Sessione ordinaria 2010
Seconda prova scritta

Ministero dell'Istruzione dell'Università e della Ricerca

M070 – ESAME DI STATO DI ISTITUTO TECNICO INDUSTRIALE

CORSO DI ORDINAMENTO

Indirizzo: INFORMATICA

Tema di: INFORMATICA GENERALE, APPLICAZIONI TECNICO SCIENTIFICHE INFORMATICA

(Testo valevole per i corsi di ordinamento e per i corsi sperimentali del Progetto "SIRIO")

Una Società di riparazione di apparecchi telefonici e televisivi desidera commissionare ad una software house la realizzazione di un sistema informativo per la gestione delle richieste di riparazione (apertura del TICKET di riparazione) degli articoli trattati da parte dei negozi specializzati (rivendite o riparatori, denominati PDA, Punti Di Accettazione).

Il processo di gestione prevede la memorizzazione di tutte le informazioni contenute nel modulo cartaceo utilizzato abitualmente dai PDA: codice identificativo dell'intervento (numero progressivo generato in automatico e non modificabile), codice del PDA, nominativo e recapito telefonico e/o e-mail del cliente, data di invio della richiesta di intervento, tipologia, marca e modello dell'apparecchio telefonico/televisivo, difetto o anomalia segnalata, ecc…

L'accesso al nuovo sistema informativo, verrà regolamentato mediante l'inserimento di username e password, in modo da garantire un accesso sicuro. Gli utenti potranno usufruire di alcune funzionalità: compilazione di un nuovo modulo di prenotazione on line, visualizzazione dello stato dei ticket (APERTO: il riparatore ha preso in carico la richiesta; IN_RIPARAZIONE: l'articolo è in riparazione; CHIUSO: l'articolo, riparato o non riparato, è pronto per la consegna al PDA), visualizzazione del tempo residuo stimato per la conclusione dell'intervento stesso.

Il candidato, formulate le opportune ipotesi aggiuntive, realizzi:

