Introduzione alle basi di dati
Introduzione

Le basi di dati ed i relativi programmi di gestione (detti **DBMS**) in prima approssimazione possono essere considerati come sistemi che permettono la gestione di grandi quantità di informazioni per raggiungere gli scopi di una qualche organizzazione (azienda, istituzione etc.), dove per gestione intendiamo le seguenti attività :

1. Raccolta, acquisizione;
2. Archiviazione, conservazione;
3. Elaborazione, trasformazione, produzione;
4. Distribuzione, comunicazione, scambio.

Queste attività di gestione delle informazioni, così come le informazioni stesse, mentre a livello umano possono essere gestite usando principalmente idee informali e linguaggio naturale, a livello informatico devono essere opportunamente formalizzate e codificate. Si noti che tali formalizzazioni sono anteriori alla realizzazione dei sistemi informatici. Nei sistemi informatici le informazioni vengono rappresentate in modo strutturato ed essenziale (mediante i dati).
Sistemi informativi, sistemi informatici e basi di dati.

È utile chiarire la distinzione fra i seguenti termini:

Sistema informativo:
: Componente di una organizzazione che gestisce (acquisisce, elabora, conserva, produce) le informazioni interessanti per l'organizzazione in questione. Si noti che, in principio, l'esistenza di un sistema informativo prescinde da qualsiasi processo di automazione. In effetti istituzioni vecchie di secoli (quali ad esempio le banche) usavano sistemi informativi prima che esistesse l'energia elettrica. Anche alcune formalizzazioni usate per il trattamento sistematico dei dati sono anteriori all'informatica (ad esempio, le schede degli archivi).

Sistema informatico:
: Parte automatizzata del sistema informativo, ovvero quella parte che gestisce le informazioni usando l'informatica.

Nei sistemi informatici, le informazioni sono rappresentate in termini di dati. È per questo che si usa la terminologia base di dati.
Basi di dati

Base di dati:
: Insieme organizzato di dati utilizzati per il supporto allo svolgimento delle attività di una organizzazione (azienda, ufficio ecc). In senso più tecnico, una base di dati identifica l'insieme dei dati gestiti da un programma DBMS (*Database Management System*).

DBMS:
: Programma che gestisce grandi insiemi di dati, in modo persistente e condiviso, in modo tale da garantire privatezza, affidabilità ed efficienza.

Vediamo i vari concetti introdotti nella definizione di DBMS:

grandi:
: molte aziende hanno oramai basi di dati dell'ordine del terabyte (1 terabyte = mille miliardi di byte).

persistente:
: i dati che si gestiscono hanno un tempo di vita maggiore di quello dei programmi che li gestiscono, deve quindi essere garantita la loro persistenza nel tempo. Si noti che in molti i casi i dati sono una risorsa strategica per un'organizzazione e la loro perdita può costituire un

danno irreparabile.

privatezza:
: visto che vengono gestiti grandi quantità di dati, di solito ci sono molti utenti che hanno accesso alla base di dati. Ogni utente deve poter accedere solo ai dati per i quali ha una specifica autorizzazione.

affidabilità:
: i DBMS devono permettere di gestire eventuali malfunzionamenti (hardware o software) mediante opportune funzionalità di salvataggio (*backup*) e ripristino (*recovery*) dei dati.

efficienza:
: l'accesso alla base di dati deve essere possibile in tempi ragionevoli, così come devono essere ragionevoli le risorse di calcolo e di memoria necessarie (in generale comunque i DBMS sono programmi abbastanza pesanti e richiedono adeguate risorse).

Fra i DBMS disponibili sul mercato ricordiamo *Access*, DB2, Oracle, Informix .

Cosa si può fare con una base di dati

Per l'utente finale l'uso più comune di una base di dati è l'esecuzione di opportune interrogazioni (*query*) predefinite. Esempi tipici di interrogazioni che molti di noi hanno già usato sono l'accesso ad un catalogo elettronico di una biblioteca, la consultazione (on-line oppure sulle apposite macchine nelle stazioni) degli orari dei treni, la prenotazione di un volo aereo, la consultazione delle informazioni relative all'andamento dei titoli in borsa ecc.

Queste interrogazioni possono essere molto semplici (ad esempio, dimmi tutti i libri di Calvino che sono presenti in biblioteca) ma possono essere anche molto complesse. Ad esempio, si potrebbe voler sapere quali sono i libri di Calvino che nel titolo hanno una parola che compare anche nel titolo di un libro di Pavese e che inoltre sono stati presi in prestito almeno 10 volte. Oppure, in ambito finanziario, si potrebbe voler conoscere i titoli che hanno perso almeno il 50% del valore negli ultimi due anni, che hanno un rapporto prezzo/utili minore di quello medio dei titoli dello stesso settore e che hanno un volume superiore a quello medio dei titoli di aziende con la stessa capitalizzazione.

Questo tipo di interrogazioni possono essere espresse usando opportuni linguaggi (*query languages*) che differiscono a seconda del modello dei dati adottato (vedi più avanti).
La possibilità di usare tali linguaggi, che in sostanza sono dei linguaggi di programmazione specializzati, costituisce la maggiore differenza fra le basi di dati ed i fogli elettronici, dato che questi ultimi invece permettono di esprimere delle interrogazioni molto più semplici (usando le formule e le funzioni).
Altra differenza rilevante fra DBMS e fogli elettronici è nella quantità dei dati che si possono trattare, molto inferiore nel caso dei fogli elettronici.

Le transazioni

Le basi di dati, date le loro dimensioni, di solito sono utilizzate da più utenti e da più programmi applicativi, permettendo così un uso condiviso dei dati. In tali circostanze possono verificarsi problemi di ridondanza (stessi dati ripetuti inutilmente) e incoerenza (visione diversa dello stesso dato). Per ovviare a questi possibili problemi si usa il cataolgo e il meccanismo basato sulle transazioni:

Catalogo.
: I dati di un DBMS sono organizzati in un opportuni file che il DBMS gestisce usando anche le funzionalità del sistema operativo. Tuttavia, a differenza di quanto avviene con un normale programma che usi dei file, nei DBMS esiste una porzione della base di dati, detta catalogo o dizionario, che contiene una descrizione centralizzata dei dati e che può essere utilizzata dalle varie applicazioni. Questo permette di ridurre i problemi di incoerenza e ridondanza dei dati.

Transazione.

> Una transazione è un insieme di operazioni sulla base di dati che devono essere considerate in modo indivisibile (o "atomico"); ovvero, o tutte le operazioni dell'insieme sono eseguite correttamente oppure nessuna.

L'effetto di una transazione deve essere corretto anche in presenza di altre transazioni concorrenti eseguite nello stesso intervallo di tempo. Questo si può garantire, ad esempio, facendo si che l'effetto di due transazioni concorrenti sia equivalente alla loro esecuzione sequenziale.

Infine gli effetti di una transazione conclusasi positivamente devono essere definitivi, ovvero devono essere garantiti anche in presenza di guasti ed di esecuzione concorrente

Come esempi di transazioni si pensi alle operazioni effettuate sulla base di dati di una banca: le operazioni di addebitamento dell'importo prelevato da un Bancomat e l'erogazione del denaro devono costituire una transazione. Analogamente, nello spostamento di fondi fra due conti, le operazioni di accreditamento su un conto e addebitamento su un altro devono costituire una transazione.

Il modello dei dati

I programmi che realizzano un DBMS fanno riferimento ad una struttura logica, detta modello dei dati, che descrive i dati ad a livello piùastratto di quello che invece corrisponde alla loro rappresentazione fisica. Questo permette una maggiore indipendenza delle applicazioni dai dettagli implementativi. Ad esempio, se si passa da un file-system ad un altro, i programmi che realizzano le applicazioni non necessariamente devono essere cambiati (saranno cambiate le componenti, di livello più basso, che accedono direttamente ai dati).

