B DB2 Universal Database

DB2 Universal Database (UDB)[1] appartiene a una storica famiglia di sistemi di gestione di basi di dati prodotti dalla IBM. Il capostipite di questa famiglia è SQL/DS, uno dei primi prodotti commerciali basati sul modello relazionale, reso disponibile dall'IBM agli inizi degli anni Ottanta. A sua volta, il sistema SQL/DS affonda le sue radici in System R, uno dei primi prototipi di DBMS relazionale sviluppato, negli anni Settanta, nei laboratori di ricerca dell'IBM di San José. Proprio nell'ambito dello sviluppo di questo prototipo è nato il linguaggio SQL che, come abbiamo detto nel Capitolo 4, è presto diventato il linguaggio di riferimento per tutti i DBMS commerciali basati sul modello relazionale.

DB2 estende il modello relazionale con un ricco pacchetto di funzionalità avanzate, tra cui:

- un supporto alla gestione di dati multimediali quali testi, immagini, audio e video;
- funzionalità evolute di *business intelligence* per il supporto alle decisioni basate sulla costruzione di *data warehouse* e su operazioni OLAP (i sistemi di data warehouse sono illustrati nel secondo volume [5]);
- la possibilità di gestire dati secondo il paradigma a oggetti e dati semistrutturati basati sul linguaggio XML (anche questi argomenti sono trattati nel secondo volume);
- un supporto completo allo sviluppo di applicazioni distribuite e basate su Web che rende possibile l'accesso a basi di dati attraverso la rete Internet;
- un supporto sia a parallelismi a memoria condivisa (shared-memory), per basi di dati memorizzate su calcolatori a multiprocessori simmetrici (SMP), che a parallelismi a memorie separate (shared-nothing), nei quali una base di dati viene partizionata tra diversi calcolatori connessi in rete (MMP).

La componente server di DB2 è disponibile su varie piattaforme software e hardware, in ambienti Windows e Unix (nelle loro varie versioni). Essa include la componente client per l'accesso locale. Per l'accesso remoto al sistema è disponibile una componente client autonoma per la maggior parte dei sistemi operativi (che comunque non è necessario installare per applicazioni basate su JDBC livello 4). Client remoti possono comunicare con la componente server mediante i più diffusi protocolli di comunicazione (TCP/IP, NetBios ecc.).

Con DB2 è possibile costruire componenti di federazioni distribuite ed eterogenee di basi di dati, usando un protocollo chiamato *Distributed Relational Database Architecture* (DRDA), adottato da diversi DBMS relazionali (le architetture distribuite sono anch'esse trattate nel secondo volume). Inoltre, DB2 fornisce un supporto per

[1] In tutto il capitolo chiameremo questo prodotto semplicemente DB2, sigla con cui è universalmente conosciuto.

i principali standard di interfacce (quali ODBC, JDBC) e aderisce allo standard corrente di SQL.
Nel resto di questa appendice descriveremo le caratteristiche generali del sistema, prestando particolare attenzione agli strumenti di base e ad alcune sue funzionalità avanzate. Per ulteriori approfondimenti su questo sistema si rimanda il lettore al testo di Chamberlin [18], uno degli inventori di SQL.

B.1 Caratteristiche generali di DB2

B.1.1 Versioni del sistema

IBM offre un ricco ventaglio di prodotti per la gestione di dati tra cui: un pacchetto di strumenti per l'integrazione di basi di dati eterogenee (IBM Information Integration), un pacchetto per lo sviluppo di data warehouse (IBM Data Warehouse), un pacchetto per l'OLAP (IBM Business Intelligence), e un pacchetto per il *content management*, ovvero per la gestione di vari tipi di contenuto digitale (IBM Enterprise Content Management). Nel seguito ci occuperemo solo delle funzionalità di *database server* offerte da IBM, ovvero del sistema che si occupa strettamente della gestione di basi di dati, e sebbene venga offerto in varie forme sotto varie denominazioni, utilizzeremo d'ora in poi per la sigla DB2 per fare riferimento a questa componente.

DB2 è disponibile in cinque versioni principali, in grado di gestire architetture di complessità crescente.

- *Everyplace*: è una versione ridotta per dispositivi mobili (palmari, smart phone, portatili con risorse di calcolo limitate), disponibile sui diversi sistemi operativi per questa fascia di calcolatori (Palm OS, Symbian, Windows Mobile). Occupa solo 200K di memoria e fornisce le funzionalità di base del sistema.
- *Personal* (PE): è una versione monoutente e monoprocessore, ma completa, disponibile in ambienti Windows e Linux. Consente la creazione e la manipolazione di basi di dati alle quali possono accedere più applicazioni che però devono risiedere localmente. Questa versione può essere anche usata come client per l'accesso a una componente server remota.
- *Workgroup* (WSE): consente l'accesso condiviso a basi di dati locali da parte di utenti/applicazioni locali e remoti ed è disponibile in ambienti Windows, Linux e Unix. È anche in grado di sfruttare le potenzialità di parallelismo di un calcolatore che contiene fino a quattro processori (SMP a quattro vie).
- *Express*: è una versione di facile installazione e semplificata rispetto alla WSE, che offre funzionalità di amministrazione automatica; è adatta allo sviluppo di applicazioni persistenti che richiedano una amministrazione minima della base di dati. Una configurazione entry-level di questa versione chiamata *Express-C* è distribuita gratuitamente da IBM.
- *Enterprise*: è la versione completa del sistema che, oltre alle funzionalità delle altre versioni, consente la gestione di federazioni di basi di dati e il partizionamento di una base di dati tra diversi calcolatori connessi da una rete di comunicazione, ognuno dei quali può essere dotato di più processori (architetture SMP e MPP).

Il sistema è infine completamente integrato con i diversi ambienti di sviluppo commercializzati da IBM (Eclipse, VisualAge, WebSphere e Rational) che, insieme a DB2, offrono un supporto per l'intero ciclo di vita di applicazioni software realizzate con i più diffusi linguaggi di programmazione (C, C++, Java ecc.).

B.1.2 Istanze e schemi di DB2

Su un medesimo calcolatore è possibile definire diverse *istanze*[2] indipendenti di server DB2, a ognuna delle quali viene assegnato un nome. Ogni istanza ha una propria configurazione e può gestire diverse basi di dati che restano di proprietà dell'istanza. È possibile in questa maniera adattare il sistema a specifiche necessità applicative; per esempio, si può definire e configurare un'istanza DB2 per applicazioni operative e un'altra, con parametri di configurazione diversi, per applicazioni di supporto alle decisioni. Esiste comunque un'istanza predefinita, creata durante l'installazione, che si chiama semplicemente DB2. Le basi di dati di una istanza sono organizzate in *schemi* aventi un nome e costituiti da collezioni di tabelle. Lo schema di default nel quale vengono inserite nuove tabelle ha lo stesso nome dell'amministratore DB2.

I client DB2 possiedono una lista delle istanze e delle basi di dati DB2 alle quali possono accedere e sono dotati di strumenti per creare nuove istanze, nuovi schemi e nuove basi di dati. Per interagire con il server, essi devono prima accedere ad una istanza di DB2 e successivamente stabilire una connessione con una base di dati di questa istanza. Un client può connettersi contemporaneamente a diverse basi di dati. Si possono poi inviare comandi DB2 sia a livello di istanza (per esempio per creare una nuova base di dati) che a livello di base di dati (tipicamente un'istruzione SQL). Nelle istruzioni SQL si può far riferimento alla tabella di uno schema anteponendo al nome della tabella il nome dello schema, separati da un punto. Se il nome dello schema non è specificato, allora ci si riferisce allo schema di default.

