Progetto di Basi Di Dati

LiquoriVeneto

Nome – Matricola

4-09-2015

Indice

1 Progettazione Concettuale

1.1 Abstract di “LiquoriVeneto”

Si vuole realizzare una Base di Dati per una ditta che si occupa della vendita all'ingrosso di alcolici. La Base di Dati contiene le informazioni relative ai prodotti e alla loro vendita, documentata attraverso fatture, attraverso le quali è possibile tenere traccia della quantità disponibile per ogni prodotto. Sono inoltre contenute le informazioni riguardanti i clienti ed il personale, compresi gli stipendi ed i rimborsi maturati dai venditori in base ai km percorsi. La ditta mette a disposizione delle autovetture ai proprio dipendenti e ne viene registrato l'utilizzo.

1.2 Analisi dei requisiti

I clienti sono soggetti fiscali dotati di Partita IVA con la quale vengono identificati univocamente, per ognuno sono memorizzati i dati relativi al nome, che può identificare una persona fisica o persona giuridica, telefono, indirizzo e CAP.

Le fatture vengono inserite ad ogni vendita, sono identificate dal loro id, per ogni fattura vengono specificati data, cliente che effettua l'acquisto, venditore che emette fattura, l'IVA ed il totale calcolato grazie alle informazioni contenute nelle righe di ogni fattura.

Riga Fattura specifica per ogni fattura il codice e la quantità di prodotto acquistato. E' identificata da fattura e prodotto.

Ogni prodotto è identificato dal codice. Per ogni prodotto, essendo alcolici e complementi per cocktails contenuti in bottiglie, sono specificati: nome, marca, tipo, nazione, disponibilità, litri, gradazione e prezzo. Ad ogni vendita varia la quantità disponibile.

Ogni dipendente è individuato grazie al codice fiscale, sono inoltre immagazzinate informazioni personali quali nome, cognome, indirizzo e cellulare.

I venditori sono la parte di dipendenti la cui mansione sia vendite, sono individuati univocamente dal codice fiscale e per ognuno vengono specificati i km correntemente percorsi con i quali viene calcolato il rimborso corrente spettante ad ogni venditore.

Lo stipendio è identificato dal codice fiscale del dipendente a cui si riferisce e per ognuno è indicato l'ammontare.

La ditta dispone di autovettiure identificate dalla targa. Viene riportato il modello di ogni autovettura.

Quando viene permesso l' utilizzo di un'autovettura aziendale si registra in utilizzo_vettura la targa del mezzo e il codice fiscale del dipendente utilizzatore.

1.3 Schema Concettuale

1.3.1 Entità

cliente: effettuano acquisti presso la ditta e ottengono fattura
-p_iva char(11)
-nome varchar(20)
-indirizzo varchar(50)
-telefono varchar(13)
-CAP decimal(5,0)

fattura: per ogni vendita specifica clinete, venditore, IVA e totale
-id varchar(5)
-data date
-venditore varchar(16)
-cliente char(11)
-totale decimal(8,2)
-IVA decimal(8,2)

riga_fattura: specifica la quantità per rogni prodotto contenuto della fattura
-fattura varchar(5)
-prodotto char(6)
-unita decimal(9,0)

prodotto: tutti gli lacolici e I complementi per cocktails nel listino della lditta
-codice varchar(6)
-nome varchar(40)
-marca varchar(20)
-tipo varchar(20)
-nazione varchar(20)
-disponibilita decimal(9,0)
-prezzo decimal(8,2)
-litri decimal(8,2)
-gradazione decimal(4,1)

venditore: (sottoinsieme di venditore) emette fattura e si sposta nel territorio al fine di effettuare vendite, per questo ottiene un rimborso
-rimborso_cor decimal(8,2)
-km_cor decimal(8,0)

dipendente: comprende tutti I dipendenti per tutte le mansioni nella ditta
-cod_fis varchar(16)
-nome varchar(20)
-cognome varchar(20)
-mansione varchar(20)
-indirizzo varchar(50)
-cellulare varchar(13)

stipendio: ammontare ensile dovuto per ogni impiegato
-dipendente varchar(16)
-ammontare decimal(8,2)

autovettura: vetture aziendali messe a disposizione dei dipendenti
-targa char(7)
-modello varchar(40)

utilizzo_vettura: registra l'autorizzazione utilizzo delle vetture aziendali
-vettura char(7)
-dipendente varchar(16)

1.3.2 Relazioni

Ottiene:
Mette in relazione cliente-fattura con cardinalità(1,N), ogni cliente ha effettuato almeno un acquisto, e fattura-cliente con cardinalità(1,1) in quanto pe rogni fatttura è specificato un unico cliente .

