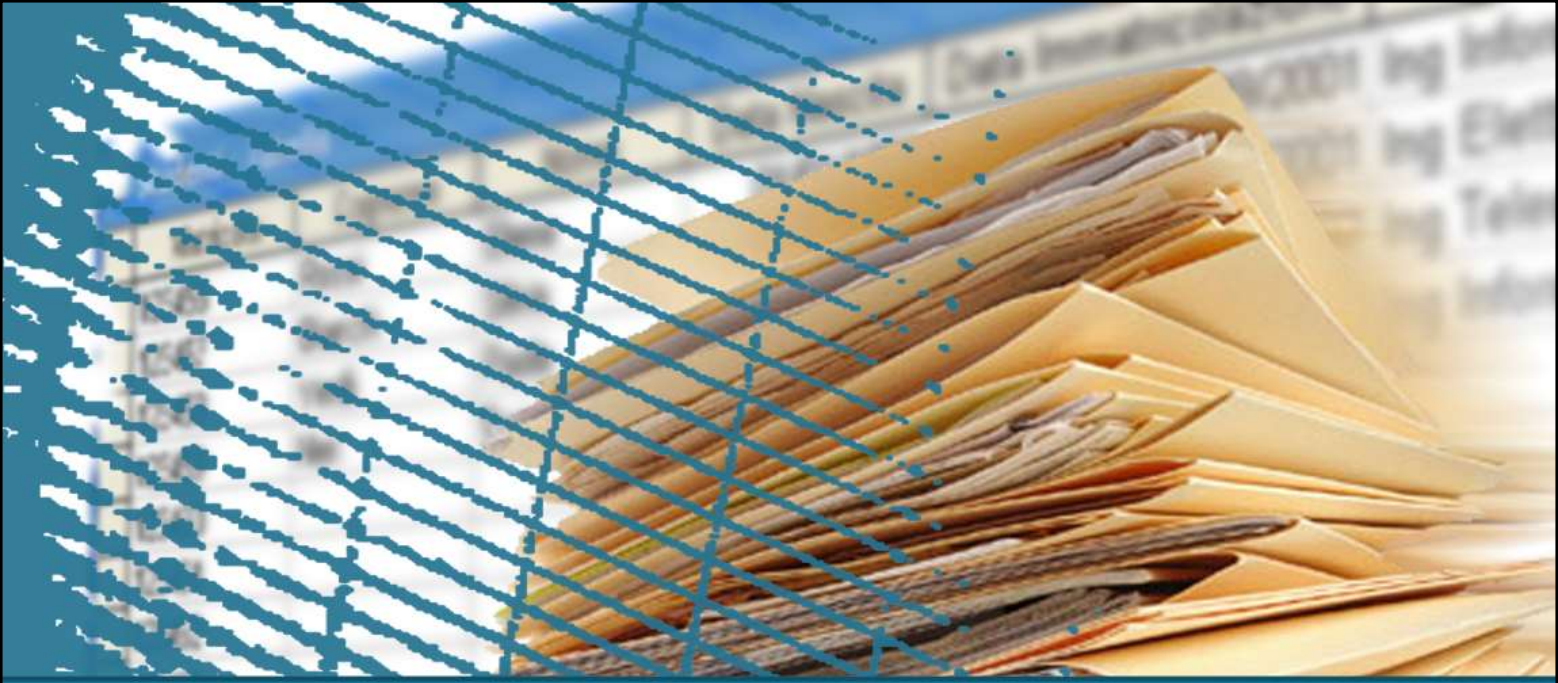


SQL per le applicazioni

Call Level Interface (CLI)

Call Level Interface

- Le richieste sono inviate al DBMS per mezzo di funzioni del linguaggio ospite
  - soluzione basata su interfacce predefinite
    - API, Application Programming Interface
  - le istruzioni SQL sono passate come parametri alle funzioni del linguaggio ospite
  - non esiste il concetto di precompilatore
- Il programma ospite contiene direttamente le chiamate alle funzioni messe a disposizione dall'API

Call Level Interface

- Esistono diverse soluzioni di tipo Call Level Interface (CLI)
  - standard SQL/CLI
  - ODBC (Open DataBase Connectivity)
    - soluzione proprietaria Microsoft di SQL/CLI
  - JDBC (Java Database Connectivity)
    - soluzione per il mondo Java
  - OLE DB
  - ADO
  - ADO.NET

Modalità d'uso

- Indipendentemente dalla soluzione CLI adottata, esiste una strutturazione comune dell'interazione con il DBMS
  - apertura della connessione con il DBMS
  - esecuzione di istruzioni SQL
  - chiusura della connessione

