Tema 9. PHP I BASES DE DADES RELACIONALS

9.1 BASES DE DADES RELACIONALS

Existeixen molts tipus de bases de dades en funció de la manera en que s'emmagatzema i s'accedeix a la informació que guarden: relacional, jeràrquica, en xarxa, orientada a objectes, etc…

Exemples de gestors de bases de dades relacionals o DBMS hi ha molts: MySQL, SQLite, Oracle, Informix, SyBase, Microsoft SQL Server, PostGres, mSQL, etc…
Bàsicament, un gestor de bases de dades relacional emmagatzema les dades en **taules**, casdascuna de les quals està formada per **files** (o registres), i aquestes, al seu torn, estan formades per **columnes** (o camps). Abans de definir una taula s'ha de **normalitzar**, procés que consisteix en evitar la redundància..

Per una altra banda, l'accés als gestors de bases de dades es fa mitjançant el llenguatge SQL, del qual ja teniu coneixements i farem un resum en aquest tema.

9.2. MySQL

D'entre els gestors mencionats amb anterioritat, l'elecció de MySQL com a gestor de bases de dades radica en que és gratuit tant per a usos privats, com comercials (només s'ha de pagar en el cas de que es desenvolupi un producte comercial que estigui basat en MySQL), en la seva disponibilitat per a diferents sistemes operatius, en que és capaç de treballar amb milions de registres i perquè, a més, és molt ràpid i no nocessita grans recursos de màquina.

El nou sistema d'emmagatzematge per a les bases de dades gestionades per MySQL és *InnoDB*, desenvolupat i mantingut per Heikki Tuuri. Aquest sistema d'emmagatzematge proporciona bloquejos de columnes, lectures no bloquejants, múltiples nivells d'aïllament, integritat referencial, recuperació automàtica i totes les garanties ACID (Atomic, Consistent, Isolated, Durable).

9.2.1. CONNEXIÓ AMB EL GESTOR DE LA BASE DE DADES

Les aplicacions que segueixen l'arquitectura client-servidor (Web, correu, dns, ftp, news, etc…) basen el seu funcionament en dos extrems: un servidor que es manté *escoltant* peticions en un port determinat i, a l'altre els clients que, quan volen contactar amb el servidor, realitzen connexions amb aquest port.

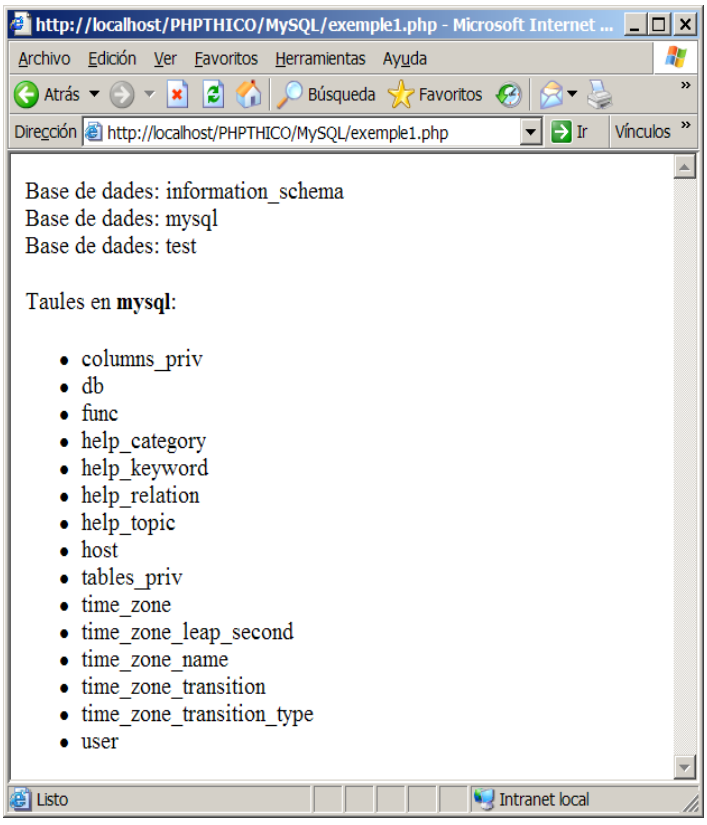
MySQL segueix aquesta mateixa arquitectura i, per tant, per a poder realitzar operacions, és necessari tenir arrencat el programa servidor. Per defecte, el servidor de MySQL escolta peticions en el port TCP 3306. A la mateixa distribució on ve el servidor es troba un programa client (*mysql*) que permet fer la connexió i treballar interactivament amb el gestor:

A la comanda se li indica l'usuari amb el que es vol connectar (`-u root`) i que la clau s'introdueix mitjançant teclat (`-p`). Les operacions que s'han executat són les corresponents a veure totes les bases de dades que conté MySQL, posar activa la base de dades `mysql` i veure les taules que conté. Per sortir de la sessió, es tecleja `exit`.

També es pot fer la connexió al servidor mitjançant una sèrie de funcions implementades en PHP, donat que les comandes que es poden introduir a la sessió amb MySQL tenen el seu equivalent com a funció de PHP (totes aquestes funcions comencen amb el prefixe `mysql_`).

L'exemple anterior es pot fer des d'un script PHP exactament igual amb les funcions `mysql_connect()`, `mysql_select_db()`, `mysql_list_tables()` i `mysql_close()`:

```
<?
  $dbd = mysql_connect('localhost', 'root', '');

  $res = mysql_list_dbs($dbd);                //show databases
  while ($fila = mysql_fetch_array($res, MYSQL_NUM))
     echo "Base de dades: $fila[0]<BR>";

  $base_dades='mysql';
  mysql_select_db($base_dades, $dbd);    //use mysql

  $res = mysql_list_tables("$base_dades");  //show tables

  echo "<BR>Taules en <B>$base_dades</B>:<UL>";
  while($fila = mysql_fetch_array($res, MYSQL_NUM))
    echo "<li>$fila[0]<BR>";
  echo "</UL>";

  mysql_close($dbd);
?>
```



La funció `mysql_connect()` retorna un descriptor de la connexió establerta amb el gestor de la bases de dades. Aquest descriptor és equivalent a l'utilitzat per les funcions que realitzen operacions amb fitxers, és a dir, serà necessari per a realitzar qualsevol operació amb el gestor.

