ISIS Keynes – *Gazzada Schianno*

MANUALE SQL – PHP/HTML

Sommario

Capitolo 1- Introduzione SQL

1.1- Tipi di Dati

Tipo	Descrizione
VARCHAR	Stringa variabile
INT	Intero
VALUTA	Per valori riguardo il prezzo (come un float)
BOOLE	Tipo logico: Valore=0 (False) Valore≠0 (Vero)
DATA	Data (gg/mm/aaaa)
YEAR	Anno
TIME	Tempo

Capitolo 2- Istruzioni DDL

Per creare un Database:

```
CREATE DATABASE <NomeDatabase>
```

Per cancellare un Database:

```
DROP DATABASE <NomeDatabase>
```

2.1- DDL - Operazioni sulle tabelle

2.1.1- Creazione di una tabella

Per creare una nuova tabella:

```
CREATE TABLE <NOMETABELLA>
        (<Attrubito1>  <Tipo1>  [<VincoloAttributo1>],
        <Attrubito2>  <Tipo2>  [<VincoloAttributo2>],
        …                …                …                …
        <AttrubitoN>  <TipoN>  [<VincoloAttributoN>],
        [<VincoliTabella>] ) ;
```

Vincoli di ogni singolo attributo

- **NOT NULL**: l'attributo deve avere per forza un valore (il campo non deve rimanere vuoto);
- **DEFAULT**: assegna un valore di default se il campo non viene riempito.
- **CHECK(<Condizione>)**: questo comando permette di inserire un valore in base a una condizione che impostiamo noi.

Operatori di confronto

Codice	Descrizione
<Attributo> **IN** (<Valore1>, …, <ValoreN>)	Richiede che il valore di <Attributo> sia tra quelli specificati in <Valore1>, …, <ValoreN>
<Attributo> **BETWEEN** <Min> **AND** <Max>	Richiede che Il valore di <Attributo> sia compreso tra i valori <Min> e <Max>
<Attributo> **NOT BETWEEN** <Min> **AND** <Max>	Richiede che il valore **non** sia compreso
<Attributo> **LIKE** <Espressione1>	Richiede che il valore di <Attributo> assuma il formato specificato da <Espressione1>
<Attributo> **NOT LIKE** <Espressione1>	Richiede che il valore **non** assuma il formato specificato

Il carattere "%" rappresenta una sequenza di più caratteri. Per il singolo carattere si usa "_" (underscore).

Vincoli della tabella

Per indicare una chiave primaria (ci possono essere più attributi che compongono la chiave primaria; gli attributi selezionati devono essere dichiarati **NOT NULL**):

PRIMARY KEY (<Attributo1>, …, <AttributoN>)

Per indicare un attributo univoco (non ci devono essere valori uguali nella stessa tabella; **non** si applica sulle chiavi primarie):

UNIQUE (<Attributo1>, …, <AttributoN>)

Per indicare una condizione:

CHECK (<Espressione1>)

CHECK (valoreA < valoreB)

Per indicare le chiavi esterne:

FOREIGN KEY (<Attributo1>, …, <AttributoN>)
REFERENCES <NOMETABELLA> (<Attr1>, ..., <AttrN>)
[[**ON DELETE** | **ON UPDATE**] **RESTRICT** | **CASCATE** | **SET NULL** | **SET DEFAULT** | **NO ACTION**]

Le operazioni sotto vengono eseguite con la modifica (ON UPDATE) o la cancellazione (ON DELETE) della tabella.

RESTRICT: l'azione si limita alla riga corrispondente.
CASCATE: avvia una cancellazione (o modifica) in cascata di tutte le righe corrispondenti in tutte le tabelle correlate.
SET NULL: mette il valore *null* alle righe corrispondenti.
SET DEFAULT: mette il valore impostato di default alle righe corrispondenti.
NO ACTION: non viene eseguita alcuna azione (**N.B.** è il parametro di default sia per ON DELETE che ON UPDATE se non viene impostato nient'altro).