B.1.3 Interazione con DB2

Come in tutti sistemi di gestione di basi di dati moderni, l'accesso a una base di dati DB2 può avvenire secondo due modalità principali.

- In maniera interattiva, nella quale si inviano, tramite una opportuna interfaccia (comunemente detta *interfaccia utente*), comandi o istruzioni SQL che vengono immediatamente eseguiti dal sistema. Esistono sia una versione semplice dell'interfaccia, puramente testuale, che una versione grafica evoluta per ambienti Windows. Per l'amministrazione del sistema (creazione di istanze, basi di dati, schemi e gestione di autorizzazioni, prestazioni ecc.) si ricorre nella maggior parte dei casi a questa modalità di interazione.

[2] Il termine "istanza" qui usato denota una installazione di DB2 e quindi non ha niente a che vedere con il concetto di istanza di base di dati introdotto nel Paragrafo 1.3.1.

- Tramite lo sviluppo di programmi in linguaggi di programmazione tradizionali, nei quali vengono immerse istruzioni SQL. È possibile sviluppare, secondo questa modalità, sia programmi statici, nei quali la struttura delle istruzioni SQL è nota a tempo di compilazione, sia programmi dinamici (Capitolo 6), nei quali le istruzioni SQL vengono generate a tempo di esecuzione.

A differenza di altri sistemi di gestione di basi di dati, DB2 non offre un linguaggio 4GL, cioè un linguaggio di sviluppo ad-hoc. Se da un lato questa scelta comporta la necessità di dover disporre, oltre che di DB2, anche di opportuni compilatori per lo sviluppo di applicazioni per basi di dati, dall'altra favorisce la realizzazione di software facilmente portabile da un sistema a un altro.

Nel prossimo paragrafo descriveremo, con maggior dettaglio, come si gestisce una base di dati DB2, secondo le suddette modalità.

B.2 Gestione di una base di dati con DB2

B.2.1 Strumenti per la gestione interattiva

La maniera più semplice per interagire con DB2 è attraverso la sua interfaccia utente. Questo avviene tipicamente in una architettura client-server classica nella quale una base di dati che risiede su un server (per esempio su una macchina Unix) viene acceduta dall'utente tramite un client locale (sullo stesso computer sul quale risiede la base di dati) o remoto (per esempio su un personal computer in ambiente Windows). L'interfaccia utente mette a disposizione diversi strumenti interattivi che sono classificati come segue.

- Strumenti di gestione generale (consentono l'amministrazione di una base di dati mediante un'interfaccia grafica di facile uso).
  - Il *Centro di controllo* è lo strumento principale e che tipicamente si invoca all'avvio del sistema. Fornisce una interfaccia per le operazioni di amministrazione più importanti quali la creazione di basi di dati e di tabelle. Dal centro di controllo è possibile poi invocare tutti gli altri strumenti.
  - Il *Giornale* tiene traccia di tutti le note informative prodotte dal sistema, inclusi i diagnostici e i messaggi di errore. È utile per verificare la presenza di un problema nel sistema.
  - Il *Centro attività* consente di pianificare attività che il sistema esegue successivamente in maniera automatica, per esempio dei backup periodici.
  - Il *Centro di replica* permette la creazione e la gestione di copie di una base di dati, che vengono tenute aggiornate in maniera automatica dal sistema.
- Strumenti riga comandi (consentono di inviare istruzioni SQL o comandi di sistema).
  - Il *Command Line Processor* (CLP) è un ambiente puramente testuale ed è disponibile su tutte le piattaforme.
  - L'*Editor di comandi* è invece dotato di un'interfaccia grafica ed è disponibile per sistemi operativi Windows.

- Strumenti di sviluppo (consentono di realizzare piccole procedure per una basi di dati).
  - Il *Centro di sviluppo* consente la definizione e il testing di oggetti applicativi quali procedure (le stored procedure, descritte nel Capitolo 6), tipi utente e funzioni utente (che verranno descritte nei Paragrafi B.3.3 e B.3.4).
  - Gli *Strumenti di distribuzione progetto* permettono di esportare su altre basi di dati, eventualmente remote, oggetti applicativi definiti nel Centro di sviluppo.
- Strumenti di informazione.
  - Il *Centro di informazioni* fornisce l'accesso rapido a tutti i manuali DB2.
  - La *Verifica aggiornamenti di DB2* serve a tenere aggiornato il sistema.
- Strumenti di controllo (per il monitoraggio del sistema).
  - Il *Centro di controllo stato* consente di individuare potenziali fonti di errore, per esempio l'eccessivo uso di memoria principale, quando certi indicatori superano soglie di "guardia".
  - L'*Analizzatore di eventi* memorizza tutti gli eventi che si verificano sul sistema e consente di monitorare il corretto funzionamento del sistema.
  - Il *Controllo attività* fornisce informazioni relative alle prestazioni del sistema e all'uso delle risorse.
  - Il *Memory Visualizer* genera grafici relativi all'uso della memoria da parte del sistema.
  - Il *Indoubt Transaction Manager* consente di analizzare transazioni a due fasi che si trovano in stati di incertezza (indoubt), cioè transazioni preparate ma per le quali non è stato effettuato il commit o il rollback (le transazioni sono descritte nel Paragrafo 5.4 e approfondite nel secondo volume).
- Strumenti di configurazione.
  - L'*Assistente di configurazione* consente di configurare facilmente il sistema per la connessione a basi di dati remote.
  - I *Primi passi* che fornisce un tutorial sull'uso del sistema.
  - La *Registrazione Visual Studio Add-in* che permette l'installazione di un plug-in in Visual Studio che consente l'invocazione diretta di DB2 da questo ambiente.
- Altri strumenti, di vario tipo.
  - Il *Centro licenze* fornisce informazioni sul tipo di licenza installata sul computer.
  - Il *Centro di gestione satelliti* consente la gestione di satelliti, ovvero di gruppi di diversi server DB2 con configurazioni simili sui quali vengono eseguite le medesime applicazioni.
  - L'*SQL assist* offre un supporto in linea per la scrittura di istruzioni SQL.

Nel seguito vengono illustrati con maggior dettaglio gli ambienti che consentono la gestione di base del sistema.

