

## Sommario

1. INTRODUZIONE .....	3
1.1 SPECIFICHE SUI DATI .....	3
1.2 IMPLEMENTAZIONE DI VARIE OPERAZIONI.....	3
2. PROGETTAZIONE CONCETTUALE.....	4
2.1 SCHEMA E-R .....	4
2.2 SCHEMA E-R FINALE.....	5
2.3 DIZIONARIO COMPATTO.....	8
3. PROGETTAZIONE LOGICA DEL MODELLO.....	9
3.1 FASE DI TRASFORMAZIONE.....	9
3.2 FASE DI TRADUZIONE.....	11
4. PROGETTAZIONE FISICA.....	13
4.1 DIMENSIONAMENTO FISICO .....	13
4.2 CREAZIONE DATABASE.....	15
4.3 POLITICHE DI SICUREZZA .....	15
4.4 CREAZIONE OGGETTI .....	16
4.4.1 CREAZIONE TABELLE.....	16
4.4.2 INSERIMENTO VALORI.....	22
4.4.3 QUERY .....	26
4.4.4 TRIGGER E STORED PROCEDURE.....	30
5. VISUALIZZAZIONI.....	34
5.1 QUERY .....	34
5.2 TRIGGER.....	37
5.3 STORED PROCEDURE .....	39
6. INDICI.....	41
7. SITO WEB / INTERFACCIA UTENTE .....	43

# 1.INTRODUZIONE

Si vuole progettare una base di dati della piattaforma in grado di supportare le funzionalità descritte (registrazione tragitti e pagamento pedaggi), più alcune utili a supportare analisi di tipo statistiche (es. tragitto più trafficato).

## 1.1 SPECIFICHE SUI DATI

Dai primi colloqui, sono emerse le seguenti specifiche:

- Riguardo i DISPOSITIVI sono conservate le informazioni relativi ad essi. Sono dotati da un codice univoco e associate al massimo a due automobili di un dato CLIENTE.
- Del CLIENTE si vogliono memorizzare i dati anagrafici e le informazioni di fatturazione (il conto corrente o la carta di credito col relativo codice).
- Per ogni DISPOSITIVO, occorre memorizzare il TRAGITTO compiuto ai fini della fatturazione al cliente, in particolare: numero del tragitto (per quel dato dispositivo), casello di ingresso, data e ora ingresso, casello di uscita, data e ora uscita.
- Il CASELLO è individuato da un codice formato dall'uscita autostradale e da un numero (es. Napoli EST 1).
- Infine, per ogni AUTOMOBILE si vuole memorizzare targa, modello, e il proprietario nel caso sia uno dei clienti.

## 1.2 IMPLEMENTAZIONE DI VARIE OPERAZIONI

In questo progetto verrà effettuata l'implementazione in SQL di stored procedure, query, viste e trigger che si ritengono necessari all'esecuzione delle operazioni richieste. Ecco un primo esempio:

- generazione della fattura a fine mese per ogni cliente.