Modello dei dati.

> Un modello dei dati è costituito da un insieme di costrutti utilizzati per organizzare i dati e per descriverne l'evoluzione. La parte fondamentale di un modello dei dati è costituita da opportuni meccanismi di strutturazione dei dati.

Si distinguono tre tipi di modelli dei dati

modelli concettuali:

> sono i modelli più astratti, usati per rappresentare i dati e, più in generale, i concetti esistenti nel mondo reale, secondo un formalismo che è completamente indipendente da ogni sistema informatico. Il modello concettuale più diffuso è il modello E-R (Entità -Relazioni). Sono usati nella prima fase della progettazione di una base di dati.

modelli logici:

> sono i modelli adottati nei DBMS per definire l'organizzazione dei dati utilizzati nei programmi, in modo indipendente dalle strutture fisiche di memorizzazione. I modelli principali sono i seguenti:
>
> 1. relazionale
> 2. a oggetti
> 3. reticolare,
> 4. gerarchico, a oggetti

I modelli più importanti oggi sono quello relazionale e quello a oggetti

Schema e istanza

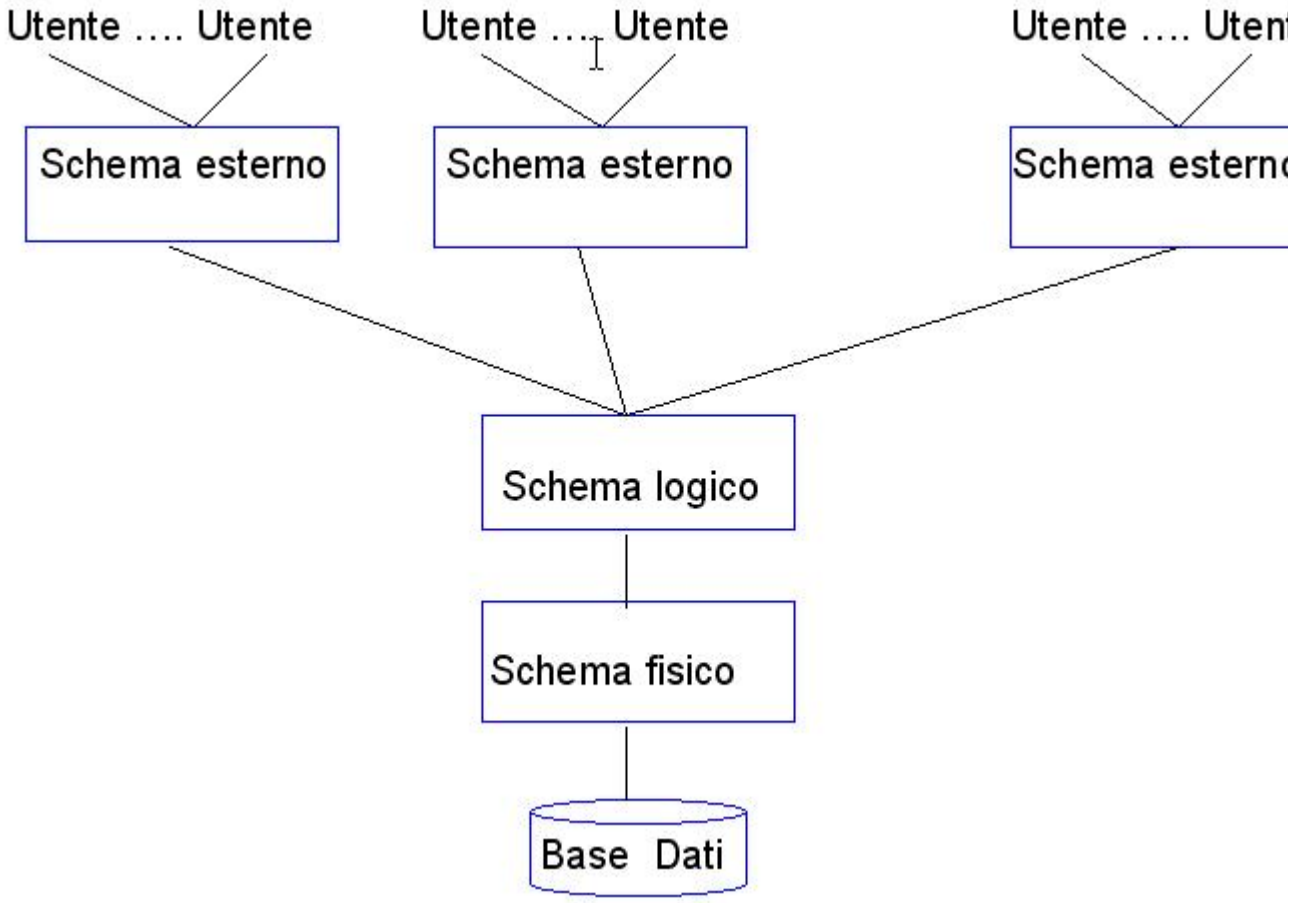
Lo **schema** di una base di dati ne descrive la struttura invariante nel tempo, ovvero l'aspetto intensionale. Distinguiamo

Schema logico:
descrizione della base di dati nel modello logico di riferimento.
Schema interno (o fisico):
descrizione delle strutture di memorizzazione usate per memorizzare lo schema logico.
Schema esterno:
descrizione, allo stesso livello di astrazione dello schema logico, di una parte della base di dati (ad esempio, la parte accessibile ad una classe di utenti)

L'istanza di una base di dati invece descrive i valori contenuti nella base di dati, che possono variare nel tempo (aspetto estensionale).

Si ha dunque lo schema nella pagina successiva

Architettura standard di una base di dati



Indipendenza dei dati

Con l'architettura vista in precedenza, che è uno standard ANSI/SPARC, l'accesso da parte degli utenti avviene solo a livello più esterno, ovvero attraverso lo schema esterno.

Questa architettura garantisce l'indipendenza dei dati in due sensi:

indipendenza fisica.
Il livello logico e quello esterno sono indipendenti da quello fisico, ovvero l'accesso ai dati avviene in modo indipendente dai dettagli di allocazione fisica e memorizzazion. Questo è molto importante in quanto, ad esempio, permette di modificare le modalità di organizzazione dei supporti di memorizzazione senza che questo influisca sui programmi delle applicazioni che accedono ai dati.

indipendenza logica.

Il livello esterno è indipendente da quello logico per cui modifiche allo schema esterno (ad esempio, per aggiungere classi di utenti o per limitare i diritti di alcuni utenti) non comportano modifiche dello schema logico.

Linguaggi per basi di dati

Nell'ambito dei DBMS si distinguono due tipi di linguaggi (anche se, nella pratica, entrambi i tipi sono presenti nello stesso formalismo).

Data Manipulation Language (DML):

linguaggi per la manipolazione di dati, ovvero per esprimere interrogazioni e aggiornamenti di (istanze di) basi di dati

Data Definition Language (DDL):

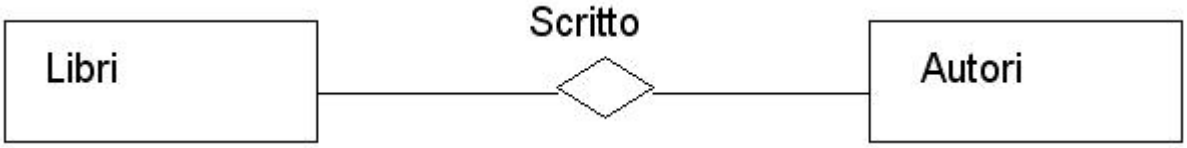
linguaggi per la per la definizione degli aspetti strutturali di una base di dati, ovvero per la definizione di schemi logici, schemi esterni, schemi fisici) ecc

I linguaggi usati dai DBMS fanno riferimento ad un particolare modello dei dati. Uno dei più diffusi è **SQL** (*Structured Query Language*), un linguaggio basato sul modello relazionale che è disponibile in varie versioni, con interfacce sia grafiche che testuali, e che può essere a sè stante, oppure immerso in linguaggio di programmazione ospite (generico come Pascal, Java e C oppure specifico per alcune applicazioni, come ad esempio per realizzare grafici).