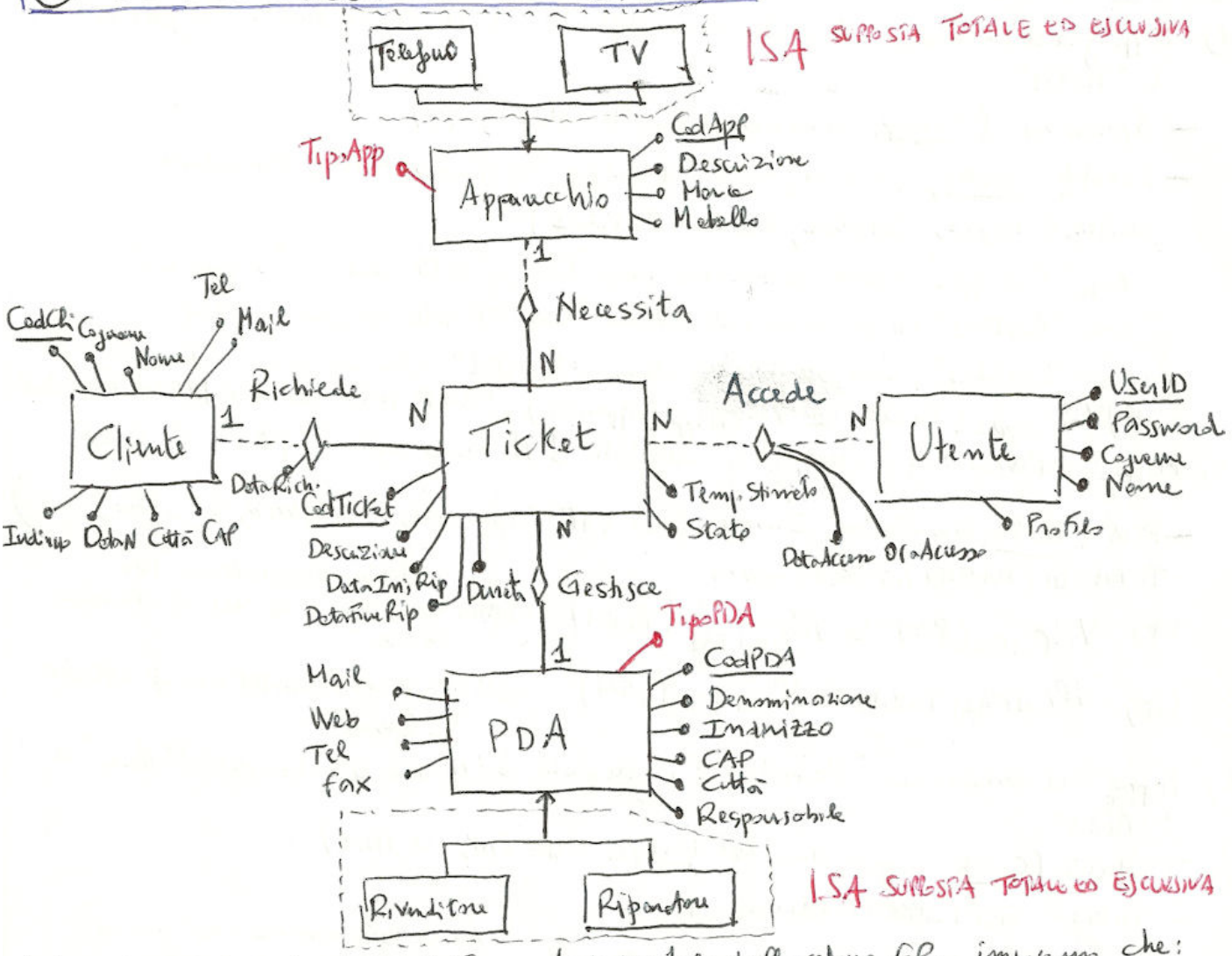
- Una analisi della realtà di riferimento, completa dello schema funzionale dell'architettura proposta.
- Uno schema concettuale ed uno schema logico del data base.
- La definizione delle relazioni e le seguenti interrogazioni espresse in linguaggio SQL:
  1. visualizzare l'elenco, in ordine cronologico, di tutti gli interventi di riparazione di telefoni cellulari del tipo "sostituzione display" effettuati nell'arco di un mese;
  2. visualizzare i dati dei clienti ai quali, dopo trenta giorni, non è stato ancora riparato l'articolo consegnato;
  3. data una marca ed un modello, visualizzare la durata media degli interventi di riparazione per i prodotti di tale marca e modello;
  4. calcolare e visualizzare quanti interventi di riparazione, andati a buon fine, sono stati effettuati, suddivisi per marca, nell'arco di un anno;
  5. calcolare e visualizzare quanti TICKET sono stati compilati da ciascun PDA e la durata media di lavorazione dei TICKET;
  6. dato il codice identificativo di un intervento, visualizzarne lo stato.
- La codifica, in un linguaggio di programmazione per il web, di un segmento significativo del progetto realizzato.

Durata massima della prova: 6 ore.

È consentito soltanto l'uso di manuali tecnici e di calcolatrici non programmabili.

Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

SCHEMA CONCETTUALE DEL DATABASE



N.B Oltre ai vincoli impliciti direttamente deducibili dallo schema ER imponiamo che:

V1: (Ticket.Stato = "APERTO" OR Stato = "IN_RIPARAZIONE" OR Stato = "CHIUSO_OK" OR Stato = "CHIUSO_NOT-OK")

V2: (Apparecchio.Tipo_All = "TELEFONO" OR Tipo_All = "TV")

V3: (PDA.TipoPDA = "RIVENDITORE" OR TipoPDA = "RIPARATORE")

V4: (Utente.Profilo = "AMMINISTRATORE" OR Profilo = "OPERATORE" OR Profilo = "OSPITE")

V5: (Ticket.DataIniRip ≤ DataFineRip)

V6: (Richiede.DataRich ≤ Ticket.DataFineRip)

I 2 attributi TipoApp e TipoPDA sono stati introdotti per mappare le 2 ISA individuate utilizzando le tecniche di mappa delle entità figlie nelle entità padri

② SCHEMA LOGICO (RELAZIONALE) DEL DATABASE

d) Mapping dell'associazione "Necessita" di molteplicità 1:N tra le entità "Apparecchio" e "Ticket"