2.1.2- Modifica di una tabella

Per modificare una tabella:

ALTER TABLE <NOMETABELLA>

Per aggiungere un nuovo attributo (colonna) alla tabella:

ALTER TABLE <NOMETABELLA>
    ADD <NomeAttributo1> <Tipo1> [**FIRST** | **AFTER** <NomeAttributo2>]

FIRST serve per inserire la colonna all'inizio della tabella e AFTER per inserirlo dopo l'attributo <NomeAttributo2>.

Di default lo aggiunge alla fine della tabella.

Per eliminare un attributo:

ALTER TABLE <NOMETABELLA>
    DROP <NomeAttrubuto>

Per modificare un attributo:

ALTER TABLE <NOMETABELLA>
    CHANGE <VecchioAttrubuto> <NuovoAttributo> <NuovoTipo>

2.1.3- Cancellazione di una tabella

Per cancellare una tabella:

DROP TABLE <NOMETABELLA> [**RESTRICT** | **CASCATE** | **SET NULL**]

RESTRICT: È di default; non permette la cancellazione della tabella se essa è associata ad altre.
CASCATE: Cancellazione in cascata di tutte le tabelle collegate.
SET NULL: Pone a NULL tutti i valori delle chiavi interessate.

2.2- Operazioni sugli indici

Per creare un indice:

CREATE [**UNIQUE**] **INDEX** <NomeIndice>
ON <NOMETABELLA> (<Attributo1>, <Attributo2>, …, <AttributoN>)

- La clausola UNIQUE crea un indice su attributi chiave.

Per eliminare un indice:

DROP INDEX <NomeIndice> **ON** <NOMETABELLA>

Capitolo 3- Istruzioni DML

3.1- DML - Operazioni sui dati

Per inserire una nuova riga in tabella:

```
INSERT INTO  <NOMETABELLA>  [<Attributo1>, <Attributo2>, …, <AttributoN>]
        VALUES (<Valore1>, <Valore2>, …, <ValoreN>)
```

Per modificare la riga di una tabella:

```
UPDATE <NOMETABELLA>
SET <Attributo1> = <Espressione1>
    <Attributo2> = <Espressione2>
        ...                ...
[WHERE <Condizione>]
```

Il WHERE è opzionale: serve per modificare un gruppo di righe.

Per cancellare le righe di una tabella:

```
DELETE FROM <NOMETABELLA>
[WHERE <Condizione>]
```

WHERE

La clausola WHERE può avere al suo interno molti operatori.

Operatore	Descrizione
AND ; OR ; NOT	Operatori logici per combinare più condizioni
IS NULL ; NOT NULL	Per indicare se l'attributo è di tipo null o no
= ; > ; < ; >= ; <= ; <>	Operatori di confronto (pag. 5)

3.2- DML - Query

Alias

Noi usiamo gli Alias per dare nomi alle tabelle e agli attributi. Per dare un alias si usa la clausola AS.

3.2.1- Operazioni relazionali

L'operatore di proiezione:

```
SELECT DISTINCT <Attributo1>, <Attributo2>, ..., <AttributoN>
FROM <NOMETABELLA>
```

- Il DISTINCT serve per eliminare le t-uple duplicate.

L'operatore di restrizione:

```
SELECT *
FROM <NOME TABELLA>
```

3.2.2- Join

La giunzione naturale (o inner join) serve per unire una o più tabelle aventi anche grado e cardinalità differenti.
La sintassi:

```
<TABELLA1> INNER JOIN <TABELLA2> ON <Condizione>
```

Inner join usando il SELECT

La sintassi:

```
SELECT <ListaAttributi>
FROM <TABELLA1>, <TABELLA2>
WHERE <TABELLA1>.<AttributoX> = <TABELLA2>.<AttributoX>
```

Left join

La sintassi:

```
<TABELLA1> LEFT JOIN <TABELLA2> ON <Condizione>
```

- Il risultato è dato dalle t-uple della <TABELLA1> e da quelle della <TABELLA2> che hanno un valore corrispondente per l'attributo comune (la condizione).

