

Capitolo 8

Applicazioni tradizionali

Copyright © 2004 – The McGraw-Hill Companies srl

SQL e applicazioni

- In applicazioni complesse, l'utente non vuole eseguire comandi SQL, ma programmi, con poche scelte
- SQL non basta, sono necessarie altre funzionalità, per gestire:
 - input (scelte dell'utente e parametri)
 - output (con dati che non sono relazioni o se si vuole una presentazione complessa)
 - per gestire il controllo

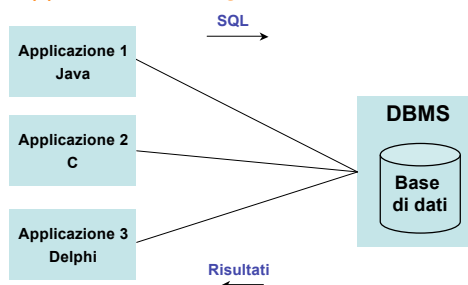
Copyright © 2004 – The McGraw-Hill Companies srl

SQL e linguaggi di programmazione

- Le applicazioni sono scritte in
 - linguaggi di programmazione tradizionali:
 - Cobol, C, Java, Fortran
 - linguaggi "ad hoc", proprietari e non:
 - PL/SQL, Informix4GL, Delphi
- Vediamo solo l'approccio "tradizionale", perché più generale

Copyright © 2004 – The McGraw-Hill Companies srl

Applicazioni ed SQL: architettura



Copyright © 2004 – The McGraw-Hill Companies srl

Una difficoltà importante

- **Conflitto di impedenza** ("disaccoppiamento di impedenza") fra base di dati e linguaggio
 - linguaggi: operazioni su singole variabili o oggetti
 - SQL: operazioni su relazioni (insiemi di ennuple)

Copyright © 2004 – The McGraw-Hill Companies srl

Altre differenze

- **Accesso ai dati e correlazione:**
 - linguaggio: dipende dal paradigma e dai tipi disponibili; ad esempio scansione di liste o "navigazione" tra oggetti
 - SQL: join (ottimizzabile)
- **tipi di base:**
 - linguaggi: numeri, stringhe, booleani
 - SQL: CHAR, VARCHAR, DATE, ...
- **costruttori di tipo:**
 - linguaggio: dipende dal paradigma
 - SQL: relazioni e ennuple

Copyright © 2004 – The McGraw-Hill Companies srl

SQL e linguaggi di programmazione: tecniche principali

- SQL immerso ("Embedded SQL")
 - sviluppata sin dagli anni '70
 - "SQL statico"
- SQL dinamico
- Call Level Interface (CLI)
 - più recente
 - SQL/CLI, ODBC, JDBC

Copyright © 2004 – The McGraw-Hill Companies srl

SQL immerso

- le istruzioni SQL sono "immerse" nel programma redatto nel linguaggio "ospite"
- un precompilatore (legato al DBMS) viene usato per analizzare il programma e tradurlo in un programma nel linguaggio ospite (sostituendo le istruzioni SQL con chiamate alle funzioni di una API del DBMS)

Copyright © 2004 – The McGraw-Hill Companies srl

SQL immerso, un esempio

```
#include<stdlib.h>
main(){
    exec sql begin declare section;
        char *NomeDip = "Manutenzione";
        char *CittaDip = "Pisa";
        int NumeroDip = 20;
    exec sql end declare section;
    exec sql connect to utente@librodb;
    if (sqlca.sqlcode != 0) {
        printf("Connessione al DB non riuscita\n"); }
    else {
        exec sql insert into Dipartimento
            values (:NomeDip, :CittaDip, :NumeroDip);
        exec sql disconnect all;
    }
}
```

Copyright © 2004 – The McGraw-Hill Companies srl

SQL immerso, commenti al codice

- EXEC SQL denota le porzioni di interesse del precompilatore:
 - definizioni dei dati
 - istruzioni SQL
- le variabili del programma possono essere usate come "parametri" nelle istruzioni SQL (precedute da ":") dove sintatticamente sono ammesse costanti

Copyright © 2004 – The McGraw-Hill Companies srl

SQL immerso, commenti al codice, 2

- `sqlca` è una struttura dati per la comunicazione fra programma e DBMS
- `sqlcode` è un campo di `sqlca` che mantiene il codice di errore dell'ultimo comando SQL eseguito:
 - zero: successo
 - altro valore: errore o anomalia

Copyright © 2004 – The McGraw-Hill Companies srl

SQL nel codice applicativo

- I comandi SQL possono essere chiamati dall'interno di un programma in un linguaggio ospite (ad esempio C++ o Java)
 - I comandi SQL possono far riferimento a variabili dell'ospite (includere variabili speciali usate per restituire uno status)
 - Deve essere incluso un comando per *connettersi* alla giusta base di dati
- Due principali approcci all'integrazione:
 - Incorporare l'SQL nel linguaggio ospite (SQL incapsulato, SQLJ)
 - Creare una speciale API per chiamare i comandi SQL (JDBC)