Excepte les operacions de connectar, desconnectar i alguna més, la resta de les operacions es fan mitjançant la funció `mysql_query($sentencia_sql, $descr)`. També són importants les funcions `mysql_error()` i `mysql_errno()` ja que indiquen la naturalesa de l'error ocorregut i el nombre de l'error, respectivament. Una altra funció que s'utilitza amb bastant freqüència és `mysql_affected_rows()`, la qual indica el nombre de registres que s'han vist *afectats* en la operació que s'acaba de realitzar, sigui la que sigui (inserció, eliminació, modificació, cerca, etc…)

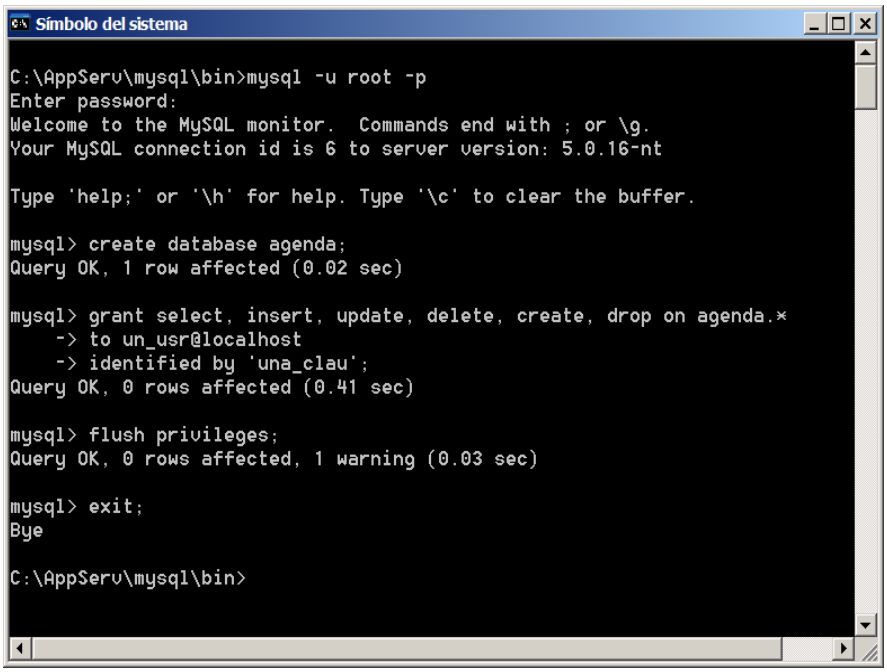
9.3. IMPLEMENTACIÓ D'UNA AGENDA AMB MySQL

Com ja coneixem, les operacions que es poden realitzar amb una base de dades són crear-la, esborrar-la i manipular les seves dades (inserir, modificar, esborrar i cercar). En aquest apartat farem un recordatori de les operacions indicades juntament amb la descripció de les funcions implementades en PHP.

9.3.1. CREACIÓ DE LA BASE DE DADES

Per seguretat, la operació de crear una base de dades està restringida a l'usuari `root`. Per això, i donat que aquesta operació només es fa una vegada, la base de dades que contindrà l'agenda es crearà des del programa client `mysql`.

En aquesta imatge podem observar la connexió amb l'usuari `root` i, una vegada a dins, la creació de la base de dades `agenda`. A continuació s'afegeix un usuari `un_usr` que només tindrà accés des de la màquina on resideix el gestor (determinat per la clàusula `TO un_usr@localhost`); tindrà com a paraula clau `una_clau`, ii tots els permisos sobre la base de dades `agenda` que s'acaba de crear:



9.3.2. CREACIÓ DE LA TAULA

A la creació de la taula, on s'emmagatzemaran les files (o registres) de la nostra base de dades s'ha d'indicar el nom de la taula, les distintes columnes que tindrà (alhora que amb el tipus de cadascun d'elles), i la descripció de quines columnes seran utilitzades com a claus primàries. La sintaxis simplificada de la sentència SQL que ens permet crear una taula és la següent:

```
CREATE TABLE taula(
[camp tipus [NOT NULL|NULL] [DEFAULT valor] [AUTOINCREMENT]]
[, PRIMARY KEY([camp])]
[, INDEX [nom_ind] (camp)]
[, UNIQUE [INDEX] [nom_ind] (camp)]
[, FOREIGN KEY camp (nomb_ind)])
[ON DELETE RESTRICT|CASCADE|SET NULL|NO ACTION|SET DEFAULT]
[ON UPDATE RESTRICT|CASCADE|SET NULL|NO ACTION|SET DEFAULT];
```

9.3.2.1. TIPUS DE DADES DE LES COLUMNES

Els identificadors de les columnes poden començar per qualsevol lletra o número, però no poden estar constituit només per números; a més, poden tenir una longitud màxima de 64 caràcters. A continuació es poden veure alguns dels diferents tipus de dades que MySQL pot manipular:

- **CHAR(L)**: s'utilitza per emmagatzemar cadenes de L caràcters. Una columna d'aquest tipus pot tenir un tamany de 255 caràcters. Internament es reserven sempre 255 caràcters. Pe: `nom CHAR(50)`
- **VARCHAR(L)**: és igual que CHAR, excepte en que internament només s'emmagatzema el tamany real de la dada. Així, CHAR és més eficient en termes de rapidesa, mentre que VARCHAR lo és en termes d'espai d'emmagatzematge. Pe: `nom VARCHAR(50)`
- **TEXT/BLOB**: ambdós tipus de dades poden emmagatemar fins a un tamany màxim de 65.535 octets (de fet, BLOB és un acrònim de *Binary Large Object*). La única diferència entre els dos tipus està en que, a l'hora de fer comparacions, TEXT distingeix entre majúscules i minúscules mentre que BLOB no.
- **INT [unsigned]**: especifica un nombre enter amb els valors que van desde -2.147.483.648 fins a 2.147.483.648. Si s'especifica `unsigned`, llavors els valors aniran desde 0 fins a 4.294.967.295. Pe: `edat INT unsigned`
- **TINYNT [unsigned]**: especifica un nombre enter petit amb els valors que van desde -127 fins a 128 o desde 0 a 255 si s'especifica `unsigned`.
- **SMALLINT [unsigned]**: especifica un nombre enter petit amb els valors que van desde -32.768 fins a 32.767 o desde 0 a 65535 si s'especifica `unsigned`.

