
WHERE Movimento.Codice = Anagrafica.Codice
AND Nome = [prefissato]
```

6. *Nome anagrafico, Data e Importo dei movimenti riferiti a un indirizzo prefissato*
Operazioni relazionali:
Selezione di Anagrafica per Indirizzo = [prefissato]
Congiunzione della tabella ottenuta su Codice e di Movimento su Codice
Proiezione della tabella ottenuta su Nome, Data, Importo

```
SELECT Nome, Data, Importo
FROM Movimento, Anagrafica
WHERE Movimento.Codice = Anagrafica.Codice
AND Indirizzo = [prefissato]
```

LE FUNZIONI DI AGGREGAZIONE

Sono funzioni predefinite che agiscono sui valori contenuti in insiemi di righe della tabella e restituiscono un valore calcolato.

- **COUNT**
  La funzione COUNT restituisce il numero di righe presenti in una tabella, incluse quelle con campi di tipo Null.

```
SELECT COUNT (*)
FROM NomeTabella
```

  La funzione COUNT restituisce il numero di righe presenti in una tabella, escluse le righe che hanno valore Null nella colonna dell'attributo specificato.

```
SELECT COUNT (Nomeattributo)
FROM NomeTabella
```

***Esempio**:*
Data la tabella: Anagrafica (Codice, Nome, Indirizzo)
Numero delle persone registrate nella tabella delle anagrafiche aventi un indirizzo prefissato

```
SELECT COUNT(*)
FROM Anagrafica
WHERE Indirizzo = [Quale Indirizzo]
```

- **SUM:** restituisce la somma di tutti i valori contenuti in una colonna specificata (*l'attributo utilizzato nel calcolo deve essere di tipo numerico*)

```
SELECT SUM (NomeAttributo)
FROM NomeTabella
```

Esempio:
Data la tabella: Movimento (Numero, Descrizione, Data, Importo, *Codice_Cliente*)
Importo totale dei movimenti riferiti a un codice cliente prefissato

```
SELECT SUM(Importo)
FROM Movimento
WHERE Codice_Cliente = [Quale Codice]
```

- **AVG:** calcola la media (*average*) dei valori (numerici) contenuti in una determinata colonna di una tabella.

N.B. Non include nel calcolo i valori di tipo *Null* presenti nella colonna.

```
SELECT AVG (NomeAttributo)
FROM NomeTabella
```

Esempio:
Data la tabella: Movimento (Numero, Descrizione, Data, Importo, *Codice*)
Importo medio dei movimenti

```
SELECT AVG(Importo)
FROM Movimento
```

- **MIN e MAX**

Restituiscono rispettivamente il valore minimo e il valore massimo tra i valori della colonna specificata come argomento della funzione (anche per campi di tipo carattere). Ignorano i campi con valore *Null*.

```
SELECT MIN (NomeAttributo), MAX(NomeAttributo)
FROM NomeTabella
```

Esempio:
Date le tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, *Codice*)
Valori minimo e massimo tra gli importi dei movimenti

```
SELECT MIN(Importo), MAX(Importo)
FROM Movimento
```

Ultimo nome dell'anagrafica

```
SELECT MIN(Nome)
FROM Anagrafica
```

ORDINAMENTO

La clausola **ORDER BY** consente di ottenere i risultati di un'interrogazione ordinati secondo i valori contenuti in una o più colonne, tra quelle elencate accanto alla parola *SELECT*.

```
SELECT Colonna1, Colonna2, ..., ColonnaK, ... ,ColonnaN
FROM NomeTabella
ORDER BY ColonnaK
```

Ordinamento crescente: **ASC:** stringhe dalla A alla Z, numeri dal minore al maggiore
Ordinamento decrescente: **DESC:** stringhe dalla Z alla A, numeri dal maggiore al minore
L'ordinamento crescente è quello di default (non occorre specificare ASC*).*

Esempio:
Data la tabella: Anagrafica (Codice, Nome, Indirizzo)
Elenco alfabetico delle anagrafiche

```
SELECT Nome, Indirizzo
FROM Anagrafica
ORDER BY Nome
```

RAGGRUPPAMENTI

La clausola **GROUP BY** serve per raggruppare un insieme di righe aventi lo stesso valore nelle colonne indicate: produce una riga di risultati per ogni raggruppamento.
Viene usata con le funzioni di aggregazione (*SUM, COUNT...)* : per ciascuna riga della tabella risultante viene prodotto un valore di raggruppamento.