Copyright © 2004 – The McGraw-Hill Companies srl

SQL nel codice applicativo (segue)

Disadattamento di impedenza:

- Le relazioni SQL sono (multi-)insiemi di record senza un numero di record fissato *a priori*. Nessuna struttura di questo tipo esiste tradizionalmente in linguaggi di programmazione procedurali come il C++ (sebbene ora ci sia l'STL)
 - SQL supporta un meccanismo chiamato *cursore* per gestire questa situazione

Copyright © 2004 – The McGraw-Hill Companies srl

SQL incapsulato

- Approccio: incapsulare l'SQL nel linguaggio ospite.
 - Un preprocessore converte i comandi SQL in speciali chiamate all'API
 - Successivamente un normale compilatore viene usato per compilare il codice
- Costrutti del linguaggio:
 - connessione a una base di dati:
EXEC SQL CONNECT
 - Dichiarazione di variabili:
EXEC SQL BEGIN (END) DECLARE SECTION
 - Comandi:
EXEC SQL comando

Copyright © 2004 – The McGraw-Hill Companies srl

SQL incapsulato: variabili

```
EXEC SQL BEGIN DECLARE SECTION
Char c_vnome[20];
Long c_vid;
Short c_esperienza;
Float c_età;
EXEC SQL END DECLARE SECTION
```

- Due speciali variabili "errore":
 - SQLCODE (long, se negativa si è verificato un errore)
 - SQLSTATE (char[6], codici predefiniti per errori comuni)

Copyright © 2004 – The McGraw-Hill Companies srl

Cursori

- Si può dichiarare un cursore su una relazione o un comando di interrogazione (che genera una relazione)
- Si può *aprire* un cursore, e ripetutamente *prelevare* una tupla, poi *muovere* il cursore fino a quando tutte le tuple siano state lette.
 - Si può usare una clausola speciale, chiamata ORDER BY, nelle interrogazioni cui si accede tramite cursore, per controllare l'ordine in cui le tuple sono restituite
 - I campi nella clausola ORDER BY devono apparire anche nella clausola SELECT
 - La clausola ORDER BY, che ordina le tuple della risposta, è permessa *solo* nel contesto di un cursore
- Si può anche modificare/cancellare la tupla puntata dal cursore

Copyright © 2004 – The McGraw-Hill Companies srl

Cursori che leggono i nomi dei velisti che hanno prenotato una barca rossa, in ordine alfabetico

```
EXEC SQL DECLARE vinfo CURSOR FOR
SELECT V.vnome
FROM Velisti V, Barche B, Prenota P
WHERE V.vid = P.vid AND P.bid = B.bid AND B.colore = 'rosso'
ORDER BY V.vnome
```

- Notate che è illegale sostituire V.vnome con, diciamo, V.vid nella clausola ORDER BY! (Perché?)
- Possiamo aggiungere V.vid alla clausola SELECT e sostituire V.vnome con V.vid nella clausola ORDER BY?

Copyright © 2004 – The McGraw-Hill Companies srl

Incapsulare l'SQL in C: un esempio

```
Char SQLSTATE[6];
EXEC SQL BEGIN DECLARE SECTION
char c_vnome[20]; short c_minesperienza; float c_età;
EXEC SQL END DECLARE SECTION
c_minesperienza = random();
EXEC SQL DECLARE vinfo CURSOR FOR
SELECT V.vnome, V.età FROM Velisti V
WHERE V.esperienza > :c_minesperienza
ORDER BY V.vnome;
Do {
    EXEC SQL FETCH vinfo INTO :c_vnome, :c_età;
    printf("%s ha %d anni\n", c_vnome, c_età);
} WHILE (SQLSTATE != '02000');
EXEC SQL CLOSE vinfo;
```

Copyright © 2004 – The McGraw-Hill Companies srl

SQL dinamico

- Le stringhe di interrogazione SQL ora sono sempre note al momento della compilazione (ad esempio fogli di lavoro o *frontend* grafici per il DBMS): possibile la costruzione di comandi SQL al volo
- Esempio:

```
char c_stringasql[] =  
    {"DELETE FROM Velisti WHERE esperienza > 5"};  
EXEC SQL PREPARE pronti FROM :c_stringasql;  
EXEC SQL EXECUTE pronti;
```

Copyright © 2004 – The McGraw-Hill Companies srl

API per basi di dati: alternative all'incapsulamento

Piuttosto che modificare il compilatore, si possono aggiungere librerie con chiamate alla base di dati (API)

- Speciale interfaccia standardizzata: procedure/oggetti
- Si passano le stringhe SQL dal linguaggio, si presentano gli insiemi risultato in maniera comprensibile
- JDBC della Sun: API Java
- Generalmente neutre rispetto al DBMS
 - Un "driver" rileva le chiamate e le traduce in codice specifico per il DBMS
 - La base di dati può essere distribuita