- Apparecchio (CodApp, Descrizione, Marca, Modello, TipoApp)
- Ticket (CodTicket, Descrizione, Stato, DataIniRip, DataFinRip, Durata, TempoStimato CodApp1, CodPDA1, DataRich, CodCli1)

con "CodApp1" chiave esterna sulla colonna "CodApp" della relazione "Apparecchio"
con "CodPDA1" chiave esterna sulla colonna "CodPDA" della relazione "PDA"
con "CodCli1" chiave esterna sulla colonna "CodCli1" della relazione "Clienti"

(*) $VR_{CodApp1}(Ticket) \subseteq VR_{CodApp}(Apparecchio)$ DALLA TOTALITÀ DELL'ASS. INVERSA "È Montato"

b) Mapping dell'associazione "Gestisce" di molteplicità 1:N tra le entità "PDA" e "Ticket"

- PDA (CodPDA, Denominazione, Indirizzo, Città, CAP, Responsabile, Mail, Web, Tel, Fax, TipoPDA)
- Ticket: GIÀ MAPPATA IN PRECEDENZA

VINCOLO REFERENZIALE CHE DERIVA DALLA TOTALITÀ DELL'A. DIRETTA "Gestisce"

(*) $VR_{CodPDA}(PDA) \subseteq VR_{CodPDA1}(Ticket)$

DERIVA DALLA TOTALITÀ DELL'A. INVERSA "È Gestito"

(*) $VR_{CodPDA1}(Ticket) \subseteq VR_{CodPDA}(PDA)$

c) Mapping dell'associazione "Richiede" di molteplicità 1:N tra le entità "Clienti" e "Ticket"

- Clienti (CodCli, Cognome, Nome, DataN, Indirizzo, Città, CAP, Tel, Mail)
- Ticket: GIÀ MAPPATA IN PRECEDENZA

VINCOLO REFERENZIALE CHE DERIVA DALLA TOTALITÀ DELL'A. INVERSA "È Richiesto"

(*) $VR_{CodCli1}(Ticket) \subseteq VR_{CodCli}(Clienti)$

d) Mapping dell'associazione "Accede" di molteplicità N:N tra le entità "Utente" e "Ticket"

- Utente (UserID, Password, Cognome, Nome, Profilo)
- Ticket GIÀ MAPPATA IN PRECEDENZA
- Accede (UserID2, CodTicket2, DataAccesso, OraAccesso)

DAL MAPPING RELAZIONALE DI UNA ASSOCIAZIONE N:N

(*) $VR_{UserID2}(Accede) \subseteq VR_{UserID}(Utente)$

(*) $VR_{CodTicket2}(Accede) \subseteq VR_{CodTicket}(Ticket)$

MAPPING DEI VINCOLI

- VINCOLI DI CHIAVE PRIMARIA → VINCOLI DI INTEGRITÀ INTRARELAZIONALI O INTERNI SU PIÙ N-PLE
- VINCOLI TOTALITÀ DELLE A. → VINCOLI DI INTEGRITÀ INTERRELAZIONALI O ESTERNI DI TIPO REFERENZIALE

V1 (Ticket) : (Stato = "APERTO" OR Stato = "IN_RIPARAZIONE" OR Stato = "CHIUSO_OK" OR Stato = "CHIUSO_NOT-OK")

V2 (Apparecchio): (TipoApp = "TELEFONO" OR TipoApp = "TV")

V3 (PDA) : (TipoPDA = "RIVENDITORE" OR TipoPDA = "RIPARATORE")

V4 (Utente) : (Profilo = "AMMINISTRATORE" OR Profilo = "OPERATORE" OR Profilo = "OSPITE")

V5 (Ticket) : (DataIniRip ≤ DataFinRip)

V6 (Ticket): (DataRich ≤ DataIniRip)

V1 è di vincolo di integrità intrarelazionale su singola n-pla su singolo attributo
V2 " " " " " " " " "
V3 " " " " " " " " "
V4 " " " " " " " " "
V5 è di vincolo di integrità intrarelazionale su singola n-pla sul dominio di più attributi
V6 " " " " " " " " "

