





## 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



2

## 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



3

## 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



4

## 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



9

## 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




10




## SQL per le applicazioni

Funzioni MySQL per PHP



## Estensione MySQLi

- ▷ MySQLi (MySQL improved) è un'estensione di PHP che consente di interfacciarsi a DB MySQL in modo efficiente
- ▷ Funzionalità supportate
  - Connessione al DB
  - Esecuzione immediata o preparata (istruzioni SQL precedentemente utilizzate e mantenute in cache per successive chiamate) di query SQL
  - Acquisizione e lettura di dati
  - Supporto per stored procedure, query multiple e transazioni



12

## 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');
```



13

## 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());
}
```



14

## 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);
```



15

## Esecuzione di istruzioni 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



16



## Esecuzione immediata

➤ Chiamata alla funzione `mysqli_query()`

- Richiede come parametro l'id della connessione e la query da eseguire, in formato stringa
- In caso di successo restituisce il risultato della query, in caso di insuccesso restituisce FALSE
- `mysqli_error()`: restituisce il testo dell'errore relativo alla funzione Mysql eseguita più recentemente

➤ 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));
```

17



## Esecuzione "preparata"

➤ Passi logici

1. Preparazione della query
2. Assegnazione delle variabili ai parametri della query
3. Esecuzione della query
4. Eventuale ripetizione di 2. e 3. con variabili diverse



18

## Preparazione della query

- ⊃ Chiamata alla funzione `mysqli_prepare()`
  - Richiede come parametri l'identificativo di connessione e la query da eseguire, in formato stringa
  - I parametri all'interno della query sono indicati con un '?'
  - La funzione invia la query a MySQL che ne controlla la validità e ne verifica la correttezza
  - In caso di successo restituisce un oggetto di tipo `mysqli_stmt`, in caso di insuccesso restituisce `FALSE`

## Binding dei parametri della query

- ⊃ Prima di eseguire la query bisogna collegare ciascun parametro con la variabile corrispondente (operazione di "binding")
- ⊃ Chiamata alla funzione `mysqli_stmt_bind_param()`
  - Richiede come parametri l'oggetto restituito da `mysqli_prepare()`, il tipo dei dati e le variabili che devono essere assegnate ai parametri della query
  - In caso di successo restituisce `TRUE`, in caso di insuccesso restituisce `FALSE`

## Esempio di binding

```
//preparazione della query
$stmt = mysqli_prepare($con, "INSERT INTO Forniture VALUES (?, ?, ?)");

$CodF= "F1";
$CodP= "P1"
$Qta= 100;

//binding dei parametri
mysqli_stmt_bind_param($stmt, "ssi", $CodF, $CodP, $Qta);
```

### ▷ Tipo del parametro

- "s": stringa
- "i": numero intero
- "d": numero reale



21

## Esecuzione della query "preparata"

### ▷ Chiamata alla funzione mysqli\_stmt\_execute()

- Richiede come parametro l'oggetto restituito da mysqli\_prepare()
- In caso di successo restituisce TRUE, in caso di insuccesso restituisce FALSE

```
//preparazione della query
$stmt = mysqli_prepare($con, "SELECT Provincia FROM Citta WHERE nome=?");
if (!$stmt)
    die ('Query error; ' . mysqli_error());

$nome= "Torino";

//binding dei parametri
mysqli_stmt_bind_param($stmt, "s", $nome);

//esecuzione della query
mysqli_stmt_execute($stmt);
```



22

## Letture 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



23

## Acquisizione di una riga

- ▷ Esistono diverse possibilità per acquisire una riga della tabella
- ▷ 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)) {
    ...
}
```



24

## Acquisizione di una riga

- ▷ 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



25

## Acquisizione di una riga

- ▷ Chiamata alla funzione `mysqli_fetch_array()`
- È la funzione più generale
  - Richiede come parametro la risorsa restituita da `mysqli_query()` e il tipo di array da riempire (scalare, associativo o entrambi)
  - `MYSQLI_ASSOC`: l'array risultante è di tipo associativo
  - `MYSQLI_NUM`: l'array risultante è di tipo scalare
  - `MYSQLI_BOTH`: l'array risultante è accessibile sia con indice numerico sia con chiave corrispondente al nome del campo



26

## 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";
}

```

27

## Esempi

NomeF	NSoci
Andrea	2
Gabriele	2

```

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

```

28

## Altre funzioni utili

▷ `int mysqli_num_rows(resource $result)`

- Restituisce il numero di righe della risorsa `$result`, o FALSE in caso di insuccesso

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



29

## 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>";
}
```



30

## Altre funzioni utili

```
if( mysqli_num_rows($result) > 0 ){  
    //Intestazione tabella  
    echo "<table border=1 cellpadding=10>";  
    echo "<tr>";  
    for ($i = 0; $i < mysqli_num_fields($result); $i++){  
        $title = mysqli_fetch_field($result);  
        $name = $title->name;  
        echo "<th> $name </th>";  
    }  
    echo "</tr>";  
    // riempimento tabella  
    while ($row = mysqli_fetch_row($result)) {  
        ...  
    }  
}
```

## 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 di connessione e TRUE o FALSE a seconda che si voglia abilitare o disabilitare la modalità auto-commit
  - In caso di successo restituisce TRUE, in caso di insuccesso restituisce FALSE



33

## 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
  - In caso di successo restituisce TRUE, in caso di insuccesso restituisce FALSE
- ▷ `bool mysqli_rollback (mysqli $link)`
  - Esegue il rollback della transazione corrente
  - In caso di successo restituisce TRUE, in caso di insuccesso restituisce FALSE



34