Interazione con il DBMS

1. Chiamata a una primitiva delle API per creare una connessione con il DBMS

Interazione con il DBMS

1. Chiamata a una primitiva delle API per creare una connessione con il DBMS
2. Invio sulla connessione di un'istruzione SQL

Interazione con il DBMS

1. Chiamata a una primitiva delle API per creare una connessione con il DBMS
2. Invio sulla connessione di un'istruzione SQL
3. Ricezione di un risultato in risposta all'istruzione inviata
   - nel caso di SELECT, di un insieme di tuple

Interazione con il DBMS

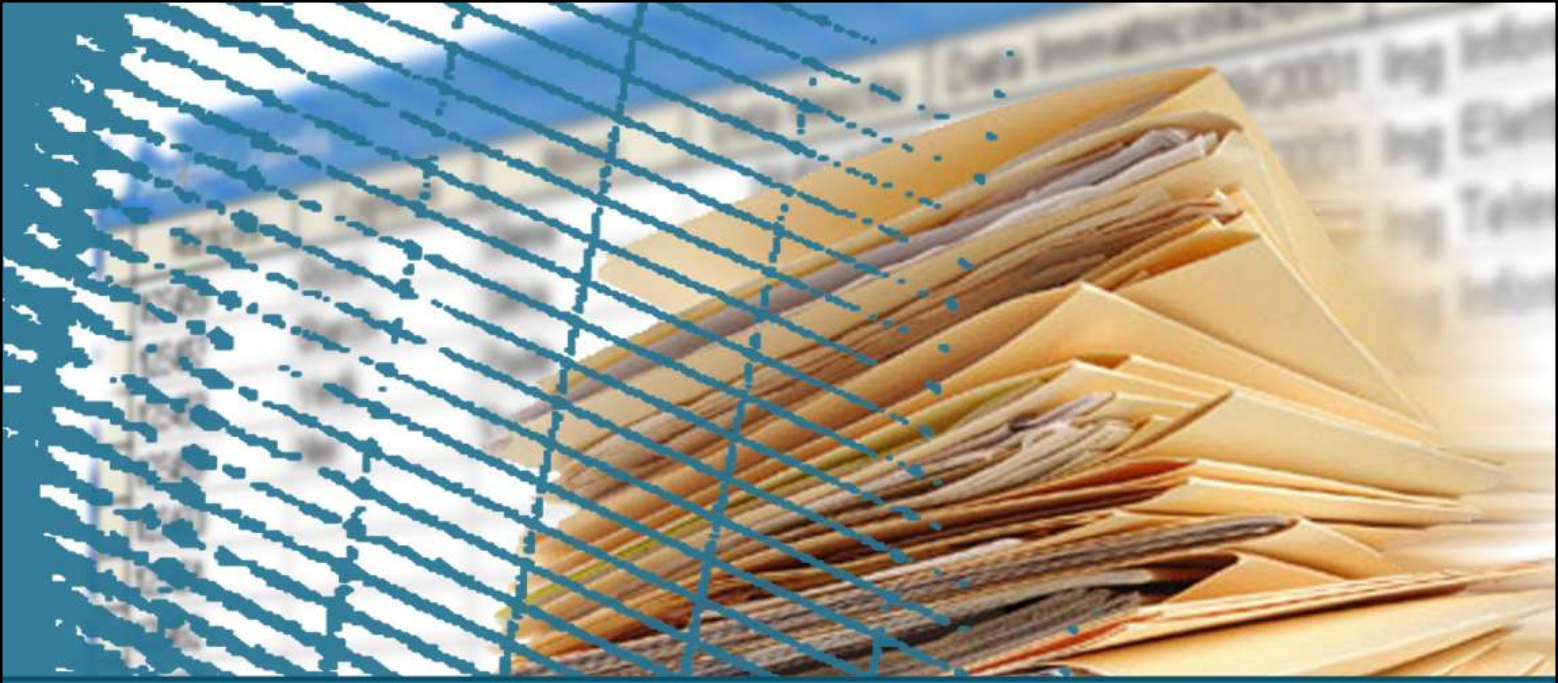
1. Chiamata a una primitiva delle API per creare una connessione con il DBMS
2. Invio sulla connessione di un'istruzione SQL
3. Ricezione di un risultato in risposta all'istruzione inviata
   - nel caso di SELECT, di un insieme di tuple
4. Elaborazione del risultato ottenuto
   - esistono apposite funzioni per leggere il risultato

Interazione con il DBMS

1. Chiamata a una primitiva delle API per creare una connessione con il DBMS
2. Invio sulla connessione di un'istruzione SQL
3. Ricezione di un risultato in risposta all'istruzione inviata
   - nel caso di SELECT, di un insieme di tuple
4. Elaborazione del risultato ottenuto
   - esistono apposite funzioni per leggere il risultato
5. Chiusura della connessione al termine della sessione di lavoro