Right join

La sintassi:

```
<TABELLA1> RIGHT JOIN <TABELLA2> ON <Condizione>
```

- Il risultato è dato dalle t-uple della <TABELLA2> e da quelle della <TABELLA1> che hanno un valore corrispondente per l'attributo comune (la condizione).

Join tra più tabelle

Si effettua un join dietro l'altro; esempio:

```
SELECT DISTINCT <tabella1>.<attributo1>
FROM ((<tabella1> INNER JOIN <tabella2> ON <tabella1>.<id1> = <tabella2>.<fkId1>) INNER JOIN
        <tabella3> ON <tabella2>.<fkId3> = <tabella3>.<id3>)
```

Con il SELECT:

```
SELECT DISTINCT <tabella>.<attributo>
FROM <tabella1>, <tabella2>, <tabella3>
WHERE (<tabella1>.<id1> = <tabella2.fkid1>) AND (<tabella2>.<fkId3> = <tabella3>.<id3>)
```

Self join

Si effettua tramite gli Alias quando abbiamo una situazione di questo tipo:

```
SELECT <tabella1.att1>, <tabella1.att2>, <tabella2.att1>, <tabella2.att2>
FROM <tabella> AS <tabella1>, <tabella> AS <tabella2>
WHERE <tabella1.cod1> = <tabella2.cod2>
```

In questo modo abbiamo usato una sola tabella e abbiamo effettuato i controlli nel *where* su due campi diversi della stessa tabella.

3.2.3- Funzioni di aggregazione

Sono funzioni che permettono di fare calcoli e conteggi. La sintassi generica è:

```
<FunzioneDiAggregazione> ([DISTINCT] <Attributo>)
```

dove <FunzioneDiAggregazione> può essere:

- **COUNT** che conta il numero di elementi della colonna <Attributo> escluse le t-uple con valore null. Scrivendo **COUNT(*)** la funzione conterà TUTTE le righe, anche quelle con valore null. In poche parole, con l'asterisco, otteniamo la cardinalità della tabella.
- **MIN** restituisce il valore minimo della colonna <Attributo>;
- **MAX** restituisce il valore massimo della colonna <Attributo>.
    - MIN e MAX funzionano anche con gli attributi CHAR e lo fanno in ordine alfabetico.
- **SUM** restituisce la somma degli elementi della colonna <Attributo>;
- **AVG** restituisce la media matematica degli elementi della colonna <Attributo>.
    - SUM e AVG non considerano i valori *null*.

Ordinamento

In SQL è possibile ordinare il risultato di una query con la clausola **ORDER BY**.
La sintassi:

ORDER BY <Attributo1> [**ASC** | **DESC**], ..., <AttributoN> [**ASC** | **DESC**]

ASC e DESC stanno rispettivamente per ordine Crescente e Decrescente. Per *default* l'ordine è crescente. Possiamo aggiungere la clausola ASC e DESC in ogni attributo.
Possiamo ordinare per più attributo perché, se esistono t-uple con <Attributo1> uguale, si fa riferimento ad <Attributo2> per ordinarli e così via.

Raggruppamento

La sintassi:

GROUP BY <Attributo1>, …, <AttributoN> [**HAVING** <CondizioneGruppo>]

Questa funzione raggruppa tutte le t-uple aventi <Attributo1>, …, <AttributoN> in comune.

3.2.4- Funzione TIMESTAMPDIFF()

TIMESTAMPDIFF() restituisce un valore dopo aver sottratto due valori datetime.

TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2);

unit
indica l'unità di misura del risultato. Può essere uno dei seguenti valori:
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR

expr1
Prima data o espressione DateTime.

expr2 –
Seconda data o espressione DateTime.

esempio:

Seleziona i maggiorenni

```
SELECT  *
FROM <tabella1>
WHERE TIMESTAMPDIFF(YEAR, <tabella1>.<dataNascita>, CURDATE( ))>=18;
```

3.2.5- Query e subquery annidate

La subquery è quella individuata dalla seconda parola chiave SELECT delimitata all'interno di parentesi tonde.