Il modello relazionale dei dati: introduzione

Il modello gerarchico e quello reticolare utilizzano riferimenti espliciti (puntatori) fra strutture di memorizzazione (*record*) per esprimere le relazioni e le dipendenze esistenti fra i dati. Il modello relazionale è invece basato unicamente sui valori, ovvero i riferimenti fra dati diversi sono realizzati usando valori.

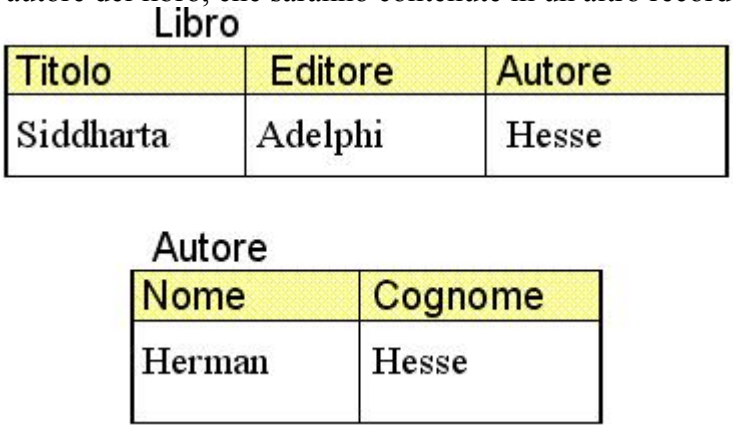
Ad esempio, supponiamo di avere la seguente situazione, descritta in termini concettuali usando il modello E-R



dove abbiamo l'entità Libro (con gli opportuni attributi quali, titolo, editore ecc.), l'entità Autore (con attributi nome, cognome, anno di nascita ecc.) e l'associazione Scritto che collega ogni libro ai suoi autori (e quindi ogni autore ai libri che ha scritto).
Questa associazione, nel modello gerarchico dei dati è rappresentata esplicitamente, mediante un riferimento (o puntatore) che collega il record contenente le informazioni su un singolo libro al record contenente le informazioni sull'autore del libro. Avremo quindi una situazione di questo tipo:

dove il tratto verticale indica il riferimento esplicito.

Nel modello relazionale invece tale riferimento è espresso implicitamente in termini di valori. Ad esempio, nel record Libro potremmo aggiungere un attributo autore dove sia memorizzato un valore che permetta di individuare univocamente un autore, e quindi di accedere alle informazioni sull'autore del libro, che saranno contenute in un altro record. La situazione è dunque la seguente



e non abbiamo più riferimenti espliciti, ma solo riferimenti impliciti dati dai valori (nel nostro caso, il cognome Hesse).

N.B. In alcuni DBMS relazionali che permettono un'interfaccia grafica (ad esempio ***Access***) si possono tracciare degli archi fra le varie tabelle. Tali archi non servono per rappresentare dei riferimenti espliciti fra i dati ma solo per abbreviare alcune operazioni (di *Join*).

Il modello relazionale dei dati: la nozione di relazione

Il modello relazionale è stato proposto da E. F. Codd nel 1970 per favorire l'indipendenza dei dati, ma è disponibile in DBMS reali nel 1981 e si basa sul concetto matematico di relazione (da cui il nome), leggermente modificato .

Dato che le relazioni possono essere rappresentate per mezzo di tabelle, nel linguaggio delle basi di dati spesso si usano i termini relazione e tabella come sinonimi.

In senso strettamente matematico una relazione su n insiemi anche non distinti D1 ,..., Dn è definita come un qualsiasi sottoinsieme del prodotto cartesiano D1 ×...× Dn .

Il prodotto Cartesiano D1 ×...× Dn è l'insieme di tutte le n-uple (d1 ,..., dn) tali che d1 è un elemento in D1 , d2 è un elemento in D2 ,..., Dn è un elemento in Dn.

Esempio. Se A è l'insieme dei cognomi, B è l'insieme dei numeri di telefono, una il seguente insieme di coppie
{ (Rossi, 06 12134), (Bianchi, 02 12115) }
è una relazione su A,B. Dal punto di vista strettamente matematico la coppia
(Rossi, 06 12134) è diversa dalla coppia (06 12134, Rossi)
dato che l'ordine dei componenti è importante. Difatti è tale ordine che ci permette di identificare il tipo dei valori (sappiamo che Rossi è un cognome, perchè appare nella prima posizione).

Per poter ottenere una rappresentazione indipendente dalla posizione possiamo attribuire ad ogni dominio un nome che ne precisa il ruolo. Abbiamo quindi una rappresentazione di tipo tabellare che nel caso dell'esempio precedente è

Cognome	**Numero Telefono**
Rossi	06 12134
Bianchi	02 12115

Si noti che tale tabella contiene esattamente la stessa informazione di

Numero Telefono	**Cognome**
06 12134	Rossi
02 12115	Bianchi

Relazioni e Tabelle

La definizione di **relazione** che si usa nel modello relazionale è dunque derivata dalla nozione matematica, introducendo dei nomi di dominio per ottenere una notazione non posizionale. Dal punto di vista pratico tale nozione coincide con quella di **tabella** se nella definizione della tabella sono rispettate le seguenti condizioni:

1. ogni colonna della tabella ha una intestazione (detta attributo) e le intestazioni delle colonne sono diverse fra di loro;
2. i valori di ogni colonna (ovvero di ogni attributo) sono omogenei fra di loro;
3. le righe sono diverse fra loro;
4. l'ordinamento tra le righe è irrilevante;
5. l'ordinamento tra le colonne è irrilevante.

Come detto in precedenza, il modello relazionale è basato sui valori: i riferimenti fra varie tabelle diverse (quali ad esempio autori e libri, dell'esempio precedente, sono realizzati mediante valori). Questo ha i seguenti vantaggi:

1. indipendenza dalle strutture fisiche di memorizzazione;
2. si rappresenta solo ciò che è rilevante dal punto di vista dell'applicazione;
3. i dati sono portabili più facilmente da un sistema ad un altro;
4. le associazioni fra dati sono rappresentate in modo non direzionale .

Il modello relazionale: Definizioni

Schema di relazione:
: uno schema di relazione è costituito da un nome R con un insieme di attributi A1 , ..., An e si indica con R(A1 ,..., An). Dal punto di vista della tabelle uno schema di relazione è costituito al nome della tabella e dal nome delle singole colonne.

Schema di base di dati:
: uno schema di base di dati è insieme di schemi di relazione R = {R1 (X1), ..., Rk (Xk)}. Dal punto di vista della tabelle, uno schema della base di dati è costituito dagli schemi di tutte le tabelle presenti.

Ennupla :

una ennupla su un insieme di attributi X èuna funzione che associa a ciascun attributo A in X un valore del dominio di A. Dal punto di vista della tabelle, una ennupla è costituita da una riga della tabella (diversa dalla prima riga che contiene le intestazioni delle colonne).

