**1.1 Sistemi informativi, informazioni e dati**

Per lo svolgimento di ogni attività, sia a livello individuale che in organizzazioni di ogni dimensione, sono essenziali la disponibilità di informazioni e la capacità di gestirle in modo efficace; ogni organizzazione è dotata di un *sistema informativo,* che organizza e gestisce le informazioni necessarie. Per indicare la parte automatizzata del sistema informativo viene di solito utilizzato il termine ***sistema informatico***.

Nei sistemi informatici, le informazioni vengono rappresentate per mezzo di dati, che hanno bisogno di essere interpretati tramite informazioni.

Dato e informazioni: i dati da soli non hanno alcun significato, ma, una volta interpretati e correlati opportunamente, essi forniscono informazioni.

Informazione: notizia , dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni e modi d’essere.

Dato: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione.

In informatica 🡪 elementi di informazione costituiti da simboli che devono essere elaborati.

**1.2 Bd e sistemi di gestione**

BASE DI DATI: Collezione di dati, utilizzati per rappresentare le informazioni di interesse per un sistema informativo. Concepite per gestire in modo integrato e flessibile le informazioni di interesse per diversi soggetti, limitando i rischi di ridondanza e incoerenza.

Approccio “convenzionale” 🡪 sfrutta la presenza di un archivio o file per memorizzare i dati in modo persistente sulla memoria di massa.

Un sistema di gestione di basi di dati (Data Base management System - DBMS) è un sistema software in grado di gestire collezioni di dati che siano *grandi*, *condivise* e *persistenti*, assicurando la loro *affidabilità* e *privatezza*. Un DBMS deve essere *efficiente* ed *efficace*.

1. Grandi: possono avere anche dimensioni enormi, in generale molto maggiori della memoria centrale disponibile. Quindi i DBMS devono prevedere una gestione dei dati in memoria secondaria.
2. Condivise: applicazioni e utenti diversi devono poter accedere, secondo opportune modalità, a dati comuni. In questo modo si evita la ridondanza dei dati, perché si evitano ripetizioni e si riduce la possibilità di inconsistenze: ogni dato è memorizzato in modo univoco. Per garantire l’accesso condiviso ai dati da parte di molti utenti che operano contemporaneamente, il DBMS dispone di un meccanismo detto *controllo di concorrenza.*
3. Persistenti: tempo di vita non limitato a quello delle singole esecuzioni dei programmi che lo utilizzano.
4. Affidabilità: capacità di conservare intatto il contenuto del database o almeno di permetterne la ricostruzione in caso di malfunzionamenti. Perciò i DBMS forniscono specifiche funzionalità di salvataggio e ripristino.
5. Privatezza: Ciascun utente viene abilitato a svolgere solo determinate azioni sui dati, attraverso meccanismi di autorizzazione.
6. Efficienza: capacità di svolgere le operazioni utilizzando un insieme di risorse accettabile per gli utenti. Questa caratteristica dipende dalle tecniche utilizzate nell’ implementazione del DBMS e dalla bontà della realizzazione della base di dati da parte dei suoi progettisti.
7. Efficacia: capacità di rendere produttive le attività degli utenti.

I DBMS utilizzano file che, a differenza dei normali file, ammettono un’organizzazione dei dati più sofisticata.

**1.3 Modelli dei dati**

*Modello dei dati*: insieme di concetti utilizzati per organizzare i dati di interesse e descriverne la struttura in modo che essa risulti comprensibile a un elaboratore.

Ogni modello fornisce *meccanismi di ristrutturazione, che permettono di definire nuovi tipi sulla base dei tipi predefiniti (elementari) e costruttori di tipo. Es. in C i nuovi tipi si costruiscono con struct, \*(puntatore) ecc...*

Il modello attualmente più diffuso è quello *relazionale*: permette di costruire tipi per mezzo del costruttore *relazione,* che consente di organizzare i dati in insiemi di record a struttura fissa. Una relazione viene spesso rappresentata tramite una tabella: le righe rappresentano i record e le colonne i campi dei rispettivi record.