Esempi:

```
SELECT <att11>, <att12>
FROM <tabella1>
WHERE <att11> > (SELECT AVG(<att21>)
                 FROM <tabella2>)
```

Risultato: abbiamo una tabella con i gli attributi *att11* e *att12* in cui tutti i valori di *att11* sono maggiori della media di *att21* preso da *tabella2*.

3.2.5.1- Conservare i risultati parziali

Possiamo salvare i risultati delle nostre query in due modi:

- Inserendo il risultato come se fosse una t-upla in una tabella;
- Creando una tabella: - Una tabella fisica che occupa memoria di massa;
  - Una Vista che non occupa memoria (vedi capitolo 4).

Le sintassi da usare sono rispettivamente:

```
INSERT INTO <NomeTabellaDestinazione>
<Query>
```

Con l'INSERT INTO, nel SELECT della query ci devono essere tutti gli attributi.

```
CREATE TABLE <NOMETABELLA>
<Query>
```

3.2.6- Tipi di subquery

Finora abbiamo visto delle subquery che producevano delle tabelle. Cosa succede se queste subquery invece producono singoli valori?

ANY e ALL

Questi due operatori si possono utilizzare solo con gli operatori relazionali di confronto (<, >, <=, >=, =, <>)!

La sintassi:

SELECT <ListaAttributi>
FROM <ListaTabelle>
WHERE <Attributo> <OperatoreRelazionale> **ANY**| **ALL** (<Sottoquery>)

Il significato:

- ANY: La clausola WHERE è vera se il valore di <Attributo> compare in almeno uno dei valori forniti dalla <Sottoquery>;
- ALL: La clausola WHERE è vera se il valore di <Attributo> compare in tutti i valori restituiti dalla <Sottoquery>.

IN e NOT IN

La sintassi:

SELECT <ListaAttributi>
FROM <ListaTabelle>
WHERE <Attributo> **IN**| **NOT IN** (<Sottoquery>)

La clausola WHERE è vera se il valore di <Attributo> appartiene (IN) o non appartiene (NOT IN) tra quelli prodotti dalla <Sottoquery>.

EXISTS e NOT EXISTS

La sintassi:

SELECT <ListaAttributi>
FROM <ListaTabelle>
WHERE <Attributo> **EXISTS**| **NOT EXISTS** (<Sottoquery>)

- EXISTS: La clausola WHERE è vera se la <Sottoquery> produce una tabella non vuota;
- NOT EXISTS: La clausola WHERE è vera se la <Sottoquery> produce una tabella vuota.

Capitolo 4- Istruzioni DCL

Il DCL (*Data Control Language*) imposta le politiche relative al controllo e alla sicurezza dei dati. Questo comprende anche stabilire i diritti di accesso e le "viste", ossia la modalità con le quali gli utenti possono vedere il DataBase.

4.1- DCL - Diritti di accesso

GRANT

Usiamo il comando GRANT per concedere a gruppi di persone i diritti di accesso ad un database o ad alcune tabelle del DB.
La sintassi:

```
GRANT <ElencoPrivilegi> ON <NOMEDATABASE>.<NOMETABELLA>
        TO <ElencoUtentiAutorizzati> [PUBLIC]
        [IDENTIFIED BY]
        [WHIT GRANT OPTION]
```

I componenti di <ElencoPrivilegi> indicano i privilegi da concedere agli utenti elencati dopo. Questi privilegi devono essere separati da virgole e sono:

- **ALL**: consente l'accesso globale alla tabella specificata;
- **SELECT**: consente l'accesso in lettura a tutti i campi della tabella specificata;
- **INSERT** [<Attributo>]: consente l'inserimento di dati (eventualmente solo sulla colonna specificata da <Attributo>);
- **UPDATE** [<Attributo>]: consente la modifica di dati (eventualmente solo sulla colonna specificata da <Attributo>);
- **DELETE**: consente la cancellazione di righe della tabella specificata;
- **REFERENCES** [<Attributo>]: consente il riferimento a tutti i campo della tabella (o eventualmente a solo quello specificato da <Attributo>);
- **INDEX**: consente di creare indici sulla tabella specificata;
- **ALTER**: consente di aggiungere o rimuovere colonne dalla tabella specificata;
- **ALL PRIVILEGES**: rappresenta simultaneamente tutti i privilegi possibili riferiti alla tabella specifica.

In <ElencoUtentiAutorizzati> mettiamo un elenco di nomi (separati da virgole) ai quali vengono concessi i privilegi. Per permettere a tutti gli utenti di avere determinati privilegi, useremo la clausola **PUBLIC**.

Con **IDENTIFIED BY** assegniamo ad un utente o un gruppo di utenti con una password i privilegi di accesso (vedi esempio).

Con **WHIT GRANT OPTION** permetteremo ad un utente o un gruppo di utenti di concedere a loro volta privilegi.

REVOKE

Usiamo il REVOKE per rimuovere i permessi di accesso. La sintassi:

REVOKE <ElencoPrivilegi> **ON** <NOMETABELLA> **FROM** <ElencoUtentiAutorizzati> [**PUBLIC**]

4.2- DCL - Le viste

Le viste sono delle tabelle (che non sono fisicamente memorizzare nel DB) dove un utente può avere una visione parziale del DB. Le viste vengono solitamente utilizzate per: *proteggere i dati*, *convertire le unità di misura*, *semplificare la costruzione di query complesse*.

Creare una vista

Una volta definita una vista, è possibile modificarla e interrogarla. Per creare una vista si utilizza la sintassi:

CREATE VIEW <NOMEVISTA> **AS** <Query>

dove:

- <NOMEVISTA> è il nome assegnato alla tabella fittizia della vista;
- <Query> è la query formulata con il comando SELECT.

Eliminare una vista

Per eliminare una vista usiamo la sintassi:

DROP VIEW <NOMEVISTA>

È possibile utilizzare anche il comando GRANT sulle viste, per dare diritti di accesso solo ad alcuni utenti.

Capitolo 5- HTML

L'HTML (HyperText Markup Language) è un linguaggio utilizzato per dichiarare la "struttura" di un documento ipertestuale.

5.1- TAG

Un documento HTML utilizza dei comandi per impostare e presentare la pagina. Questi comandi prendono il nome di TAG e sono racchiusi tra parentesi angolari.

Impostazione pagina

TAG DEGLI ELEMENTI	EFFETTO
<html> …… </html>	Inizio e chiusura del documento
<head> …… </head>	Contiene le informazioni dichiarative, informative, o di impostazione globale del documento
<title> …. </title>	Titolo del documento. Appare sella barra del titolo della finestra del browser
<!-- …….. -->	Racchiude un commento
<body> … </body>	Racchiude il documento vero e proprio

Strutturazione del testo

TAG DEGLI ELEMENTI	EFFETTO	
….	Racchiude un paragrafo. Tra un paragrafo e l'altro c'è un interlinea maggiore. **Attributi**	
	align="left"	Allinea il testo del paragrafo a sinistra
	align="right"	Allinea il testo del paragrafo a destra
	align="center"	Allinea il testo del paragrafo al centro
	align="justify"	Allinea il testo del paragrafo a sinistra e a destra
 	Provoca un ritorno a capo	
<hr>	Inserisce un separatore orizzontale (linea) **Attributi**	
	width="valore%"	Lunghezza della linea rispetto alla pagina

	Size=”valore”	Spessore della linea
	align	*Vedi sopra*
	noshade	*Elimina la tridimensionalità*
<pre> …. </pre>	Permette di riportare esattamente come è scritto, tenendo quindi conto di spaziature, indentazioni, tabulazioni, ecc.	