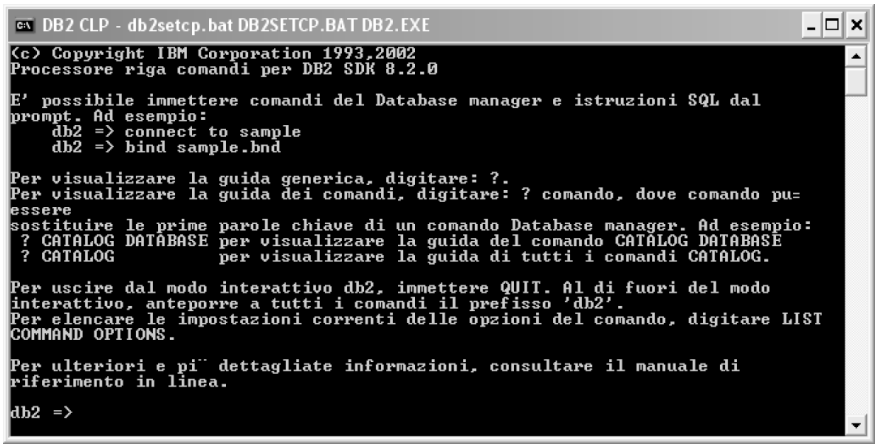

Figura B.1 Il Command Line Processor di DB2

Command Line Processor La versione spartana ma completa dell'interfaccia utente di DB2 è costituita da un ambiente testuale chiamato *Command Line Processor* (CLP) ed è disponibile su tutte le piattaforme. Il CLP si può invocare a livello di sistema operativo con il comando `db2`. Un esempio di interazione con il CLP è riportato in Figura B.1. Il prompt `db2 =>` indica che il sistema è pronto a ricevere comandi. Una volta entrati nell'ambiente si deve avviare il sistema con il comando `db2start` e poi si possono inviare, oltre alle istruzioni SQL, una serie di comandi per la gestione completa del sistema. Il comando inviato viene eseguito immediatamente e il risultato viene mostrato sul video. Se i comandi sono più lunghi di una riga, bisogna segnalare la continuazione sulla riga successiva con il carattere di *backslash* (\). Tutti i comandi possono essere inviati direttamente da sistema operativo premettendo il prefisso `db2`.

Centro di controllo Permette di amministrare una base di dati mediante un'interfaccia grafica che offre la possibilità di specificare operazioni di definizione di tabelle e di vincoli di integrità, controllo delle autorizzazioni, backup, ripristini ecc. Come si può vedere in Figura B.2, l'interfaccia del Centro di controllo ha tre finestre principali. In quella di sinistra vengono presentati, secondo un'organizzazione gerarchica, gli oggetti DB2 (istanze, basi di dati, tabelle ecc.) che l'utente ha a disposizione. Questi oggetti possono essere dichiarati esplicitamente dall'utente, oppure si può richiedere al Control Center di cercare su rete tutti gli oggetti DB2 ai quali è possibile accedere.

Nel nostro caso, si può osservare che è disponibile un sistema con oggetti DB2 chiamato `CARAVAGGIO`. Le informazioni relative a questo sistema sono state espanse, come indicato dal segno '−' davanti alla relativa icona. Su questo sistema risiedono due istanze DB2: una è quella di default, l'altra si chiama `MyDB2`. La prima contiene diverse basi di dati tra cui una chiamata `EMP`. Per questa base

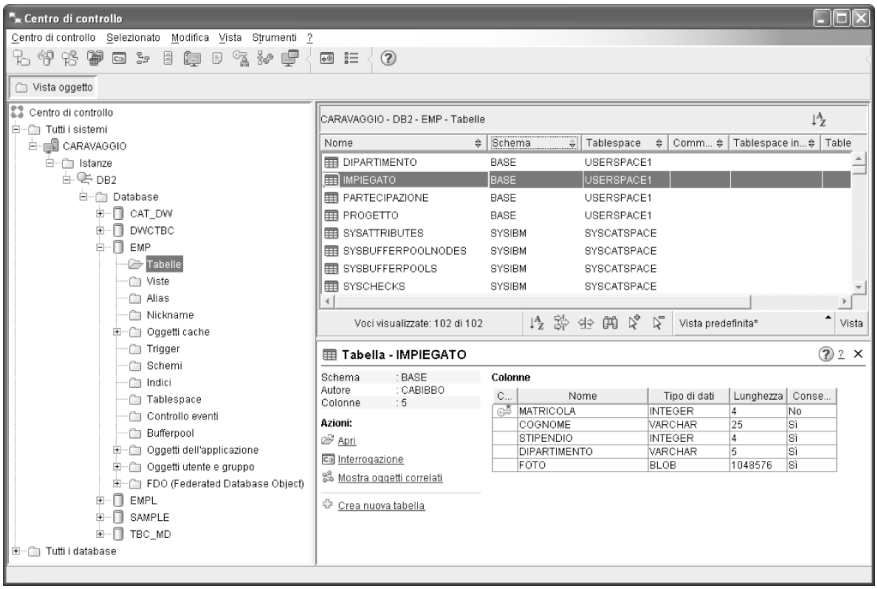

Figura B.2 Il Centro di controllo di DB2

di dati vengono mostrate tutte le sue componenti, mentre le informazioni relative alle altre non sono state espanse, come indicato dal segno '+' davanti alle relative icone. La finestra in alto a destra contiene informazioni dell'oggetto selezionato nella finestra di sinistra. Nel nostro caso, sono mostrate tutte le tabelle della base di dati `EMP`. Questa base di dati contiene due schemi: lo schema `BASE` e lo schema `SYSIBM`. Le tabelle dello schema `SYSIBM` sono tabelle di sistema. Nella finestra in basso a destra vengono infine riportate informazioni sulla tabella selezionata. Nel nostro esempio sono mostrati i dettagli della tabella `Impiegato`. Se si clicca con il bottone di destra del mouse su un qualunque oggetto sullo schermo, compare un menu che consente di effettuare una serie di azioni sull'oggetto puntato. Per esempio, puntando l'icona `Tabelle` di una base di dati, è possibile creare una nuova tabella. Il sistema guida l'utente nell'esecuzione di queste operazioni.

Editor comandi Questo strumento consente di digitare ed eseguire istruzioni SQL e di comporre *script*, ovvero sequenze di istruzioni SQL da eseguire eventualmente in momenti prestabiliti. Può essere invocato dal Centro di controllo cliccando sull'icona `Interrogazione` della finestra in basso a destra. L'Editor comandi è costituito da tre ambienti tra loro integrati: *Comandi*, *Risultati* e *Plan di accesso*; per passare da un ambiente all'altro è sufficiente cliccare sul corrispondente nome. Nel primo è possibile digitare singole istruzioni SQL, script o comandi DB2 di amministrazione. In Figura B.3 viene mostrato un esempio di interrogazione SQL formulata nell'Editor comandi di DB2 (invocato dal Centro di controllo).

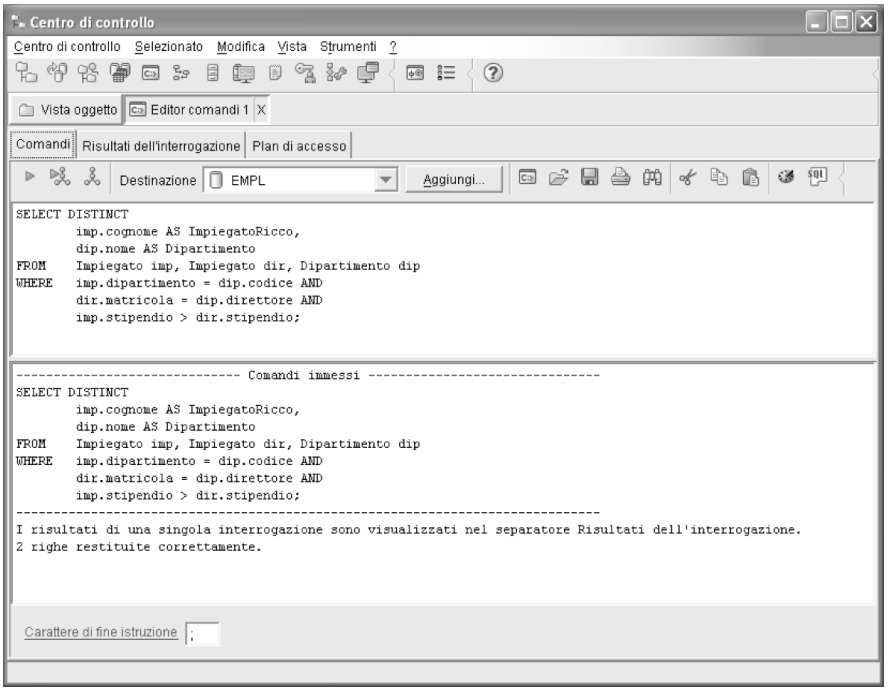

Figura B.3 Una interrogazione SQL fatta con l'Editor comandi di DB2

Si tratta di un'interrogazione che cerca gli impiegati che guadagnano più del rispettivo direttore, in uno schema contenente le tabelle:

IMPIEGATO(Matricola, Cognome, Dipartimento, Stipendio)
DIPARTIMENTO(Codice, Nome, Sede, Direttore)

Per eseguire i comandi digitati è sufficiente cliccare sull'icona di esecuzione, costituita da un triangolo: un messaggio relativo all'esecuzione dell'istruzione (completamento con successo o eventuali diagnostici di errore) viene riportato nella finestra in basso, mentre il risultato viene visualizzato automaticamente nell'ambiente *Risultati dell'interrogazione*. Un esempio del contenuto di questo ambiente dopo aver eseguito l'interrogazione in Figura B.3 viene mostrato in Figura B.4.

Passando dal primo al secondo ambiente è quindi possibile interagire in maniera interattiva con una base di dati. I comandi (o gli script) digitati nel primo ambiente possono essere salvati su un file per successive esecuzioni. Il terzo ambiente visualizza i piani di accesso creati dall'ottimizzatore di DB2 per l'esecuzione di istruzioni SQL digitate nel primo ambiente. I piani di accesso hanno la forma di alberi: le foglie degli alberi corrispondono alle tabelle coinvolte nell'istruzione SQL e i nodi interi a operazioni effettuate dal sistema; per esempio, la scansione di una tabella, un certo tipo di join tra due tabelle, l'ordinamento di un risultato intermedio. La creazione di questi grafi deve essere richiesta esplicitamente in

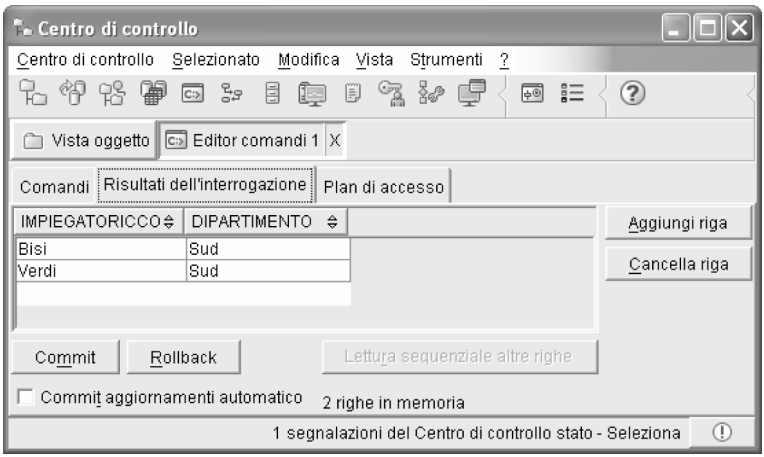

Figura B.4 Il risultato di una interrogazione SQL nell'Editor comandi di DB2

fase di esecuzione al sistema cliccando sull'icona che raffigura un albero. In Figura B.5 viene mostrato una porzione del piano di accesso dell'interrogazione di Figura B.3.

Nella finestra sono indicate delle operazioni di preparazione (scan) che precedono le operazioni di join tra le tabelle DIPARTIMENTO e IMPIEGATO e le operazioni che seguono (sort e scan) per la preparazione del risultato finale. Il valore indicato nei nodi fornisce una stima del costo previsto per l'operazione. Cliccando sui singoli nodi è possibile visualizzare altri dati relativi all'operazione corrispondente, quali stime più dettagliate sui costi di esecuzione e sulla cardinalità dei risultati. L'analisi dei piani di accesso consente di ottimizzare interrogazioni, per esempio introducendo opportunamente degli indici per evitare un'operazione di ordinamento durante la sua esecuzione. Questi aspetti sono illustrati in dettaglio nel secondo volume.

Giornale Costituisce un utile strumento di verifica perché mantiene una traccia di tutte le operazioni svolte su una base di dati. In particolare, è possibile visualizzare con questo strumento l'esecuzione di attività pianificate (pannello `Cronologia attività`), le operazioni di backup/ripristino (pannello `Cronologia database`), tutti i messaggi generati dal sistema in seguito all'esecuzione delle varie operazioni svolte (pannello `Messaggi`) e i diagnostici di errore prodotti dal sistema (pannello `Registrazione notifiche`).

Centro di controllo stato Si tratta di un altro utile strumento che consente di analizzare lo stato del sistema e risolvere preventivamente potenziali problemi. In particolare è in grado di segnalare situazioni "limite", che non costituiscono ancora veri e propri errori, ma possono indicare possibili malfunzionamenti del siste-

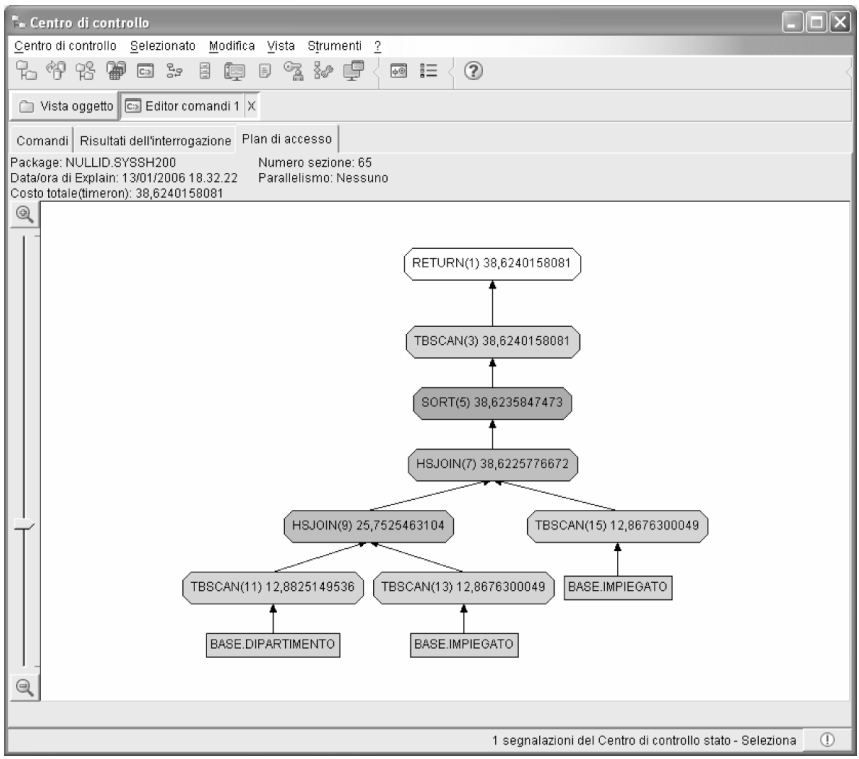

Figura B.5 Il piano di accesso generato da DB2 per una interrogazione SQL

ma. Questi avvisi vengono generati quando alcuni indicatori di stato (per esempio la dimensioni di memoria principale occupata) superano dei valori di soglia predefiniti (detti *threshold*). Un esempio di contenuto del Centro di controllo viene mostrato di Figura B.6.