Comprende:
Mette in relazione fattura - riga fattura con cardinalità (1,N), ogni fattura contiene una o più righe, riga fattura – fattura con cardinalità (1,1), ogni riga compete ad una sola fattura.

Contiene:
Mette in relazione riga fattura – prodotto con cardinalità (1,1), ogni riga contiene un solo prodotto, mette in relazione prodotti – riga fattura con cardinalità (0,N), ogni prodotto compare in nessuna oppure molte righe, in quanto può essere acquistato da qualsiasi cliente in qualsiasi fattura.

Emette:
Mette in relazione fattura – venditore con cardinalità (1,1), ogni fatture è emessa da un solo venditore. Mette in relazione venditore – fattura con cardinalità (0,N), un venditore può non aver emesso fatture, o averne emesse un certo numero.

Compete:
Mette in relazione dipendente – stipendio con cardinalità (1,1), mette in relazione riga stipendio – dipendente con cardinalità (1,1), ad ogni dipendente compete un solo stipendio mensile.

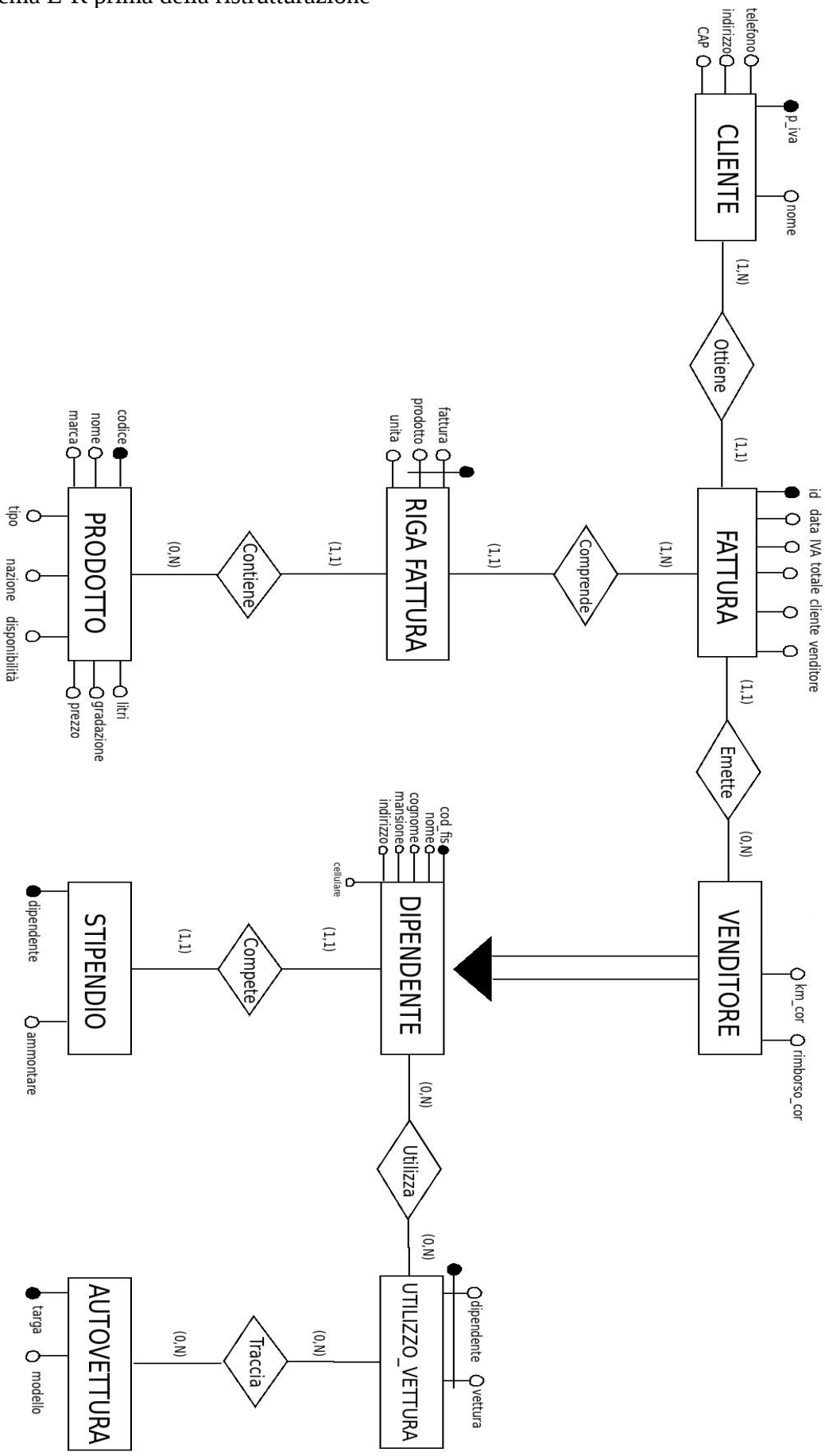
Utilizza:
Mette in relazione dipendente – utilizzo vettura con cardinalità (0,N), mette in relazione utilizzo vettura – dipendente con cardinalità (0,N), ogni dipendente può essere autorizzato ad utilizzare qualsiasi vettura. Ogni vettura può rimanere inuitilizzata o essere utilizzata da uno o più dipendenti .