Istanza di relazione :

una istanza di una relazione su uno schema R(X) è costituita da un insieme di ennuple su X. Dal punto di vista della tabelle, una istanza di relazione è costituita dall'insieme delle righe di una tabella.

Istanza di base di dati:

una istanza di una base di dati su uno schema R= {R1 (X1), ..., Rn (Xn)} è costituita da un inisieme di istanze di relazioni r = {r1 ,..., rn}, dove ri è una relazione su Ri . Dal punto di vista della tabelle, una istanza di base di dati è costituita da un insieme di istanze di tabelle, con una istanza per ogni schema di tabella presente nello schema della base di dati.

Esempio

Consideriamo il caso (molto semplificato) della gestione di una biblioteca, dove si vogliano memorizzare delle informazioni sui libri, sugli utenti e sui prestiti. Qui e nel seguito useremo una notazione tabellare dove il nome della tabella è posto immediatamente sopra la medesima. Lo schema della base di dati è costituito dai seguenti tre schemi

Libro Titolo Autore Codice ISBN

Utente Nome Indirizzo Codice Fiscale

Prestito Libro Utente Data Prestito

Uno specifica istanza della base di dati sarà ottenuta inserendo dei valori nelle righe delle celle. Quella qui sotto è un esempio di istanza (i codici ISBN e fiscali sono fittizi).

Libro

Titolo	Autore	Codice ISBN
Siddharta	Hesse	ISBN 1456
La luna e i falò	Pavese	ISBN 1347
Macbeth	*Shakespeare*	ISBN 1237

Utente

Nome	Indirizzo	Codice Fiscale
Rossi	Via Vivaldi	MRZRSSI 134
Bianchi	Via Lulli	ANMBCI 125
Smith	Via Corelli	JOHSMH 167

Libro	Utente	Data Prestito
ISBN 1456	ANMBCI 125	12 Dicembre 2002
ISBN 1237	MRZRSSI 134	3 Gennaio 2003

Esempio

Si noti come nell'esempio precedente, i valori contenuti nella tabella Prestito permettano di identificare univocamente le righe della tabella Libro mediante il codice ISBN e le righe della tabella Utente mediante il codice fiscale. Inoltre i valori contenuti nelle righe della tabella Prestito realizzano le seguenti associazioni fra le righe delle tabelle Libro e Utente

Libro

Titolo	Autore	Codice ISBN
Siddharta	Hesse	ISBN 1456
La luna e i falo'	Pavese	ISBN 1347
Macbeth	Shakespeare	ISBN 1237

Utente

Nome	Indirizzo	Codice Fiscale
Rossi	Via Vivaldi	MRZRSSI 134
Bianchi	Via Lulli	ANMBCI 125
Smith	Via Corelli	JOHSMH 167

Come detto in precedenza, l'uso dei valori è l'unica possibilità che si ha nel modello relazionale per esprimere dei riferimenti.

Chiave

Come visto nell'esempio precedente, spesso è utile poter disporre di un attributo (ad esempio il codice fiscale) che permetta di identificare univocamnente le righe di una tabella. Un tale attributo è una chiave. Con più precisione, possiamo definire una chiave come segue:

1. Una **chiave** è un insieme minimale di attributi che permette di identificare univocamente le ennuple di una relazione (ovvero le righe di una tabella).

Nell'esempio precedente, il Codice ISBN è una chiave per la tabella Libro, perchè non ci sono due libri diversi con lo stesso codice ISBN. Se supponiamo che la nostra base di dati non contenga due libri che abbiano stesso titolo e stesso autore, allora anche l'insieme di attributi {Titolo, Autore} costituisce una chiave. L'insieme {Titolo, Autore, Codice ISBN} invece non è una chiave perchè non è minimale, ovvero se togliamo un attributo da tale insieme l'insieme rimasto permette ancora di identificare le righe della tabella (tecnicamente si dice che {Titolo, Autore, Codice ISBN} è una super-chiave).

Per ogni tabella esiste sempre una chiave: infatti dato che le righe di una tabella sono considerate come un insieme, non possono esistere in una tabella due righe diverse che hanno gli stessi valori per tutte le colonne. Quindi l'insieme di tutti gli attributi (di tutte le intestazioni delle colonne cioè) permette di identificare univocamente ogni riga della tabella. Se un tale insieme è minimale allora è una chiave, se non è minimale conterrà un insieme più piccolo che permette ancora di identificare univocamente ogni riga della tabella. Ripetendo lo stesso ragionamento su tale insieme più piccolo si arriva ad un insieme minimale che costituisce la chiave.

L'esistenza delle chiavi è fondamentale perchè la chiave è l'unico meccanismo che garantisce l'accessibilità a ciascun dato della base di dati e permette di correlare i dati presenti in tabelle diverse (si ricordi che i riferimenti nel modello relazionale sono basati solo sui valori).

Fra tutte le chiavi ne viene scelta una, detta **chiave primaria**: sugli attributi della chiave primaria non sono permessi valori nulli (ovvero tutti i valori devono essere specificati).

I vincoli d'integrità

Per cercare di limitare l'immissione di dati scorretti nella base di dati si usano i vincoli d'integrità . Questi permettono di formulare delle proprietà che devono essere soddisfatte dai dati presenti nella base di dati. I vincoli vengono definiti a livello di schema, ovvero devono essere soddisfatti da tutte le possibili istanze corrette dello schema. Con i vincoli quindi si cerca di modellare delle caratteristiche rilevanti della realtà che vogliamo rappresentare nella base di dati.

Ci sono tre tipi di vincoli di integrità :

1. vincoli su valori (o di dominio)
2. vincoli di ennupla (o di riga)
3. vincoli di integrità referenziale

I vincoli di dominio esprimono delle condizioni sui valori di un singolo attributo (ovvero di una singola colonna) di una singola ennupla (o riga).
Ad esempio, considerando l'attributo Data Prestito nella precedrnte tabella Prestiti possiamo imporre il vincolo che la data sia inferiore (ovvero più vecchia) della data odierna (i DBMS reali di solito offrono particolari rappresentazioni per i valori data, con un opportuno ordinamento sui medesimi)

I vincoli di ennupla esprimono delle condizioni sui valori di una singola ennupla (riga).
Ad esempio, nella precedente tabella Utenti possiamo imporre il vincolo che l'attributo Codice Fiscale contenga una sequenza di caratteri che corrispondono all'attributo Nome secondo una certa regola .

I vincoli di integrità referenziale infine permettono di correlare i dati in tabelle diverse. Più precisamente, un vincolo di integrità referenziale fra gli attributi X di una relazione R1 e un'altra relazione R2 impone ai valori su X in R1 di comparire come valori della chiave primaria di R2. Ad esempio, nelle precedenti tabelle Prestito e Utenti possiamo imporre il vincolo che i valori dell' attributo Utente che compaiono nella tabella Prestito compaiano anche come valori dell'attributo Codice Fiscale nella tabella Utenti. Questo fa sì che non si possano inserire delle informazioni relative al prestito per un utente non registrato.

