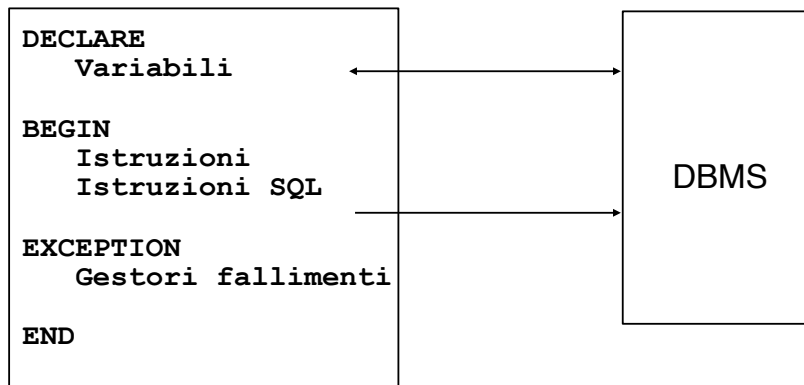


- Come collegarsi alla BD
- Come trattare gli operatori SQL
- Come trattare il risultato di un comando SQL (relazioni)
- Come scambiare informazioni sull'esito delle operazioni.



## "IMPEDENCE MISMATCH"

- È necessario convertire i dati dal "modello dei dati" relazionale al "modello dei dati" del linguaggio di programmazione
- Tutti i linguaggi di programmazione hanno modelli dei dati diversi
- Anche la logica di funzionamento è diversa:
  - Concetto di transazione
  - Si lavora su insiemi, in maniera dichiarativa
  - Una modifica da parte del programma non comporta necessariamente una scrittura su disco (cioè una modifica del database)
- Ad oggi non vi sono soluzioni riconosciute da tutti come "ideali"

- Linguaggio integrato (dati e DML)

Linguaggio disegnato ad-hoc per usare SQL. I comandi SQL sono controllati staticamente dal traduttore ed eseguiti dal DBMS.

- Linguaggio convenzionale + API

Linguaggio convenzionale che usa delle funzioni di una libreria predefinita per usare SQL. I comandi SQL sono **stringhe** passate come parametri alle funzioni che poi vengono controllate dinamicamente dal DBMS prima di eseguirle.

- Linguaggio che ospita l'SQL

Linguaggio convenzionale esteso con un nuovo costrutto per marcare i comandi SQL. Occorre un **pre-compilatore** che controlla i comandi SQL, li sostituisce con chiamate a funzioni predefinite e genera un programma nel linguaggio convenzionale + API.

- Mapping oggetti-relazionali (Object Relational Mapping, ORM)

Linguaggio ad oggetti convenzionale con uno strato intermedio di software (middleware) che "mappa" entità in oggetti (e viceversa) in maniera semi-trasparente.

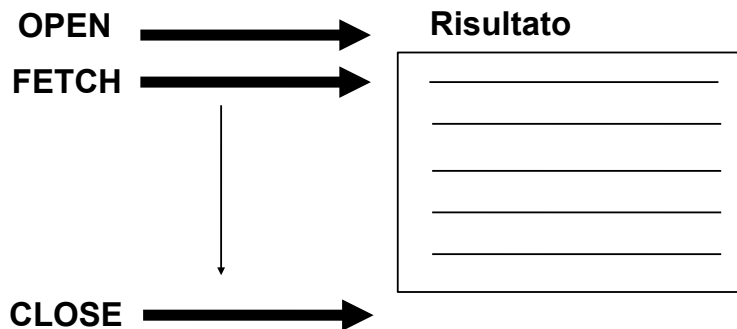
Query particolari restituiscono oggetti, che sono "ricreati" a partire dalle entità ogni volta che serve (e resi unici). Si lavora su di esse e alla fine della "sessione" di lavoro il software li riscrive su disco.