Traccia:
Mette in relazione utilizzo vettura – autovettura con cardinalità (0,N) e mette in relazione autovettura – utilizzo vettura con cardinalità (0,N). Ogni vettura può non essere utilizzata o essere utilizzata da qualsiasi numero di dipendenti, ogni autorizzazione è tracciata inserendo targa e codice fiscale del dipendente utilizzatore.

1.3.3 Gerarchie

Nello schema l'Entità DIPENDENTE è la generalizzazione che rappresenta I dipendenti della ditta con assegnata qualsiasi mansione. Venditore è un sottoinsieme di tale generalizzazione che non è totale, infatti comprende solo i dipendenti aventi mansione “vendite”. Il sottoinsieme VENDITORE essendo separato dall'entita DIPENDENTE permette di affettuare agevolmente le operazioni di inserimento dei km percorsi ed il conteggio dei rimborsi essendo operazioni rigiuardanti soltato questa categoria di dipendenti.

1.3.4 Schema E-R prima della ristrutturazione



2 Progettazione logica

2.1 Modello Relazionale

CLIENTE(p_iva, nome, telefono, indirizzo, CAP)

FATTURA(id, data, cliente, venditore, IVA, totale)

RIGA_FATTURA(fattura, prodotto, unita)

PRODOTTO(codice, nome, marca, tipo, nazione, disponibilita, litri, gradazione, prezzo)

VENDITORE(cod_fis, km_cor, rimborso_cor)

DIPENDENTE(cod_fis, nome, cognome, mansione, indirizzo, cellulare)

STIPENDIO(dipendente, ammontare)

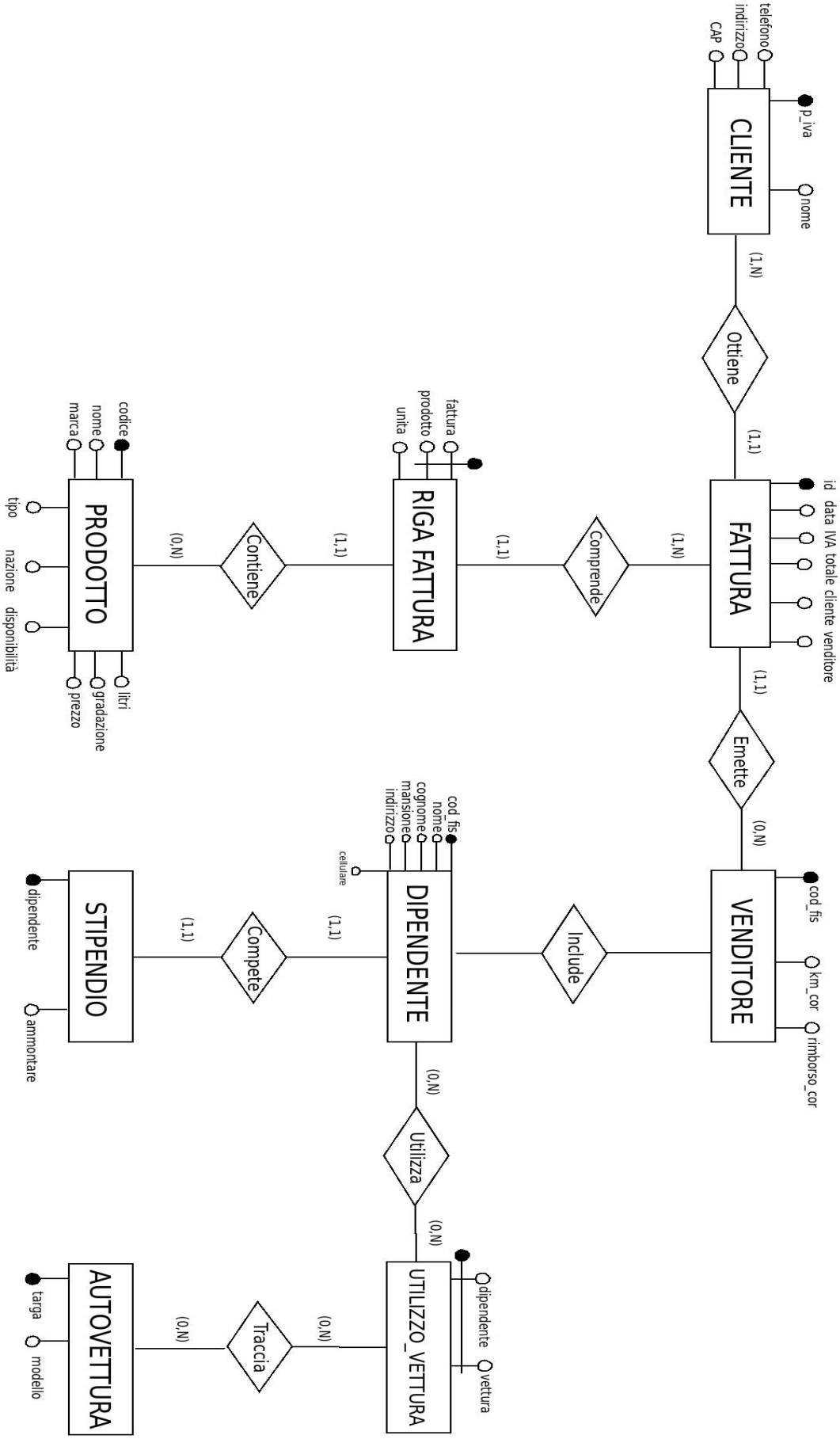
UTILIZZO_VETTURA(dipendente, vettura)