- **FLOAT [(E,D)]**: especifica valors en coma flotant de precissió simple. Donat que hi ha una part entera i una altra real en aquests nombres, es pot especificar el total de dígits per a les dues quantitats. Pe: `mitja_aritmet FLOAT, temperatura FLOAT(2,1), alçada FLOAT(1,2)`

- **DOUBLE[(E,D)]** o **REAL[(E,D)]**: és igual que FLOAT, excepte en que especifica valors en coma flotant de precissió doble.

- **DECIMAL[(E,D)]**: igual que FLOAT, excepte que no fa arrodoniment de la quantitat donat que emmagatzema els números com a cadenes de caràcters (és útil, per exemple, per a tractar quantitats de diners).

- **DATE**: emmagatzema valors de tipus data. Els valors que admet són cadenes de caràcters amb diferents formats: YYY-MM-DD, YY-MM-DD o YYMMDD. Els valors que pot prendre van desde 1000-01-01 fins a 9999-31-12.

- **TIME**: Emmagatzema valors de tipus hora. Admet els format HH:MM:SS, HHMMSS o HHMM.

- **YEAR**: emmagatzema valors de tipus any amb el format YYYY.

- **TIMESTAMP**: emmagatzema valors de tipus *instant* amb el format YYYYMMDDhhmmss.

- **ENUM**: especifica una columna que només podrà contenir un únic valor d'entre una llista. Pe: `estat_civil ENUM('solter', 'casat', 'altres').`

- **SET**: és igual que l'anterior excepte que a la columna es podrà tenir qualsevol combinació d'un o més valors. Pe: `aficions SET('snowboard', 'natació', 'senderisme', 'viatjar').`

En especificar quins tipus de dades poden emmagatzemar les columnes, es pot, a més, indicar una sèrie de característiques que poden complir com ara:

- **PRIMARY KEY**: Si una columna té aquesta característica, no podrà haver 2 registres que tinguin el mateix valor en aquesta columna. Donat que aquest camp és **clau primària** de la taula, MySQL indexarà automàticament la taula en funció d'aquesta columna.

- **AUTOINCREMENT**: Només s'aplica a camps de tipus enter ja el seu efecte és, en inserir un registre a la taula assigna al camp el valor més gran que hi hagi en aquesta columna més un. No es pot definir més d'una columna d'aquest tipus per taula.

- **DEFAULT valor_per_defecte**: en inserir una fila, s'assignarà valor_per_defecte quan s'especifiqui NULL com a valor a assignar.

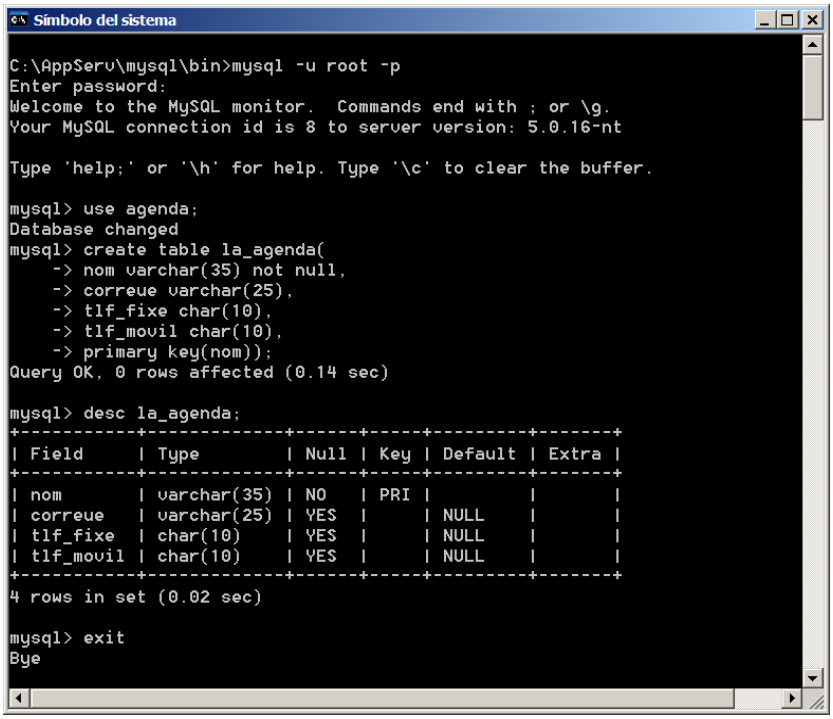
- **NOT NULL**: no es pot assignar un valor NULL al camp que tingui aquesta caracaterística.

- **NULL**: són camps amb valors NULL.

9.3.2.2. CREACIÓ DE LA TAULA

La operació que resulta més *complexa* és la de crear la taula donat que s'ha d'indicar expressament el nom, tipus i longitud de cada camp, i quin camps serà l'utilitzat com a clau de la taula.