② DEFINIZIONE DELLE RELAZIONI IN SQL

```
CREATE DATABASE Centro_riparazioni;
CREATE DOMAIN MioTipoApp AS CHAR(8) CHECK (VALUE IN ("TELEFONO", "TV")); //V2
CREATE DOMAIN MioTipoPDA AS CHAR(11) CHECK (VALUE IN ("RIVENDITORE", "RIPARATORE")); // V3
CREATE DOMAIN MioStato AS CHAR(14) CHECK (VALUE IN ("APERTO", "IN RIPARAZIONE",
    "CHIUSO_OK", "CHIUSO_NOT_OK")); // V1
CREATE DOMAIN MioProfilo AS CHAR(14) CHECK (VALUE IN ("AMMINISTRATORE",
    "OPERATORE", "OSPITE")); // V4

CREATE TABLE Apparecchio
(
CodApp      CHAR(10) NOT NULL,
Descrizione VARCHAR(1000) NOT NULL,
Marca       CHAR(255) NOT NULL,
Modello     CHAR(255) NOT NULL,
TipoApp     MioTipoApp NOT NULL,
PRIMARY KEY (CodApp)
);
```

```
CREATE TABLE PDA
(
CodPDA      CHAR(10) NOT NULL,
Denominaz   CHAR(255) NOT NULL,
Indirizzo   CHAR(255) NOT NULL,
Citta       CHAR(50) NOT NULL,
Responsab   CHAR(255) NOT NULL,
Mail        CHAR(50) NOT NULL,
Web         CHAR(255),
Tel         CHAR(30) NOT NULL,
Fax         CHAR(30) NOT NULL,
TipoPDA     MioTipoPDA NOT NULL,
PRIMARY KEY(CodPDA)
);

CREATE TABLE Clienti
(CodCli     CHAR(10) NOT NULL,
Cognome     CHAR(50) NOT NULL,
Nome        CHAR(50) NOT NULL,
DataN       DATE     NOT NULL,
Indirizzo   CHAR(100) NOT NULL,
CAP         CHAR(5) NOT NULL
Citta       CHAR(50) NOT NULL,
Tel         CHAR(30) NOT NULL,
Mail        CHAR(50),
PRIMARY KEY(CodCli)
);

CREATE TABLE Utenti
(
UserID      CHAR(10) NOT NULL,
Password    CHAR(10) NOT NULL,
Cognome     CHAR(50) NOT NULL,
Nome        CHAR(50) NOT NULL,
PRIMARY KEY(UserID)
);
```

```
CREATE TABLE Accede
(
 UserID2  CHAR(10) NOT NULL,
 CodTicket2 CHAR(10) NOT NULL,
 DataAccesso  DATE,
 OraAccesso  TIME,
PRIMARY KEY (UserID2, CodTicket2),
FOREIGN KEY (UserID2) REFERENCES ON Utente (UserID),     // VR FK
FOREIGN KEY (CodTicket2) REFERENCES ON Ticket(CodTicket)   // VR FK
);

CREATE TABLE Ticket
(
 CodTicket  CHAR (10) NOT NULL,
 DescrizioneVARCHAR (1000) NOT NULL,
 Stato MioStato  NOT NULL,
 DataInizio  DATE  NOT NULL,
 DataFine  DATE
 Durata  TIME (0 anche DECIMAL (5,2)),
 Tempo Stimato TIME ( "     "     " ),
 CodApp1  CHAR (10) NOT NULL,
 CodPDA1  CHAR (10) NOT NULL,
 DataRich  DATE  NOT NULL,
 CodCli  CHAR (10) NOT NULL,
 PRIMARY KEY (CodTicket),
 FOREIGN KEY (CodApp1) REFERENCES ON Applicativo (CodApp), //VR FK
 FOREIGN KEY (CodPDA1) REFERENCES ON PDA (CodPDA),  //VR FK
 FOREIGN KEY (CodCli) REFERENCES ON Clienti (CodCli), //VR FK
 CHECK (DataInizio ≤ DataFine),   // V5
 CHECK ( DataRich ≤ DataInizio)    // V6
);
```

⑥ QUERY SQL (OVE POSSIBILE IN ALGEBRA RELAZIONALE)

Q_1: VISUALIZZARE IN ORDINE CRONOLOGICO L'ELENCO DI TUTTI GLI INTERVENTI DI RIPARAZIONE DI TELEFONI CELLULARI DEL TIPO "Sostituzione display" EFFETTUATI NELL'ARCO DI UN MESE

```
SELECT *
FROM Apparecchio, Ticket
WHERE (CodApp = CodApp1) AND (TipoApp = "TELEFONO") AND
      (Descrizione = "Sostituzione Display") AND (DataIniRip ≥ [StartMese])
      AND (DataIniRip ≤ [EndMese])
ORDER BY DataIniRip ;
```