AUTOVETTURA(targa, modello)

2.2 Ristrutturazione

Nello schema la generalizzazione DIPENDENTE → VENDITORE non è totale, comprende solo I dipendenti aventi mansione “vendite”, è stata sostituita da due Entità in quanto esistono operazioni riguardanti solo l'entità figlia VENDITORE. Venditore essendo un sottoinsieme di dipendente mantiene l'attributo cod_fis per permettere l'identificazione dei venditori e comprende gli attributi km_cor e rimborso_cor con I quali si svolgono le operazioni di conteggio dei rimborsi, operazioni riguardanti soltato questa categoria di dipendenti. Non mantiene gli attributi riguardanti I dati personali dei dipendenti inutili al fine di tale calcolo.

2.3 Schema E-R dopo la ristrutturazione



2.4 Entità

cliente: effettuano acquisti presso la ditta e ottengono fattura
-p_iva char(11)
-nome varchar(20)
-indirizzo varchar(50)
-telefono varchar(13)
-CAP decimal(5,0)

fattura: per ogni vendita specifica clinete, venditore, IVA e totale
-id varchar(5)
-data date
-venditore varchar(16)
-cliente char(11)
-totale decimal(8,2)
-IVA decimal(8,2)

riga_fattura: specifica la quantità per rogni prodotto contenuto della fattura
-fattura varchar(5)
-prodotto char(6)
-unita decimal(9,0)

prodotto: tutti gli lacolici e I complementi per cocktails nel listino della lditta
-codice varchar(6)
-nome varchar(40)
-marca varchar(20)
-tipo varchar(20)
-nazione varchar(20)
-disponibilita decimal(9,0)
-prezzo decimal(8,2)
-litri decimal(8,2)
-gradazione decimal(4,1)

venditore: emettono fattura e si spostano nel territorio al fine di effettuare vendite, per questo ottengono rimborsi
-cod_fis varchar(16)
-rimborso_cor decimal(8,2)
-km_cor decimal(8,0)

dipendente: comprende tutti I dipendenti per tutte le mansioni nella ditta
-cod_fis varchar(16)
-nome varchar(20)
-cognome varchar(20)
-mansione varchar(20)
-indirizzo varchar(50)
-cellulare varchar(13)

stipendio: ammontare ensile dovuto per ogni impiegato
-dipendente varchar(16)
-ammontare decimal(8,2)

autovettura: vetture aziendali messe a disposizione dei dipendenti
-targa char(7)
-modello varchar(40)

utilizzo_vettura: registra l'autorizzazione utilizzo delle vetture aziendali
-vettura char(7)
-dipendente varchar(16)