In questo esempio viene segnalato un eccessivo utilizzo della memoria riservata al controllo (92% del valore massimo) e la necessità di eseguire dei backup. Selezionando un avviso, vengono suggerite delle azioni da intraprendere per risolvere la segnalazione.

Centro informazioni Fornisce un ricco insieme di informazioni sul sistema DB2. Tramite questo strumento è possibile accedere a spiegazioni su come effettuare certe operazioni (per esempio, "creazione di una tabella" o "backup di una base di dati"), a tutorial, ai vari manuali disponibili in linea, a chiarimenti sulla messaggistica di DB2 e sui codici di errore, e a una lista di programmi di esempio che illustrano le varie funzionalità di DB2.

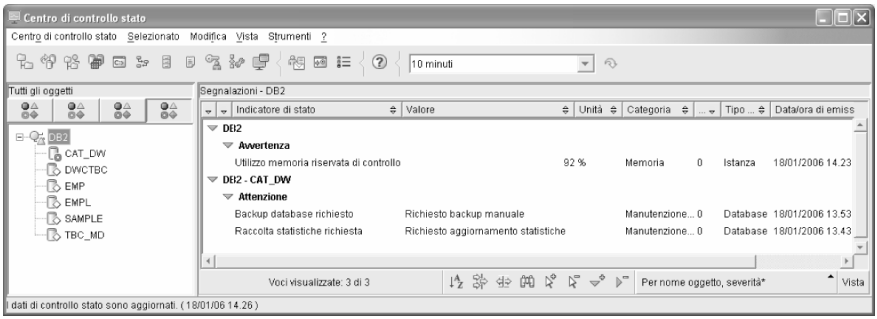

Figura B.6 Il centro di controllo stato di DB2

B.2.2 Applicazioni

DB2 mette a disposizione diversi strumenti per lo sviluppo di applicazioni per basi di dati, nei quali le istruzioni SQL di interazione con la base di dati vengono usate insieme a istruzioni di linguaggi di programmazione tradizionali, detti linguaggi *host*, come C, C++, Java, FORTRAN e COBOL. Come descritto nel Capitolo 6, esistono due modalità principali: l'SQL embedded (statico e dinamico) e l'uso delle Call Level Interface (CLI) quali ODBC e JDBC. Ricordiamo che nell'SQL statico i nomi delle relazioni e degli attributi coinvolti nelle istruzioni SQL sono fissati a priori; l'unica parte delle istruzioni che può rimanere non nota a tempo di compilazione sono gli specifici valori da ricercare o da aggiornare. Nell'SQL dinamico invece, le istruzioni SQL sono generate a tempo di esecuzione; non è quindi necessario specificare relazioni e attributi coinvolti in una istruzione SQL prima della sua esecuzione.

Nel seguito parleremo prevalentemente dell'SQL statico, mentre accenneremo agli strumenti offerti da DB2 per realizzare programmi secondo le altre modalità. Per rendere la trattazione più autocontenuta, vengono ripetute alcune nozioni già presentate nel Capitolo 6.

SQL statico Un esempio completo di un programma SQL statico immerso in C per DB2 è riportato in Figura B.7. Questo programma accede a una base di dati contenente le tabelle IMPIEGATO e DIPARTIMENTO citate nel paragrafo precedente, legge il nome di una località da input e stampa nome e stipendio degli impiegati dei dipartimenti che si trovano in tale località, ordinati per dipartimento.

Si può innanzitutto osservare che il prefisso utilizzato in DB2 per segnalare una istruzione SQL è `exec sql`. Il programma inizia con una serie di direttive di compilazione che includono la libreria di sistema DB2 (chiamata `sqlenv.h`) e il record `sqlca` nel quale vengono memorizzate automaticamente, di volta in volta, informazioni sullo stato del sistema. In particolare, il campo `sqlcode` di `sqlca`