Altri modelli logici sono:

* Gerarchico: uso di strutture ad albero;
* Reticolare: uso di grafi;
* A oggetti: evoluzione del modello relazionale, che utilizza una programmazione ad oggetti;
* Xml: rivisitazione del modello gerarchico. I dati vengono rappresentati insieme alla loro descrizione e non devono sottostare a un’unica struttura logica (auto-descrittivo e semi-strutturato).

Esistono, inoltre, i modelli concettuali, non disponibili nei DBMS commerciali. Essi descrivono i concetti del mondo reale, piuttosto che i dati.

Es. modello Entità – Relazione.

**1.3.1 Schemi e istanze**

Schema: parte invariante nel tempo, costituita dalle caratteristiche dei dati. *Componente intensionale* del DB. Nome della relazione seguito dai suoi attributi. Es. docenza (Corso, NomeDocente).

Istanza o stato: valori effettivi, varianti nel tempo, delle sue righe. *Componente estensionale* del DB.

**1.3.2 Livelli di astrazione nei DBMS**

DBMS articolato su tre livelli:

1. Schema logico: descrizione dell’intero DB tramite modello logico scelto (quelli precedentemente citati).
2. Schema interno: rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione.
3. Schema esterno: descrizione di una porzione della BD, tramite modello logico. Organizzazione dei dati diverse rispetto a quelle dello schema logico, che riflettono il punto di vista di uno o più utenti. Quindi uno schema logico = più schemi esterni.

**1.3.3 Indipendenza dei dati**

Proprietà principale dei DBMS:

1. Indipendenza fisica: consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati. In base a questa proprietà è possibile modificare le strutture fisiche senza influire sui programmi che usano i dati stessi.
2. Indipendenza logica: consente di interagire con il livello esterno della Bd indipendentemente dal livello logico e viceversa.

N.B gli accessi alla BD avvengono solo tramite livello esterno. Il DBMS traduce le operazioni in termini dei livelli sottostanti.

**1.4 Linguaggi e utenti delle basi di dati**

**1.4.1 Linguaggi**

Si dividono in:

1. DDL (Data Definition Language): per definire schemi logici, esterni e fisici e le autorizzazioni per l’accesso;
2. DML (Data Manipulation Language): per la manipolazione e l’aggiornamento delle istanze di BD.

**1.5 Vantaggi e svantaggi dei DBMS**

Vantaggi:

* Permettono di considerare i dati come una risorsa comune di una organizzazione, a disposizione di tutte le sue componenti;
* La Bd fornisce un modello utilizzabile nelle applicazioni attuali e, con estensione, future;
* Controllo centralizzato dei dati;
* La condivisione permette di ridurre ridondanze e inconsistenze;
* L’indipendenza dei dati favorisce lo sviluppo di applicazioni più flessibili e facilmente modificabili.

Svantaggi:

* Costosi e complessi;
* Forniscono, in forma integrata, una serie di servizi associati a un costo. Se alcuni di questi servizi sono necessari è difficile scorporare quelli effettivamente richiesti dagli altri. Ciò può comportare un calo di prestazioni.

**2.1 Il modello relazionale: strutture**

**2.1.1 Modelli logici nei sistemi di Bd**

Il modello relazionale si basa sui concetti di relazione e tabella. Esso risponde al requisito di indipendenza dei dati; gli utenti che accedono ai dati e i programmatori che sviluppano le applicazioni fanno riferimento solo al livello logico; i dati descritti al livello logico sono poi realizzati per mezzo di opportune strutture fisiche, ma per accedere ai dati non è necessario conoscere le strutture fisiche stesse. Gli altri modelli logici includevano espliciti riferimenti alla sottostante struttura realizzativa.

relazione matematica, secondo la definizione normalmente data nella teoria degli insiemi elementare.