- Un linguaggio per manipolare basi di dati che integra DML (SQL) con il linguaggio ospite
- Un linguaggio a blocchi con una struttura del controllo completa che contiene l'SQL come sottolinguaggio
- Permette:
  - Di definire variabili di tipo scalare, record (annidato), insieme di scalari, insieme di record piatti, cursore
  - Di definire i tipi delle variabili a partire da quelli della base di dati
  - Di eseguire interrogazioni SQL ed esplorarne il risultato
  - Di modificare la base di dati
  - Di definire procedure e moduli
  - Di gestire il flusso del controllo, le transazioni, le eccezioni

## UNA PROCEDURA IN PL/SQL

```
CREATE
PROCEDURE Esempio1(
    p_Mat IN Studenti.Matricola%TYPE) IS
DECLARE
--    Identificatori per lo scambio dati
    XNome CHAR;
    XProvincia Studenti.Provincia%TYPE;
    XAttributi Studenti%ROWTYPE;
    prv_manca EXCEPTION;
BEGIN
-- ricerca di enupla : stampa Nome e Provincia
SELECT Nome, Provincia INTO XNome, XProvincia
FROM Studenti WHERE Matricola = p_Mat;
IF XProvincia IS NULL THEN
    RAISE prv_manca
ELSE PRINT ...
END IF
EXCEPTION
    WHEN prv_manca THEN <gestione eccezione>
END;
```

- E' il meccanismo per ottenere uno alla volta gli elementi di una relazione
- Un cursore viene definito con un'espressione SQL, poi
  - si apre per far calcolare al DBMS il risultato e poi
  - con un opportuno comando si trasferiscono i campi delle ennuple in opportune variabili del programma.



## USO DEI DATI DA PROGRAMMI (cont.)

### Cursore + FETCH

```
PROCEDURE Esempio2 (Prov IN Studenti.Provincia%TYPE) IS
DECLARE
    CURSOR c IS
        SELECT Nome, AnnoNascita
        FROM Studenti WHERE Provincia = Prov;
    Stud_Rec c%ROWTYPE;
BEGIN
    -- ricerca di insieme di ennuple : stampa Nome e
    -- AnnoNascita degli studenti di Pisa
    OPEN c
    LOOP
        FETCH c INTO Stud_Rec;
        EXIT WHEN c%NOTFOUND;
        PRINT ... Stud_Rec.Nome ... Stud_Rec.Provincia
    END LOOP;
    CLOSE c -- rilascio del cursore
END
```

```
PROCEDURE Esempio3 (Prov IN Studenti.Provincia%TYPE) IS
BEGIN
-- ricerca di insieme di ennuple: stampa Nome e
-- AnnoNascita degli studenti di una certa provincia */

FOR Stud_Rec IN
    (SELECT Nome, AnnoNascita
     FROM Studenti WHERE Provincia = Prov)
LOOP
    PRINT ... Stud_Rec.Nome ... Stud_Rec.Provincia
END LOOP; -- rilascio del cursore

END
```

## LINGUAGGIO CON INTERFACCIA API

10

- Invece di modificare il compilatore di un linguaggio, si usa una libreria di funzioni/oggetti che operano su basi di dati (API) alle quali si passa come parametro stringhe SQL e ritornano il risultato sul quale si opera con una logica ad iteratori.
  - Microsoft ODBC è C/C++ standard su Windows
  - Sun JDBC è l'equivalente in Java
  - Dovrebbero essere indipendenti dal DBMS
    - un "driver" gestisce le richieste e le traduce in un codice specifico di un DBMS
    - la BD può essere in rete

```
class StampaNomiStudenti{
public static void main(String argv[]){
Class.forName("driver per DBMS");
Connection con = // connect
    DriverManager.getConnection("url", "login", "pass");
Statement stmt = con.createStatement(); // crea un oggetto per comando SQL
String query = "SELECT Nome
                FROM Studenti WHERE Provincia =\'" + argv[0] + " \';"
ResultSet iter = stmt.executeQuery(query);
System.out.println("Nomi trovati:");
try { // gestore eccezioni
    // ciclo sul risultato
    while (iter.next()) {
        String nome = iter.getString("Nome");
        int anno = iter.getInt("AnnoNascita");
        System.out.println(" Nome: " + nome + "; AnnoNascita: " + anno);
    }
} catch(SQLException ex) {
    System.out.println(ex.getMessage()+ex.getSQLState()+ex.getErrorCode());
}
stmt.close(); con.close();
}}
```