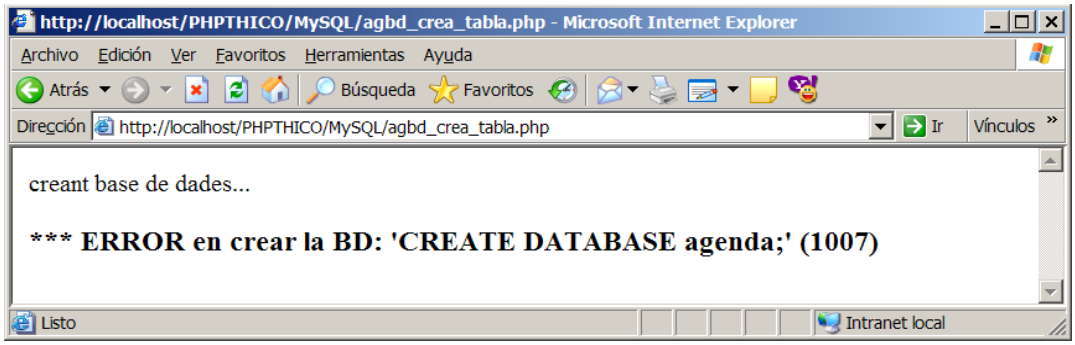
Donat que en el punt anterior en crear la base de dades `agenda`, s'ha creat l'usuari `un_usr` amb privilegi de creació de taules, crearem la taula `my_agenda`. També, en aquest cas, aquesta operació només s'ha de fer una vegada, per tant, la podem fer mitjançant el client mysql:

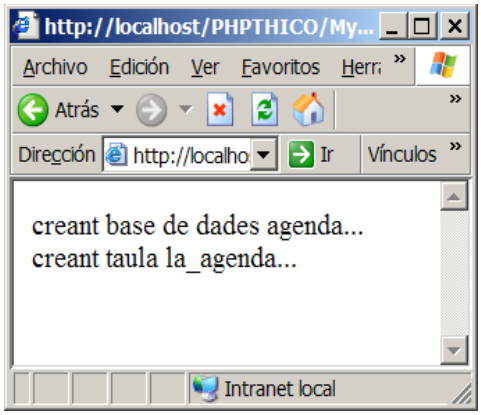


A continuació podem veure l'script PHP que fa la mateixa operació:

```
<?
$la_bd    = "agenda";
$la_taula = "la_agenda";

$db=mysql_connect("localhost", "un_usr", "una_clau");
if (!$db)
  die ("<h3>*** ERROR en connectar...");

echo "creant base de dades...";

$sql="CREATE DATABASE $la_bd;";
if (!$result=mysql_query($sql, $db))
  die ("<h3>*** ERROR en crear la BD: '$sql' (".mysql_errno().")");

if (!mysql_select_db($la_bd, $db))
  die("<h3>ERROR: en seleccionar BD $la_bd</h3>".mysql_errno().")");
```

```
echo "creant taula $la_taula...";
$sql="CREATE TABLE $la_taula (
      nom VARCHAR(35) NOT NULL,
      correue VARCHAR(25),
      tlf_fixe CHAR(10),
      tlf_mobil CHAR(10),
      PRIMARY KEY(nom)
      );";
if (!$result=mysql_query($sql, $db))
  die ("<h3>*** ERROR en executar: '$sql' (".mysql_error().")");

mysql_close($db);
?>
```





9.3.3. FITXER DE RECOLZAMENT

Donat que hi haurà una sèrie de rutines que seran utilitzades per quasi tots els scripts, es poden situar en un fitxer apart (`agbd_dades.inc`) que serà importat per aquests fitxers.

La primera rutina és la funció d'establir una connexió amb el servidor MySQL (on s'haurà d'indicar l'usuari i la contrasenya) que retornarà un descriptor de la connexió.

```
<?
function connecta()
{
  $dbd=mysql_connect("localhost", "un_usr", "una_clau");
  if (!$dbd)
    die ("<h3>*** ERROR al conectar... :((");

  if (!mysql_select_db("agenda", $dbd))
    die("<h3>ERROR: al seleccionar</h3>".mysql_errno());
  return $dbd;
}
```

Es poden definir 3 funcions més relacionades amb la generació de text HTML:

`pinta_entrada_dades()`: permet treure per pantalla el formulari on l'usuari introduirà les dades per a ser inserides o modificades a la base de dades

`pagina_anterior()`: imprimeix en el navegador un enllaç per a tornar a la pàgina anterior

`posa_peu_de_tornada()`: treurà un altre enllaç que ens portarà a la pàgina principal

```
function pinta_entrada_dades($el_php, $bot_submit,  $la_victima, $el_n, $el_c, $el_tf, $el_tm)
{
echo "<form action='$el_php' method='get'>";

if ($la_victima)
  echo "<input type='hidden' name='victima_modificacio' value='$la_victima'>";

echo "
<table border='0'>
<tr>
  <td>Nom:
  <td><input type='text' size='35' name='nom' value='$el_n'>
</tr>
<tr>
  <td>Correu-e:
  <td><input type='text' size='25' name='correue' value='$el_c'>
</tr>
<tr>
  <td>Telèfon fixe:
  <td><input type='text' size='10' name='tlf_fixe' value='$el_tf'>
</tr>
<tr>
  <td>Telèfon mòbil:
  <td><input type='text' size='10' name='tlf_mobil' value='$el_tm'>
</tr>
```

```
<tr>
  <td colspan='2'><input type='submit' value='$bot_submit'>
</tr>

</table>
</form>
";
}

function pagina_anterior()
{
  echo "<a href='javascript:history.go(-1)'>&lt;&lt;tornar enrera</a><br>";
}

function posa_peu_de_tornada()
{
  echo "<br><a href='agbd_opers.php'>Tornar a l'Agenda</a>";
}
?>
```

9.3.4. LLISTAT DE REGISTRES

La sentència SQL que permet realitzar cerques de dades que donin resposta a una sèrie de condicions és la sentència SELECT. La seva sintaxis bàsica és:

```
SELECT llista_de_camps
FROM taula
WHERE condició
[GROUP BY camps [HAVING condició]]
[ORDER BY camps [ASC|DESC]];
```

A l'exemple de l'agenda que estem fent, hi ha una sèrie d'operacions amb els registres de la base de dades que necessiten un llistat de registres: per a escollir un d'entre tots per a modificar-lo, per a escollir tots aquells que es vulguin esborrar, per a mostrar el resultat d'una cerca o, simplement, per a mostrar-los. Per a aquest motiu, podem implementar en un únic script (`agbd_llistats.php`) la generació de tots aquests llistats.