```
SELECT Colonna, ... , Funzione(nome_colonna)
FROM NomeTabella
GROUP BY Colonna
```

Esempio:
Data la tabella: Movimento (Numero, Descrizione, Data, Importo, *Codice_Cliente*)
Totale degli importi dei movimenti per ciascun codice anagrafico

```
SELECT Codice_Cliente, SUM(Importo)
FROM Movimento
GROUP BY Codice_Cliente
```

Condizioni sui raggruppamenti

L'uso della clausola **HAVING** consente di sottoporre al controllo di una o più condizioni i gruppi creati con la clausola *GROUP BY*.
La condizione scritta dopo *HAVING* normalmente controlla il valore restituito dalle funzioni di aggregazione (*COUNT*, *SUM*, *AVG*, *MIN*, *MAX*).

```
SELECT Colonna, Funzione
FROM NomeTabella
GROUP BY Colonna
HAVING Condizione
```

Esempio:
Data la tabella: Movimento (Numero, Descrizione, Data, Importo, *Codice_Cliente*)
Importo medio dei movimenti per i clienti (codice_cliente) aventi più di 20 movimenti registrati

```
SELECT Codice_Cliente, AVG(Importo)
FROM Movimento
GROUP BY Codice_Cliente
HAVING COUNT(*) >20
```

Attenzione alla differenza tra WHERE e HAVING:

- con WHERE vengono poste condizioni sulle righe di una tabella
- con HAVING il controllo delle condizioni è fatto sui risultati delle funzioni di aggregazione applicate a gruppi di righe.

LE CONDIZIONI DI RICERCA

Il linguaggio SQL utilizza operatori e predicati insieme alle clausole WHERE e HAVING per determinare i criteri di selezione rispettivamente delle righe e dei raggruppamenti.

- Segni del confronto =, <, >, <> (diverso), >=, <=.
- Più condizioni legate tra loro con gli operatori AND e OR, precedute eventualmente dall'operazione NOT.
- Predicati: BETWEEN, LIKE, IN.

♦ **BETWEEN:** controlla se un valore è compreso all'interno di un intervallo di valori, inclusi gli estremi.
N:B: è possibile valutare la condizione opposta, anteponendo al BETWEEN l'operatore NOT.

Esempio:
Data la tabella: Movimento (Numero, Descrizione, Data, Importo, *Codice*)
Elenco dei movimenti con importo compreso tra 100 e 200

```
SELECT *
FROM Movimento
WHERE Importo BEETWEN 100 AND 200
```

♦ **IN:** controlla le righe che hanno i valori di un attributo compresi in una lista di valori indicati dopo la parola *IN*. Anche in questo caso è possibile valutare la condizione opposta: NOT IN.

Esempio:
Data la tabella: Anagrafica (Codice, Nome, Indirizzo)
Elenco delle anagrafiche con indirizzo Milano, Torino o Napoli

```
SELECT *
FROM Anagrafica
WHERE Indirizzo IN ('Milano', 'Torino', 'Napoli')
```

♦ **LIKE**

Il predicato **LIKE** confronta il valore di un attributo di tipo carattere con un modello di stringa che può contenere caratteri jolly:
_ **(*underscore*)** per indicare un singolo carattere qualsiasi in quella posizione della stringa;
% (*percento*) per indicare una sequenza qualsiasi di caratteri in quella posizione della stringa.

Esempio:
LIKE 'xyz%' vengono ricercate tutte le stringhe che iniziano con i caratteri 'xyz';
LIKE '%xyz' serve a ricercare tutte le stringhe che finiscono con i caratteri 'xyz';
LIKE '%xyz%'' per tutte le stringhe che contengono al loro interno i caratteri 'xyz';
LIKE '_xyz' controlla le stringhe di 4 caratteri che finiscono con xyz.

Esempio:
Data la tabella: Anagrafica (Codice, Nome, Indirizzo)
Elenco delle anagrafiche con nome che inizia con 'Ros' (Rossi, Rosi, Rossini,...)