Interazione con il DBMS

- ODBC (Open DataBase Connectivity)
  - Metodo di accesso standard verso una base dati
  - Scopo: rendere il protocollo di accesso al database indipendente dal tipo di database utilizzato
  - PHP mette a disposizione del programmatore una libreria che consente di accedere via ODBC ad una base dati
- Metodi di accesso mirati ad un DBMS specifico
  - MySQL, Postgres, Microsoft SQL server, …
  - PHP mette a disposizione del programmatore librerie specifiche per gran parte dei DBMS



SQL per le applicazioni

Funzioni MySQL per PHP

Estensione MySQLi

- MySQLi *(MySQL improved)* è un'estensione di PHP che consente di interfacciarsi al DB MySQL in modo efficiente
- Funzionalità principali
  - connessione al DB
  - esecuzione di query SQL in modalità *immediata* o *preparata* (istruzioni SQL precedentemente utilizzate e mantenute in cache per successive chiamate)
  - acquisizione e lettura di dati
  - supporto per stored procedure, query multiple e transazioni

Creazione di una connessione

- Chiamata alla funzione mysqli_connect()
  - Richiede quattro parametri: "hostname" (nome della macchina che ospita il DBMS MySQL a cui si desidera fare la connessione), "username", "password", "dbname" (nome del DB)
  - In caso di successo restituisce un identificativo di connessione MySQL, in caso di insuccesso restituisce FALSE
- Esempio;

```
//Connessione a MySQL tramite mysqli_connect()
$con = mysqli_connect('localhost','joe','xyz','dbname');
```

Creazione di una connessione

- Esempio con controllo di eventuali errori di connessione
  - die(): arresta l'esecuzione dello script e stampa un messaggio
  - mysqli_connect_errno(): restituisce il codice dell'errore di connessione
  - mysqli_connect_error(): restituisce l'errore di connessione

```
if (mysqli_connect_errno())
  {
  die ('Failed to connect to MySQL: ' . mysqli_connect_error());
  }
```

Chiusura di una connessione

- Deve essere eseguita quando non è più necessario interagire con il DBMS
  - chiude il collegamento con il DBMS e rilascia le relative risorse
- Chiamata alla funzione mysqli_close(...)
  - parametro (opzionale): identificativo della connessione
  - se non viene indicato nessun parametro viene chiusa la connessione aperta più recentemente

```
//chiusura della connessione
mysqli_close($con);
```

Esecuzione immediata di SQL

- Esecuzione *immediata* dell'istruzione
  - Il server compila ed esegue *immediatamente* l'istruzione SQL ricevuta
- Esecuzione "preparata" dell'istruzione
  - L'istruzione SQL
    - è compilata (preparata) una volta sola e il suo piano di esecuzione è memorizzato dal DBMS
    - è eseguita molte volte durante la sessione
  - Utile quando si deve eseguire la stessa istruzione SQL più volte nella stessa sessione di lavoro
    - varia solo il valore di alcuni parametri

Esecuzione immediata di SQL

- Chiamata alla funzione mysqli_query(...)
  - richiede come parametri
    - l'id della connessione
    - la query da eseguire, in formato stringa
  - restituisce
    - *il risultato della query* in caso di successo
    - *FALSE* in caso di insuccesso

Esecuzione immediata di SQL

- Chiamata alla funzione mysqli_error(...)
  - richiede come parametro
    - l'id della connessione
  - restituisce la descrizione dell'ultimo errore occorso nella connessione specificata
- Esempio:

```
/* QUERY SQL */
$sql   = " SELECT autore.cognome, opera.nome,
           FROM autore, opera
           WHERE autore.coda = opera.autore "
$result = mysqli_query($con,$sql);

if( !$result )
  die('Query error: ' . mysqli_error($con));
```

Lettura del risultato

- Il risultato della funzione mysqli_query() viene memorizzato in una variabile di tipo "resource"
  - una variabile speciale, che contiene il riferimento ad una risorsa esterna
- La lettura del risultato avviene riga per riga: ciclo che prevede due fasi
  - acquisizione di una riga della tabella (utilizzo di un cursore)
  - lettura della riga acquisita

NomeF	NSoci
Andrea	2
Gabriele	2

Cursore

Acquisizione di una riga (modo 1)

- Chiamata alla funzione mysqli_fetch_row()
  - richiede come parametro la risorsa restituita da mysqli_query()
  - restituisce l'array corrispondente alla riga corrente, o FALSE nel caso in cui non ci siano righe disponibili
  - ciascuna colonna del risultato viene memorizzata in un elemento dell'array, a partire dall'indice "0"

```
while ($row = mysqli_fetch_row($result)) {
      ...
      }
```