**2.1.2 Relazioni e tabelle**

Dati n > 0 insiemi D1, D2 , ... , Dn, non necessariamente distinti, il prodotto cartesiano di D1, D2 , ... , Dn, indicato con D1xD2 x ... x Dn, è costituito dall'insieme delle n-uple (v1, v2, …, vn) tali che vi, appartiene a Di , per 1 <= i <= n. Una relazione matematica sui domini D1, D2, ..., Dn è un sottoinsieme del prodotto cartesiano D1 x D2 x ... x Dn. Il numero n delle componenti del prodotto cartesiano (e quindi di ogni n-upla) viene detto *grado* del prodotto cartesiano e della relazione. Il numero di elementi (cioè di n-uple) della relazione viene chiamato *cardinalità* della relazione.

**2.1.3 Relazioni con attributi**

Le informazioni organizzate nelle relazioni hanno una struttura che si può ricondurre a quella dei record.

Utilizzo di notazioni non posizionali, che si differenziano dalle notazioni posizionali degli array.

Nel caso dei record a ogni campo è associato un nome, dunque, a ciascuna occorrenza di dominio nella relazione associamo un nome detto “attributo”. L’ordinamento degli attributi è irrilevante.

Per formalizzare i concetti, indichiamo con V l'insieme dei domini e specifichiamo la corrispondenza fra attributi e domini, nell'ambito di una relazione, per mezzo di una funzione *dom* : X 🡪 D, che associa a ciascun attributo A ϵ X un dominio *dom*(A) ϵ D. Poi, diciamo che una t-upla su un insieme di attributi X è una funzione t che associa a ciascun attributo A ϵ X un valore del dominio *dom*{A).

Nuova definizione di relazione: Una relazione su un insieme X è un insieme di t-uple su X. Mentre nella relazione matematica abbiamo n-uple i cui elementi sono individuati per posizione, nelle t-uple della nuova definizione gli elementi sono individuati tramite attributi, quindi con notazione non posizionale.

**2.1.4 Relazioni e basi di dati**

In generale una base di dati è costituita da più relazioni, le cui t-uple contengono valori comuni, ove necessario per stabilire corrispondenze.

Negli altri modelli logici, reticolare e gerarchico, le corrispondenze si realizzano in modo esplicito tramite puntatori e vengono perciò detti “basati su record e puntatori”.

I vantaggi del modello relazionale, basato su valori, sono:

* Richiede di rappresentare solo ciò che è rilevante dal punto di vista dell’applicazione (dell’utente); i puntatori sono legati ad aspetti realizzativi;
* La rappresentazione logica non fa riferimento a quella fisica che può cambiare nel tempo. Permette quindi di avere indipendenza fisica dei dati;
* Poiché l’informazione è tutta contenuta nei valori è più semplice trasferire i dati da un contesto all’altro; con i puntatori è più complesso.

Possiamo a questo punto riassumere le definizioni relative al modello relazionale, con un po' di precisione, distinguendo il livello degli schemi da quello delle istanze.

* Uno *schema di relazione* è costituito da un simbolo R, detto nome della relazione, e da un insieme di (nomi di) attributi X = {A1, A2, ..., An}, il tutto di solito indicato con R(X). A ciascun attributo è associato un dominio.
* Uno *schema di base di dati* è un insieme di schemi di relazione con nomi diversi: R={R1(X1),R2(X2), ..., Rn (Xn ) } I nomi di relazione hanno come scopo principale quello di distinguere le varie relazioni nella base di dati.
* Una *istanza di relazione* (o semplicemente relazione) su uno schema R(X) è un insieme r di tuple su X.
* Una *istanza di base di dati* (o semplicemente base di dati) su uno schema R = {R1(X1), R2(X2), ..., Rn(Xn)} è un insieme di relazioni r = {r1, r2, …, rn), dove ogni ri, per 1 < =i <= n, è una relazione sullo schema Ri(Xi).

