TEMA DI INFORMATICA

Prof.ssa Alessandra Valpiani
Prof. Antonio Garavaglia
Docenti di informatica presso ITIS LAGRANGE di Milano

SOLUZIONE
Si tratta di un classico problema di progettazione ed implementazione di un database che richiede da parte del tecnico un'analisi puntuale dei bisogni in modo da ottimizzare non solo i tempi di accesso, ma anche l'occupazione di spazio sul supporto di memorizzazione.

In particolare risulta fondamentale analizzare lo schema logico del database in modo da ridurre al minimo la duplicazione dei dati, mantenendo però una struttura che, in caso di crash, consenta di recuperare, se non tutti, almeno buona parte dei dati.

Trattandosi di un archivio piuttosto complesso in cui gli elementi fondamentali, libri e clienti, risultano essere presenti in numero piuttosto consistente appare indispensabile prevedere per il sistema un sistema di backup automatico che consenta di ridurre al minimo le possibilità di perdite di dati (ad esempio prevedendo l'esistenza di un secondo server di supporto, su cui periodicamente viene aggiornata una copia del database con le variazioni apportate nell'ultimo periodo).

a.1)
Da un punto di vista logico, dopo aver adeguatamente analizzato la traccia, appare evidente che l'intero database si articola intorno alle seguenti entità fondamentali:

Libro, contenente tutti i dati relativi al libro in giacenza
Cliente, contenente tutti i dati relativi al cliente abilitato agli acquisti on line
Ordine, contenente i dati relativi agli ordini

Questa prima conclusione deriva dal fatto che i dati relativi ai reparti possono concettualmente essere visti come attributi del singolo libro, mentre il carrello Acquisti è chiaramente collegato al singolo ordine che, contabilmente, riprende i medesimi dati.
La traccia ministeriale, comunque, fornisce di per sé alcune utili indicazioni sulla linea da seguire nell'ottimizzare lo sviluppo del database, suddividendo le entità fondamentali in entità più semplici correlate tra loro mediante relazioni 1:1 e 1:N.

In base a queste considerazioni possiamo individuare le seguenti entità:

Reparto
- Codice
- Descrizione

Tipologia
- Codice
- Descrizione

Tipologia carta
- Codice
- Descrizione

Libro
- ISBN
- Titolo
- Autore
- Prezzo
- Codice Tip
- Cod Rep
- Dati supplementari
- Anno
- Editore

Collana
Data archiviazione
Foto

Cliente
- Codice fiscale
- Cognome
- Nome
- Indirizzo

Cap
- Città
- Nazione
- Telefono/Fax

Email
Password
Codice carta
Numero carta
Scadenza

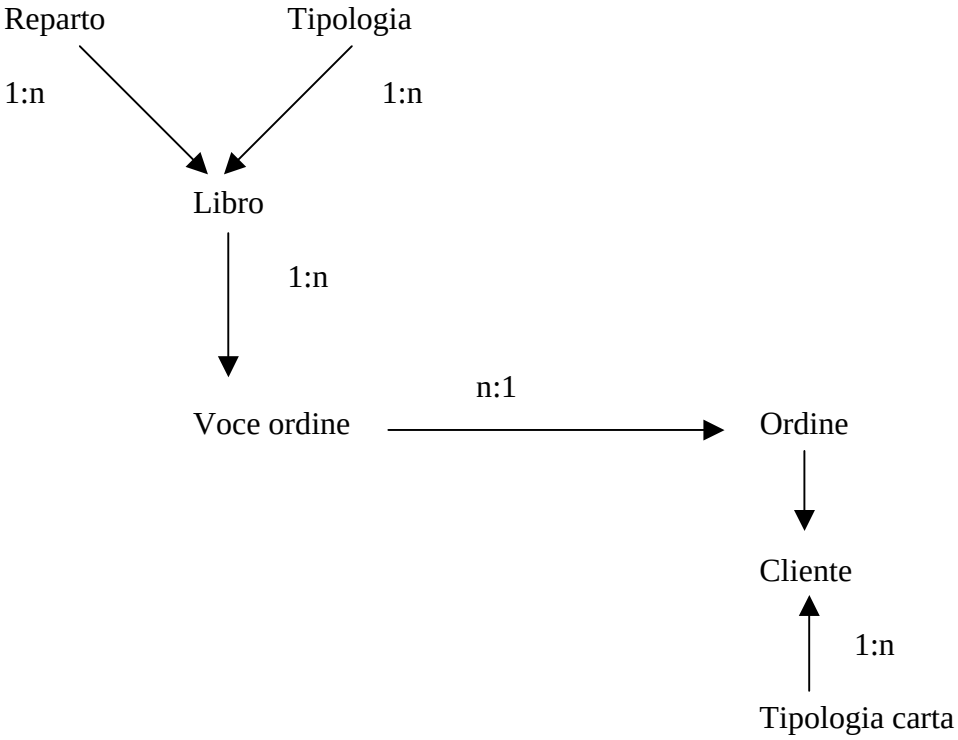
Ordine
- Numero
- Data
- codice fiscale cliente
- stato dell'ordine