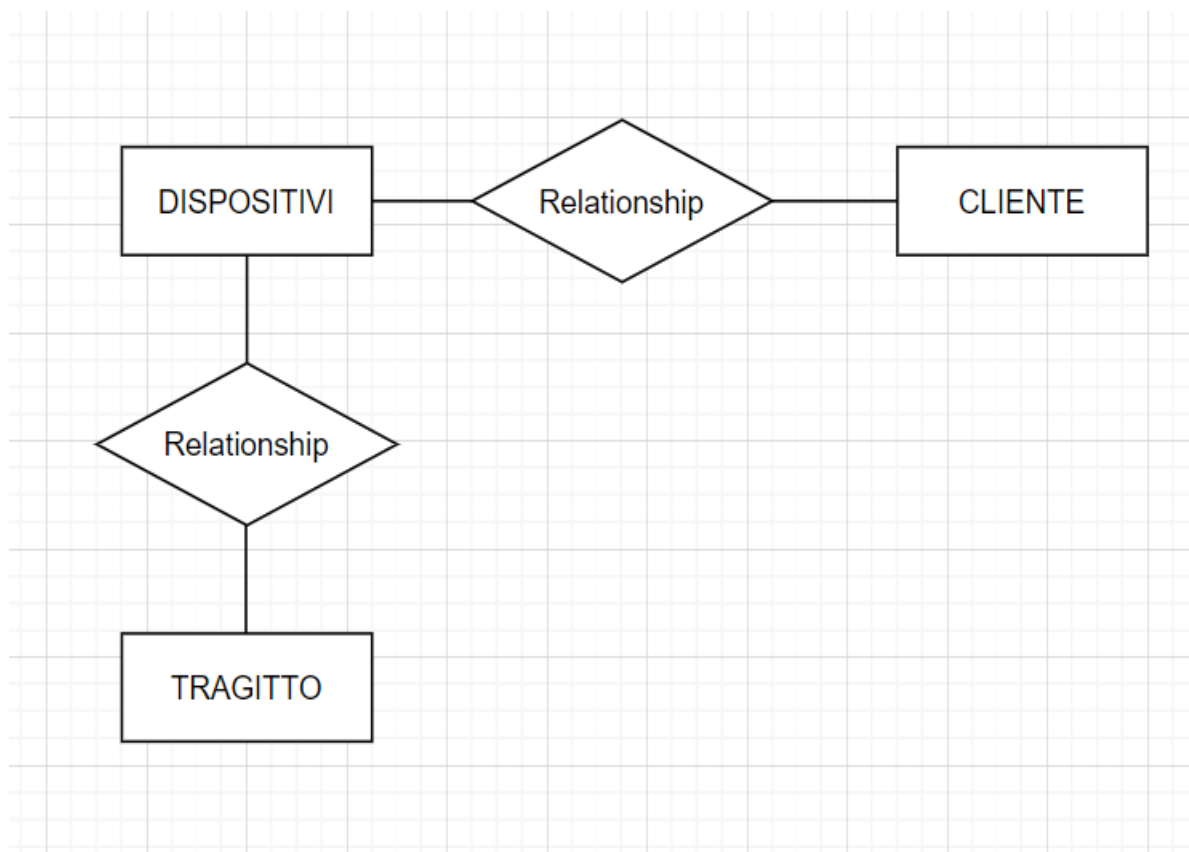
## 2.PROGETTAZIONE CONCETTUALE

### 2.1 SCHEMA E-R

Il primo passo per la progettazione della base di dati è la definizione dello schema E/R, individuando i concetti fondamentali (schema E/R portante) e, successivamente, andando ad estendere e raffinare tale schema aggiungendo le specifiche analizzate nel capitolo precedente al fine di giungere allo schema E/R finale, punto centrale della progettazione concettuale.

Principalmente, la prima cosa che si può notare istantaneamente, è quella di considerare fin dall'inizio tre entità principali, che sono: DISPOSITIVI, CLIENTE e TRAGITTO.

Prima di andare a dare dei nomi alle varie associazioni e ad andare ad individuare eventualmente nuove entità, vediamo un primo schema grafico della nostra piattaforma informatica.



Dopo uno studio più approfondito del progetto in questione, sono state individuate nuove entità e i vari collegamenti tra di loro.

Le **entità** fondamentali che si possono individuare, nel nostro caso, sono 5:

1. Dispositivi;
2. Automobili;
3. Clienti;
4. Caselli;
5. Tragitti.

## 2.2 SCHEMA E-R FINALE

Una volta individuate le entità all'interno del nostro progetto, si passa all'analisi delle varie associazioni tra essi.

Con il termine **associazione** si fa riferimento al legame concettuale fra due o più entità. Invece, con il termine **grado di associazione** si indica il numero di entità che partecipano all'associazione stessa. Da questo, introduciamo nuovi termini: l'associazione è detta BINARIA se fa riferimento a due entità, TERNARIA se fa riferimento a tre entità, e così via.