**2.1.5 Informazione incompleta e valori nulli**

Per rappresentare in modo semplice, ma comodo, la NON disponibilità di valori, il concetto di relazione viene esteso includendo il fatto che una t-upla possa assumere, su ciascun attributo, o un valore del dominio (sconsigliato perché porta a confusione), oppure un valore speciale, detto *valore nullo*, che denota l’assenza di informazione ma è esterno al dominio (in simboli NULL).

Viene considerato nullo, un valore: *inesistente*, *sconosciuto* o *senza informazione*.

**2.2 Vincoli di integrità**

Es. voto esame=36 🡪 valore non ammesso

Vincolo di integrità: predicato che associa a ogni istanza il valore vero o falso. Vero se soddisfa il vincolo.

* Vincolo intrarelazionale: soddisfacimento definito rispetto a singole relazioni
* Vincolo di t-upla: può essere valutato su ciascuna tupla indipendentemente dalle altre;
* Vincolo su valori o di dominio: impone una restrizione sul dominio dell’attributo (es voto compreso tra 18 e 30).
* Vincolo interrelazionale: coinvolge più relazioni (es. una matricola compare in esami solo se è presente tra gli studenti).

**2.2.2 Chiavi**

Insieme di attributi utilizzato per identificare univocamente le t-uple di una relazione.

* Un insieme K di attributi è *superchiave* di una relazione r se r non contiene due t-uple distinte t1 e t2 con t1 [K]= t2 [K];
* K è chiave di r se è una *superchiave minimale* di r (ovvero non esiste un’altra superchiave K’ di r che sia contenuta in K come sottoinsieme proprio). Es. matricola è superchiave e chiave minimale perché anche da sola identifica in modo univoco; nome, cognome e nascita è superchiave ma nessuno dei suoi sottoinsiemi è superchiave poiché possono esistere persone con nome e cognome uguali, con nome e nascita e con cognome e nascita uguali. Matricola e corso è superchiave ma non superchiave minimale perché esiste un sottoinsieme proprio che è chiave minimale (matricola), quindi corso non è necessario nella coppia matricola, corso.

Su ciascuno schema di relazione può essere definita almeno una chiave, ciò garantisce l’accessibilità di tutti i valori di una base di dati e la loro univoca identificabilità.

**2.2.3 Chiavi e valori nulli**

È necessario porre dei limiti alla presenza di valori nulli nelle chiavi delle relazioni in quanto le t-uple il cui attributo identificatore dovesse assumere valore nullo, non sarebbero più identificabili.

Per fare ciò, su una delle chiavi, detta chiave primaria, si vieta la presenza di valori nulli; sulle altre, i valori nulli sono in genere ammessi. Gli attributi che costituiscono la chiave primaria vengono spesso evidenziati attraverso sottolineatura.

**2.2.4 Vincoli di integrità referenziale**

Un *vincolo di integrità referenziale* (*Foreign key* o *referential integrity constraint*, nella letteratura inglese) fra un insieme di attributi X di una relazione R1 e un’altra relazione R2 è soddisfatto se i valori su X di ciascuna tupla dell’istanza R1 compaiono come valori della chiave (primaria) dell’istanza di R2.

Se la chiave di R2 è unica e composta di un solo attributo B (quindi l’insieme X è a sua volta costituito da un solo attributo A) il vincolo di integrità referenziale fra l’attributo A di R1 e la relazione R2 è soddisfatto se, per ogni t-upla t1 in R1 per cui t1 [A] non è nullo, esiste una t-upla t2 in R2 tale che t1 [A]= t2 [B]. Es. nella tabella incidenti devono essere presenti solo targhe di auto contenute nella tabella auto.

