Finanziato dall'Unione europea NextGenerationEU

Agenzia Nazionale Politiche Attive del Lavoro

REGIONE PIEMONTE

GOL Garanzia Occupabilità Lavoratori

SLIDES PHP E SQL

IMMAGINAZIONE E LAVORO DAL 1978

Pagine web statiche e dinamiche

- In una pagina web statica il contenuto viene stabilito nel momento in cui si crea e si memorizza la pagina sul web server.

- In una pagina web dinamica i contenuti della pagina variano in funzione delle informazioni passate dall'utente come input.

Il motore di PHP

- Il codice della pagina dinamica è nascosto all'utente, che riceve sempre una pagina in formato HTML.
- Il web server trasforma la pagina dinamica in una pagina statica generando al volo (on-the-fly) il codice HTML.
- Questa elaborazione è effettuata da un apposito modulo o componente software del server web definito scripting engine (motore di scripting), che assolve alla funzione di interprete del linguaggio.

Request e Response

- Quando l'utente richiede una pagina dinamica effettua una richiesta al server web (Request).
- A questo punto il server, dopo aver elaborato la pagina, restituirà (Response) una pagina il cui contenuto può variare in relazione alla richiesta e alla elaborazione che è stata eseguita sul server da parte dello script dinamico.

Script lato server

- Perché le pagine diventino dinamiche vi si possono inserire degli script.
- Uno script è un codice di programma che viene interpretato ed eseguito.
- Si possono avere script lato client, che vengono interpretati ed eseguiti dal browser, e script lato server, in cui il server, sulla base delle istruzioni di scripting, confeziona una pagina HTML e la invia al client.
- PHP è un linguaggio di scripting lato server.
- Applicazioni tipiche degli script lato server sono le interrogazioni a basi di dati remote.

Architettura LAMP o WAMP

- Per utilizzare PHP è necessario dotarsi di alcuni elementi:
- Web Server (p.e. Apache)
- Interpreter (chiamato engine o motore, p.e. PHP)
- Database Administrator (p.e. MySQL)
- Client Browser (p.e. Firefox)

- L'insieme di tali componenti funzionanti in un server prende il nome di architettura LAMP (in ambiente Linux) o WAMP (in ambiente Windows)

Server HTTP

- Un server HTTP gestisce due flussi di informazioni:

HTTP request: le richieste al server in arrivo dal client;

HTTP response: le risposte del server inviate al client.

- Gli script lato server possono interagire con vari oggetti della conversazione client-server, tra cui HTTP request e HTTP response.

Variabili

- I loro nomi devono cominciare sempre con il carattere $ (dollaro)
- Il linguaggio PHP è "case sensitive": differenzia i caratteri maiuscoli dai caratteri minuscoli.

Pertanto il nome $totale è diverso dal nome $Totale e anche dal nome $TOTALE.

I tre nomi indicano cioè tre variabili diverse.

- In PHP le variabili non sono tipizzate.

Questo significa che:

- non è necessario dichiararne nome e tipo per poterle utilizzare, ma è sufficiente assegnare loro un valore.

- il tipo di dato contenuto in esse può variare durante lo svolgimento dello script.

Assegnazione di valori alle variabili

- Si può assegnare un valore ad una variabile con l'operatore di assegnazione = (uguale) .
- A sinistra dell'operatore di assegnazione deve trovarsi SEMPRE E SOLO il nome di una variabile.
- A destra si può trovare:

un valore (di qualsiasi tipo) ;

un'altra variabile ;

un'espressione (combinazione di valori, variabili, operatori aritmetici, parentesi tonde) .

- L'istruzione di assegnazione deve terminare SEMPRE con ; (punto e virgola) .

Variabili numeriche

```
$a = 5;

$b = 567.32;

$c = $a;

$d = $b - ( ($b/100) * 20 );

$quoziente = $a / 2;

$resto = $a % 2;
```

- I decimali sono separati dagli interi dal carattere . (punto decimale) .
- Gli operatori aritmetici sono + (più) , - (meno) , * (per) , / (diviso) .
- L'operatore aritmetico % (modulo) indica il resto intero di una divisione, ed è preceduto dal dividendo e seguito dal divisore.