Q_2: VISUALIZZARE I DATI DEI CLIENTI AI QUALI DOPO 30 GIORNI NON È STATO ANCORA RIPARATO L'ARTICOLO CONSEGNATO

OSS: Stato = "IN-RIPARAZIONE"

```
SELECT *
FROM Clienti, Ticket
WHERE (CodCli = CodCli1) AND (Stato = "IN-RIPARAZIONE")
      AND ([DataImmag] - DataIniRip > 30) ;
```

Q_3: Determinare marca e modello, visualizzare la durata media degli interventi di riparazione di tale marca e modello

```
SELECT AVG(Durata) AS mediaIntervento
FROM Apparecchio, Ticket
WHERE (CodApp = CodApp1) AND ((Stato = "CHIUSO_OK") OR Stato =
      "CHIUSO_NOT-OK")) AND (Marca = [MarcaAmpia]) AND
      (Modello = [ModelloAmpio]) ;
```

Q_4: CALCOLARE E VISUALIZZARE QUANTI INTERVENTI DI RIPARAZIONE ANDATI A BUON FINE SONO STATI EFFETTUATI, SUDDIVISI PER MARCA, NELL'ARCO DI UN ANNO

```
SELECT Marca, COUNT (*) AS NumInterventi
FROM Apparecchio, Ticket
WHERE (CodApp = CodApp1) AND (Stato = "CHIUSO_OK") AND
      (DataIniRip ≥ [StartDate]) AND (DataFineRip ≤ [EndDate]) ;
GROUP BY Marca;
```

Q_5: CALCOLARE E VISUALIZZARE QUANTI TICKET SONO STATI COMPILATI DA CIASCUN PDA E LA DURATA MEDIA DI LAVORAZIONE DEI TICKET

```
SELECT CodPDA, COUNT(*) AS NumTicket, AVG(Durata) AS MediaLavorazione
FROM PDA, Ticket
WHERE (CodPDA = CodPDA1) AND ((Stato = "CHIUSO") OR (Stato = "CHIUSO-NOK"))
GROUP BY CodPDA;
```

Q_6: DATO IL CODICE IDENTIFICATIVO DI UN INTERVENTO, VISUALIZZARNE LO STATO

```
SELECT Stato
FROM Ticket
WHERE (CodTicket = [Codice Immesso]);
```

$$\Pi_A(\sigma_P(\text{Ticket}))$$

$A = \{\text{Stato}\}$

$P = \{\text{CodTicket} = [\text{Codice Immesso}]\}$

⑥ Codifica in un linguaggio di programmazione per il Web, di una semplice funzionalità del progetto realizzato.

Questo punto richiederebbe uno sviluppo molto articolato che però a mio avviso va al di là delle possibilità fornite al candidato dal tempo a disposizione. Mi limiterò quindi ad indicare dei criteri di carattere generale.
Dato che viene richiesto la gestione dei ticket di assistenza attraverso un portale, il database deve essere disponibile su un server on line. Per far ciò si può pensare di utilizzare uno dei tanti tool presenti sul mercato per generare la parte grafica e uno dei tanti linguaggi attualmente esistente per la realizzazione di pagine dinamiche per il codice.
Molte sono le opzioni disponibili per lo sviluppo della componente server-side. Nella soluzione che presentiamo, abbiamo deciso di utilizzare il linguaggio di programmazione PHP che consente di arricchire le pagine Web di codice script che sarà eseguito direttamente sul server.
In particolare, PHP consente di implementare un motore di scripting server side molto diffuso e multipiattaforma con un buon supporto della connettività verso database diversi (ad es. dBase, Oracle, MySQL) attraverso componenti standard.
Nello specifico ipotizzeremo di interfacciarsi verso un DBMS MySQL, anch'esso multipiattaforma e piuttosto semplice da utilizzare
Ovviamente, per potere utilizzare PHP è necessario aver installato sul proprio sistema un Web Server (come ad esempio Apache).
Riepilogando si è deciso di utilizzare quindi per la realizzazione del portale e del relativo servizo Web una piattaforma **WAMP** basata su:
Sistema Operativo: **W**indows 2003 server
Web Server: **A**PACHE
Database Server: **M**YSQL
Linguaggio di programmazione lato server: **P**HP
mentre si utilizzerà il **linguaggio SQL** per la creazione delle tabelle e per le interrogazioni.
Esistono piattaforme complete open-source assolutamente gratuite che contengono il web server APACHE, il linguaggio di scripting lato server PHP ed il database relazionale MYSQL, (Ad esempio EasyPHP, PHPMyAdmin oppure Apache2triad) sia per ambiente Windows sia per ambiente Linux.
Naturalmente l'uso di una piattaforma **LAMP** (ossia con un operativo server anch'esso open source come **L**inux) sebbene da un lato porti l'indiscutibile vantaggio della totale gratuità rispetto agli alti costi di gestione della Microsoft, dall'altro può creare a tutti quegli utenti "non addetti ai lavori" che decidano di farne uso e che non abbiano un grosso livello di competenza tecnica informatica specifica più di qualche problemino di configurazione, di customizzazione e di utilizzo non sempre compatibili con le ragioni di efficienza aziendale
Infine va notato che per garantire un discreto margine di sicurezza, sarebbe opportuno far sì che le informazioni riservate, come ad esempio le password, non compaiano nel sorgente HTML della pagina, e che inoltre, tramite opportuni meccanismi, viaggino criptate attraverso la rete.