Nel caso più generale, dobbiamo fare attenzione al fatto che ciascuno degli attributi in X deve corrispondere a un preciso attributo nella chiave primaria K di R2. Allo scopo deve essere specificato un ordinamento sia nell’insieme X sia in K. Indicando gli attributi in ordine, X = A1A2…Ap e K = B1B2…Bp, il vincolo è soddisfatto se per ogni t-upla t1 in R1 senza nulli su X esiste una t-upla t2 in R2 con t1[Ai]=t2[Bi], per ogni i compreso fra 1 e p.

**3.1 Algebra relazionale**

Linguaggio procedurale basato su concetti di tipo algebrico. Costituito da un insieme di operatori, definiti su relazioni e che producono relazioni come risultati.

**3.1.1 Unione, intersezione, differenza**

Una relazione non è un insieme di t-uple generico, ma un insieme di t-uple omogeneo, cioè definite sugli stessi attributi. Pertanto, è possibile definire gli operatori in questione su qualunque coppia di relazioni, ma non ha senso, dal punto di vista del modello relazionale, definirli con riferimento a relazioni su attributi diversi. Dunque, si considerano ammissibili, nell’algebra relazionale, solo applicazioni degli operatori di unione, intersezione e differenza a coppie di operandi definite sugli stessi attributi.

* L’unione di due relazioni r1e r2 definite sullo stesso insieme di attributi X è indicata con r1Ur2 ed è una relazione ancora su X contenente le t-uple che appartengono a r1 oppure a r2, oppure a entrambe;
* L’intersezione di r1(X) e r2(X) è indicata con r1∩r2 ed è una relazione su X contenente solo le t-uple che appartengono sia a r1 sia a r2;
* La differenza di r1(X) e r2(X) è indicata con r1 - r2 ed è una relazione su X contenente le t-uple che appartengono a r1 e non appartengono a r2.

**3.1.2 Ridenominazione**

Tale operatore cambia il nome degli attributi lasciando inalterato il contenuto delle relazioni.

In forma generale. Sia r una relazione definita sull’insieme di attributi X e sia Y un altro insieme di attributi con la stessa cardinalità. Inoltre, siano A1A2…Ak e B1B2…Bk rispettivamente un ordinamento per gli attributi in X e un ordinamento per quelli in Y. Allora la ridenominazione ρ B1B2…Bk 🡨 A1A2…Ak (r) contiene una t-upla t’ per ciascuna t-upla t in r: t’ è una t-upla su Y e t’[Bi]=t[Ai], per i=1, …, k.

**3.1.3 Selezione**

Selezione e proiezione svolgono funzioni complementari. Sono entrambe definite su un operando e producono come risultato una porzione dell’operando che soddisfa le condizioni. Più precisamente, la selezione produce un sottoinsieme delle tuple, su tutti gli attributi(decomposizione orizzontale), mentre la proiezione dà un risultato cui contribuiscono tutte le tuple, ma su un sottoinsieme degli attributi (decomposizione verticale). Le condizioni di selezione possono prevedere confronti fra attributi e confronti fra attributi e costanti, e possono essere complesse, ottenute combinando condizioni semplici con i connettivi logici *(and)*, *v(or)* e *¬(not).*

Definizione: la selezione σF(r) produce una relazione sugli stessi attributi di r che contiene le tuple di r su cui F è vera.

**3.1.4 Proiezione**

Dati una relazione r(X) e un sottoinsieme Y di X, la proiezione di r su Y (indicata con ΠY (r)) è l’insieme di t-uple su Y ottenute dalle t-uple di r considerando solo i valori su Y: ΠY (r)={t[Y] | t ϵ r}

Essendo le relazioni definite come insiemi, non possono in esse comparire più t-uple uguali fra loro: i contributi “uguali” collassano in una sola t-upla.

In generale, il risultato di una proiezione contiene al più tante tuple quante l’operando, ma può contenerne di meno. Inoltre, ΠY (r) contiene lo stesso numero di t-uple di r se e solo se Y è superchiave per r. In questo caso, infatti, r non può contenere t-uple uguali su Y, quindi ogni t-upla dà un contributo diverso alla proiezione.