3 Creazione del Database tramite codice SQL

3.1 Creazione Tabelle

```
CREATE TABLE IF NOT EXISTS `cliente` (
  `p_iva` char(11) PRIMARY KEY ,
  `nome` varchar(20) ,
  `indirizzo` varchar(50) ,
  `telefono` varchar(13) ,
  `CAP` decimal(5,0) ,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `dipendente` (
  `cod_fis` varchar(16) PRIMARY KEY ,
  `nome` varchar(20) ,
  `cognome` varchar(20) ,
  `mansione` varchar(20) ,
  `indirizzo` varchar(50) ,
  `cellulare` varchar(13) ,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE IF NOT EXISTS `venditore` (
  `cod_fis` varchar(16) PRIMARY KEY ,
  `rimborso_cor` decimal(8,2) DEFAULT NULL,
  `km_cor` decimal(8,0) DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `fattura` (
  `id` varchar(5)  PRIMARY KEY ,
  `data` date NOT NULL DEFAULT '0000-00-00',
  `venditore` varchar(16) references venditore(cod_fis),
  `cliente` char(11) references cliente(p_iva),
  `totale` decimal(8,2) DEFAULT NULL,
  `IVA` decimal(8,2) DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `riga_fattura` (
  `fattura` varchar(5) NOT NULL,
  `prodotto` char(6) NOT NULL,
  `unita` decimal(9,0) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE IF NOT EXISTS `prodotto` (
  `codice` varchar(6) PRIMARY KEY ,
  `nome` varchar(40) ,
  `marca` varchar(20) ,
  `tipo` varchar(20) ,
  `nazione` varchar(20) ,
  `disponibilita` decimal(9,0) NOT NULL,
  `prezzo` decimal(8,2) NOT NULL,
  `litri` decimal(8,2) ,
  `gradazione` decimal(4,1) ,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `stipendio` (
  `dipendente` varchar(16) references dipendente(cod_fis),
  `ammontare` decimal(8,2) ,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `autovettura` (
  `targa` char(7) PRIMARY KEY ,
  `modello` varchar(40) ,
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `utilizzo_vettura` (
  `vettura` char(7) references autovettura(targa),
  `dipendente` varchar(16) references dipendente(cod_fis)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

3.2 Triggers

Il trigger calcoloiva permette il calcolo automatico dell'IVA, al 20% per tutti I prodotti, nella tabella FATTURA all'inserimento del totale nella tabella FATTURA. Il trigger calcoloagg si occupa di ricalcolare il campo IVA quado viene fatto un'aggiornamento.

```
DROP TRIGGER IF EXISTS `calcoloiva`;
DELIMITER //
CREATE TRIGGER `calcoloiva` BEFORE INSERT ON `fattura`
 FOR EACH ROW set new.IVA=new.totale*20/100
//
DELIMITER ;


DROP TRIGGER IF EXISTS `calcoloagg`;
DELIMITER //
CREATE TRIGGER `calcoloagg` BEFORE UPDATE ON `fattura`
 FOR EACH ROW set new.IVA=new.totale*20/100
//
DELIMITER ;
```

Il trigger rimborsi calcola in modo automatico il rimborso dovuto ad ogni venditore in funzione dei km percorsi. Ad ogni inserimento nella tabella VENDITORI il trigger calcola il campo rimborso_cor nella stessa tabella moltiplicando il valore del campo km_cor per una costante(1,60).
Il trigger rimborsiagg si occupa di effettuare lo stesso calcolo ad ogni modifica in modo da avere un rimborso corrente esatto.

```
DROP TRIGGER IF EXISTS `rimborsi`;
DELIMITER //
CREATE TRIGGER `rimborsi` BEFORE INSERT ON `venditore`
 FOR EACH ROW set new.rimborso_cor=new.km_cor*1.60
//
DELIMITER ;


DROP TRIGGER IF EXISTS `rimborsiagg`;
DELIMITER //
CREATE TRIGGER `rimborsiagg` BEFORE UPDATE ON `venditore`
 FOR EACH ROW set new.rimborso_cor=new.km_cor*1.60
//
DELIMITER ;
```

Il trigger scorte permette di effettuare un controllo sulle scorte diponibili per un certo prodotto quando viene inserita la riga di una fattura in RIGA_FATTURA, se il prodotto non fosse disponibile in quantità sufficiente per la vendita verrebbe segnalato. Per effettuare il calcolo viene fatto un confronto fra il valore inserito come “unita” e il valore “disponibilita” nella tabella PRODOTTO in corrispondenza del “codice” del prodotto da vendere corrispondente al valore inserito in “prodotto” nella riga_fattura. Nel caso la disponibilità di tale prodotto sia superiore alla quantita da vendere l'inserimento verrà permesso.