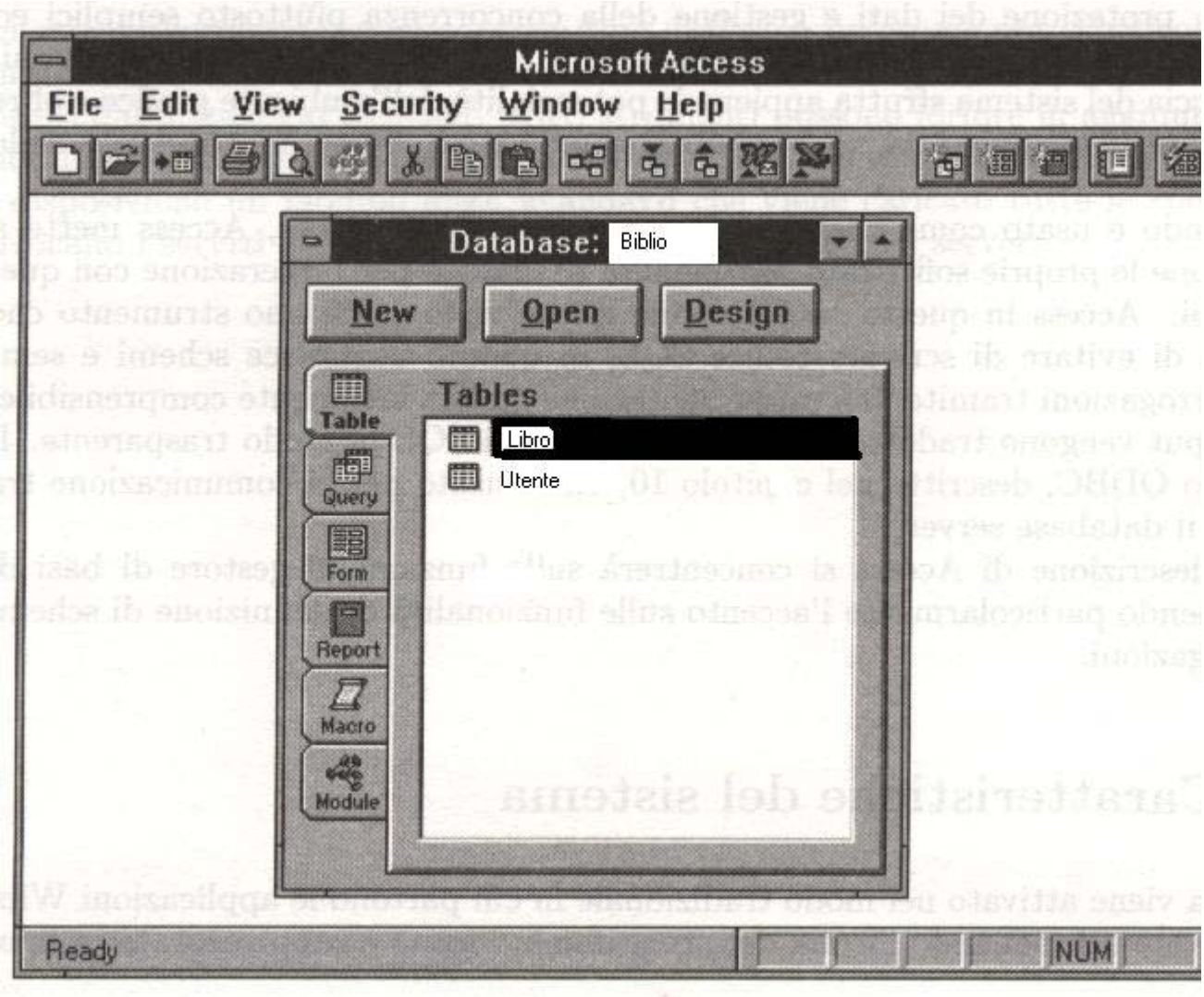
Come si crea una base di dati

Tralasciando la fase (in realtà molto importante) della progettazione concettuale che dovrebbe procedere la fase di effettiva realizzazione di una base di dati, la progettazione di una base di dati relazionale avviene definendo le tabelle specifiche per la particolare applicazione che si vuole realizzare e quindi gli eventuali vincoli di integrità che vogliamo imporre. Questa fase, detta progettazione logica, fa solo riferimento al modello dei dati usato (quello relazionale nel nostro caso) e prescinde dal particolare DBMS usato.

Una volta definite le tabelle si passa alla loro realizzazione fisica usando il DDL (*Data Definition Language*) dello specifico DBMS che useremo. Tutti i moderni DBMS offrono un linguaggio di definizione dei dati con interfaccia grafica, di uso relativamente semplice.

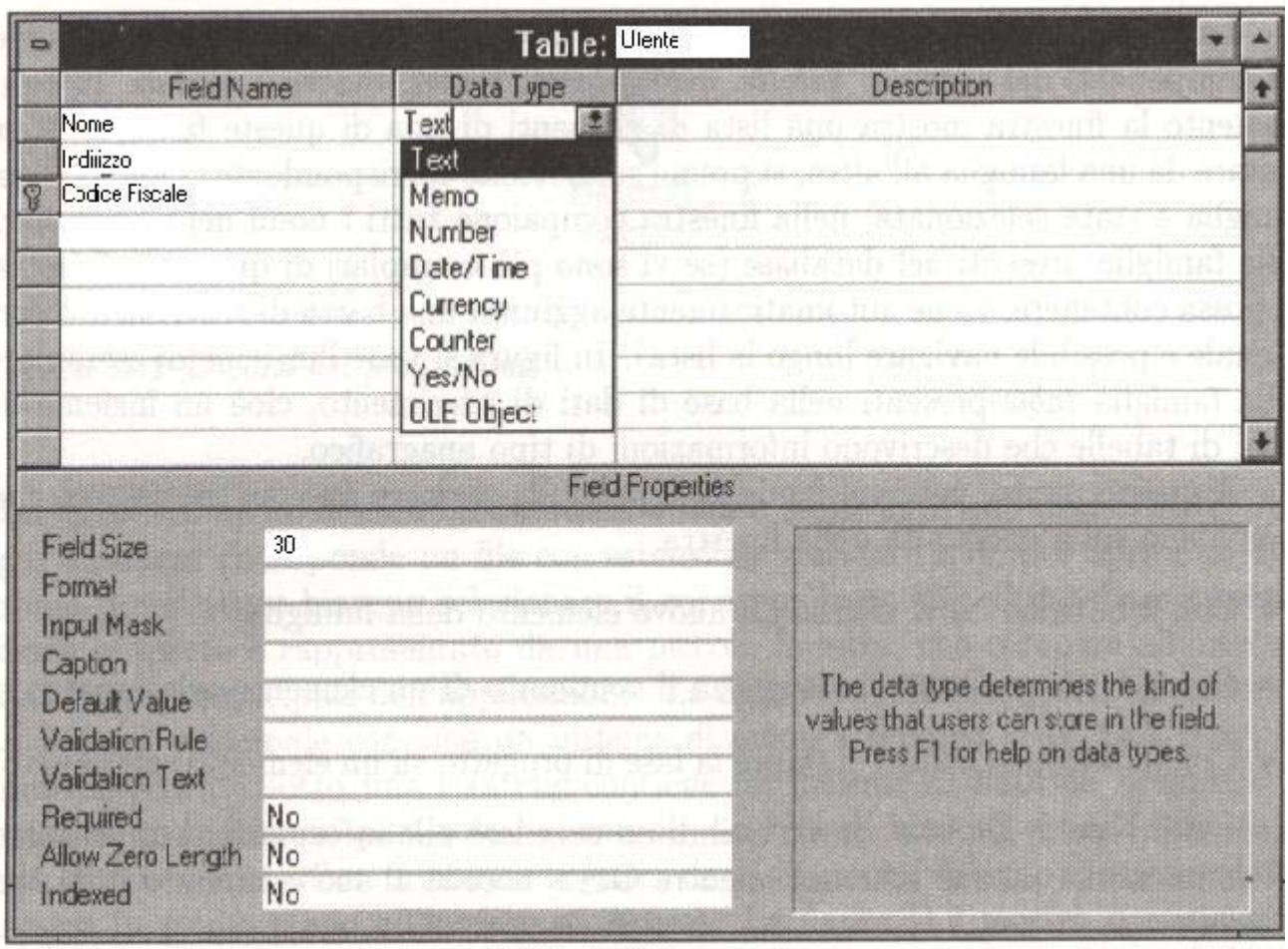
Come si crea una base di dati in Access

In *Access*, una volta fatto partire il programma e creata una nuova base di dati con il comando *New Database* del menu *File*, otteniamo una finestra (come quella nella figura seguente) nella cui parte sinistra abbiamo un elenco dei principali componenti di sistema: *Table , Query, Form, Report, Macro* e *Module*, mentre nella parte superiore abbiamo tre comandi *New, Open* e *Design* che servono per creare un nuovo componente, per visualizzare il contenuto di un componente e per modificare il progetto di un componente (lo schema, se si tratta di tabelle), rispettivamente.