Variabili alfanumeriche

```
$a = "Mario";

$b = "";

$c = "Cinzia e " . $a;
```

- Le sequenze di caratteri alfanumerici sono dette "stringhe".
- Sono assegnate alle variabili delimitandole all'inizio e alla fine con i caratteri " (doppio apice) o ' (singolo apice).

Per evitare confusione si consiglia di usare, quando possibile, il doppio apice.

- Assegnando valori alfanumerici ad una variabile si possono concatenare più stringhe usando l'operatore . (punto), che in questo caso assume questa funzione.

C-like

- Il linguaggio PHP è C-like, ovvero deriva dal linguaggio C, come altri linguaggi ampiamente diffusi a tutt'oggi (Java, JavaScript, JSP, C#, C++) .
- Come esso è "case sensitive" anche per quanto riguarda la sintassi delle istruzioni.
- Pertanto tutti i costrutti utilizzati si attengono alla sintassi originaria del C: test condizionali (if), scelte multiple (switch), iterazioni pre e post condizionali (while , do...while), iterazioni enumerative (for) .
- I blocchi di istruzioni ad essi sottesi sono delimitati da { e } (parentesi graffe) .
- Non essendo il PHP un linguaggio tipizzato, l'utilizzo di funzioni definite dall'utente assume la generica forma function NomeFunzione() {...} .

Condizioni

- Una condizione è composta SEMPRE da un operatore di confronto che separa due entità le quali possono essere valori, variabili o espressioni di qualsiasi tipo.
- Gli operatori di confronto possono essere : == (uguale) , <= (minore o uguale) , >= (maggiore o uguale) , != (non uguale) , < (minore) , > maggiore .

```
if ( $a == 3 ) { ... }

if ( $b > $a ) { ... }

if ( $a != ( $z*3 ) ) { ... }

if ( ( $totale + $iva ) <= ( $tettomassimo * 2 ) ) { ... }
```

- Il risultato di una condizione posta può essere SEMPRE E SOLO un valore VERO o FALSO. In un costrutto if ciò determina quale dei due blocchi di istruzioni verrà eseguito (uno solo dei due).

Il costrutto if

- Per vincolare lo svolgimento di codice ad una condizione è valida la seguente sintassi:

```
if ( condizione )
{
   ...eseguo blocco istruzioni per condizione vera...
}
else
{
   ...eseguo blocco istruzioni per condizione falsa...
}
```

Scelte multiple

- E' possibile fare eseguire diversi blocchi di codice a seconda del valore contenuto in una variabile di controllo.
- Tali blocchi è opportuno siano mutualmente esclusivi, e cioè permettano al flusso dello script di seguire solo una strada tra quelle possibili.
- A tale scopo si gestisce ciascuno dei rami della scelta multipla terminando il percorso con una istruzione di interruzione (“break”).

Il costrutto switch

- Per vincolare lo svolgimento di codice al valore di una variabile è valida la seguente sintassi:

```
switch ( variabile )
{
  case <primo> :
  {
    ...eseguo blocco istruzioni per il valore <primo> della variabile...
  break ;
  }
  …altri casi...
}
```

Iterazioni

- E' possibile in uno script la ripetizione di un blocco di istruzioni.
- Tale ripetizione è vincolata al risultato di una condizione, solitamente per vero (true) piuttosto che per falso (false).
- Si possono classificare le iterazioni principalmente in

pre-condizionali (condizione posta prima del blocco da ripetere);

post-condizionali (condizione posta dopo il blocco da ripetere);

enumerative (ripetizione del blocco vincolata ad un contatore o a un elenco di dati).

I costrutti while e do...while

- Per vincolare l'iterazione del codice ad una condizione sono valide le seguenti sintassi:

```
while ( condizione )  {
   ...blocco istruzioni finché condizione vera...
}

do  {
   ...blocco istruzioni finché condizione vera...
}
while ( condizione );
```

I costrutti for e foreach

- Per vincolare l'iterazione del codice al valore di un contatore o a un elenco di dati sono validi le seguenti sintassi:

```
for ( valore di partenza del contatore; condizione; incremento del contatore )  {
   ...eseguo blocco istruzioni finché condizione vera...
}

foreach (elenco dati as dato corrente)  {
   ...eseguo blocco istruzioni per ciascun dato corrente...
}
```

I moduli HTML

- In una pagina HTML è possibile inserire degli oggetti facenti funzione di modulo (form).
- Tali oggetti sono utilizzati per contenere altri oggetti HTML come caselle di testo (text), bottoni (submit) ed altri, i quali permettono l'invio di dati da parte dell'utente al web server per poter elaborare pagine dinamiche in funzione di essi.
- In un form occorre necessariamente specificare alcuni attributi :

il nome (name);

il metodo di invio dei dati (method);

la pagina contenente lo script che riceverà i dati (action).