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <sqlenv.h>
exec sql include sqlca;
void main() {
   char  CurrDept[10];              /* Dichiarazione variabile programma */
   exec sql begin declare section;      /* Dichiarazione variabili host */
      char  CognomeImp[10];                        /* Cognome Impiegato */
      char  NomeDip[10];                           /* Nome Dipartimento */
      char  SedeDip[15];                           /* Sede Dipartimento */
      long  StipendioImp;                        /* Stipendio Impiegato */
      char  msgbuffer[500];         /* Buffer per messaggi di errore DB2 */
   exec sql end declare section;
   exec sql declare C1 cursor for              /* Dichiarazione cursore */
        select distinct imp.cognome, imp.stipendio, dip.nome
        from     impiegato imp, dipartimento dip
        where    imp.departmento = dip.codice and
                 dip.sede = :SedeDip
        order by dip.nome;
   exec sql whenever sqlerror go to PrintError;
   exec sql connect to EMP;         /* Connessione alla basi di dati EMP */
   printf("Digita una localita':"); scanf("%s", SedeDip);
   exec sql open C1;                              /* Apre il cursore C1 */
   exec sql fetch C1 into :CognomeImp, :StipendioImp, :NomeDip;
   if (sqlca.sqlcode==100) printf("Nessun dipartimento trovato.\n");
   while (sqlca.sqlcode == 0) {
      if (strcmp(NomeDip,CurrDept)!=0) {
         printf("\nDipartimento: %s\nImpiegato\tStipendio\n",NomeDip);
         strcpy(CurrDept,NomeDip);
      }              /* end if */
      printf("%s\t%d\n", CognomeImp, StipendioImp);
      exec sql fetch C1 into :CognomeImp, :StipendioImp, :NomeDip;
   }               /* end while */
   exec sql close C1;                              /* Chiude il cursore C1 */
   exec sql connect reset;                       /* Rilascia la connessione */
   return;
PrintError:                     /* Trova e stampa un messaggio di errore */
   sqlaintp(msgbuffer, 500, 70, &sqlca);
   printf("Errore DB2 inatteso: %s\n", msgbuffer);
   return;
}               /*  end of main */
```

Figura B.7 Un programma SQL immerso in C per DB2

contiene un numero intero che codifica il comportamento dell'ultima istruzione SQL eseguita.

Fra le dichiarazioni delle variabili, notiamo che ve ne sono alcune racchiuse in una sezione delimitata dalle parole chiave `declare section`. Queste variabili sono dette *host* e sono quelle variabili del programma che possono essere usate nelle istruzioni SQL: tali variabili realizzano l'interfaccia tra programma e base di dati. Si osservi che quando una variabile host è usata in SQL, è preceduta da un prefisso (`:`); questo prefisso serve a distinguere le variabili dai nomi di attributo. L'istruzione `whenever` permette di stabilire il comportamento del programma a seguito dell'esecuzione di ogni istruzione SQL non andata a buon fine. L'istruzione non è quindi associata a una specifica operazione SQL, ma all'intero programma. Tre opzioni sono possibili per questa istruzione: `not found`,

sqlerror e sqlwarning. Nel primo caso si può specificare cosa fare quando il risultato di una interrogazione è vuoto (sqlcode = 100); nel secondo, quando si è verificato un errore (sqlcode < 0); nell'ultimo, quando si è verificato un *warning*, vale a dire la segnalazione di un evento imprevisto ma non grave (sqlcode > 0). Nel nostro programma è stato stabilito che, quando si verifica un errore, il controllo passi a una routine che carica in un buffer il messaggio di errore generato, lo stampa e poi termina il programma. Nulla è stato specificato per risultati nulli e warning.

Prima di interagire con una base di dati, va stabilita una connessione con essa tramite l'istruzione connect to. Successivamente, si possono eseguire liberamente istruzioni SQL i cui effetti sulla base di dati divengono però permanenti solo dopo l'esecuzione dell'istruzione commit. (Il programma di esempio non prevede istruzioni di aggiornamento della base di dati e quindi non fa uso di tale istruzione). Per eseguire le operazioni previste, il programma fa uso della tecnica del cursore presentata nel Capitolo 6. Viene inizialmente definito un cursore su una interrogazione SQL che contiene la variabile host SedeDip. Successivamente, dopo aver memorizzato in questa variabile un dato letto da input, il cursore viene aperto, l'interrogazione eseguita e, tramite l'istruzione fetch, il risultato viene copiato, una tupla alla volta, in variabili di host per poter essere visualizzato. Questa operazione viene coordinata esaminando il valore assunto dalla variabile di stato sqlcode. In particolare, viene visualizzato un messaggio quando il risultato è vuoto (sqlcode pari a 100) e viene ripetuta l'istruzione di copia e stampa finché ci sono ancora tuple da visualizzare (sqlcode pari a 0). Il cursore viene quindi chiuso e la connessione rilasciata.

Per creare una applicazione eseguibile per DB2, il programma va prima precompilato inviando il comando prep <nome file> nel CLP o nel Editor comandi. Il risultato della precompilazione va poi compilato e collegato alle librerie usate secondo le modalità adottate per un qualunque programma scritto nel linguaggio host scelto.

SQL dinamico Nell'SQL dinamico vengono messe a disposizione speciali istruzioni per compiere le seguenti operazioni principali:

- preparazione di un'istruzione SQL con invocazione dell'ottimizzatore DB2 che crea il relativo piano di accesso;
- descrizione del risultato di un'istruzione SQL con specifica del numero e del tipo degli attributi;
- esecuzione di un'istruzione SQL precedentemente preparata, con assegnamento di valori a eventuali variabili usate nell'istruzione;
- caricamento del risultato, una tupla alla volta, in variabili di programma per il loro successivo uso.

Queste operazioni possono essere immerse in linguaggi di programmazione secondo le medesime modalità dell'SQL statico (in particolare mediante l'uso del prefisso exec sql).

La fase di preparazione di una istruzione SQL si realizza con il comando `prepare <nome> from <variabile>`, dove la variabile è di tipo stringa e tipicamente memorizza istruzioni SQL. Questa istruzione SQL può contenere dei punti interrogativi che indicano la presenza di parametri che verranno passati in fase di esecuzione. Nel caso di interrogazioni, la gestione del risultato si realizza con il meccanismo del cursore, come avviene con l'SQL statico.

La fase di descrizione si fonda sull'uso di un *descrittore*, ovvero di una struttura di dati che descrive tipo, lunghezza e nome di un numero variabile di attributi del risultato di un'interrogazione. Il sistema DB2 mette a disposizione a questo scopo il descrittore predefinito `sqlda`, che è un record contenente un numero variabile di campi di tipo `sqlvar`, uno per ogni attributo da descrivere. Il numero di questi campi di tipo viene memorizzato nel campo `sqld` di `sqlda`. I campi `sqlvar` sono a loro volta dei record contenenti, tra l'altro, un campo `sqltype`, che codifica il tipo dell'attributo, un campo `sqllen`, che memorizza la lunghezza dell'attributo, e un campo `sqlname`, che memorizza il nome dell'attributo. Nella fase di preparazione, si può indicare l'uso di questo descrittore con la seguente sintassi: `prepare <nome> into sqlda from <variabile>`. Dopo l'esecuzione di questa istruzione, in `sqlda` viene caricata la descrizione dell'istruzione memorizzata nella variabile specificata. Mediante questa tecnica, è possibile implementare per esempio un'interfaccia utente personalizzata, in grado di accettare ed eseguire istruzioni SQL arbitrarie.

Per la fase di esecuzione e caricamento si possono utilizzare le medesime istruzioni viste per l'SQL statico, compreso l'uso di cursori. Accedendo alle informazioni contenute nel descrittore è possibile generare opportunamente la stampa dei risultati di un'interrogazione.

Call Level Interface DB2 offre delle interfacce CLI (Call Level Interface) basate sugli standard ODBC (Open DataBase Connectivity) e JDBC (Java DataBase Connectivity), illustrate nel Capitolo 6.

Ricordiamo che l'interfaccia basata su ODBC è disponibile per vari linguaggi di programmazione (escluso Java). Tale interfaccia mette a disposizione una serie di funzioni che possono essere direttamente invocate dai programmi per accedere a una base di dati. Per esempio la funzione `sqlconnect()` consente di connettersi a una base di dati DB2, la funzione `sqlprepare()` prepara un'istruzione SQL alla sua esecuzione, la funzione `sqlexecute()` esegue un'istruzione SQL precedentemente preparata e infine la funzione `sqlfetch()` carica una tupla in variabili host del programma.

L'interfaccia JDBC si basa sullo stesso principio della ODBC, ma è dedicata allo sviluppo di programmi in linguaggio Java ed è quindi orientata agli oggetti. In particolare, questa interfaccia è dotata del metodo `executequery(String)` che prende in ingresso una stringa contenente una istruzione SQL e restituisce un oggetto della classe predefinita `ResultSet`, costituito da un insieme di tuple. Questa classe possiede una serie di metodi che permettono di manipolare l'insieme di tuple contenute negli oggetti della classe. Oltre che applicazioni standard Java, questa interfaccia permette di sviluppare *applets*, ovvero programmi che possono essere caricati ed eseguiti da un browser Web abilitato allo scopo. In

questa maniera, basi di dati DB2 possono essere accedute da una qualunque computer connesso a Internet, senza bisogno di installare su di esso la componente client del sistema DB2.

Il grosso vantaggio dei programmi che usano l'interfaccia CLI è che non hanno bisogno di essere precompilati; inoltre, essi possono essere utilizzati su qualunque DBMS che fornisce un supporto per gli standard citati.

B.3 Funzionalità avanzate di DB2

Come accennato all'inizio del capitolo, DB2 offre tutta una serie di funzionalità avanzate. Pur non essendo tutte standardizzate, queste funzionalità ci danno una interessante indicazione delle caratteristiche dei sistemi di gestione di basi di dati reali. Ne citiamo alcune.

B.3.1 Dati complessi

DB2 mette a disposizione tre tipi di dati predefiniti che possono essere usati per memorizzare in una tabella dati complessi (per esempio documenti o immagini), che vengono chiamati genericamente LOB (Large OBjects).

- *Blob (Binary Large Object)*: rappresenta un dato in formato binario, grande fino a due gigabyte. I dati di tipo Blob non possono essere assegnati o confrontati con dati di altro tipo.
- *Clob (Character Large Object)*: rappresenta un dato composto da una sequenza di caratteri di un byte, grande fino a due gigabyte. I dati di tipo Clob possono essere confrontati con dati di tipo stringa (Char e Varchar).
- *Dbclob (Double-Byte Character Large Object)*: rappresenta un dato composto da una sequenza di caratteri di due byte, grande fino a due gigabyte. I dati di tipo Dbclob possono essere usati solo su basi di dati con una configurazione apposita.

Una possibile definizione di tabella che contiene dati di tipo LOB è la seguente.