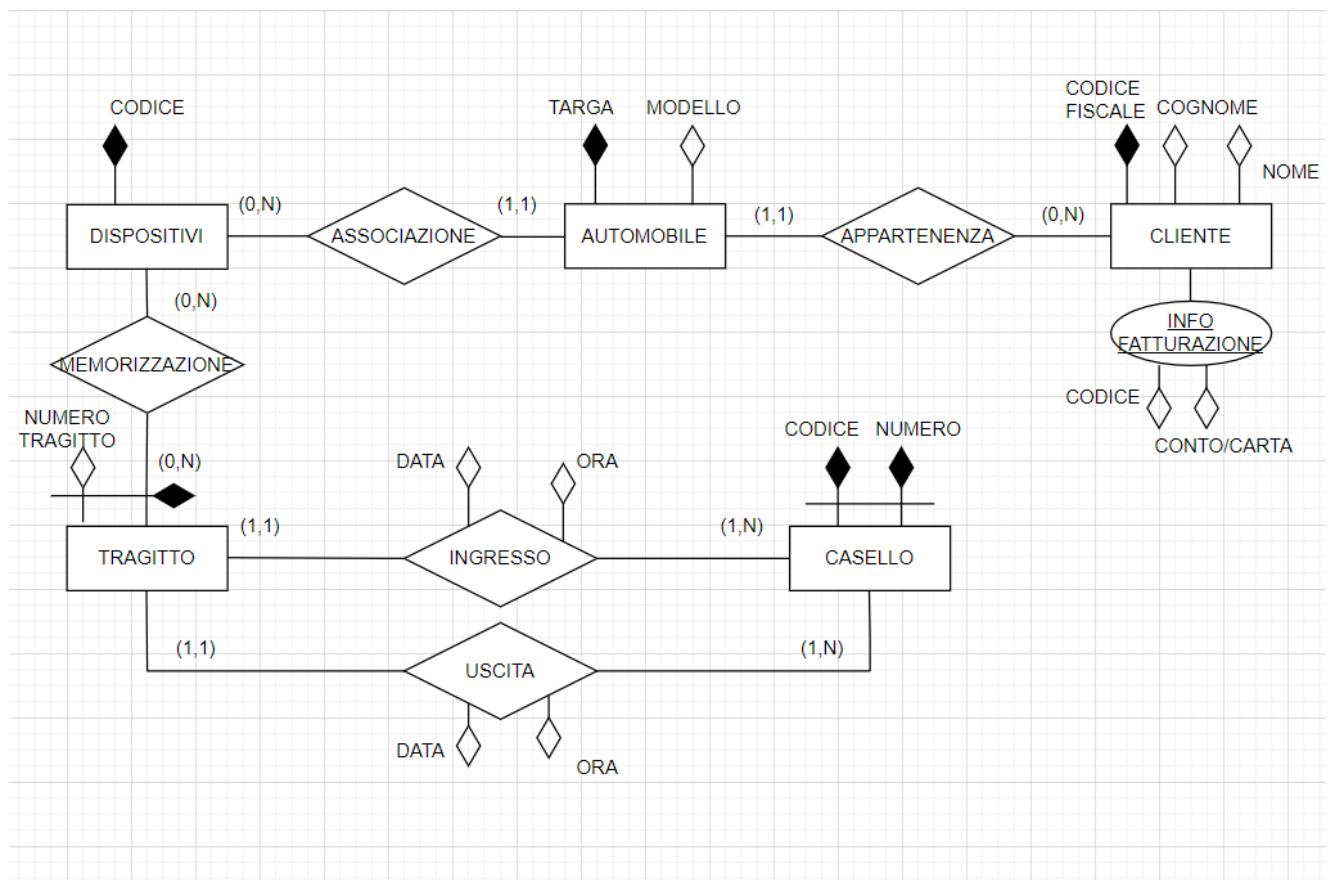
Vediamo adesso tutte le varie scelte riguardanti le associazioni e le cardinalità delle associazioni, ma prima di fare ciò, andiamo a spiegare cosa si intende con il termine cardinalità.

La **cardinalità** in una associazione è una coppia di valori che specifica il numero (rispettivamente massimo e minimo) delle occorrenze di associazioni.

Con riferimento a ciò, possiamo parlare di rapporto di cardinalità, introducendo così i seguenti rapporti:

- **Uno ad uno:** un'entità corrisponde a una ed una sola occorrenza dell'altra entità;
- **Uno a molti:** ad un'occorrenza di un'entità sono associate più occorrenze delle altre entità;
- **Molti a molti:** non solo dice che ad un'occorrenza di un'entità sono associate più occorrenze delle altre entità, ma vale anche il viceversa.

Per quanto riguarda il progetto, è stato optato un diagramma E-R di questo tipo:



## NOTA

*L'UTENTE è identificato dal suo Codice Fiscale, ma il Codice Fiscale può cambiare se l'utente cambia nome nel corso della sua vita. Sarebbe, quindi, ancora più preciso identificarlo con una matricola posta come identificatore esterno.*

## Andiamo adesso a spiegare le scelte fatte dal team sul modello E-R:

### INIZIAMO CON LE CARDINALITÀ:

In base all'associazione tra dispositivi e automobile, come si può notare, è stato scelto di optare come grado di cardinalità in riferimento ai dispositivi, il tipo (0, N) perché, come dice la traccia, il dispositivo può essere associato ad un

massimo di due automobili del cliente. Invece per quanto riguarda le automobili, esse possono essere associate al massimo ad uno e un solo dispositivo.