L'array POST

- L'array POST è una struttura contenente un elenco di dati provenienti dagli oggetti contenuti in un form HTML.
- Per poterne disporre è necessario che il metodo di uscita dei dati dal form sia “post” .
- Tali dati sono ricevibili nella pagina di destinazione indicata dal form .
- Ciascun dato è accessibile per mezzo del suo nome originario secondo la sintassi: $_POST[' <nome> '] .

Utilizzo dei dati di un array POST

- In uno script PHP, è possibile utilizzare i dati provenienti da un post a seconda del tipo di oggetto che li ha generati.
- Se si tratta di “submit”, è possibile verificarne l'esistenza per mezzo della funzione isset() .

Qualora si sia raggiunta la pagina per mezzo di un bottone con uno specifico nome, la condizione :

isset($_POST [' <nomesubmit> '] sarà vera, altrimenti no.

- Se si tratta di “text”, è possibile recuperare il valore in esse contenute in formato stringa, e utilizzarlo.

In una assegnazione : $a = $_POST [' <nometext> '] ;

Valori numerici casuali

- Per generare valori numerici casuali compresi in un intervallo esiste la funzione :

rand(<inferiore><superiore>);

- Per mezzo di questa è possibile ottenere un valore intero generato in modalità casuale ad ogni chiamata e comprso nell'intervallo <inferiore>...<superiore> estremi inclusi.

Tabelle HTML

- Per visualizzare dati in forma tabellare è possibile avvalersi in HTML delle tabelle
- Queste utilizzano tre tags in modalità gerarchica:

<TABLE border=' larghezza' >…</TABLE> come tabella

<TR>…</TR> come riga

<TD> ...dato... </TD> come colonna (o cella)

- Ciascuno di essi contiene il tag gerarchicamente inferiore

Tabelle generate dinamicamente

- Utilizzando opportunamente due iterazioni for() nidificate, la più esterna per le righe (<TR>) e la più interna per le colonne (<TD>), è possibile creare dinamicamente tabelle HTML di R righe per C colonne per mezzo di uno script PHP. I tag di tabella (<TABLE>) sono esterni alle iterazioni.

```
echo(“<TABLE border='1'>”);
for($i=0;$i<10;$i++)  {
  echo (“<TR>”);
  for($j=0;$j<10;$j++)  {
       echo (“<TD> </TD>”);   }
  echo (“</TR>”);  }
echo(“</TABLE>”);
```

Arrays

- Gli arrays sono strutture che permettono di gestire gruppi di dati con lo stesso identificatore.
- Ciascun singolo dato è localizzabile per mezzo di un indice, solitamente numerico ma volendo anche alfanumerico.
- L'indice viene rappresentato racchiudendolo tra parentesi quadre [ ] .
- In PHP gli arrays possono contenere tipi di dati eterogenei, anche nulli.
- La lunghezza di un array (quantità di elementi) può variare durante l'utilizzo dell'applicazione, e si attesta alla quantità degli elementi effettivamente esistenti.
- Gli arrays possono avere anche più dimensioni. Per ciascuna dimensione occorrerà un indice che la rappresenti.

Arrays impliciti ed espliciti

- E' possibile creare un array contenente dati in forma esplicita:

```
$vett = [ 34, 12, 10, 26 ] ;
```

- oppure in forma implicita:

```
$vett = array();
$vett[0] = 34;
$vett[1] = 12;
$vett[2] = 10;
$vett[3] = 26;
```

- In entrambi i casi evidenziati l'array conterrà 4 elementi, a partire dall'elemento 0 fino all'elemento 3.