```
create table Impiegato (
   Codice         integer not null unique,
   Nome           varchar(20),
   Stipendio      decimal(7,3),
   DataAssunzione date,
   Foto           blob(5M) compact,
   Curriculum     clob(500K)
)
```

In questa tabella la colonna Foto conterrà immagini e la colonna Curriculum testi. La dimensione specificata tra parentesi indica un valore massimo (si possono usare i suffissi `K`, `M`, e `G` per indicare kilobyte, megabyte e gigabyte). L'opzione `compact` specifica che il dato LOB dovrà occupare il minimo spazio sul disco, a costo di una minore efficienza nella sua gestione.

Esistono diverse limitazioni nell'uso di dati LOB in istruzioni SQL. In particolare non è possibile confrontare direttamente colonne di tipo LOB mediante operatori come =, >, < o `in`. È però possibile far uso dell'operatore `like`. Per esempio, l'istruzione SQL:

```
select Codice, Nome
from Impiegato
where Curriculum like '%DBA%'
```

trova gli impiegati per i quali compare la stringa 'DBA' nel curriculum.

I dati LOB sono gestiti dal sistema in maniera da minimizzare i loro spostamenti da una locazione di memoria a un'altra. In particolare, i LOB possono essere manipolati nelle applicazioni per mezzo di opportune variabili dette *locatori*. I locatori rappresentano un LOB senza memorizzarlo fisicamente. Usando in maniera opportuna i locatori è possibile differire o addirittura evitare il caricamento di un LOB in un programma. Per esempio, la copia di un LOB avviene semplicemente copiandone il locatore. Un locatore si definisce nella sezione di dichiarazione delle variabili host di un programma con il comando: `sql type is <tipo di LOB> <nome locatore>`. Successivamente, il locatore può essere usato come tutte le altre variabili host per gestire un LOB del tipo associato. È inoltre possibile manipolare locatori con funzioni speciali tra cui `posstr`, che trova la posizione della prima occorrenza di un pattern in un LOB, e `substr`, che restituisce la sottostringa di un LOB compresa tra le posizioni specificate.

B.3.2 Extender

Oltre ai tipi predefiniti di base, DB2 mette a disposizione, come prodotti ausiliari, degli *extender*, ovvero ulteriori tipi di dato, più specifici dei LOB, per la gestione di dati non tradizionali. A ciascun extender sono associate particolari funzioni ausiliarie per la manipolazione dei relativi tipi di dato. Generalmente queste funzioni possono essere utilizzate liberamente in comandi SQL.

Gli extender sono tipicamente sviluppati direttamente dalla IBM ma esistono anche extender sviluppati da partner commerciali della IBM.

Tra gli extenders DB2 più interessanti citiamo i seguenti.

- Gli *AIV (Audio, Image, Video) extenders* per la gestione di dati multimediali. Questi extender offrono funzionalità evolute di manipolazione di dati multimediali (tra cui la la riproduzione di suoni e video) e di ricerca avanzata (per esempio ricerche basate su proprietà dei dati quali la presenza di un certo campione audio o immagine).
- Il *text extender* per la gestione di basi di dati documentali. Questo extender offre funzionalità tipiche dell'*information retrieval* come l'indicizzazione di testi e la ricerca basata su parole chiavi.
- Lo *spatial extender* per la gestione di dati spaziali. Vengono offerti a tale riguardo diversi tipi di dato spaziale che possono essere usati per modellare oggetti del mondo reale quali mappe, confini geografici, corsi di fiumi ecc.

- L'*XML extender* per la gestione di dati in formato XML. Questo extender consente di memorizzare documenti XML in attributi di tabelle relazionali. Le funzioni messe a disposizione permettono di effettuare ricerche su interi documenti XML o su loro porzioni e di trasformare dati XML in basi di dati relazionali e viceversa.
- Il *Net search extender* per la gestione di ricerche testuali evolute. Offre un ricco insieme di funzioni di ricerca su testi e di indicizzazione di documenti memorizzati in una base di dati. Questi strumenti consentono la costruzione di motori di ricerca.

B.3.3 Tipi utente

Un tipo di dato utente (denominato *distinct* in DB2) è un tipo definito a partire dai tipi di dato di base e corrisponde al concetto di definizione di dominio utente introdotto nel Capitolo 4. Per esempio, la definizione dei tipi MONEY, IMAGE e TEXT può essere specificata con le seguenti istruzioni:

```
create distinct type Money as decimal(7,2)
                           with comparisons;
create distinct type Image as blob(100M);
create distinct type Text  as clob(500K) compact;
```

La clausola `as` specifica il tipo *sorgente* del tipo utente, mentre la clausola `with` serve a specificare che i relativi dati vanno confrontati come se fossero dati dei rispettivi tipi sorgente. Si possono usare come tipi sorgente solo tipi di dato DB2 predefiniti e non sono ammesse nidificazioni nelle definizioni.

Una volta definiti, i tipi utente possono essere liberamente usati nelle creazioni di tabelle. Per esempio, la precedente definizione della tabella IMPIEGATO può essere riscritta come segue:

```
create table Impiegato (
   Codice          integer not null unique,
   Nome            varchar(20),
   Stipendio       money,
   DataAssunzione  date,
   Foto            image,
   Curriculum      text
)
```

Sugli attributi definiti su un tipo utente non è in generale possibile applicare le medesime operazioni che si possono applicare sui rispettivi tipi sorgente. Per esempio, non è possibile sommare due dati di tipo Money. Si può però ovviare a questa limitazione con la definizione di opportune funzioni utente, come descritto nel seguito.

B.3.4 Funzioni utente

Le funzioni utente corrispondono alle procedure (o *stored procedure*) introdotte nel Capitolo 6. In DB2 possono essere dichiarate in maniera esplicita mediante l'istruzione `create function` che assegna un nome alla funzione e ne definisce la semantica. Tali funzioni sono associate a una base di dati e possono essere utilizzate solo nel contesto della rispettiva base di dati. Una caratteristica importante delle funzioni utente DB2 è che aderiscono al principio di *overloading* della programmazione orientata agli oggetti. È possibile definire cioè funzioni con lo stesso nome purché i parametri di ingresso delle varie definizioni differiscano in tipo e/o numero. In base allo stesso principio è anche possibile ridefinire una funzione DB2 predefinita, per esempio gli operatori aritmetici.
Le funzioni utente DB2 possono essere classificate come segue.

- *Funzioni interne*: si costruiscono sulla base di funzioni DB2 predefinite, dette funzioni *sorgente*, in maniera simile a quanto avviene per i tipi utente.
- *Funzioni esterne*: corrispondono a programmi esterni alla base di dati e scritti in linguaggi di programmazione di alto livello (C o Java) che possono contenere o meno istruzioni SQL. La dichiarazione di una funzione esterna contiene la specifica della locazione su disco dove risiede il codice che implementa la funzione. Esistono due tipi di funzioni esterne.
  - *Funzioni scalari*: possono ricevere diversi valori in ingresso ma restituiscono un solo valore in uscita; se il nome della funzione ridefinisce un operatore (per esempio "+") allora tali funzioni possono essere invocate utilizzando la notazione infissa.
  - *Funzioni tabella*: sono in grado di restituire una collezione di ennuple di valori, che può essere assimilata a una tabella: ogni volta che queste funzioni vengono invocate, esse restituiscono una nuova ennupla (vista come una riga di tabella), oppure un codice speciale che indica il completamento dell'operazione.

Le funzioni interne si usano principalmente con i tipi utente per poter applicare, in maniera selettiva, alcune funzioni dei relativi tipi sorgente. Per esempio, con riferimento al tipo utente Money precedentemente definito, possiamo definire alcune funzioni interne con le seguenti dichiarazioni.