Copyright © 2004 – The McGraw-Hill Companies srl

JDBC: architettura

- Quattro componenti architetturali:
 - Applicazione (inizia e termina le connessioni, invia i comandi SQL)
 - Gestore di driver (carica il driver JDBC)
 - Driver (si connette alla sorgente di dati, trasmette le richieste e restituisce/traduce i risultati e i codici di errore)
 - Sorgente di dati (elabora i comandi SQL)

Copyright © 2004 – The McGraw-Hill Companies srl

Architettura JDBC (segue)

Quattro tipi di driver:

- Bridge:
 - Traduce i comandi SQL in API non nativa
 - Esempio: Ponte JDBC-ODBC. Il codice per i driver ODBC e JDBC deve essere disponibile su ogni client
- Traduzione diretta in API nativa, driver non Java:
 - traduce i comandi SQL nell'API nativa della sorgente di dati. Ha bisogno di codice binario specifico per il SO di ciascun client
- Bridge di rete:
 - Manda i comandi sulla rete a un server intermedio che comunica con la sorgente di dati. Ha solo bisogno di un piccolo driver JDBC su ciascun client
- Traduzione diretta in API nativa tramite driver Java:
 - Converte le chiamate JDBC direttamente nel protocollo di rete usato dal DBMS. Ha bisogno di un driver Java specifico per il DBMS su ciascun client

Copyright © 2004 – The McGraw-Hill Companies srl

Classi e interfacce JDBC

Passi per inviare una interrogazione ad una base di dati:

- Caricare il driver JDBC
- Connettersi alla sorgente di dati
- Eseguire i comandi SQL

Copyright © 2004 – The McGraw-Hill Companies srl

Gestione del driver JDBC

- Tutti i driver sono gestiti dalla classe DriverManager
- Caricamento di un driver JDBC:
 - Nel codice Java:
`Class.forName("oracle/jdbc.driver.OracleDriver");`
 - All'avvio dell'applicazione Java:
`-Djdbc.drivers=oracle/jdbc.driver`

Copyright © 2004 – The McGraw-Hill Companies srl

Connessioni in JDBC

Si interagisce con una sorgente di dati tramite sessioni. Ogni connessione identifica una sessione logica.

- URL JDBC:
jdbc:<subprotocollo>:<altriParametri>

Esempio:
String url="jdbc:oracle:www.bookstore.com:3083";
Connection con;
try{
 con = DriverManager.getConnection(url, userID,
 password);
} catch SQLException excp {...}

Copyright © 2004 – The McGraw-Hill Companies srl

Interfaccia della classe Connection

- Public int getTransactionIsolation() e
Void setTransactionIsolation(int livello)
Imposta il livello di isolamento per la connessione corrente
- Public boolean getReadOnly() e
Void setReadOnly(boolean b)
Specifica se le transazioni in questa connessione sono di sola lettura
- Public boolean getAutoCommit() e
Void setAutoCommit(boolean b)
Se autocommit è attivo, allora ciascun comando SQL viene considerato come una transazione a sé stante. Altrimenti, una transazione termina usando commit(), o viene interrotta usando rollback()
- Public boolean isClosed()
Controlla se la transazione è ancora aperta

Copyright © 2004 – The McGraw-Hill Companies srl

Esecuzione di comandi SQL

- Tre diversi modi di eseguire i comandi SQL:
 - Statement (sia comandi di SQL statico che dinamico)
 - PreparedStatement (comandi di SQL semi-statico)
 - CallableStatement (stored procedure)
- Classe PreparedStatement:
- Comandi SQL parametrizzati, precompilati
 - La struttura è fissa
 - I valori dei parametri sono determinati al momento dell'esecuzione

Copyright © 2004 – The McGraw-Hill Companies srl

Esecuzione di comandi SQL (segue)

```
String sql="INSERT INTO Velisti VALUES(?,?,?,?)";  
PreparedStatement pstmt=con.prepareStatement(sql);  
pstmt.clearParameters();  
pstmt.setInt(1, vid);  
pstmt.setString(2, vnome);  
pstmt.setInt(3, esperienza);  
pstmt.setFloat(4, età);  
// sappiamo che non vengono restituite righe, quindi  
// usiamo executeUpdate()  
int numRighe = pstmt.executeUpdate();
```

Copyright © 2004 – The McGraw-Hill Companies srl

Insiemi risultato

- PreparedStatement.executeUpdate restituisce solo il numero di record modificati
- PreparedStatement.executeQuery restituisce dati, incapsulati in un oggetto ResultSet (un cursore)

```
ResultSet rs=pstmt.executeQuery(sql);  
// rs ora è un cursore  
While (rs.next()) {  
    // elabora i dati  
}
```

Copyright © 2004 – The McGraw-Hill Companies srl

Insiemi risultato (segue)

Un ResultSet è un cursore molto potente:

- previous(): muove indietro di una riga
- absolute(int num): muove alla riga col numero specificato
- relative(int num): muove avanti o indietro
- first() e last()

Copyright © 2004 – The McGraw-Hill Companies srl