Formattazione del testo

TAG DEGLI ELEMENTI	**EFFETTO**	
<u> …. </u>	Testo sottolineato	
….	Testo in grassetto	
….	Testo in corsivo	
<strike> … </strike>	Testo barrato	
_{….}	Testo in pedice	
^{….}	Testo in apice	
<blink> …. </blink>	Testo lampeggiante	
 …	Permette di selezionare il font **Attributi**	
	Size=valore (*max* 7)	Grandezza del carattere
	Face=”tipo”	Tipo di carattere (es. Tahoma …)
	Color=colore	Colore carattere: attraverso il nome (es. red, yellow, …) attraverso il codice (es. #FF0000) 3 bytes ex. Attraverso il formato RGB(0,0,255)

Titoli articolati su livelli gerarchici

TAG DEGLI ELEMENTI	EFFETTO	
<h*n* *attributi*> …. </h*n*>	Permette di scrivere intestazione su vari livelli (con caratteri sempre più piccoli), dal primo livello (*n*=1) al sesto livello (*n*=6) **Attributi**	
	align	*Vedi sopra*

Elenchi numerati

TAG DEGLI ELEMENTI	EFFETTO	
<ol type="tipo"> ……	Inizio elenco **Type**	
	"1" (*default)*	Numeri arabi
	"a"	Alfabeto minuscolo
	"A"	Alfabeto maiuscolo
	"i"	Numeri romani minuscoli
	"I"	Numeri romani maiuscoli
	Definizione voce in elenco	

Elenchi puntati

TAG DEGLI ELEMENTI	EFFETTO	
<ul type="tipo"> ……	Inizio elenco **Type**	
	"disc" (*default)*	Pallino pieno
	"circle"	Cerchio vuoto
	"square"	Quadrato pieno
	Definizione voce in elenco	

Tabelle

TAG DEGLI ELEMENTI	EFFETTO	
<table *attributi* > …. <table>	Inizio, fine definizione tabella **Attributi**	
	border=”n_pixel”	Spessore del bordo
	cellspacing=”n_pixel”	Spaziatura tra celle
	cellpadding=”n_pixel”	Spazio dal bordo della cella
	bgcolor=”colore”	Colore dello sfondo
	bordercolor=”colore”	Colore del bordo
	align=”*vedi sopra*”	Posizione della tabella rispetto alla finestra
	width=”n_pixel”	Larghezza in pixel
	width=”n%”	Larghezza in percentuale (*raccomandato*)
<tr> …. </tr>	Inizio, fine riga	
<td *attributi*> …. </td>	Inizio, fine definizione cella	
	colspan=n	Numero di colonne (n) da unire
	rowspan=n	Numero di righe (n) da unire
	align=”vedi sopra”	Posizione del testo all’interno della cella
<th *attributi* >…..</th>	Inizio, fine definizione intestazione colonna (*attributi vedi <td>*)	
<caption *attributi*> .. </caption>	Titolo di una tabella	
	align=”**top**”	Titolo superiore
	align=”**bottom**”	Titolo inferiore

Immagini

TAG DEGLI ELEMENTI	EFFETTO	
<img *attributi* >	Inserimento di un immagine nel documento **attributi**	
	src=”immagine”	Percorso file immagine
	allign=”vedi sopra”	Posizione dell’immagine rispetto alla finestra
	alt=”messaggio”	Messaggio visualizzato nel caso sia impossibile visualizzare l’immagine
	height=”n° di pixel”	Dimensiona l’immagine in altezza
	Width=”n° di pixel”	Dimensiona l’immagine in larghezza
	hspace=”n° di pixel”	Spaziatura in orizzontale
	vspace=”n° di pixel”	Spaziatura in verticale
	border=”n° di pixel”	Larghezza del bordo