Iterazioni sugli arrays

- Solitamente gli arrays vengono percorsi per mezzo di una iterazione enumerativa (for), in cui il contatore rappresenta l'indice della posizione corrente dell'array ($vett[$i]):

```
for($i=0; $i<count($vett); $i++)
{
   echo $vett[$i].”<BR>”;
}
```

La funzione count(<array>) restituisce la quantità di elementi di un array.

Arrays di oggetti HTML

- E' possibile utilizzare arrays di oggetti HTML, specificando per ciascuno di essi un identificatore comune e un indice che contraddistingua il singolo elemento.
- Tali oggetti sono recuperabili nell'array di POST semplicemente specificandone l'indice, oltre che il nome.
- Possono essere creati sia staticamente in HTML, sia dinamicamente, avvalendosi di uno script iterativo.
- Qualsiasi oggetto HTML può essere creato in forma di array.

Array di oggetti HTML statico

- Può essere creato nel seguente modo:

```
<FORM name='F1' method='post' action='index.php'>
  <INPUT type='text' name='A[0]' value='0'><BR>
  <INPUT type='text' name='A[1]' value='0'><BR>
  <INPUT type='text' name='A[2]' value='0'><BR>
  <INPUT type='text' name='A[3]' value='0'><BR>
  <INPUT type='submit' name='B1' value='Invia'>
</FORM>
```

Array di oggetti HTML dinamico

- Può essere creato nel seguente modo:

```
<FORM name='F1' method='post' action='index.php'>
  <?php
  for ($i=0; $i<4; $i++ )
  {
    echo “<INPUT type='text' name='A[“ . $i . ”]' value='0'><BR>”;
  }
 ?>
  <INPUT type='submit' name='B1' value='Invia'>
</FORM>
```

Recupero arrays HTML da POST

- Per recuperare i dati di un array HTML da un array di POST si può utilizzare un'iterazione enumerativa (for) :

```
for($i=0; $i<count( $_POST [ 'A' ]); $i++)
{
   echo $_POST[ 'A' ][$i].”<BR>”;
}
```

In questo caso i dati sono leggibili come alfanumerici perchè provenienti da oggetti TEXT.

Arrays HTML bidimensionali

- Utilizzando un secondo indice si possono gestire arrays bidimensionali di oggetti HTML, come peraltro di arrays di dati PHP.

```
for ($i=0; $i<4; $i++ )
  for ($j=0; $j<5; $j++ )
    echo “<INPUT type='text' name='A[“ . $i . ”][“ . $j . ”]' value='0'><BR>”;
```

- La lettura dall'array di POST avviene specificando entrambi gli indici

```
for ($i=0; $i<count($_POST[ 'A' ]); $i++ )
  for ($j=0; $j<count($_POST[ 'A' ][$i]); $j++ )
    echo $_POST[ 'A' ][$i][$j].”<BR>”;
```

Oggetti HTML nascosti

- E' possibile utilizzare oggetti nascosti (HIDDEN) all'interno di un FORM per far transitare dati non visibili dall'utente.

```
<FORM name='F1' method='post' action='index.php'>
<?php
  for ($i=0; $i<4; $i++ )
      echo “<INPUT type='hidden' name='H[“ . $i . ”]' value='0'><BR>”;
?>
<INPUT type='submit' name='B1' value='Invia'>
</FORM>
```

La lettura dall'array di POST avviene esattamente come per gli oggetti visibili (TEXT).

```
for ($i=0; $i<count($_POST[ 'H' ]); $i++ )
     echo $_POST[ 'H' ][$i].”<BR>”;
```

Variabili di sessione

- Per far transitare i dati tra le pagine di una applicazione senza passare da un FORM HTML, è possibile utilizzare le variabili di sessione.
- E' necessario comunicare alla pagina PHP l'accesso all'utilizzo di tal variabili per mezzo della funzione session_start() ad inizio pagina.

```
<?php
  session_start();
  $_SESSION['livello'] = 1;
```

Visibilità delle variabili $_SESSION

- Analogamente all'array $_POST da un FORM è quindi possibile far esistere $_SESSION, il quale può contenere singoli dati o arrays di dati non tipizzati, utilizzabili in lettura e scrittura in tutte le pagine PHP aperte chiamando la funzione session_start() all'avvio.
- Una variabile di sessione comincia ad esistere nel momento in cui gli viene attribuito un valore:

```
$_SESSION['livello'] = 1;
```