No obstant, cadascun d'aquests llistats té un requeriments diferents. Per exemple, no és el mateix generar un llistat que tingui la possibilitat d'escollir un grup de registres per a ser eliminats que un llistat normal de registres: mentre que en aquest últim no es requerix res especial, el primer ha de tenir elements de formulari de tipus *checkbox* per a que l'usuari pugui marcar-los segons les seves necessitats i, per suposat, aquest formulari ha d'indicar el nom de l'script que farà efectiva l'eliminació. Per tant, les crides a aquest script s'hauran de parametritzar adequadament per saber quin tipus de llistat s'ha de generar i amb quines característiques.

A la pràctica, això significa que a les crides d'aquest programa s'expressaran aquestes particularitats en forma de valors donat en el QUERY_STRING. Així, les dades que obtindrem seran: el tipus d'operació a realitzar, la cadena de caràcters a cercar (en el seu cas), el camp pel qual es vol fer la ordenació dels registres i el tipus d'ordenació (ascendent o descendent).

```
$operacio=$_GET['oper'];
switch($operacio) {
  case 'LLISTAR' :
       $capcalera="Agenda: Llistat d'entrades a l'agenda";
       $cgi='';
       $elem_form='';
       $llegenda='';
       $a_cercar='';
       break;
  case 'CERCAR' :
       $capcalera="Agenda: cerca de registres";
       $cgi='';
       $elem_form='';
       $llegenda='';
       $a_cercar=$_GET['a_buscar'];
       break;
  case 'MODIFICAR' :
       $capcalera="Agenda: Selecció d'Entrades a Modificar";
       $cgi='agbd_modif_plantilla.php';
       $elem_form='r';
       $llegenda='Modificar!';
       $a_cercar='';
       break;
  case 'ESBORRAR' :
       $capcalera="Agenda: Selecció d'Entrades a Esborrar";
       $cgi='agbd_esborrar.php';
       $elem_form='c';
       $llegenda='Esborrar!';
       $a_cercar='';
       break;
}
echo "<title>$capcalera</title> </head>
      <body>
      <h2>$capcalera</h2>";

if (isset($_GET['campo'])){
  $campo_ord=$_GET['campo'];
  $sentido=$_GET['dir'];
} else {
  $campo_ord='nom';
  $sentido='ASC';
}

include("agbd_dades.inc");
$dbd=connecta();
$sql="SELECT * from la_agenda ";
if ($a_cercar)
  $sql .= "WHERE nom LIKE '%$a_cercar%' ";
$sql .= "ORDER BY ($campo_ord) $sentido";
$res = mysql_query($sql, $dbd);
if (!$res)
  die ("*** ERROR en la cerca: " . mysql_error());


pinta_llistat($res, $operacio, $cgi, $elem_form, $llegenda, $a_cercar);
posa_peu_de_tornada();
```

Per tant, una vegada units els diferents valors i feta la consulta a la base de dades, es crida una funció que s'ha declarat al mateix script (amb el nom `pinta_llistat()`) per a que, amb el resultat de la consulta, fagi la impresssió del llistat. Aquesta funció està definida amb els següents paràmetres:

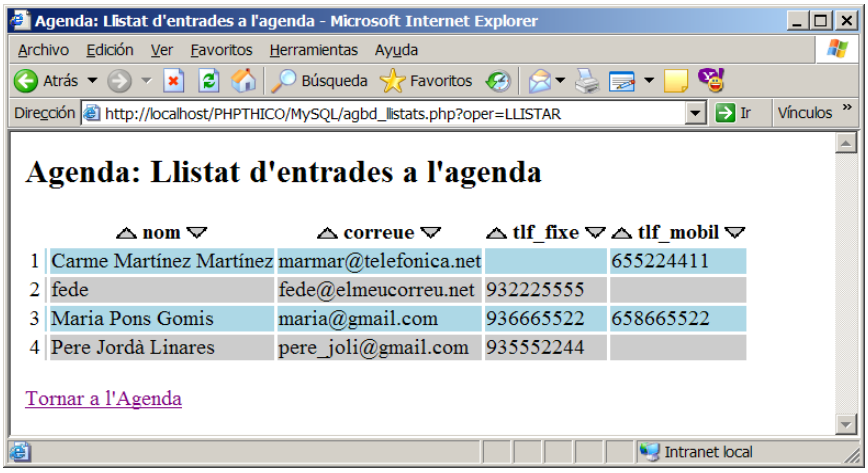
```
pinta_llistat($cursor, $operac, $cgi, $elem_form, $bot_submit, $a_cercar)
```

Els paràmetres que s'han de passar són el cursor on estan els resultats de la cerca realitzada, el tipus d'operació o llistat a realitzar (LLISTAR, CERCAR, MODIFICAR o ESBORRAR), el nom del programa que tractarà les dades del formulari (si és el cas), el tipus d'element de formujlari que apareixerà (checkbox, radio o res) a la primera columna de la taula, la llegenda que apareixerà en el botó de tipus *submit* que efectua la crida al següent script PHP (si és el cas) i, per últim, la cadena a cercar, si es que la operació és de cerca.

Llavors, per generar un simple llistat de totes les entrades a la base de dades o un llistat del resultat d'una cerca, es crida a aquesta funció de la següent forma:

```
pinta_llistat($res, 'LLISTAT', '', '', '' , '')
```

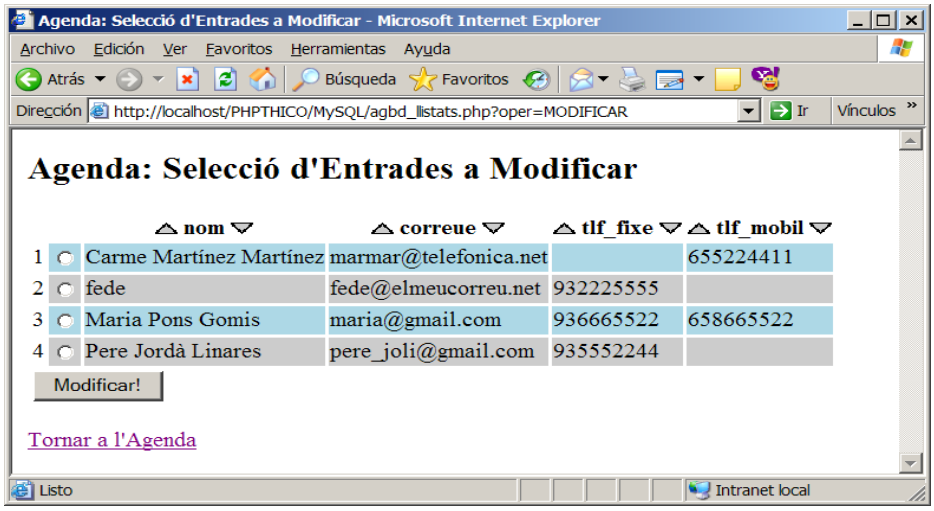
i el resultat seria:



La crida que generarà un llistat de totes les entrades per a poder seleccionar aquella que volem modificar serà de la següent forma:

```
pinta_llistat($res, 'MODIFICAR', 'agbd_modif_plantilla.php','r', 'Modificar!', '')
```

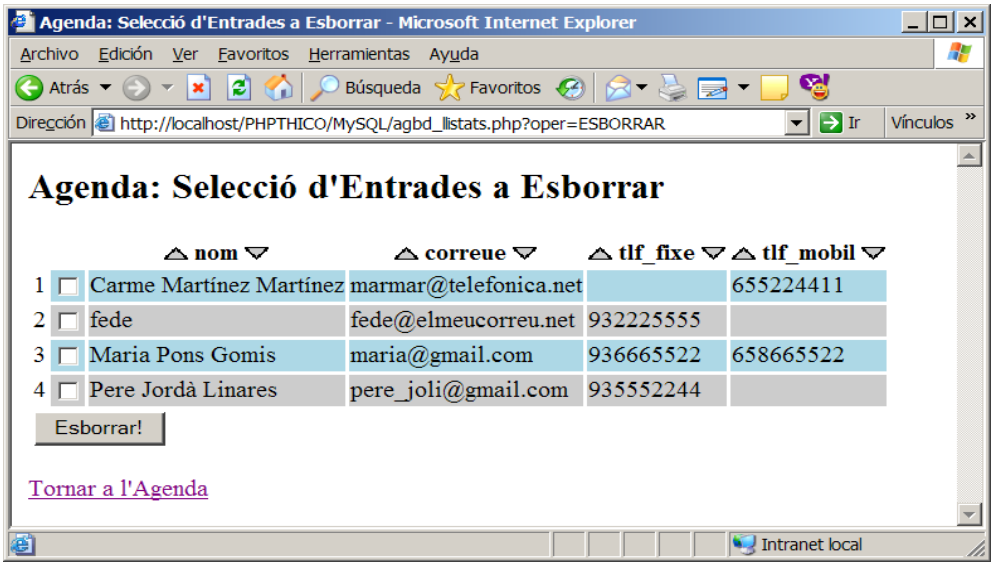
que generarà:



Per a esborrar registres, a l'usuari se li mostra un llistat de tots els que hi ha i així pugui seleccionar (mitjançant checkboxes) aquells que es vulguin esborrar. En aquest cas, la crida a la funció seria:

```
pinta_llistat($res, 'ESBORRAR', 'agbd_esborrar.php','c', 'Esborrar!', '')
```

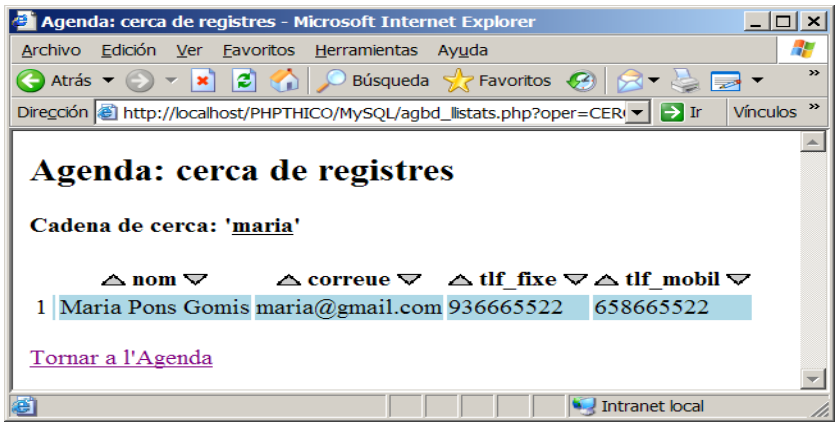
amb el resultat:



Per últim, la crida a aquesta funció per a obtenir el llistat d'aquells registres que continguin una determinada cadena en el camp `nom`, per exemple, 'ito', seria:

```
pinta_llistat($res, 'CERCAR', '','', '', 'ito')
```

i el resultat seria:



El codi corresponent a aquesta funció és molt senzill, bàsicament es tracta d'imprimir una taula HTML on cada fila es correspon amb un registre i s'ha de tenir en compte el tipus d'element de formulari que s'ha d'imprimir per a l'elecció de registres:

```
function pinta_llistat($cursor, $operac, $cgi, $elem_form, $bot_submit, $busco)
{
 global $campos;
 $colores_filas=array('#cccccc', 'lightblue');
 $ind_colores=0;
 $cont_lineas=1;

 echo "<table>";
 if ($cgi) echo "<form action='$cgi' method='post'>";

 if ($busco)
   echo "<h4>Cadena de cerca: '<u>$busco</u>'</h4>";

 pinta_capcalera_taula($operac, $busco);

 while ($reg = mysql_fetch_array($cursor, MYSQL_ASSOC))
 {
  $ind_colores++;
  $ind_colores %= 2;
  echo "<tr bgcolor=${colores_filas[$ind_colores]}>";
  switch($elem_form){
    case 'c':
      $valor=$reg{'nom'};
      $seg_col="
       <input type='checkbox' name='victima$cont_lineas' value='$valor'>";
      break;
    case 'r':
      $valor=$reg{'nom'};
      $seg_col="<input type='radio' name='victima' value='$valor'>";
      break;
    default: $seg_col="";  // cas de llistar i cercar
  }

  echo "<td bgcolor='white'>$cont_lineas</td><td>$seg_col</td>";
  for ($i=0; $i<count($reg); $i++) {
    echo "<td>", $reg{$campos[$i]}, "</td>";
  }
  echo "</tr>";
  $cont_lineas++;
 }
 if ($cgi)
  echo "<tr>
        <td colspan='6'><input type='submit' value='$bot_submit'></td>
        </tr>
        </form>";
 echo "</table>";
}
```

En el cas d'una selecció de registres, el que retorna la funció `mysql_query()` és un *recurs* o *cursor*, és a dir, un grup de files emmagatzemades en una estructura interna de la que s'hauran d'extreure amb la funció:

```
mysql_fetch_array($recurs, TIPUS_ARRAY)
```

Si es vol que es retornin les files en forma d'array associatiu, numèric o amb ambdós tipus d'indexs, es posarà com a segon paràmetre: MYSQL_ASSOC, MYSQL_NUM o MYSQL_BOTH respectivament.