```
DROP TRIGGER IF EXISTS `scorte`;
DELIMITER //
CREATE TRIGGER `scorte` BEFORE INSERT ON `riga_fattura`
 FOR EACH ROW begin
if new.unita>(select disponibilita
                              from prodotto
          where new.prodotto=prodotto.codice)

then
         insert into riga_fattura select * from riga_fattura LIMIT 1;
end if;
end
//
DELIMITER ;
```

Il trigger riduzionescorte permette di gestire dinamicamente le quantità diponibili per ogni prodotto in base alle vendite. Qundo l'inserimento di una riga in RIGA_FATTURA ha successo il trigger si attiva e aggiorna la quantità di prodotto disponibile per il prodotto inserito, il valore “disponibilita” nell atabella PRODOTTO subisce una diminuzione pari al valore di “unita” appena inserito in RIGA_FATTURA.
Il trigger riduzioneagg ripropone il calcolo qunado una riga di RIGA_FATTURA subisce un'aggiornamneto. Un au,ento del valore di “unita” in RIGA_FATTURA causa una diminuzione del valore di disponibilita in corrispondenza del codice prodotto inserito, a cui somma il valore precedentemente contenuto in “unita” in modo da eseguire un calcolo corretto. Senza la somma della vecchia quantità subiremmo una diminuzione errata, formata sia dal vecchio valore sia dal nuovo, l'obiettivo è una diminuzione pari solo al valore aggiornato. Ciò permette anche un calcolo corretto in caso di diminuzione della quantità venduta.

```
DROP TRIGGER IF EXISTS `riduzionescorte`;
DELIMITER //
CREATE TRIGGER `riduzionescorte` AFTER INSERT ON `riga_fattura`
 FOR EACH ROW update prodotto set prodotto.disponibilita=disponibilita-new.unita where
prodotto.codice=new.prodotto
//
DELIMITER ;


DROP TRIGGER IF EXISTS `riduzioneagg`;
DELIMITER //
CREATE TRIGGER `riduzioneagg` AFTER UPDATE ON `riga_fattura`
 FOR EACH ROW update prodotto set prodotto.disponibilita=disponibilita-new.unita+old.unita
where prodotto.codice=new.prodotto
//
DELIMITER ;
```

3.3 Procedures

La procedure nuovafattura permette l'inserimento di nuove voci nella tabella FATTURA e automatizza il calcolo di alcuni suoi campi. Accetta in input 'fid', 'fdata', 'fvenditore' e 'fcliente' che vengono rispettivamente inseriti nella tabella FATTURA nei campi 'id', 'data', 'venditore' e 'cliente'.
Dichiara ftotale come decimal(8,2) in cui sarà inserito il totale di FATTURA.
Seleziona i codici dei prodotti presenti nelle righe della fattura 'fid', trova il loro prezzo e lo moltiplica per le unità ordinate. Somma questi valori e li memorizza in 'ftotale'.
Il campo IVA è calcolato dinamicamente attraverso il trigger 'calcoloiva'.

```
DELIMITER $$
CREATE PROCEDURE `nuovafattura`(IN `fid` VARCHAR(5), IN `fdata` DATE, IN `fvenditore`
VARCHAR(16), IN `fcliente` CHAR(11))
begin
declare ftotale decimal(8,2);
                        SELECT sum(totu) into ftotale
                        FROM (SELECT fattura, prodotto, prezzo * unita as totu
                                FROM prodotto, riga_fattura
                                WHERE prodotto.codice = riga_fattura.prodotto
                                GROUP BY fattura, prodotto) AS p
                        WHERE fattura=fid
insert into fattura(id,data,venditore,cliente,totale) values(fid,fdata,fvenditore,fcliente,ftotale);
end$$
```

La procedure nuovovenditore permette l'inserimento di una nuova entry nella tabella VENDITORE. Accetta in input un codice fiscale di un dipendente in 'vcod_fis' e i km da esso percorsi in 'vkm_cor'. La procedure esegue un controllo sul codice fiscale, se è presente nella tabella DIPENDENTE controlla il campo mansione: se equivale a 'vendite' l'inserimento viene permesso. In quel caso si attiva il trigger 'rimborsi' che calcola ed inserisce il rimborso per tale venditore.

```
CREATE PROCEDURE `nuovovenditore`(IN `vcod_fis` VARCHAR(16), IN `vkm_cor`
DECIMAL(8,2))
begin
if exists(select * from dipendente where cod_fis=vcod_fis and mansione="vendite")
then insert into venditore(cod_fis,km_cor) values(vcod_fis,vkm_cor);
else insert into venditore select* from venditore limit 1; end if; end
```

3.4 Queries

1. Dati personali dei clienti che hanno comprato sempre e solo prodotti analcolici.

```
select*
from cliente
where p_iva not in(select cliente
            from fattura
            where id in(select fattura
                  from riga_fattura
                  where prodotto in(select codice
                              from prodotto
                              where gradazione <> 0)))
```

OUTPUT

p_iva	nome	indirizzo	telefono	CAP
00488410010	Hotel Torino	via Romano 56	0495326345	35100

2. Per ogni tipo di prodotto, le quantità vendute.