5.2- Struttura di un documento HTML

```
<html>
        <head>
                <!-- Qui si inseriscono le informazioni dichiarative o impostazioni globali del documento -->

                <title> Titolo del documento </title>
        </head>
        <body>
                <!-- Qui si inserisce il testo del documento, le immagini, i link, ecc. -->
        </body>
</html>
```

5.3- Gli <input>

Con il tag <input …> possiamo creare textbox, bottoni, checkbox, ecc. La sintassi:

<**input** **type**=”*tipo-di-input*” **name**=”*nome-variabile*” **value**=”*nome-visualizzato*”>

Tipi di type	Descrizione
text	È un textbox.
checkbox	Crea una casellina di spunta.
radio	Ti fa scegliere una sola opzione fra tante. In questo caso si creano tante righe di <input> con lo stesso *name* quante sono le opzioni.
hidden	È un valore nascosto.
submit	È un comando che serve per confermare il <form> (vedi 5.4).

<Textarea>

È una casella di testo più grande, dove possiamo decidere noi il numero di righe e di colonne per la grandezza del box. La sintassi:

```
<textarea name="nome-variabile" cols="n_colonne" rows="n_righe">
        Valore di default
</textarea>
```

<Select>

È una combobox dove viene selezionata una sola voce. La sintassi:

```
<select name='nome-variabile'>
        <option> nome selezione1 </option>
        <option> nome selezione2 </option>
</select>
```

5.4- Il <form>

Il form è un tag per richiamare un'altra pagina. Al termine di un form c'è sempre un comando di tipo "submit" che indica una conferma. La sintassi:

```
<form  action="Nome-pagina-successiva.php" method="get/post" > …. </form>
```

action: indichiamo la pagina successiva da aprire.

method: la differenza tra il post e il get è che con il post non vengono visualizzate nell'URL del browser le variabili che passiamo alla pagina successiva, invece col get si.

5.5- Collegamento ipertestuale

Un collegamento ipertestuale richiama un'altra pagina quando viene cliccato. La sintassi:

```
<a href="URL"> testo visualizzato </a>
```

Composizione URL

L'URL è composto da un sito web (nel nostro caso una pagina .php) e dalle variabili.

? → serve per separare il sito o la pagina dalle variabili.
& → serve per separare le variabili una dall'altra.
\" → serve per far passare all'html le " la stessa cosa succede con ' (quando siamo in php).

Capitolo 6- PHP

6.1- Introduzione e principali funzioni

Il PHP è un linguaggio lato server utilizzato per sviluppare le pagine web dinamiche.

TAG / Funzioni	Descrizione
<?php …. ?>	Tag per aprire e chiudere il php.
$nome_variabile	Per definire le variabili.
$_REQUEST[“*nome_variabile*”]	Per estrapolare dall’URL il valore di *nome_variabile.*
echo”….”	Funzione di output.

6.2- Gli array

Gli array in php sono dinamici. Gli indici degli array in php sono alfanumerici, quindi possono essere sia numeri che caratteri.

Esempio:

```
$a["zero"]="ValoreZero";      //l’indice è “zero”
$a["uno"]="ValoreUno";        //l’indice è “uno”
$a[]="prossimo";        //se non metto niente il php gli assegna indici numerici che partono da 0
```

Gli array li posso creare in tre modi: assegnando dei valori manualmente, prelevandolo da una pagina con form e con la funzione *array*.

La funzione array

```
$a=array("zero "=>"ValoreZero "," uno "=>" ValoreUno ",1000=>"Valore1000");
//" zero " è l'indice ; " ValoreZero " è il contenuto
```

6.2.1- Cicli di visualizzazione degli array

Con il while, finito l'array il ciclo finisce. Se voglio che si ferma devo aggiungere && e la condizione.

```
<?php
$a=array("zero "=>"ValoreZero "," uno "=>" ValoreUno ",1000=>"Valore1000");
// Salva in $indice l'indice e in $valore il valore
while (list($indice,$valore)=each($a) )
        echo "($indice) $valore<br>";
?>
```