Selezionando *Table* e dando quindi il comando *New* nella finestra sopra mostrata otteniamo la finestra mostrata nell'immagine seguente che permette di definire lo schema della tabella, come specificato più avanti..

Per riempire di dati una tabella, una volta che questa sia già stata definita, basta selezionare nella finestra iniziale la tabella voluta e quindi usare il comando *Open*. Si otterrà una tabella (con i nomi degli attributi come intestazioni) nella quale si potranno inserire i dati riga per riga usando il cursore (spostando il cursore fuori da una riga si indica che l'inserimento per quella riga è terminato). Se i dati inseriti non rispettano i vincoli imposti al momento di definizione della tabella verrà impedito l'inserimento e visualizzato un opportuno messaggio di errore.
Ovviamente si possono riempire le tabelle anche in modo non manuale, usando un opportuno programma che prelevi ad esempio i dati da un file.



Definizione delle tabelle in Access

Dalla finestra vista in precedenza possiamo definire gli attributi (cioè le intestazioni delle colonne) specificandone il nome ed il tipo.

Il tipo fornisce una sorta di primo vincolo di integrità di dominio, in quanto permette di specificare il fatto che i valori per un certo attributo possono esser numerici, testuali ecc. Si possono specificare ulteriormente le caratteristiche di un tipo utilizzando la parte in basso della finestra, parte che varia a seconda del tipo selezionato.

Per inserire il nome di un attributo basta digitarlo nella casella opportuna, mentre per inserire i tipi si possono utilizzare i menu a tendina. Si noti che nel gergo di *Access* un attributo è detto *Field*, cioè campo, e nella rappresentazione dello schema di una base di dati (così come nella fase di definizione del medesimo) gli attributi sono elencati su una stessa colonna (su righe diverse) invece che su una stessa riga come nel modello a tabelle che abbiamo visto.

Dopo avere definito gli attributi per completare la definizione dello schema della tabella occorre indicare quali di essi costituiscono la chiave primaria. Questo si può fare usando il comando *Set Primary Key* nel menu *Edit*, oppure premendo il bottone che rappresenta un chiave nella barra degli strumenti, dopo aver selezionato (con il mouse) gli attributi rilevanti.

Infine, usando il comando *Table Properties* nel menu *View*, si possono specificare dei vincoli di integrità di ennupla (*Validation Rule*) e l'eventuale messaggio che deve essere visualizzato se un vincolo di integrità e violato (*Validation Text*).

NOTA: per specificare i vincoli di integrità si usa sostanzialmente la logica proposizionale, ovvero si usano proposizioni booleane (cioè proposizioni che possono essere vere o false) e operatori logici

(*AND*, *OR*, *NOT* ecc). La sintassi che si usa per esprimere i vincoli è la stessa usata per esprimere le condizioni nel linguaggio delle interrogazioni (QBE, vedi più avanti). è disponibile un menu che facilita la scrittura di tali condizioni.

Tipi e parametri in Access

Fra i tipi che *Access* permette ricordiamo i seguenti:

Text:
testo, ovvero sequenze di caratteri;
Number:
rappresenta l'insieme dei tipi numerici, che si possono specificare ulteriormente utilizzando la parte in basso della finestra;
Date/Time:
tipi che permettono di rappresentare date ed intervalli temporali;
Counter:
è un tipo che associa automaticamente ad ogni riga della tabella un valore numerico che viene incrementato per ogni nuova riga inserita;
OLE *Object*:
rappresenta un generico oggetto (testo, immagine, informazione multimediale) che può essere gestito tramite OLE (*Object Linking and Embedding*). Come già detto nella parte introduttiva generale, gli oggetti OLE sono gestiti dall'applicazione che li creati, ovvero quando viene selzionato un oggetto OLE viene invocata l'applicazione che lo ha creato.

Fra i parametri che si possono specificare per ogni attributo (utilizzando la parte in basso della finestra) ricordiamo :

Field size:
dimensione dell'attributo (per i tipi *Text* e *Number*). Per *text* la dimensione specifica il numero massimo di caratteri che si possono memorizzare (1 carattere = 1 byte) mentre per *Number* possiamo specificare
Byte:
interi su 1 byte
Integer:
interi su 2 byte
Long Integer:
interi su 4 byte
Single:
permette di memorizzare numeri reali in virgola mobile su 32 bit, single fa riferimento alla precisione (singola o doppia)
Double:
come sopra con 64 bit (doppia precisione).
Format:
permette di descrivere vari formati di visualizzazione degli attributi
Default Value:
permette di inserire un valore di default che verrà dato all 'attributo automaticamente se l'utente non ne specifica alcuno.
Validation Rule:
permette di definire un vincolo di integrità di dominio per l'attributo in questione.
Required:
con questo parametro si specifica se l'immissione di un valore per l'attributo è obbligatoria o no.
Indexed:
con questo parametro si specifica se deve essere costruito un indice sull'attributo. Un indice è

una struttura dati ausiliaria che permette un accesso efficiente ai dati e che, di norma, è costruita sugli attributi che costituiscono la chiave primaria.

Le interrogazioni e i relativi linguaggi

Lo scopo principale di una base di dati è quello di permettere interrogazioni, dette anche *query*, con le quali accedere al contenuto della basi di dati stessa.
Le interrogazioni si possono esprimere usando un opportuno DML (*Data Manipulation Language*). Di norma le interrogazioni sono definite dai progettisti della base di dati e l'utente finale le usa attraverso delle maschere nella quale vengono immessi i parametri in base ai quali verrà fatta l'interrogazione (come, ad esempio, quando si consulta un orario ferroviario on-line). Non è comunque troppo difficile formulare proprie *query* o modificarne di esistenti, usando un DML (specie se si tratta di linguaggi con interfacce grafiche). Nel seguito vedremo quindi le caratteristiche essenziali dei DML relazionali e quindi il linguaggio QBE usato in *Access*.

Nell'ambito delle basi di dati relazionali possiamo distinguere i seguenti formalismi principali per esprimere interrogazioni:

1. formalismi teorici quali algebra relazionale e calcolo relazionale;
2. SQL (*Structured Query Language*): il più diffuso linguaggio per DBMS relazionali;
3. QBE (*query by example*): linguaggi di tipo grafico abbastanza semplice (il QBE è usato ad esempio in *Access*).

Tutti questi formalismi sostanzialmente permettono di manipolare relazioni usando opportuni operatori. Inoltre, gli operatori principali che si usano sono gli stessi in tutti i formalismi, ovvero sono gli operatori dell'algebra relazionale.

Algebra relazionale

L'**algebra relazionale** è costituita da un insieme di operatori su relazioni (ovvero tabelle) che producono come risultati altre relazioni (o tabelle). Questi operatori possono essere composti per formare espressioni complesse analogamente a quello che possiamo fare con gli operatori aritmetici. Nel seguito, per non appesantire la notazione, useremo spesso il termine relazione (o tabella) come sinonimo di istanza di relazione (o istanza di tabella).