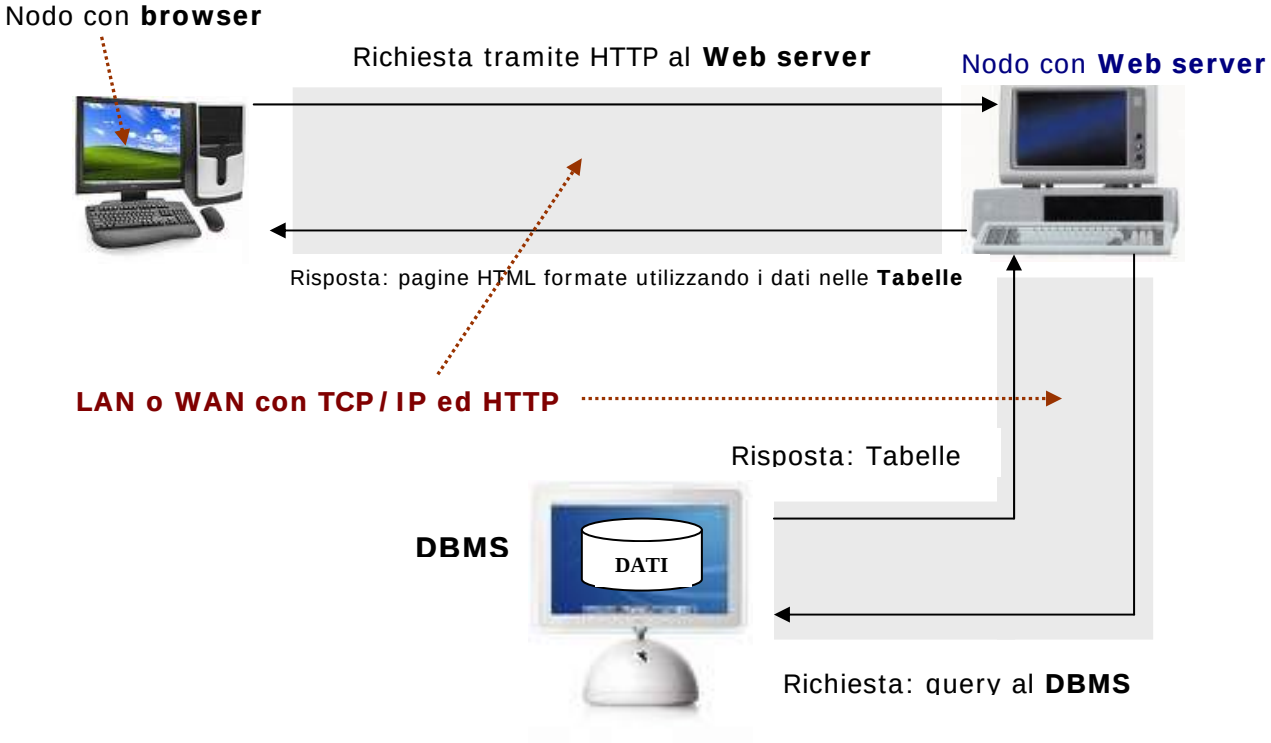
SCHEMA DI MASSIMA file stati.php (senza dettagli HTML embedded)

```
<? php //Segnalatore di inzio codice PHP
/*****************************************/
/* Valorizzazione parametri del database */
/*****************************************/
$db_host = "62.154.198.216"; // IP ADDRESS del database server che ospita i dati
$db_user = "root";
$db_pswd = "1234"; //per semplicità la pswd è in chiaro e non crittografata
$db_name = "miodb";
/*****************************************/
/* Connessione al database */
/*****************************************/
$connessione = mysql_connect($db_host, $db_user, $db_pswd)
or die("Connessione non riuscita" . mysql_error());
/*****************************************/
/* Selezione del database */
/*****************************************/
mysql_select_db($db_name) or die("Selezione del database non riuscita" . mysql_error());
/*****************************************/
/* Preparazione della query SQL */
/*****************************************/
// QUERY Q6: occorre costruire nella variabile $query, secondo la sintassi SQL, la query di interesse che dato un codice ticket
// ne visualizzi lo stato
//SELECT Stato
//FROM Ticket
//WHERE ( (CodTicket = $_POST[“CodiceTicket”]);
/*****************************************/
/* Esecuzione di una query SQL */
/*****************************************/
$risultato = mysql_query($query) or die("Query fallita" . mysql_error());
/*****************************************/
/* Calcolo del numero di occorrenze */
/*****************************************/
$numrighe=mysql_num_rows($risultato);
/*****************************************************************************************************/
/* Visualizzazione delle occorrenze in una pagina HTML usando i relativi TAG in modo embedded (funzione “echo”)*/
/*****************************************************************************************************/
if ($numrighe = = 1)
  {
  /*****************************************/
  /* Ciclo sul numero di record trovati dalla query */
  /*****************************************/
  for ($i=0;$i<$numrighe;$i++)
     {
      // esecuzione del codice PHP che prelevi tutti gi stati possibili nella tabella Ticket presenti nel db e che
       // costruisca una opportuna tabella di riepilogo (usando i TAG HTML “table
      }
   }
else
  {
   echo "CREDENZIALI UTENTE NON RICONOSCIUTE<br>";
   }
/*********************************************/
/* Liberazione delle risorse del risultato della query */
/*********************************************/
mysql_free_result($risultato) or die("Liberazione risorse fallita" . mysql_error());
/*****************************************/
/* Chiusura della connessione al database */
/*****************************************/
mysql_close($connessione) or die("Chiusura connessione fallita" . mysql_error());
?> //Segnalatore di fine codice PHP
```