9.3.4.1. ORDENACIÓ DE REGISTRES

Com es pot veure en el codi anterior, des de la funció `pinta_llistat()` es fa la crida a la funció `pinta_capcalera_taula()` la qual el que fa és imprimir la capçalera de la taula amb els noms dels camps i un parell d'icones que representen la ordenació dels registres en funció d'aquests camps, però en ordre ascendent o descendent:

La operació d'ordenació és molt simple ja que la sentència `SELECT` la fa de forma automàtica gràcies a la clàusula `ORDER BY camp [ASC|DESC]` que fa que el resultat de la consulta estigui ordenat alfabèticament en ordre ascendent (ASC) o descendent (DESC).

Podem veure un extracte del codi mostrat anteriorment on es veu com es contrueix la ordre de cerca o llistat de registres:

```
$sql="SELECT * from la_agenda ";
if ($a_cercar)
  $sql .= "WHERE nom LIKE '%$a_cercar%' ";
$sql .= "ORDER BY ($campo_ord) $sentido";
```

Donat que la funció `pinta_capcalera_taula()` genera els enllaços per a obtenir el mateix llistat que tenim, però ordenat per un altre camp i/o en un altre sentit, es necessita per a això conèixer les dades de la consulta, és a dir, el tipus de llistat i la cadena de cerca. Per aquest motiu, s'ha declarat amb dos paràmetres que es corresponen amb el tipus de llistat i la cadena de cerca.

```
function pinta_capcalera_taula($op, $op2)
{
  global $campos;

  echo "<tr><td></td><td></td>";  // 2 cel·les: numeració & elem de formulari

  if ($op2) {
    $param="&a_buscar=$op2";
  } else {
    $param="";
  }
```

```
foreach ($campos as $un_campo) {
   $prefijo  = "<a href='".$_SERVER['PHP_SELF']."?oper=$op&campo=$un_campo";
   $pref_arriba   = "$prefijo&dir=ASC$param'>";
   $pref_abajo    = "$prefijo&dir=DESC$param'>";
   $sufijo_arriba = "<img src='up.gif' border='0'></a>";
   $sufijo_abajo  = "<img src='down.gif' border='0'></a>";
   echo "<th> $pref_arriba$sufijo_arriba $un_campo $pref_abajo$sufijo_abajo</th>";
 }
 echo "</tr>";
}
```

9.3.5. ESBORRAR UN REGISTRE

La sintaxis simplificada de la sentència SQL que ens permet esborrar files d'una taula de la base de dades és:

```
DELETE FROM taula WHERE condició;
```

On la clàusula `condició` és idèntica a la que s'indica quan es fa una cerca; esborrar implica fer un primer examen dels registres que compleixen la condició indicada i, després, executar la operació d'esborrar sobre aquells registres trobats.

A l'agenda, per a esborrar registres, cridarem al programa de llistats: `agbd_llistats?op=ESBORRAR`, el qual generarà un llistat de tots els registres de l'agenda cridant a la rutina `pinta_llistat()` de la següent forma:

```
pinta_llistat($res,     'ESBORRAR',     'agbd_esborrar.php',     'c',
'Esborrar!', '');
```

Com ja sabem, es genera un formulari amb caselles de verificació (checkbox) per a que l'usuari seleccioni els registres que han de ser esborrats. A continuació es mostra un extracte de la funció `pinta_llistat()` per a veure com es generen aquests elements de formulari, donat que és important conèixer com és el mecanisme pel qual es seleccionen els registres que es volen eliminar:

```
switch($elem_form){
     case 'c':
       $valor=$reg{'nom'};
       $seg_col="
        <input type='checkbox' name='victima$cont_lineas' value='$valor'>";
       break;
     case 'r':
       $valor=$reg{'nom'};
       $seg_col="<input type='radio' name='victima' value='$valor'>";
       break;
```

Com es veu la crida de pinta_llistat, el nom de l'script que farà efectiva la eliminació és `agbd_esborrar.php`:

```
<html>
<head> <title>Agenda: Esborrar entrades</title> </head>
<body>
<h2>Agenda: Esborrar entrades</h2>
<?php
include("agbd_dades.inc");

if (count($_POST) > 0) {
  $sql="DELETE FROM la_agenda WHERE ";
  foreach ($_POST as $clave=>$valor) {
    $sql.= "nom='$valor' OR ";
  }
  $sql=substr($sql, 0, strlen($sql)-3); // treiem l'últim 'OR '

  $dbd = connecta();
  $res = mysql_query($sql, $dbd);
  $total_victimas=mysql_affected_rows($dbd);
  echo "<h3>Esborrat/s <u>$total_victimas</u> registre/s </h3>";
} else {
  echo "<h4>NO ha seleccionat cap registre per a esborrar :(</h4>";
  pagina_anterior();
}

posa_peu_de_tornada();
?>
```

9.3.6. MODIFICAR REGISTRES

Per a realitzar la operació de modificar o actualitzar dades, SQL proporciona la sentència `UPDATE`, la sintaxis simplificada de la qual és:

```
UPDATE taula
SET camp1 = valor1, …
[WHERE condició];
```