Acquisizione di una riga (modo 2)

- Chiamata alla funzione mysqli_fetch_assoc()
  - richiede come parametro la risorsa restituita da mysqli_query()
  - restituisce l'array associativo corrispondente alla riga corrente, o FALSE nel caso in cui non ci siano righe disponibili
  - ciascuna colonna del risultato viene memorizzata in un elemento dell'array associativo in corrispondenza alla chiave definita dal nome del campo
  - non viene definito un indice numerico

Acquisizione di una riga (modo 3)

- Chiamata alla funzione mysqli_fetch_array()
  - è la funzione più generale
  - richiede, come parametri
    - la risorsa restituita da mysqli_query()
    - il tipo di array da riempire; una costante che può assumere i seguenti valori:
      - MYSQLI_ASSOC: l'array risultante è di tipo *associativo* (accesso con chiave)
      - MYSQLI_NUM: l'array risultante è di tipo *scalare* (accesso con indice numerico)
      - MYSQLI_BOTH: l'array risultante è accessibile sia con *indice numerico* sia con *chiave* corrispondente al *nome del campo*

Esempio (modo 1 di acquisizione)

NomeF	NSoci
Andrea	2
Gabriele	2

```
$sql = "SELECT NomeF, NSoci FROM F";
```

```
$result = mysqli_query($con, $sql);
```

```
// mysqli_fetch_row($result) assegna alla variabile $row,
// UNA RIGA PER VOLTA, le righe del risultato $result;
// quando non ci sono più righe, la funzione assegna
// a $row il valore FALSE
while($row = mysqli_fetch_row($result)) {
  // $row[0] e $row[1] sono i due elementi dell'array row[]
  // in cui vengono inseriti i due campi di indici 0 e 1
  // del risultato dell'istruzione SQL (NomeF e NSoci)
  echo "$row[0] $row[1]<br>";
}
```

Altri esempi

NomeF	NSoci
Andrea	2
Gabriele	2

```
while ($row = mysqli_fetch_row($result)) {
        echo "<tr>";
        echo "<td>$row[0]</td><td>$row[1]</td>";
        echo "</tr>";
}
```

```
while ($row = mysqli_fetch_row($result)) {
        echo "\t<tr>\n";
        foreach ($row as $cell) {
                echo "\t\t<td>$cell</td>\n";
        }
        echo "\t</tr>\n";
}
```

```
while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr>";
        echo "<td>" . $row["NomeF"] . "</td><td>" . $row["NSoci"] . "</td>";
        echo "</tr>";
}
```

Altre funzioni utili

- int mysqli_num_rows(*resource $result*)
  - restituisce il numero di righe della risorsa *$result*, oppure FALSE in caso di insuccesso

```
if ( mysqli_num_rows($result) <=0) {
        echo "<h5>Nessun risultato</h5>";
}
else {
// accedi alle righe del risultato
...
}
```

Altre funzioni utili

- int mysqli_num_fields(*resource $result*)
  - restituisce il numero di campi (attributi) della risorsa $result, o FALSE in caso di insuccesso
- string mysqli_fetch_field(*resource $result*)
  - restituisce la prossima colonna come oggetto. Per ottenerne il nome occorre selezionarne la proprietà "name"

```
for ($i = 0; $i < mysqli_num_fields($result); $i++){
  $title  =  mysqli_fetch_field($result);
  $name = $title->name;
  echo "<th> $name </th>";
}
```

Le transazioni

- Le connessioni avvengono implicitamente in modalità auto-commit
  - dopo l'esecuzione con successo di ogni istruzione SQL, è eseguito automaticamente commit
- Quando è necessario eseguire commit solo dopo aver eseguito con successo una sequenza di istruzioni SQL
  - il commit deve essere gestito in modo non automatico
  - si esegue un solo commit alla fine dell'esecuzione di tutte le istruzioni

Gestione delle transazioni

- bool mysqli_autocommit (mysqli $link , *bool $mode*)
  - abilita o disabilita la modalità *auto-commit*
  - richiede come parametri
    - l'identificativo della connessione
    - TRUE o FALSE a seconda che si vogli abilitare o disabilitare la modalità auto-commit
  - restituisce
    - TRUE in caso di successo
    - FALSE in caso di insuccesso

Gestione delle transazioni

- Se si disabilita l'autocommit le operazioni di commit e rollback devono essere richieste esplicitamente
- bool mysqli_commit (*mysqli $link*)
  - esegue il commit della transazione corrente
- bool mysqli_rollback (*mysqli $link*)
  - esegue il rollback della transazione corrente

Le due funzioni restituiscono

- TRUE in caso di successo
- FALSE in caso di insuccesso