In questo script abbiamo utilizzato la funzione che riceve come parametro il nome dell'host, il nome dell'utente e la password.
$connessione = ***mysql_connect****($host, $ user, $ pswd)*
per effettuare il collegamento con un host su cui gira un'istanza di MySQL.
Questa funzione restituisce un oggetto che rappresenta integralmente il nostro host e con il quale è possibile utilizzare la funzione
mysql_select_db*($database,[$connessione])*
per selezionare l'istanza corretta del database (essenziale in quanto l'istanza del database server può contenere più di un database).
Dopo aver selezionato il database, è possibile effettuare query su qualsiasi tabella tramite le funzione
$risultato = ***mysql_query****($query, [$connessione]);*
A questo punto l'oggetto *$risultato* contiene il risultato della query.
Per l'estrazione dei dati dalla select abbiamo utilizzato la funzione
mysql_result*($risultato, $i, $nomecampo)*
che restituisce i contenuti di una cella da un risultato MySQL dove l'argomento campo può essere l'indice della riga contenente tutti i dati. Per ottenre solo numeri di riga validi si è utilizzata la funzione
mysql_num_rows ($risultato)
che restituisce il numero di righe in un risultato. Questo comando è valido solo per le istruzioni SELECT.
Questa query ha estratto tutti i record della tabella Utente e con le opportune istruzioni HTML li abbiamo visualizzati all'interno di una tabella.
Per terminare correttamente una sessione di interrogazione ad MYSQL dobbiamo utilizzare le funzioni
mysql_free_result ($risultato);
mysql_clos e($connessione);
che rispettivamente libera tutta la memoria associata all'identificativo del risultato $*risultato e* chiude la connessione al server MySQL associata all'identificativo di connessione $connessione specificato permettendo eventualmente ad un altro client di utilizzarla.