- Si può assegnare ad una variabile locale il valore di una variabile di sessione:

```
$a = $_SESSION['livello'];
```

Esistenza di variabili $_SESSION

- Per verificare l'esistenza di variabili di sessione è possibile usare la funzione isset():

```
if ( ! isset ($_SESSION['A'] )
{
   $_SESSION['A'] = ”esiste”;
}
```

- E' possibile annullare l'esistenza di una variabile di sessione ponendola a null:

```
$_SESSION['A'] = null;
```

Arrays associativi

- L'array associativo è un array i cui elementi sono accessibili mediante nomi, quindi stringhe anziché indici puramente numerici. Questo non comporta però l'obbligo di utilizzare solo un tipo di indice: alcuni elementi dell'array possono avere un indice numerico, altri un indice di tipo stringa.

```
$auto['marca'] = “FIAT”;
$auto['modello'] = “500L”;
$auto['colore'] = “Blu”;
$auto['anno'] = 1956;
$auto['revisionata'] = true;
```

Arrays associativi tabellari

- Un utilizzo comune degli arrays associativi in PHP è quello in forma tabellare, in cui l'indice delle colonne è rappresentato da nomi (campi) e quello delle righe da numeri.

```
$giocatori = array(
array("nome"=>"Marco","vincita"=>0,"importo"=>0),
array("nome"=>"Franco","vincita"=>0,"importo"=>0),
array("nome"=>"Giorgio","vincita"=>0,"importo"=>0),
array("nome"=>"Piero","vincita"=>0,"importo"=>0),
array("nome"=>"Gianni","vincita"=>0,"importo"=>0)
);
```

Visualizzare arrays associativi

- Analogamente ai semplici arrays, si è soliti visualizzare gli arrays associativi percorrendoli per l'indice numerico delle righe per mezzo di una iterazione enumerativa, e accedendo a ciascuna colonna (campo) esplicitandola:

```
for($i=0;$i<count($giocatori);$i++)    {
   echo $i.” ”;
   echo $giocatori[$i]['nome']." ";
   echo $giocatori[$i]['vincita']." ";
   echo $giocatori[$i]['importo']."<BR> ";
}
```

Oggetto tendina in HTML

- E' possibile creare oggetti tendina in HTML per mezzo del tag <SELECT name='nome'>...</SELECT>. In tali oggetti l'utente potrà selezionare tra varie voci, per ciascuna delle quali sarà indicato un valore e una denominazione.
- Il valore associato alla denominazione scelta risulterà quello selezionato al recupero dell'oggetto nella pagina di destinazione. Ciascuna coppia valore-denominazione viene associata all'oggetto tendina per mezzo del tag <OPTION>...</OPTION>.

```
<SELECT name='SOCI'>
  <OPTION value='0' >Non socio</OPTION>
  <OPTION value='1' >Ordinario</OPTION>
  <OPTION value='2' >Benemerito</OPTION>
</SELECT>
```

Creazione dinamica di tendine

- E' possibile in PHP creare dinamicamente oggetti tendina in un form recuperandone le coppie valore-denominazione da arrays, preferibilmente associativi.

```
echo “<SELECT 'nazione'>”;
for($i=0;$i<count($elenco);$i++) {
   echo “<OPTION value = ' ”.$elenco[$i]['sigla'].” ' >”;
   echo $elenco[$i]['nazione'];
   echo “</OPTION>”;
}
echo “</SELECT>”;
```

Il linguaggio SQL

- Il linguaggio SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- creare e modificare schemi di database (DDL - Data Definition Language);

- inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);

- interrogare i dati memorizzati (DQL - Data Query Language);

- creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

- Nonostante il nome, non si tratta dunque solo di un semplice linguaggio di interrogazione, ma alcuni suoi sottoinsiemi si occupano della creazione, della gestione e dell'amministrazione del database.