**3.1.5 Join**

È l’operatore che permette di correlare dati contenuti in relazioni diverse, confrontando i valori contenuti in esse. Esistono sostanzialmente due versioni dell’operatore: join naturale e theta-join.

***Join naturale.*** È un operatore che correla dati in relazioni diverse, sulla base di valori uguali in attributi con lo stesso nome. Il grado della relazione ottenuta come risultato di join è minore o uguale della somma dei gradi dei due operandi, perché gli attributi omonimi degli operandi compaiono una sola volta nel risultato.

***Join completi e incompleti.*** Ciascuno degli operandi contribuisce ad almeno una t-upla del risultato. Il join si dice in questo caso *completo*: per ogni t-upla t1 di r1, esiste una t-upla t in r1 join r2 tale che t[X1] = t1 (analogamente per r2). Questa proprietà non è sempre verificata, perché richiede una corrispondenza fra le t-uple delle due relazioni. Infatti, se una delle relazioni non contiene t-uple con gli stessi valori sull’attributo comune essa non viene riportata nel risultato del join. In inglese tali t-uple vengono dette *dangling* (“dondolanti”). Nel caso peggiore è ovviamente possibile che nessuna delle t-uple degli operandi sia combinabile, quindi il risultato del join è la relazione vuota. Nel caso migliore, invece, è possibile che ciascuna delle t-uple di ciascuno degli operandi sia componibile con tutte le t-uple dell’altro. In tal caso, il risultato contiene un numero di t-uple pari al prodotto delle cardinalità degli operandi e cioè |r1|x|r2| t-uple.

Riassumendo possiamo dire che il join di r1 e r2 contiene un numero di t-uple compreso fra 0 e |r1|x|r2|.

Inoltre:

* Se il join di r1 e r2 è completo, allora contiene almeno un numero di tuple pari al massimo fra |r1| e |r2|;
* Se X1 ∩ X2 contiene una chiave per r2, allora il join di r1(X1) e r2(X2) contiene al più |r1| tuple;
* Se X1 ∩ X2 coincide con una chiave per r2 e sussiste il vincolo di riferimento fra X1 ∩ X2 in r1 e la chiave di r2, allora il join di r1(X1) e r2(X2) contiene esattamente |r1| tuple.

***Join esterno (outer join).*** Esistono tre varianti: join esterno sinistro che prevede che tutte le t-uple del primo operando partecipino al risultato, eventualmente con valori nulli ove non vi siano controparti opportune; join esterno destro che estende le t-uple del secondo operando; join esterno completo, che le estende tutte.

***Join n-ario, intersezione e prodotto cartesiano.*** Il join naturale è commutativo e associativo. Nel caso in cui i due insiemi su cui sono definite le relazioni sono uguali (X1=X2), il join coincide con l’intersezione:

r1(X1) join r2(X2)= r1 (X1)∩r2 (X1).

Se i due insiemi sono disgiunti (nessun elemento comune) il risultato è sempre definito sull’unione di X1X2 e ciascuna t-upla deriva sempre da due t-uple, una per ciascuno degli operandi ma, poiché tali t-uple non hanno attributi comuni, non viene richiesta a esse nessuna condizione per partecipare al join insieme: la condizione degenera in una condizione sempre verificata. Quindi il risultato contiene le t-uple ottenute combinando, in tutti i modi possibili, le t-uple degli operandi. Il join diventa un prodotto cartesiano (operatore definito su relazioni senza attributi comuni). Termine improprio perché il prodotto cartesiano è un insieme di coppie, mentre qui abbiamo t-uple.

***Theta-join ed equi-join.*** Un prodotto cartesiano ha poca utilità ed è solitamente seguito da una selezione. Per questo viene definito un operatore derivato, composto da prodotto cartesiano seguito da selezione, il theta-join.