La operació de modificar un registre es farà en 3 passos:

a) primer s'executa `agbd_llistats?op=MODIFICAR`, per tant, la generació del llistat de registres serà:

```
pinta_llistat($res, 'MODIFICAR', 'agbd_modif_plantilla.php', 'r',
'Modificar!', '');
```

b) a continuació l'script `agbd_modif_plantilla.php` (que podem veure a continuació) mostrarà el contingut actual del registre seleccionat per a que es puguin introduir els valors oportuns:

```
<html>
<head>
<title>Agenda: seleccionar registre per a modificar-lo</title>
</head>
<body>
<h2>Agenda: Seleccionar elements per a la seva modificació</h2>
<?php
include("agbd_dades.inc");

if (!isset($_POST['victima'])) {
  echo "<h4>NO ha seleccionat cap registre per a modificar :(</h4>";
  //pagina_atras();
  pagina_anterior();
  posa_peu_de_tornada();
  exit;
}
$victima=$_POST['victima'];
$dbd = connecta();
$sql = "SELECT * FROM la_agenda WHERE nom='$victima'";
$res = mysql_query($sql, $dbd);
if (! $res){
  echo "*** ERROR en el SELECT: ", mysql_error();
  posa_peu_de_tornada();
  exit;
}

$campos=mysql_fetch_array($res, MYSQL_ASSOC);
pinta_entrada_dades("agbd_modificar.php", "Modifica!",
                    "$victima",
                    "${campos['nom']}",   "${campos['correue']}",
                          "${campos['tlf_fixe']}", "${campos['tlf_mobil']}");

pagina_anterior();
posa_peu_de_tornada();
?>
```

c) per últim, igual que a la funció d'inserir dades, l'script que fa efectiva la modificació també es beneficia de la característica de que el camp `nom` sigui clau primària i de que, en no existir caràcters *no permesos*, no serà necessari comprovar allò que ha teclejat l'usuari en el formulari previ. Aquest script es diu `agbd_modificar.php` i es pot veure el seu contingut a continuació:

```
<html>
<head> <title>Agenda: Modificació d'entrades</title> </head>
<body>
<h2>Agenda: Modificació d'entrades</h2>
<?php

$nom    = $_POST['nom'];
if (strlen(trim($nom))==0) {
  echo ("*** ERROR: el camp nom no pot estar buit<br>");
  pagina_anterior;
  posa_peu_de_tornada();
  exit;
}
$victima_modificacio = $_POST['victima_modificacio'];
$correue             = $_POST['correue'];
$tlf_fixe            = $_POST['tlf_fixe'];
$tlf_mobil           = $_POST['tlf_mobil'];

include("agbd_dades.inc");
$dbd=connecta();

$sql="UPDATE la_agenda SET ";
$sql.="nom='$nom', correue='$correue', tlf_fixe='$tlf_fixe', tlf_mobil='$tlf_mobil' ";
$sql.="WHERE nom='$victima_modificacio'";

$res=mysql_query($sql, $dbd);
if (! $res) {
  echo "*** ERROR en actualitzar $victima_modificacio: ", mysql_error();
  posa_peu_de_tornada();
  exit;
}
echo "Modificat <u>", mysql_affected_rows(), "</u> registre/s<br>";
posa_peu_de_tornada();
?>
```

9.3.7. INSERIR REGISTRES

La sentència SQL `INSERT` presenta la següent sintaxis:

```
INSERT INTO taula [(camp1, camp2, …)]
VALUES (valor1, valor2, …);
```

Si no es fa ús de la part condicional on s'han d'indicar els camps a inserir, la llista de valors haurà de contemplar tots els elements de la fila en el mateix ordre en que es van declarar en el moment de creació de la taula. En cas d'indicar els camps a inserir, la llista podrà tenir qualsevol ordre i, fins i tot, no incloure algun camp, sempre tenint en compte que s'associaran els valors als seus respectius camps seguint l'ordre en que apareguin aquests últims.

En inserir un valor de tipus cadena de caràcters (CHAR, VARCHAR, TEXT, etc…) s'ha d'englobar entre cometes.

Com el `nom` és clau primària, no haurem de fer una funció específica per evitar la repetició, ja que el propi gestor farà aquesta feina i generarà un error que s'haurà de capturar i mostrar a l'usuari. La única cosa que s'ha de tenir en compte és recollir adequadament els valors introduits per l'usuari en el formulari previ i introduir-lo a la base de dades:

```
<?
include('agbd_dades.inc');

if (strlen(trim($nom))==0) {
  echo ("*** ERROR: el camp nom no pot estar buit<br>");
  pagina_anterior();
  posa_peu_de_tornada();
  exit;
}

$correue   = $_POST['correue'];
$tlf_fixe  = $_POST['tlf_fixe'];
$tlf_mobil = $_POST['tlf_mobil'];


$dbd=connecta();

$sql="INSERT INTO la_agenda VALUES ('$nom', '$correue', '$tlf_fixe', '$tlf_mobil')";

$res = mysql_query($sql, $dbd);
if ($res)
  echo "Afegit ", mysql_affected_rows(), "registre/s a la bd<br>";
else
  echo "ERROR en afegir: ", mysql_error();

posa_peu_de_tornada();
?>
```

9.3.8. TOTAL DE REGISTRES

Per a mostrar el total de registres que hi ha a la base de dades s'utilitza una de les funcions agregades que té la sentència `SELECT`:

- `COUNT(*)`: retorna el total de files seleccionades
- `COUNT(DISTINCT camp)`: retorna el total de files seleccionades sense tenir en compte aquelles on el `camp` estigui repetit.
- `AVG(camp)`: retorna la mitja aritmètica de `camp`.
- `MAX(camp)`: retorna el valor màxim.
- `MIN(camp)`: retorna el valor mínim.

Per tant, avariguar el número total de registres també és molt senzill si es fa ús de la clàusula `COUNT`:

```
<html>
<head> <title>Agenda: Total de registres</title> </head>
<body>
<h2>Agenda: Total de registres a l'agenda</h2>
<?php
include('agbd_dades.inc');
$dbd=connecta();

$res = mysql_query("SELECT COUNT(*) from la_agenda", $dbd);
$total=mysql_fetch_array($res);
echo "<h4>Total de registres a l' agenda: <u>",$total[0], "</u></h4>";

posa_peu_de_tornada();
?>
```