Le tabelle SQL

- Un database SQL è a tutti gli effetti un insieme di tabelle contenenti dati relazionati tra loro.
- Ciascuna tabella può contenere diversi record, accomunati da una struttura fissa per la quale vengono a priori definiti i campi.
- Ciascun campo di una tabella viene definito almeno per tipo e dimensione, oltre che per altri parametri non sempre necessari.
- Un particolare tipo di campo è la chiave primaria numerica autoincrementale. Questo tipo di campo conterrà un valore che il database assegnerà in maniera automatica e progressiva ad ogni inserimento di un nuovo record, in modo da garantirne l'univocità.

Creare una tabella SQL

- Per creare una tabella SQL occorre definirne la struttura del record, secondo uno schema del quale questo è un esempio:

```
CREATE TABLE utenti (
ID_utenti INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
Nome VARCHAR(50) NOT NULL,
Cognome VARCHAR (50) NOT NULL,
Email VARCHAR(60)
);
```

Inserire records in una tabella SQL

- Per inserire records in una tabella SQL si deve utilizzare il comando INSERT, del quale il seguente è un esempio:

```
INSERT INTO utenti (Nome,Cognome,Email)
VALUES ('Giuseppe','Verdi','gverdi@gmail.com');
```

Se si intende inserire record in una tabella dove è presente una chiave primaria numerica autoincrementale, non sarà necessario specificarne il valore poiché questo verrà generato automaticamente in maniera univoca dal database, sebbene sia comunque possibile forzarlo (in alcuni specifici casi).

Connettersi a un database da PHP

- L'interprete PHP dispone di una serie di funzioni per operare su un dbms MySQL.
- Le prime operazioni da svolgere per interfacciarsi con un dbms MySQL da PHP sono la creazione di una connessione e l'apertura del database su cui si intende operare:

```
$mysqli = new mysqli("localhost", "root", "", "anagrafica");
```

Esecuzione di una query da PHP

- Qualsiasi operazione di manipolazione o selezione dati su tabelle SQL può essere effettuata inviando la stringa della query da svolgere al dbms MySQL per mezzo di una apposita funzione:

```
$query = “SELECT Nome,Cognome FROM utenti”;

if (!$risultato = $mysqli->query($query)) { echo $query; }
```

Nel caso questa sia una query di selezione il dbms MySQL restituirà un recordset contenente i dati, il quale verrà recepito da PHP come un array associativo in cui la struttura (campi) di ciascun elemento sarà quella imposta nella query.

Visualizzazione di un recordset

- Essendo il recordset generato da una query di selezione recepito come un array associativo, sarà possibile visualizzarlo in PHP nelle stesse modalità utilizzate in precedenza.
- In particolare è possibile visualizzarlo in forma tabellare, utilizzando i tag delle tabelle HTML, in cui ciascun record costituisce una riga e ciascun campo del record una colonna.
- L'array associativo contenente il recordset deve essere percorso un record alla volta, e per ciascuna delle letture della singola riga, che generano a loro volta un altro array associativo, occorre recuperare i singoli campi scorrendo tale array.

Recordset in forma tabellare

```
echo "<TABLE border='1'>";
echo "<TR>";
for($i=0;$i<$risultato->field_count;$i++)
{
    echo "<TD><B>".$risultato->fetch_field_direct($i)->name."</B></TD>";
}
echo "</TR>";
while ($row=$risultato->fetch_row())
{
   echo "<TR>";
   for($i=0;$i<$risultato->field_count;$i++)
   {
      echo "<TD>".$row[$i]."</TD>";
   }
   echo "</TR>";
}
echo "</TABLE>";
```

Cancellare records da una tabella SQL

- Per cancellare records da una tabella SQL si deve utilizzare il comando DELETE, del quale il seguente è un esempio:

```
DELETE FROM utenti WHERE ID_utenti=2;
```

La clausola WHERE determina un criterio per la cancellazione di uno o più record inclusi nel filtro specificato; nell'esempio soprastante verrà cancellato un unico record, poiché filtrato per mezzo di un valore della chiave primaria, notoriamente univoca.

```
DELETE FROM utenti WHERE Cognome='Rossi';
```

Nell'esempio soprastante invece verranno cancellati tutti i records che soddisfano il filtro per cognome, che potrebbero anche essere più di uno.