Gli operatori dell'algebra relazionale sono i seguenti

1. operatori insiemistici (unione, intersezione, differenza)
2. ridenominazione
3. selezione
4. proiezione
5. *join*

Gli operatori insiemistici semplicemente permettono di applicare le usuali operazioni insiemistiche alle tabelle, visto che quest'ultime possono essere considerate come insiemi (ricordiamo che una relazione è un insieme di ennuple, ovvero una tabella è costituita dall'insieme delle sue righe). Si noti che si possono applicare gli operatori insiemistici solo a tabelle definite sugli stessi attributi (ovvero che hanno le stesse intestazioni).

Ridenominazione

L'operatore di **ridenominazione** è un operatore unario (con un solo operando) che modifica lo schema della tabella, ridenominando il nome di alcuni attributi (cioè alcune intestazioni) e lascia inalterata l'istanza dell'operando.

Ad esempio, facendo riferimento alle tabelle dell'esempio della biblioteca viste in precedenza, possiamo applicare la ridenominazione alla tabella Prestito cambiando il nome di attributo Utente in Codice Utente. Il risultato di questa operazione sarà un nuova tabella con le stesse righe della tabella originaria, salvo la riga di intestazione che sarà cambiata come detto. Quindi il risultato dell'operazione

Ridenomina{Utente -> Codice Utente} (Presito)

sarà la tabella

Libro	Codice Utente	Data Prestito
ISBN 1456	ANMBCI 125	12 Dicembre 2002
ISBN 1237	MRZRSSI 134	3 Gennaio 2003

Selezione

L'operatore di **selezione** è un operatore unario che, applicato ad un tabella X, produce come risultato una tabella che ha lo stesso schema di X e che contiene solo quelle righe di X che soddisfano una determinata condizione (specificata dall'operazione di selezione stessa).

Ad esempio, sempre facendo riferimento alla tabella Prestito, potremmo voler selezionare solo le righe per cui la Data Prestito è maggiore del 1 Gennaio 2003 (supponendo di avere un opportuno operatore di confronto fra date). Una tale operazione, che possiamo indicare con

Seleziona(Data Prestito > 1 Gennaio 2003) (Prestito)

produce come risultato la tabella

Libro	Utente	Data Prestito
ISBN 1237	MRZRSSI 134	3 Gennaio 2003

mentre

Seleziona(Autore = Hesse *OR* Autore = Pavese) (Libro)

produce come risultato la tabella

Titolo	Autore	Codice ISBN
Siddharta	Hesse	ISBN 1456
La luna e i falò	Pavese	ISBN 1347

Proiezione

L'operatore di **proiezione** è un operatore unario che, applicato ad un tabella X, produce come risultato una tabella che ha uno schema costituito da un sottoinsieme degli attributi della tabella X (quali attributi devono rimanere è specificato nell'operazione, ovviamente). Il risultato inoltre contiene tutte le righe di X , limitatamente agli attributi (ovvero alle colonne) che sono rimasti.

Ad esempio, facendo riferimento alla tabella Utente, se non ci interessa il codice fiscale possiamo fare una proiezione di tale tabella sugli attributi Nome e Indirizzo. Una tale operazione, che possiamo indicare con

Proietta (Nome, Indirizzo) (Utente)

produce come risultato la tabella

Nome	Indirizzo
Rossi	Via Vivaldi
Bianchi	Via Lulli
Smith	Via Corelli

Quindi, mentre l'operatore di selezione opera una decomposizione orizzontale, l'operatore di proiezione effettua una decomposizione verticale.

Join

L'operatore di ***join*** è l'operatore più importante dato che è quello che permette di mettere in relazione dati in tabelle diverse. Diamo qui una definizione semi-formale per un operatore di *join* che è detto usualmente theta-*join* e che di solito in algebra relazionale è visto come operatore derivato dall'operatore di *join* naturale. Preferiamo definire il theta-*join* perchè questo è l'operatore più rilevante dal punto dal vista pratico.

Supponiamo di avere due relazioni A, con attributi X, e B, con attributi Y, dove X e Y si suppongono disgiunti (ovvero non ci sono attributi in comune: questo è facilmente ottenibile per mezzo della ridenominazione, inoltre nei sistemi reali, i nome di attributi di tabelle diverse sono considerati diversi). Il risultato dell'operazione

join Cond(A,B)

è una tabella definita sugli attributi X,Y, le cui righe sono ottenute come segue:
si costruisce il prodotto cartesiano di A x B, ovvero si costruiscono le righe ottenute concatenando le righe di A e quelle di B in tutti i modi possibili. Da queste righe si selezionano quindi quelle che soddisfano la condizione Cond e solo queste appariranno nel risultato finale.

Di solito la condizione è una eguaglianza fra attributi di A e B, nel qual caso si parla di equi-*join*.
Vediamo un esempio (sempre con riferimento all'esempio della biblioteca).
Se facciamo il *join* della tabella Utenti e della tabella Prestito, specificando la condizione CodiceFiscale = Utente, colleghiamo le informazioni sugli utenti a quelle sui prestiti in modo tale da sapere quali utenti hanno preso in prestito dei libri ed in quale data. Dunque facendo

Utente *join* (CodiceFicale = Utente) Prestito

otteniamo la tabella

Nome	Indirizzo	Codice Fiscale	Libro	Utente	DataPrestito
Rossi	Via Vivaldi	MRZRSSI 134	ISBN 1237	MRZRSSI 134	3 Gennaio 2003
Bianchi	Via Lulli	ANMBCI 125	ISBN 1456	ANMBCI 125	12 Dicembre 2002

si noti che alcune righe non contribuiscono al risultato (ad esempio la riga dell'utente *Smith*) in quanto non soddisfano la condizione. Esiste la possibilità di inserire anche queste ennuple nel risultato usando il *join* esterno che estende, con valori nulli, le ennuple che nel Theta-*join* normale non sarebbero considerate. Esistono tre versioni del *join* esterno:

sinistro:
: mantiene tutte le ennuple del primo operando, estendendole con valori nulli, se necessario

destro:
: come sopra per il secondo operando ;

completo:
: come sopra per entrambi gli operandi.

NOTA: Nei sistemi reali di solito il nome di una attributo è preceduto dal nome della tabella e da un punto (ad esempio, Utente.Nome indica l'attributo Nome della tabella Utenti). Questo fa si che attributi di tabelle diverse siano considerati diversi. Noi omettiamo il nome della tabella perchè tutti i nomi di attributo usati nelle tabelle degli esempi sono diversi.

Interrogazioni

Usando gli operatori visti in precedenza è possibile formulare interrogazioni anche complesse. Vediamo qualche esempio, sempre facendo riferimento al nostro esempio della biblioteca.

1) Vogliamo trovare mentre tutti i titoli dei libri (presenti nella nostra base di dati) il cui autore è Pavese.
Dovremo quindi selezionare le righe della tabella Libro, specificando nella condizione che l'autore deve essere Pavese; inoltre, dato che ci interessa solo il titolo, dovremmo fare una proiezione sull'attributo Titolo. Dovremmo quindi fare

Proietta{Titolo} (Seleziona(Autore = Pavese) (Libro))

dove gli apici indicano il fatto che Pavese è un valore e non un nome. Il risultato sarà la tabella

Titolo
La luna e i falò

2) Vogliamo adesso visualizzare le date in cui Rossi ha preso in prestito dei libri. Come prima cosa dovremo selezionare la riga che riguarda l'utente Rossi nella tabella Utente (visto che ci interessa solo lui). Poi, come visto in precedenza, per poter collegare le informazioni presenti nella tabella Utente con quelle nella tabella Prestito dobbiamo fare un *join*. Infine dovremo fare una proiezione per restringerci agli attributi che ci interessano. Dunque avremo:

Proietta{Nome, DataPrestito} (Seleziona Nome = Rossi (Utente) *join* (CodiceFiscale = Utente) Prestito)

ed il risultato sarà la tabella

Nome	**DataPrestito**
Rossi	3 Gennaio 2003.