```
select tipo, sum(unita) as qtvenduta
from riga_fattura left join prodotto on riga_fattura.prodotto=prodotto.codice
group by tipo
order by qtvenduta desc
```

OUTPUT

tipo	qtvenduta
vodka	319
gin	219
bitter	192
amaro	144
aperitivo	140
liquore	102
sciroppo	42
zucchero	33
acquavite	32
brandy	20
cognac	14
rum	13
crema	8

3. Il ricavo che I singoli venditori sono riusciti a procurare all'azienda.

```
select venditore, sum(totale) as ricavo
from fattura
group by venditore
order by ricavo desc
```

OUTPUT

venditore	ricavo
FSSLNZ89C29F342D	10354.00
NTONDR79T23G456T	9726.65
NRELRT80D14Q231F	1241.50

4. Partita iva e nome dei clienti che hanno acquistato da un unico venditore di cui specifichiamo il codice fiscale.

```
select p_iva, nome, venditore
from (select count(venditore)as ven , cliente, venditore
    from fattura
    group by cliente) as v join cliente on v.cliente=cliente.p_iva
where ven=1
```

OUTPUT

p_iva	nome	venditore
00137770210	Sergio Celli	NRELRT80D14Q231F
00195530043	Marino Galli	FSSLNZ89C29F342D
00488410010	Hotel Torino	NRELRT80D14Q231F
03231640487	Rossi Mario	FSSLNZ89C29F342D
05927271006	Piero Bitti	FSSLNZ89C29F342D
12283130156	Bar Nazionale	NTONDR79T23G456T
12654765431	Ciro Silli	FSSLNZ89C29F342D
12717510155	Hotel Garden	NTONDR79T23G456T
12934530150	Gino Luci	NRELRT80D14Q231F
20128183028	Paolo Rossini	NTONDR79T23G456T
22454410289	Da Umberto	FSSLNZ89C29F342D

5. Da quante nazioni provengono I prodotti di ogni fattura.

```
select fattura, count(nazione) as nazioni
from riga_fattura left join prodotto on riga_fattura.prodotto=prodotto.codice
group by fattura
```

OUTPUT

fattura	nazioni
160	1
161	2
162	5
163	4
164	4
165	1
166	4
167	1
168	1
169	4
170	3
171	4
172	4
173	4
174	8
175	4
176	5
177	6
178	5
179	7
180	8
181	2
182	1
183	2
184	1
189	1

6. Quanti prodotti ci sono per ogni nazione.

```
select nazione, count(codice) as qtprodotti
from prodotto
group by nazione
```

OUTPUT

nazione	qtprodotti
Barbados	1
Brasile	2
Colombia	1
Francia	14
Germania	1
Giappone	1
Guatemala	1
Irlanda	2
Italia	18
Polonia	3
Repubblica dominicana	1
Russia	6
Scozia	3
spagna	1
Svezia	4
Trinidad e Tobago	1
UK	5
USA	2

4 Interfaccia Web

Il sito è dotato un file stile.css che permette l'inserimento di un logo aziendale e un menù di navigazione per spostarsi fra le pagine in modo veloce e semplice.

```
body {
    background-color: #FFDBE0;
}
.logo{
text-indent:-999999px;
background: url('logo.png');
max-width: 119px; height: 115px;
}

nav ul
{
background-color:534640;
overflow:hidden;
text-align:inline;
color:white;
padding:10px;
}
a{
text-decoration:none;
color:white;
font-family: "Arial", Helvetica, sans-serif;
}
nav ul li
{
padding:10px;
display:inline-block;
}
nav ul li:hover
{
background-color:broown;
padding:10px;
}

h1 {
color: #700000 ;
text-align: center;
}
```

Il file prod.php presenta un listino prezzo contenente tutti I prodotti in vendita presso la ditta LiquoriVeneto ed il loro prezzo.

```
<html>
<head>
	<title>Prodotti</title>
	<link rel="stylesheet" type="text/css" href="stile.css">
</head>

<body><h1 class="logo"></h1></body>

<nav>
<ul>
<li><a href="prod.php">LISTINO</a></li>
<li><a href="quatt.php">INTERROGAZIONE DATABASE</a></li>
<li><a href="vend.php">INSERIMENTO VENDITORI</a></li>
<li><a href="fatt.php">INSERIMENTO FATTURE</a></li>
</ul>
</nav>


<body>
<?php
$link = mysql_connect('localhost', '----, '------');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}

$db_selected = mysql_select_db('LiquoriVeneto', $link);
if (!$db_selected) {
   die ('Can\'t use LiquoriVeneto : ' . mysql_error());
}
$risultato = mysql_query("SELECT * from prodotto")
   or die("Query non valida: " . mysql_error());

echo "<pre> Prodotto                    Prezzo</pre>";
echo"<table>";
while ($row = mysql_fetch_assoc($risultato)) {
$nome=$row['nome'];
$prezzo=$row['prezzo'];

echo"<tr><td>".$nome."</td><td>".$prezzo."</td></tr>";

}
echo"</table>";
echo nl2br("\n");

?>
</body>
</html>
```

La pagina quatt.php permette di interagire con il Database e visualizzare il risultato di query.