Con il foreach, il ciclo non si ferma e bisogna usare il break o una condizione.

```
<?php
$a=array("zero "=>"ValoreZero "," uno "=>" ValoreUno ",1000=>"Valore1000");
// Salva in $indice l'indice e in $valore il valore
foreach ($a as $indice=>$valore )
        echo "($indice) $valore<br>";
?>
```

6.2.2- Algoritmi di ordinamento degli array

1) Rispetto al valore:

Ordinamento alfabetico/numerico crescente:
`sort($nome_array);`

N.B.: Se l'array ha indici alfabetici, quando lo ordino con sort l'indice alfabetico lo perdo e mi viene posto un indice numerico!

Ordinamento decrescente:
`rsort($nome_array);`

Ordinamento alfabetico/numerico crescente che mantiene l'indice associato al valore:
`asort($nome_array);`

Ordinamento decrescente mantenendo l'indice:
`arsort($nome_array);`

2) Rispetto all'indice:

Ordinamento crescente:
`ksort($nomea_array);`

Ordinamento decrescente:
`krsort($nome_array);`

6.3- Le sessioni

Le **sessioni** operano sul server creando dei files dove vengono salvati alcuni dati importanti relativi alla sessione di navigazione. Una volta che la sessione sarà terminata (ossia quando l'utente chiude il browser), i files creati vengono eliminati. Le sessioni vengono utilizzate, ad esempio, nella gestione del login.

Per aprire una sessione utilizziamo la funzione **session_start()** che, tipicamente, viene inserita come prima riga di ogni pagina (nell'head del documento html).

6.3.1- Funzioni delle sessioni

Utilizziamo un esempio per vedere come facciamo a salvare in una variabile di sessione la stringa di login e password.

Capitolo 7- PHP & SQL

Siccome il PHP è un linguaggio lato server, possiamo connetterci ad un database e effettuare le operazioni tipiche del linguaggio SQL attraverso le pagine web.

7.1- Connessione al database

Per connetterci ad un database usiamo questa sintassi:

```
$conn = new msqli("localhost","Utente_mysql","Password_mysql");
If($conn -> connect_error)
        die("FALLITO <BR>");
```

La funzione DIE termina la trasmissione della pagina

7.2- Esecuzione delle query

Per eseguire le query tramite una pagina web utilizziamo la funzione:

```
$sql = "SELECT …. FROM …. WHERE …. ;";
$result = $conn -> query($sql);
```

- *$result* contiene il risultato della query;
- *$sql* contiene la stringa SQL da eseguire;

7.3- Visualizzazione del risultato della query

```
If($result -> num_rows > 0)
{
        while($row = $result -> fetch_assoc())
        {
                echo "id: ". $row["id"]." - Name: ". $row["firstname"]." ". $row["lastname"]. "<br>";
        }
}
```

7.4- Chiudere la connessione

```
$conn -> close();
```

7.5- Connessione al database con PDO

Per connetterci ad un database usiamo questa sintassi:

```
$conn = new PDO("mysql:host=$ipServer;dbname=$nomeDB", $username, $password);
```

- *$ipServer* contiene ip o hostname del server;
- *$nomeDb* contiene il nome del database;

7.6- Esecuzione delle query con PDO

```
$sql  = $conn->query("SELECT * FROM nomeTabella");
$result = $sql->fetchAll();
```

7.7- Esecuzione delle query prepared con PDO

Per eseguire le query tramite una pagina web utilizziamo la funzione:

```
$sql = $conn->prepare('SELECT * FROM nomeTabella WHERE nomeColonna = :parametro');
$sql->execute([“parametro” => "valore"]);
$result = $sql->fetchAll();
```

- *$result* contiene il risultato della query;
- *$sql* contiene l’oggetto prepared statement da eseguire;

7.8- Visualizzazione del risultato della query con PDO

```
foreach($result as $row) {
        echo $row[“nomeColonna”];
}
```

7.9- Chiudere la connessione con PDO

```
$conn = null;
```