3) Infine vogliamo sapere i nomi degli utenti che hanno in prestito dei libri di Hesse ed il titolo di tali libri. Notiamo che adesso, per collegare le informazioni nella tabella Libro con quelle nella tabella Utente, dobbiamo usareanche la tabella Prestito. Quindi dovremo fare due operazioni di *join* e ripetendo per il resto gli stessi ragionamenti di prima avremo:

Proietta{Nome,Titolo} ((Seleziona Autore = Hesse (Libro) *join* (CodiceISBN = Libro) Prestito) *join* (CodiceFiscale = Utente) Utente)

che produrrà come risultato la tabella

Nome	**Titolo**
Bianchi	Siddharta

Le interrogazioni in Access

In *Access* è disponibile un'interfaccia grafica per la formulazione di interrogazioni, detta QBE (*Query By Example*) che permette di esprimere in modo molto semplice ed intuitivo gli operatori dell'algebra relazionale visti in precedenza.

Per definire una nuova interrogazioni (o *Query*) in *Access* dalla solita finestra iniziale si seleziona *Query* e quindi si usa il comando *New*. Questo apre una nuova finestra nella quale è possibile scegliere fra varie modalità di definizione della *query*, in particolare è possibile usare una creazione guidata (usando un cosiddetto *Wizard*) oppure, selezionando *Design View*, è possibile procedere alla definizione diretta della *query* senza alcun aiuto da parte del sistema. Scegliendo questa seconda modalità viene aperta una finestra per la progettazione di una nuova *query* che consiste di due parti. Nella parte superiore si inseriscono (selezionandole da una finestra che appare separatamente) le tabelle rilevanti per la *query* che dobbiamo progettare. Nella parte inferiore della finestra invece vengono visualizzate un insieme di righe e colonne nella quali dovremo inserire gli operatori opportuni per costruire la *query*. Più precisamente, ogni colonna nella parte inferiore sarà usata per definire delle condizioni su un singolo attributo, usando le righe che compaiono e che hanno le seguenti intestazioni (almeno nella versione più semplice):

Field.
: Qui va inserito (per ogni colonna, ovvero per ogni attributo) il nome dell'attributo che ci interessa;

Table.
: Qui va inserito il nome della tabella nella quale compare l'attributo (può anche essere omesso in alcuni casi);

Show.
: Questa riga contiene, per ogni attributo, un campo booleano rappresentato da un quadratino. Cliccando nel quadratino il medesimo viene riempito da una crocetta, il che sta a indicare che l'attributo indicato nella stessa colonna sarà visualizzato nel risultato finale. In pratica questa operazione corrisponde alla Proiezione dell'algebra relazionale.

Sort.
: In questa riga possiamo inserire delle condizioni di ordinamento sui valori dell'attributo della stessa colona.

Criteria.
: Nella celle di questa riga sono inserite le condizioni che devono essere soddisfatte dalle righe del risultato. Ad esempio, se l'attributo indicato nel campo *Field* è Nome, l'attributo indicato nel campo Table è Utenti e (nella stessa colonna) inseriamo nella cella Criteria il valore Rossi, vengono selezionate solo le righe della tabella Utenti per cui il valore dell'attributo Nome è Rossi. Questa riga quindi permettte di esprimere l'operatore di Selezione dell'algebra relazionale. Se vogliamo usare più condizioni in *AND*, ovvero vogliamo indicare più condizioni che devono essere tutte soddisfate dalle righe del risultato, dobbiamo riempire più campi della riga Criteria (riempendo anche le altre righe in modo opportuno, con i nomi degli attributi e delle tabelle giusti). Se non è indicato alcun operatore (come nell'esempio) si intende =, altrimenti possono essere indicati operatori quali >, < etc e possono essere composte condizioni complesse usando gli operatori logici.

Or.
: In questa riga possono essere indicate condizioni aggiuntive che vengono considerate in *OR* rispetto alla riga precedente. Ad esempio nel caso detto sopra, se aggiungiamo anche il valore Bianchi nella cella riga *OR* (stessa colonna), vengono selezionate solo le righe della tabella Utenti per cui il valore dell'attributo Nome è Rossi oppure è Bianchi.

Per modificare invece la struttura di una *query* già definita si usa il bottone Design sempre dalla finestra iniziale.

Join ed esecuzione delle interrogazioni in Access

L'operatore di *Join* dell'algebra relazionale si esprime in *Access* inserendo nella cella Criteria il nome dell'attributo che si vuole eguagliare a quello presente nella riga *Field*. Ad esempio, per effettuare il *join*

Utente *join* (CodiceFiscale = Utente) Prestito

visto in precedenza, dovremo indicare nella parte alta della finestra le tabelle Utente e Prestito, e nella parte inferiore dovremo scrivere

Field	Codice Fiscale
Table	Utente
Sort	
Show	
Criteria	[Prestito.Utente]
OR	

dove la parte in azzurro è quello che appare nella finestra di definizione delle *query* (i nomi di attributi devono essere racchiusi fra parentesi quadre). Si noti che gli attributi che vogliamo che siano visualizzati, come detto in precedenza, devono essere opportunamente indicati nella riga Show. Ad esempio, per effettuare la *query*

Proietta{Nome,Titolo} ((Seleziona Autore = Hesse (Libro) *join* (CodiceISBN = Libro) Prestito) *join* (CodiceFiscale = Utente) Utente)

dovremo indicare nella parte alta le tabelle Libro, Utente e Prestito e nella parte bassa dovremo scrivere

Field	Nome	Titolo	Autore	Libro	Utente
Table	Utente	Libro	Libro	CodiceISBN	CodiceFiscale
Sort					
Show	[X]	[X]			
Criteria			Hesse	[Prestito.Libro]	[Prestito.Utente]
OR					

dove la seconda e la terza colonna corrispondono alla proiezione, la quarta alla selezione, la quinta al primo *join* e la sesta al secondo *join*.

In pratica l'interprete di *Access* per eseguire una *query* effettua il prodotto cartesiano di tutte le tabelle che sono state indicate nella fase di progettazione della *query*, quindi seelziona le righe opportune in base alle condizioni specificate ed infine visualizza le rughe del risultato sugli attributi indicati.

Dato che spesso si usando le stesse operazioni di *join* in più *query*, in *Access* si può indicare un cosidetto *join* predefinito, in modo grafico, direttamente sullo schema della base di dati: selezionando *Relationship* nel menu *Edit* si ottiene una finestra contente gli schemi delle tabelle già definite; qui si possono definire degli archi che collegano due attributi di due tabelle diverse, cliccando con il mouse su un attributo e quindi trascinandolo (tenendo sempre premuto il mouse) fino all'altro. Fatto questo si apre una finestra nella quale è possibile specificare ulteriori caratteristiche del *join* (ad esempio si può indicare un *join* esterno). Una volta dato il comando OK in questa ultima finestra viene inserito un *join* fra le due tabelle. Ad esempio, se abbiamo unito come detto sopra gli attributi Libro della tabella Prestito e Codice ISBN della tabella Libro, tutte le volte che useremo queste due tabelle avremo implicitamente un'operazione di *join* che corrisponde a quella indicata nella quinta colonna dello schema visto poc'anzi.

Una volta che è stata completata la definizione di una *query*, la sua esecuzione avviene premendo il bottone che contiene il punto esclamativo dalla barra degli strumenti, oppure selezionandola dalla finestra iniziale e quindi usando il bottone *Open*.