```
SELECT *
FROM Anagrafica
WHERE Nome LIKE 'Ros%'
```

♦ **IS NULL**

Questo predicato confronta il valore di una colonna con il valore NULL.
L'uso di questo predicato è il solo modo per controllare la presenza del valore NULL in una colonna.
Si può usare l'operatore NOT per verificare la condizione opposta.
L'operatore IS viene usato solo con la parola NULL.

Esempio:
Data la tabella: Anagrafica (Codice, Nome, Indirizzo)

```
SELECT cognome, nome
FROM Anagrafica
WHERE Indirizzo IS NOT NULL
```

Esercizio [ES1]

Facendo riferimento al DB con le tabelle Anagrafica e Movimento, risolvere le seguenti interrogazioni in SQL.

Tabelle: Anagrafica (Codice, Nome, Indirizzo)
Movimento (Numero, Descrizione, Data, Importo, *Codice*)

1. *Calcolare il numero dei movimenti con importo superiore a una cifra prefissata.*

```
SELECT COUNT(*)
FROM Movimento
WHERE Importo >[importoinput]
```

2. *Calcolare la somma degli importi per i movimenti che si riferiscono alle anagrafiche di un indirizzo prefissato.*

```
SELECT SUM(Importo)
FROM Movimento, Anagrafica
WHERE
Movimento.Codice=Anagrafica.Codice
AND Indirizzo = [indirizzoinput]
```

3. *Calcolare la media degli importi per i movimenti aventi una descrizione prefissata.*

```
SELECT AVG(Importo)
FROM Movimento
WHERE Descrizione = [descrizioneinput]
```

4. *Calcolare il valore massimo per gli importi dei movimenti di un'anagrafica avente un nome prefissato.*

```
SELECT MAX(Importo)
FROM Movimento, Anagrafica
WHERE
Movimento.Codice=Anagrafica.Codice
AND Nome = [nomeinput]
```

5. *Produrre l'elenco dei movimenti con importo superiore a 300 in ordine crescente di importo.*

```
SELECT *
FROM Movimento
WHERE Importo > 300
ORDER BY Importo
```

6. *Raggruppare le anagrafiche per indirizzo e fornire il numero per ogni indirizzo.*

```
SELECT Indirizzo, COUNT(*)
FROM Anagrafica
GROUP BY Indirizzo
```

7. *Nome delle anagrafiche che hanno almeno 30 movimenti registrati nella tabella dei movimenti.*

```
SELECT Nome
FROM Movimento, Anagrafica
WHERE
Movimento.Codice=Anagrafica.Codice
GROUP BY Nome
HAVING COUNT(*) > 29
```

8. *Elenco delle anagrafiche che hanno l'iniziale del nome uguale ad A.*

```
SELECT *
FROM Anagrafica
WHERE Nome LIKE 'A%'
```

***Attenzione*:**
Il predicato **IS NULL** confronta il valore in una colonna con il valore NULL. L'uso di questo predicato è il solo modo per controllare la presenza del valore NULL in una colonna. Si può anche usare l'operatore NOT per valutare la condizione opposta.
Elenco con cognome e nome degli studenti per i quali è indicata la provincia nella tabella Studenti(matricola, cognome, nome, ind, città, prov)

```
SELECT cognome, nome
FROM Studenti
WHERE prov IS NOT NULL
```

Verifiche assegnate

VERIFICA DI INFORMATICA **Classe**

Cognome e Nome:_____

1. Spiegare il significato della clausola per il raggruppamento (max 6 righe)

2. Quali sono le caratteristiche delle funzioni di aggregazione SUM e AVG? (max 6 righe)

3. Qual è il significato del valore Null? (max 3 righe)

4. Si supponga di avere una tabella Personale(coddip, cognome,nome,funzione,livello,ind,prov)
Che differenza c'è fra queste due interrogazioni SQL? (max 5 righe)