scadenza prevista

Voce ordine
Numero ordine
ISBN
Quantitativo
Disponibile/In attesa

Possiamo notare che, nonostante le singole entità proposte non prevedano direttamente tutti gli attributi previsti dalla traccia, attraverso adeguate relazioni ed opportune query è possibile rispondere a tutte le richieste della traccia. Ad esempio per visualizzare il prezzo scontato di un libro sarà sufficiente applicare, nella visualizzazione dei dati, l'opportuno parametro di sconto derivante dalla categoria di appartenenza. Risulta evidente che il sistema potrebbe essere ulteriormente complicato consentendo al gestore del negozio di personalizzare lo sconto per ogni titolo: in tal caso occorrerebbe prevedere un campo specifico all'interno dell'entità libro.

Evidenziando le relazioni esistenti tra le varie entità possiamo schematizzare il database nel modo seguente:



a.2)
Definiamo ora nei suoi aspetti principali, a titolo esemplificativo, una delle relazioni contenute nel database, dato che risulterebbe semplicemente nozionistica la definizione di tutte le tabelle. Abbiamo scelto una tabella che contiene anche chiavi esterne di correlazione in modo da evidenziarne le caratteristiche.

```
CREATE TABLE libro
(       ISBN CHAR(10),
        titolo CHAR(10) not null,
        autore CHAR(10) not null,
        prezzo FLOAT not null,
        codice_tipologia INT,
        codice_reparto INT,
        ...
        primary key (ISBN),
        foreign key (codice_tipologia) references tipologia,
        foreign key (codice_reparto) references reparto,
        on delete restrict,
        on update cascade);
```

b.1)
La prima query richiesta e' un'interrogazione parametrica, nella quale i parametri dipendono dall'interpretazione che si può dare al quesito. Se si vogliono evidenziare, come appare logico supporre, tutti i dettagli del libro il codice SQL risulta essere del tipo:

```
SELECT libro.titolo, libro.autore, libro.prezzo, tipologia.descrizione
FROM   reparto, tipologia, libro
WHERE libro.tipologia = tipologia.codice AND libro.reparto = reparto.codice AND
(tipologia.descrizione = 'da non perdere' OR tipologia.descrizione = 'piu' venduti)'
AND reparto.descrizione = [inserire il nome del reparto]
ORDER BY tipologia.descrizione;
```

se invece si volesse optare per una soluzione più semplice, che si limita a fornire il numero dei libri appartenenti ad una certa categoria, potremmo scrivere:

```
SELECT COUNT(*)
FROM   reparto, tipologia, libro
WHERE libro.tipologia = tipologia.codice AND libro.reparto = reparto.codice AND
tipologia.descrizione = 'da non perdere'
GROUP BY reparto.descrizione = [inserire il nome del reparto];
```

b.2)
La query richiesta può essere realizzata nel modo seguente:

```
SELECT ordine.numero, ordine.data, ordine.stato,
ordine.scadenza, ordine.isbn, ordine.quantitativo, ordine.disponibile/in attesa
FROM ordine, cliente
WHERE ordine.stato='in corso' AND
cliente.codicefiscale=ordine.codicefiscale AND
cliente.codicefiscale = [inserire il codice del cliente]
ORDER BY ordine.numero;
```