Un theta-join in cui la condizione di selezione sia una congiunzione di atomi di uguaglianza, con un attributo della prima relazione e uno della seconda, viene chiamato equi-join.

Date due relazioni r1(ABC) e r2(BCD), il join naturale di r1 e r2 può essere espresso per mezzo degli altri operatori in tre passi:

1. Ri-denominando gli attributi in modo da ottenere relazioni su schemi disgiunti:

ρB’C’ 🡨 BC(r2);

1. Effettuando equi-join : r1join B=B’ᴧ C=C’ ρB’C’ 🡨 BC(r2);
2. Concludendo con una proiezione che elimina gli attributi “doppioni”:

ΠABCD (r1join B=B’ᴧ C=C’ ρB’C’ 🡨 BC(r2)).

**3.1.6 Interrogazioni in algebra relazionale**

In generale è una funzione che, applicata a istanze di basi di dati, produce relazioni. Più precisamente, dato uno schema R di basi di dati, un’interrogazione è una funzione che, per ogni istanza r di R, produce una relazione su un dato insieme di attributi X.

**3.1.7 Equivalenza di espressioni algebriche**

L’algebra relazionale permette di formulare espressioni fra loro *equivalenti,* cioè che producono lo stesso risultato. Questo aspetto risulta molto importante dal punto di vista applicativo, nella fase di esecuzione delle interrogazioni. Infatti, le interrogazioni, specificate in SQL, vengono tradotte in algebra relazionale e, con riferimento all’algebra viene valutato il costo, in termini di dimensioni dei risultati intermedi. Vengono spesso utilizzate *trasformazioni di equivalenza*, cioè operazioni che sostituiscono un’espressione con una equivalente. Risultano importanti le trasformazioni che riducono le dimensioni dei risultati intermedi e quelle che preparano un’espressione alla trasformazione che ne riduce le dimensioni dei risultati intermedi.

**3.1.8 Algebra con valori nulli**

Alcune relazioni possono avere valori nulli. Per rendere più efficienti le interrogazioni su queste relazioni è stato proposto di utilizzare una logica a tre valori, in cui il predicato può essere vero, falso, o assumere valore *unknown* (sconosciuto), rappresentato con U. Un predicato assume questo valore quando almeno uno dei termini del confronto assume il valore nullo.

Si introducono due nuove forme di condizioni atomiche di selezione, per verificare se un valore è specificato o nullo:

IS NULL 🡪 vero su una t-upla t se il valore di t su A è nullo e falso se è specificato;

IS NOT NULL 🡪 vero su una t-upla t se il valore di t su A è specificato e falso se il valore è nullo.

**3.1.9 Viste**

In una base di dati relazionale possono esistere relazioni di base , il cui contenuto è autonomo, e relazioni derivate, il cui contenuto è funzione di quello di altre relazioni.

Possono esistere due tipo di relazioni derivate:

* *Viste materializzate:* relazioni derivate effettivamente memorizzate nella base di dati.Immediatamente disponibili per le interrogazioni, ma oneroso mantenerle allineate con le relazioni da cui derivano. Convenienti quando gli aggiornamenti sono rari e il calcolo della vista è complesso;
* *Relazioni virtuali:* (viste)relazioni definite per mezzo di funzioni non memorizzate nella base di dati, ma utilizzabili nelle interrogazioni. Ricalcolate per ogni interrogazione ma non presentano problemi di allineamento.

L’uso delle viste ha molti vantaggi:

1. Un utente interessato solo a una porzione di una Bd può evitare di considerare le componenti non rilevanti;
2. Espressioni molto complesse possono essere definite tramite viste;
3. Meccanismi di protezione tramite la definizione di autorizzazione di accesso alle viste;
4. Ristrutturando una base di dati, può risultare conveniente definire viste che corrispondano a relazioni sostituite da altre e perciò non più presenti dopo la ristrutturazione stessa, ma ricavabili dalle nuove relazioni. In questo modo, le applicazioni scritte con riferimento alla versione precedente dello schema possono essere utilizzate sul nuovo senza bisogno di modifiche.