```
create function "*"(Money, Decimal()) returns Money
   source "*"(Decimal(), Decimal())
create function Total(Money) returns Money
   source sum(Decimal())
```

In queste dichiarazioni viene definito il nome della funzione, i tipi dei parametri di ingresso (che possono essere anche tipi utente), il tipo del dato restituito e la funzione *sorgente*, ovvero la funzione DB2 che va applicata quando viene invocata la funzione utente che si sta definendo. Nel nostro caso è stato ridefinito l'operatore di prodotto ed è stata definita una nuova funzione basata sull'operatore aggregativo `sum` di SQL. Queste dichiarazioni rendono legali istruzioni SQL come le seguenti.

```
select Dipartimento, Total(Stipendio)
from Impiegato
group by Dipartimento;

update Impiegato
set Stipendio =  Stipendio * 1.1
where Dipartimento = 'Produzione';
```

Non sono invece valide espressioni che coinvolgono, per esempio, la somma di uno stipendio e di un numero decimale perché l'operatore di somma risulta indefinito per il tipo Money.

Supponiamo ora di voler definire una funzione scalare esterna che calcola lo stipendio di riferimento dovuto a un impiegato in base alla sua anzianità di servizio, valutando l'anzianità sulla base della data di assunzione. Una possibile definizione per tale funzione esterna è la seguente:

```
create function StandardSalary(Date) returns Money
   external name '/usr/db2/bin/salary.exe!StdSal'
   deterministic
   no external action
   language c parameter style db2sql
   no sql;
```

questa dichiarazione definisce la funzione esterna `StandardSalary` che riceve una data in ingresso e restituisce un dato di tipo Money. La clausola `external` indica il nome e la posizione su disco del file che contiene il codice della funzione. Il nome dopo il punto esclamativo indica il modulo (funzione C nel nostro caso), all'interno del file, che implementa la funzione utente. La clausola `deterministic` specifica che diverse invocazioni della funzione sullo stesso valore restituiscono sempre lo stesso risultato. La clausola `external action` serve a specificare se la funzione coinvolge azioni esterne alla base di dati, come per esempio la scrittura in un file. Questa informazione può essere utile all'ottimizzatore per decidere se limitare il numero di invocazioni della funzione stessa. La clausola `language` specifica il linguaggio di programmazione usato per implementare la funzione esterna e le modalità di passaggio di parametri con essa. La modalità standard per il linguaggio C è denominata `db2sql`, mentre quello per il linguaggio Java è `db2general`. Infine, l'ultima clausola indica che la funzione non coinvolge accessi a una base di dati.

Non discuteremo qui le modalità secondo le quali si implementa una funzione esterna. Accenniamo solo al fatto che esistono delle convenzioni per trasformare tipi di dato SQL in tipi di dato del linguaggio di programmazione prescelto e viceversa, e che l'implementazione ha in effetti più parametri della funzione esterna. Questi parametri aggiuntivi servono per scambiare ulteriori dati di servizio, come eventuali messaggi di errore.

Una volta definita e implementata una funzione esterna, è possibile usarla in istruzioni SQL nella stessa maniera in cui si usano le funzioni predefinite. Per

esempio, la seguente istruzione trova cognomi e stipendi degli impiegati che guadagnano più del 20% del rispettivo stipendio di riferimento, mostrando anche tale valore.

```
select Nome, Stipendio, StandardSalary(DataAssunzione)
from Impiegato
where Stipendio > StandardSalary(DataAssunzione) * 1.2;
```

Si osservi che la condizione nella clausola `where` è valida perché: (a) la funzione utente restituisce un dato di tipo Money, (b) il prodotto tra questo tipo di dato e un decimale è stato definito, (c) il valore restituito dal prodotto è di tipo Money e può quindi essere confrontato con l'attributo Stipendio, come stabilito all'atto della definizione del tipo utente Money.

Le funzioni tabella costituiscono una funzionalità particolarmente interessante di DB2. Permettono infatti di trasformare facilmente una qualunque sorgente di dati esterna in una tabella manipolabile con SQL. L'unica cosa da fare è scrivere un programma che accede alla sorgente, per esempio un file di testo o un file MS Excel, filtra eventualmente dei dati in base a parametri passati in ingresso, e infine li restituisce, una riga alla volta. La restituzione delle varie righe avviene mediante una tecnica particolare: viene allocata in maniera automatica una zona di memoria (detta *scratchpad*) che preserva il suo contenuto tra un'invocazione e un'altra di una funzione. In questa maniera, un'invocazione può accedere a informazioni relative all'invocazione precedente; per esempio, l'ultima posizione del file acceduta.
Supponiamo di avere a disposizione una funzione di questo tipo che prende il nome di una località in ingresso e restituisce una ennupla di valori memorizzati su un file remoto, corrispondenti a dati di vendita di negozi che si trovano in tale località. La definizione della corrispondente funzione tabella potrebbe essere la seguente:

```
create function Vendite(Char(20))
   returns table (Negozio  char(20),
                  Prodotto char(20),
                  Incasso  Integer)
   external name '/usr/db2/bin/sales'
   deterministic
   no external action
   language c parameter style db2sql no sql
   scratchpad
   final call disallow parallel;
```

Si può notare che, a parte alcune clausole specifiche per le funzioni tabella, la definizione è simile alla definizione di funzioni scalari esterne. La differenza principale è che il risultato della funzione è composto da più parametri, che vengono interpretati come attributi di una tabella.

Una possibile istruzione SQL che fa uso della funzione tabella sopra definita è la seguente:

```
select Negozio, SUM(Incasso)
from table(Vendite('Roma')) as VenditeRoma
where Product = 'Giocattoli'
group by Negozio;
```

Questa istruzione restituisce l'incasso totale dei negozi di Roma relativo alle vendite di giocattoli. La tabella interrogata non è memorizzata nella base di dati (come indicato dalla parola chiave `table`), ma viene generata dalla funzione tabella `Vendite`, alla quale viene passato, come parametro di ingresso la stringa `'Roma'`.