c)
Dato che viene richiesto un negozio on line, il database deve essre disponibile su un server on line. Per organizzare il sito di presentazione del negozio 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.

```
<html>
        <head>
                <title> Accesso al database negozio on line libri </title>
```

```
	</head>
	<body>
	<?php
		$host = 'localhost';	//ipotizzando di accedere ad un server locale.
		$database = 'libri';

		$db = mysql_connect($host)
		or die (“Impossibile connettersi al server $host”);

		mysql_select_db($database, $db)
		or die (“Impossibile connettersi al database $database”);

		$query = “select nome_reparto from reparto”;
			$dbResult = mysql_query($query, $db);

		print “<table border=\”1\”>\n”;
		while ($line = mysql_fetch_array($dbResult, MYSQL_NUM))
		{
			print “\t<tr>\n”;
			foreach ($line as $col_value)
			{
				print “\t\t<td>$col_value</td>\n”;
			}
			print “\t<tr>\n”;
		}
		print “</table>\n”;
		mysql_free_result($dbResult);




		mysql_close($db);
	?>
	</body>
</html>
```

Nello script, abbiamo utilizzato la funzione che riceve come parametro il nome dell'host

```
	$db = mysql_connect($host)
```

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, $db)
```

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

```
$dbResult = mysql_query($query, $db);
```

A questo punto l'oggetto $dbResult contiene il risultato della query.
Questa query ha estratto tutti i record della tabella reparti e li ha visualizzati all'interno di una tabella HTML. Per l'etrazione abbiamo utilizzato la funzione

```
mysql_fetch_array($dbResult, tipo_array)
```

che restituisce l'i-esimo record del sresult set e sposta in avanti il cursore di indicizzazione. Il risultato dell'esecuzione di questa funzione è un array costituito da tutti gli elementi che sono stati estratti (tipo_array è MYSQL_NUM in quanto la chiave dell'array è un numero intero).
Al termine dell'esecuzione è necessario utilizzare la funzione

```
mysql_close($db)
```

per rilasciare la connessione al database e permettere eventualmente ad un altro client di utilizzarla.

Per l’accesso riservato agli utenti registrati è necessario predisporre a monte una pagina di autenticazione dell’utente, attraverso la quale (tramite un normale form html), l’utente introdurra' il proprio user name e la propria password, che saranno validate lato server (ad esempio da uno script di autenticazione php). Se la validazione ha successo, verrà chiamata la funzione mysql_connect con i valori di utente e password previsti per l'accesso di utenti registrati

```
$db = mysql_connect($host, $user, $password)
or die (“Impossibile connettersi al server $host”);
```

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.

Per eseguire query diverse da quella che, nello script proposto, produce l'elenco dei reparti è sufficiente inserire il codice nella query che deve essere eseguita e, quindi

```
$query = “ select ... “;
```

Inoltre, per questioni di sicurezza e per garantire la consistenza dei dati, sarebbe opportuno definire diverse tipologie di utenti del negozio on line (ad esempio utenti semplici, utenti registrati, gestori del magazzino libri e cosi' via), definendo appropriate modalita' di accesso ai dati. Determinare i livelli di accesso e i gruppi di utenti riconosciuti e' compito dell'amministratore della base dati; la regola di massima è quella di concedere i permessi di accesso strettamente necessari. Di norma, tali privilegi vengono assegnati attraverso il comando GRANT, disponibile in MySQL.

Tramite questo comando

```
GRANT SELECT ON libri.* TO pippo@localhost IDENTIFIED BY 'password' WITH
GRANT OPTION
```

Si concede all'utente l'autorizzazione all'esecuzione di query di estrazione dati, ma non la possibilita' di modificarli. Il permesso di inserire nuovi record viene, ad esempio, concesso, invece, tramite il comando GRANT INSERT.

COMMENTI CONCLUSIVI

Ovviamente la soluzione presentata può risultare poco efficace e non ottimizzata, ma la nostra scelta è stata quella di proporre una *soluzione realmente fattibile nel tempo assegnato* (6 ore)
Di conseguenza ci siamo limitati a prendere in considerazione gli aspetti essenziali del testo, tralasciando i problemi di connettività con gli uffici interni del negozio e con le procedure di aggiornamento delle tabelle che, per quanto riguarda i libri e i clienti, risultano ovviamente essere per il cliente a sola lettura. Sarà compito di altre parti del programma interagire attivamente con queste tabelle aggiornando la giacenza di magazzino, l’evasione degli ordini, la modica delle categorie di appartenenza dei singoli titoli, l’aggiornamento dei dati del cliente, e così via.