**4.2 Definizione dei dati in SQL**

* Le parentesi angolari (<,>) permettono di isolare un termine della sintassi. Le parentesi quadre ([,]) indicano che il termine all'interno è opzionale, ossia può non comparire o comparire una sola volta.
* Le parentesi graffe ({,}) indicano invece che il termine racchiuso può non comparire o essere ripetuto un numero arbitrario di volte.
* Le barre verticali (|) indicano che deve essere scelto uno tra i termini separati dalle barre; un elenco di termini in alternativa può essere racchiuso tra parentesi angolari.

**4.2.1 I domini elementari**

Caratteri. Character permette di rappresentare singoli caratteri oppure stringhe. La lunghezza delle stringhe può essere fissa o variabile. Per ogni schema si può definire una famiglia di caratteri di default.

Character varying (1000) character set Greek = stringa di lunghezza variabile con lunghezza massima 1000, la cui famiglia di caratteri di default è l’alfabeto greco.

SQL ammette anche le forme compatte char e varchar corrispondenti a character e varying character.

Tipi numerici esatti. SQL mette a disposizione quattro diversi tipi numerici esatti:

* numeric [(Precisione[, Scala])] precisione = numero di cifre totali, prima e dopo la virgola; scala = numero di cifre dopo la virgola;
* decimal [(Precisione[, Scala])];
* integer;
* smallint.

Tipi numerici approssimati. Valori reali approssimati:

* Float[(Precisione)]
* Real
* Double precision

Istanti temporali. Tre forme:

* date = hour, minute e second
* time [(Precisione)] [with time zone]
* timestamp [(Precisione)] [with time zone]

Boolean. Valori true e false.

Bigint. Si aggiunge ai domini numerici esatti smallint e integer. Non ci sono limiti alla rappresentazione del dominio, ma impone che essi non siano inferiori a quelli del dominio integer.

**4.2.2 Definizione di schema**

Schema = collezione di oggetti (tabelle, domini, viste ecc.).

Sintassi 🡪 create schema [NomeSchema] [[authorization] Autorizzazione] {DefElementoSchema}

Autorizzazione = utente proprietario. Se omesso indica come proprietario l’utente che lancia il comando.

Nome schema= se omesso si assume come nome quello del proprietario.

**4.2.3 Definizione delle tabelle**

Collezione di attributi e insieme di vincoli (eventualmente vuoto).

Sintassi : create table NomeTabella (

NomeAttributo Dominio [ValoreDiDefault][Vincoli] {, NomeAttributo Dominio [ValoreDiDefault] [Vincoli]}

AltriVincoli )

**4.2.4 Definizione dei domini**

Si può far riferimento a domini definiti dall’utente a partire dai domini predefiniti, tramite la primitiva.

Create domain NomeDominio as TipoDiDato

[ValoreDiDefault]

[Vincolo]

**4.2.5 Specifica dei valori di default**

Sintassi: default <GenericoValore |user|null>

GenericoValore rappresenta un valore compatibile con il dominio. User impone come valore di default l’identificativo dell’utente che esegue il comando di aggiornamento della tabella. Null corrisponde al valore di default di base.

**4.2.6 Vincoli intrarelazionali**

Il costrutto migliore per specificare vincoli, sia interrelazionali che intrarelazionali, è il costrutto check che richiede però di formulare delle interrogazioni sulla base di dati.

I piú semplici vincoli di tipo intrarelazionale sono not null, unique e primary key.

Not null. Indica che il valore nullo non è ammesso come valore dell’attributo. Quindi l’attributo non puó non avere valore assegnato (il valore di default è concesso).