```
<html>
	<link rel="stylesheet" type="text/css" href="stile.css">
	<head><title>Area Dispositiva</title></head>
	<body>
		<h1 class="logo"></h1>
	</body>
<nav>
	<ul>
		<li><a href="prod.php">LISTINO</a></li>
		<li><a href="quatt.php">INTERROGAZIONE DATABASE</a></li>
		<li><a href="vend.php">INSERIMENTO VENDITORI</a></li>
		<li><a href="fatt.php">INSERIMENTO FATTURE</a></li>
	</ul>
</nav>
<body>
	<form method="post" action="">
	<input type="text" name="value">
	<input type="submit">
</form>
<?php
$prova = $_POST['value'];

// we connect to localhost
$link = mysql_connect('localhost', '-----', '------');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';

$db_selected = mysql_select_db('LiquoriVeneto', $link);
if (!$db_selected) {
   die ('Can\'t use LiquoriVeneto : ' . mysql_error());
}
echo nl2br(" \n and Successfully selected db LiquoriVeneto\n");


$risultato = mysql_query($prova)
   or die("Query non valida: " . mysql_error());

$num_righe = mysql_num_rows($risultato);

echo nl2br("La query ha prodotto  $num_righe Righe\n");

while ($row = mysql_fetch_array($risultato, MYSQL_ASSOC)) {
   print_r($row);
	echo nl2br(" \n");
}
?>
</body>
</html>
```

La pagina vend.php facilita l'inserimento di nuovi venditori facendo uso della procedure nuovovenditore.

```
<html>
<head>
	<title>Inserimento Venditori</title>
	<link rel="stylesheet" type="text/css" href="stile.css">
</head>

<body><h1 class="logo"></h1></body>

<nav>
<ul>
<li><a href="prod.php">LISTINO</a></li>
<li><a href="quatt.php">INTERROGAZIONE DATABASE</a></li>
<li><a href="vend.php">INSERIMENTO VENDITORI</a></li>
<li><a href="fatt.php">INSERIMENTO FATTURE</a></li>
</ul>
</nav>

<?php
echo nl2br("<pre> Codice Fiscale	Km Percorsi</pre>");
?>
<body>
<form method="post" action="">
	<input type="text" name="codice">
	<input type="text" name="kilometers">
	<input type="submit">
</form>

<?php
$cod = $_POST['codice'];
$km = $_POST['kilometers'];

$link = mysql_connect('localhost', '-----', '--------');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
$db_selected = mysql_select_db('LiquoriVeneto', $link);
if (!$db_selected) {
   die ('Can\'t use LiquoriVeneto : ' . mysql_error());
}
$risultato = mysql_query("call nuovovenditore('$cod','$km')")
   or die("Query non valida: " . mysql_error());
echo"</table>";
echo nl2br("\n");
?>
</body>
</html>
```

La pagina fatt.php permette l'inserimento di nuove fatture nel database. Essa fa uso della procedure nuovafattura.

```
<html>
<head>
	<title>Inserimento Fatture</title>
	<link rel="stylesheet" type="text/css" href="stile.css">
</head>
<body><h1 class="logo"></h1></body>
<nav>
<ul>
<li><a href="prod.php">LISTINO</a></li>
<li><a href="quatt.php">INTERROGAZIONE DATABASE</a></li>
<li><a href="vend.php">INSERIMENTO VENDITORI</a></li>
<li><a href="fatt.php">INSERIMENTO FATTURE</a></li>
</ul>
</nav>
<body>
<?php
echo nl2br("<pre> ID		Data		Venditore		Cliente</pre>");
?>
<form method="post" action="">
	<input type="text" name="fid">
	<input type="text" name="fdata">
	<input type="text" name="fvenditore">
	<input type="text" name="fcliente">
	<input type="submit">
</form>

<?php
$id = $_POST['fid'];
$data = $_POST['fdata'];
$venditore = $_POST['fvenditore'];
$cliente = $_POST['fcliente'];
$link = mysql_connect('localhost', '----', '------');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
$db_selected = mysql_select_db('LiquoriVeneto', $link);
if (!$db_selected) {
   die ('Can\'t use LiquoriVeneto : ' . mysql_error());
}
$risultato = mysql_query("call nuovafattura('$id','$data','$venditore','$cliente')")
   or die("Query non valida: " . mysql_error());
echo"</table>";
echo nl2br("\n");
?>
</body>
</html>
```