Modificare records di una tabella SQL

- Per modificare records di una tabella SQL si deve utilizzare il comando UPDATE, del quale il seguente è un esempio:

```
UPDATE utenti SET Cognome='Rossini' WHERE Cognome='Rossi';
```

Nell'esempio soprastante il campo Cognome verrà rimpiazzato con un nuovo valore per tutti i record che soddisfano il filtro per cognome.

```
UPDATE dati SET importo=importo+10, qta=qta+1
WHERE Cognome LIKE '%ss%';
```

Nell'esempio soprastante i campi importo e quantità vengono variati aumentandoli di un certo valore, e i record a cui verrà applicato l'aggiornamento saranno quelli che soddisferanno il filtro WHERE...LIKE, che prevede in questo caso tutti i cognomi contenenti la stringa 'ss', preceduta o seguita da qualsiasi carattere.

Creazione di tendine da una tabella SQL

- E' possibile in PHP creare dinamicamente oggetti tendina in un form recuperandone le coppie valore-denominazione da un recordset proveniente da una query di selezione, essendo questo in forma di array associativo.

```
$query = "SELECT ID_utenti, Nome FROM utenti";
if (!$risultato = $mysqli->query($query)) {   echo $query;  }
echo "<SELECT name='NOMI'>";
while ($row=$risultato->fetch_row())    {
        if($m==$row[0])
            echo "<OPTION value=".$row[0]." SELECTED>".$row[1]."</OPTION>";
        else
            echo "<OPTION value=".$row[0].">".$row[1]."</OPTION>";
}
echo "</SELECT><BR>";
```

Relazioni tra tabelle SQL

- Una relazione è un legame logico che permette di aggregare dati memorizzati in due distinte tabelle.
- La relazione viene definita tra due campi di raccordo definiti nelle strutture delle due tabelle.
- Tale relazione permette il collegamento dei dati presenti nelle due tabelle per l'effettuazione di query di selezione. Il recordset risultante conterrà pertanto dati provenienti da entrambe le tabelle.
- Il tipo di relazione più comune è uno-a-molti, dove un record presente in una prima tabella può essere collegato a più records presenti in una seconda tabella (es.: studenti-voti) .
- La tecnica più comune di collegamento è l'utilizzo della chiave primaria (PK) come campo di raccordo nella prima tabella con una chiave esterna (FK) come campo di raccordo della seconda tabella.

JOIN tra tabelle SQL

- Una query di selezione può recuperare dati da due distinte tabelle definendo una relazione (JOIN) tra due campi presenti nelle strutture:

```
SELECT studenti.Cognome, voti.Voto FROM voti
INNER JOIN studenti
ON voti.ID_studenti=studenti.ID_studenti;
```

- Nell'esempio soprastante verranno recuperati il cognome dello studente dalla tabella studenti e i voti dello studente dalla tabella voti (un record nel recordset per ciascun voto), stabilendo che le tabelle voti (molti) e studenti (uno) saranno collegati per mezzo dei campi voti.ID_studenti (chiave esterna presente in voti) e studenti.ID_studenti (chiave primaria presente in studenti) .
- Utilizzando più tabelle è necessario far precedere il nome di ciascun campo dal nome della tabella in cui si trova.

Raggruppamento di dati in query SELECT

- E' possibile raggruppare i dati in operazioni di conteggio, somma, media, ecc. utilizzando la clausola GROUP BY .

```
SELECT studenti.Cognome, AVG(voti.Voto) AS media
FROM voti INNER JOIN studenti
ON voti.ID_studenti=studenti.ID_studenti
GROUP BY studenti.Cognome;
```

- Nell'esempio soprastante verrà calcolata la media dei voti per ciascuno studente (un record nel recordset per ciascuno studente), poiché nella clausola GROUP BY utilizzata è stato specificato come criterio di raggruppamento il cognome dello studente.

Funzioni di aggregazione in SQL

- Le principali funzioni di aggregazione dei dati nelle query di selezione sono le seguenti:
- Somma: SUM(voti.voto) AS somma
- Conteggio: COUNT(voti.ID_voti) AS totali
- Media: AVG(voti.voto) AS media
- Massimo: MAX(voti.voto) AS massimo
- Minimo: MIN(voti.voto) AS minimo
- Tali funzioni sono anche combinabili in espressioni strutturabili per l'ottenimento di campi calcolati.