Per quanto riguarda l'architettura di riferimento da utilizzare per accedere un database in rete per questa soluzione implementativa facciamo le seguente ipotesi di partenza:

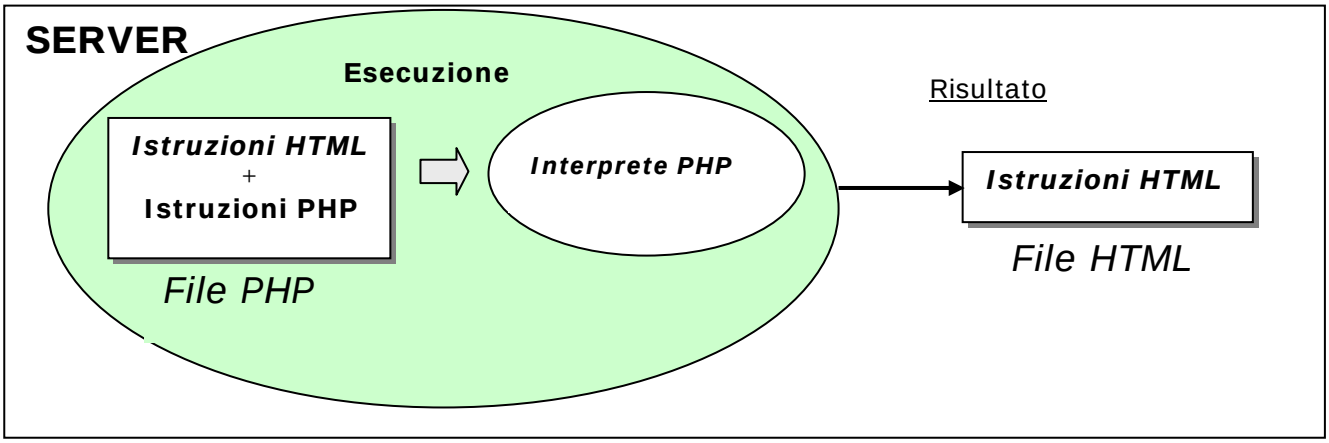
1) consideriamo il modello concettuale di **networking** (ovvero tutto ciò che riguarda la comunicazione tra reti diverse) semplificato a 4 livelli;
2) consideriamo i **protocolli** di comunicazione della famiglia **TCP/IP**, il **browser** come client universale ed il **Web server** come server universale



Per semplicità implementativa, in caso di una quantità di dati non eccessivamente numerosa, possiamo implementare sia il Web server (ad esempio APACHE) sia il Database server (ad esempio MYSQL) **sullo stesso nodo**.
Sempre per semplicità e chiarezza possiamo fare in modo di utilizzare l'**SQL** come linguaggio standard di interrogazione *e* manipolazione di un database ed utilizzeremo anche un **server SQL relazionale** (ad esempio MYSQL).
Dato l'**ambiente client-server** così ipotizzato in un'architettura di **protocolli TCP/IP ed HTTP** possiamo utilizzare per sviluppare tutte le funzionalità previste dall'applicazione complessiva, la **programmazione lato server** ossia possiamo sviluppare una serie di programmi applicativi che andranno in esecuzione prevalentemente sul *server*, accettando le richieste dal client e fornendo a quest'ultimo i risultati dell'elaborazione sottoforma di pagine HTML;

Tra i linguaggi di programmazione lato server possiamo utilizzareil PHP ossia un **linguaggio di scripting lato server** (come **PERL** ed **ASP).** Un linguaggio di scripting si differenzia dai *linguaggi di programmazione veri e propri lato server* perché non ha vita *autonoma* ma può essere utilizzato *esclusivamente* in quel contesto.

Il PHP è **u**n **linguaggio interpretato lato server:** infatti un **file di comandi PHP** è un classico esempio di programma interpretato lato server poiché infatti il Web server associa a tale file l'interprete PHP che deve essere mandato in esecuzione per poterne eseguire i comandi.



Per questo motivo è possibile avere *comandi PHP all'interno di pagine HTML* oppure *pagine PHP pure*.