```
SELECT COUNT(funzione)          SELECT COUNT(*)
FROM personale                  FROM personale
```

5. Quale delle seguenti frasi SQL rappresenta una congiunzione?

a. SELECT * FROM Tab1
b. SELECT a1, a2 FROM Tab1
c. SELECT * FROM Tab1, Tab2 WHERE Tab1.codice=Tab2.codice
d. SELECT * FROM Tab2 WHERE Tab2.a1=" valore_input"

6. **Quale delle seguenti frasi SQL consente di ottenere l'elenco cronologico delle fatture di un fornitore prefissato?**
   a. SELECT * FROM fatture WHERE codforni="codice_in_input" ORDER BY datafatt
   b. SELECT * FROM fatture WHERE codforni="codice_in_input" GROUP BY datafatt
   c. SELECT codforni FROM fatture WHERE codforni="codice_in_input" ORDER BY datafatt
   d. SELECT codforni, datafatt FROM fatture WHERE codforni="codice_ _input" ORDER BY datafatt

7. **Quale delle seguenti frasi SQL consente di ottenere il nome delle città da cui provengono meno di 50 studenti?**
   a. SELECT città FROM studenti GROUP BY città HAVING COUNT(*)<=50
   b. SELECT Studenti FROM città GROUP BY città HAVING COUNT(*)<50
   c. SELECT Studenti FROM città GROUP BY città WHERE COUNT(*)<50
   d. SELECT città FROM studenti GROUP BY città HAVING COUNT(*)<50

8. **Quale delle seguenti frasi SQL rappresenta una proiezione?**
   a. SELECT * FROM Tab1
   b. SELECT a1, a2 FROM Tab1
   c. SELECT * FROM Tab1, Tab2 WHERE Tab1.codice=Tab2.codice
   d. SELECT * FROM Tab2 WHERE Tab2.a1="valore_input"

9. **Quale delle seguenti frasi SQL consente di ottenere il numero dei partecipanti ad una determinata manifestazione sportiva?**
   a. SELECT COUNT FROM manifestazioni,iscrizioni WHERE manifestazioni.codmanifestazione=iscrizioni.codmanifestazione AND descrizione="manifestazione_input"
   b. SELECT * COUNT FROM manifestazioni,iscrizioni WHERE manifestazioni.codmanifestazione=iscrizioni.codmanifestazione AND descrizione="manifestazione_input"
   c. SELECT COUNT(*) FROM manifestazioni,iscrizioni WHERE manifestazioni.codmanifestazione=iscrizioni.codmanifestazione AND descrizione="manifestazione_input"
   d. SELECT FROM manifestazioni,iscrizioni WHERE manifestazioni.codmanifestazione=iscrizioni.codmanifestazione AND descrizione="manifestazione_input" AND COUNT(*)

10. **Quale delle seguenti frasi SQL rappresenta una selezione?**
    a. SELECT * FROM Tab1
    b. SELECT a1, a2 FROM Tab1
    c. SELECT * FROM Tab1, Tab2 WHERE Tab1.codice=Tab2.codice
    d. SELECT * FROM Tab2 WHERE Tab2.a1=" valore_input"

Griglia di valutazione

Domanda	Punti	Valutazione	Commenti
1	12		
2	18		
3	10		
4	14		
5	7		
6	7		
7	9		
8	7		
9	9		
10	7		
TOTALE	100		

VERIFICA DI INFORMATICA Classe 5^ Data:
Cognome e Nome:_____

Un'agenzia di viaggi vuole organizzare in una base di dati tutte le informazioni riguardanti la sua attività: i viaggi o vacanze, le organizzazioni turistiche o tour operator, i clienti dei propri servizi, le nazioni o località o città che sono destinazione dei viaggi/vacanze.
Tutte le attività proposte ai clienti sono dei pacchetti: ciascuno di essi si riferisce ad un'organizzazione turistica, e riguardano una precisa zona del mondo o nazione o città. Ci possono essere offerte di viaggi con la stessa destinazione da parte di organizzazioni diverse.
Di ogni pacchetto offerto occorre conoscere modalità e prezzi; di ogni località o nazione è opportuno tenere memorizzate informazioni circa le condizioni climatiche, la moneta corrente, ecc. Un cliente può ovviamente acquistare in date diverse molti viaggi o vacanze.

1. Rapppresentare il modello ER
2. Derivare il modello logico
3. Rappresentare le seguenti interrogazioni in SQL:
    a. Elenco dei viaggi/vacanza con prezzo compreso fra due valori dati in input;
    b. Elenco dei soggiorni e relativo prezzo in una località prefissata;
    c. Elenco dei clienti che hanno fatto un viaggio con una determinata destinazione;
    d. Elenco dei clienti che hanno acquistato un viaggio da una organizzazione data in input.

Bibliografia di riferimento

- Garavagila - Petracchi – Forte "*Informatica vol.3 Archivi classici e database*" Zanichelli Editore
- Garavagila – Mazzucchelli - Petracchi "*Informatica vol.3 Database con applicazioni in Access*" Zanichelli Editore
- C.N. Prague – M. R. Irwin "*Microsoft Access 2000*" Tecniche Nuove

Web Link

Siti di Riferimento	Metodi di appprendimento
1) Tutorial di MySQL 2) Tutorial di MySQL Sito ufficiale MySQL Lezioni Access	1) Cooperative Learning 2) Cooperative Learning