Nell'associazione tra automobili e cliente, invece, è stata inserita una cardinalità di tipo uno a molti, ovvero l'automobile appartiene ad uno e un solo cliente; invece, al cliente possono appartenere da 0 fino a N automobili.

Nella relazione tra dispositivi e tragitto, chiamata con il nome memorizzazione, è stata inserita una cardinalità del tipo molti a molti. Cioè, ai dispositivi possono essere associati più tragitti e i tragitti possono essere memorizzati da più dispositivi.

Invece, tra tragitto e casello, è stata introdotta una doppia relazione, una che fa riferimento all'ingresso, l'altra invece all'uscita, ambe con attributo di data ed ora. L'associazione, per entrambi, è di tipo uno a molti, perché al tragitto può essere associato uno e un solo ingresso al casello (stessa cosa per quanto riguarda l'uscita), ma in quello specifico casello, ci possono essere più ingressi (o uscite) da tragitti differenti.

#### VALUTIAMO ADESSO GLI ATTRIBUTI SCELTI:

In “dispositivi” troviamo un unico attributo, scelto come chiave primaria dell'entità, che è appunto il codice del dispositivo.

Nell'entità automobile, troviamo due attributi; la targa, scelta come chiave primaria, e il modello dell'automobile.

Nell'entità casello troviamo anche qui due attributi, scelti entrambi come chiave primaria (chiave allora composta da due attributi) che sono il codice e il numero del casello.

In “tragitto” troviamo una particolarità diversa dalle altre entità, cioè, la chiave primaria, il numero del tragitto, è posizionata in questo modo perché quel particolare numero fa riferimento al dispositivo memorizzato.

Per quanto riguarda l'entità cliente, la sua chiave primaria è il codice fiscale (viene spiegato nella nota che esiste un problema raro con questa chiave), poi troviamo nome e cognome e un attributo composto con il codice e il conto/carta del cliente.

Troviamo anche degli attributi sulle relazioni, in particolare, come già spiegato sopra, tra tragitto e casello, che sono appunto la data e l'ora.

## 2.3 DIZIONARIO COMPATTO

Vediamo adesso, tramite una tabella grafica, tutte le varie entità con i loro attributi e le varie associazioni:

ENTITA'	DESCRIZIONE	ATTRIBUTI	CHIAVE PRIMARIA
Cliente	Colui che effettua i pagamenti tramite conto oppure carta	Dati anagrafici Informazioni di fatturazione	Codice Fiscale
Dispositivi	Apparecchi associati al massimo a due automobili	Codice del dispositivo	Codice
Automobile	Veicolo del cliente con il quale viene effettuato il tragitto	Targa Modello	Targa
Tragitto	Entità da memorizzare che riguarda ogni dispositivo del cliente	Numero tragitto	Numero tragitto
Casello	Serve per identificare l'uscita autostradale dal numero dove ha percorso il cliente	Codice Numero	La coppia {codice, numero}

RELAZIONE	ATTRIBUTI	ENTITA' PARTECIPANTI
Appartenenza	-	Cliente Automobile
Associazione	-	Automobile Dispositivo
Memorizzazione	-	Dispositivo Tragitto
Ingresso	Data e ora	Tragitto Casello
Uscita	Data e ora	Tragitto Casello

### 3.PROGETTAZIONE LOGICA DEL MODELLO

Il passaggio dal modello concettuale al modello logico si chiama **progettazione logica**, che consiste nel costruire uno schema orientato al modello relazionale. Essa si identifica in due fasi:

la prima fase è detta di **trasformazione** (con l'obiettivo di semplificare alcuni costrutti), invece la seconda fase è detta di **traduzione** (che traduce lo schema in un insieme di relazioni e vincoli).

#### 3.1 FASE DI TRASFORMAZIONE

La fase di trasformazione consiste nella trasformazione di elementi non direttamente traducibili nel modello logico relazionale, come ad esempio gli attributi **multi-valore** oppure attributi **composti**.

Nel nostro caso, non ci sono generalizzazioni di entità e nemmeno attributi multi-valore, ma troviamo solo un attributo composto.

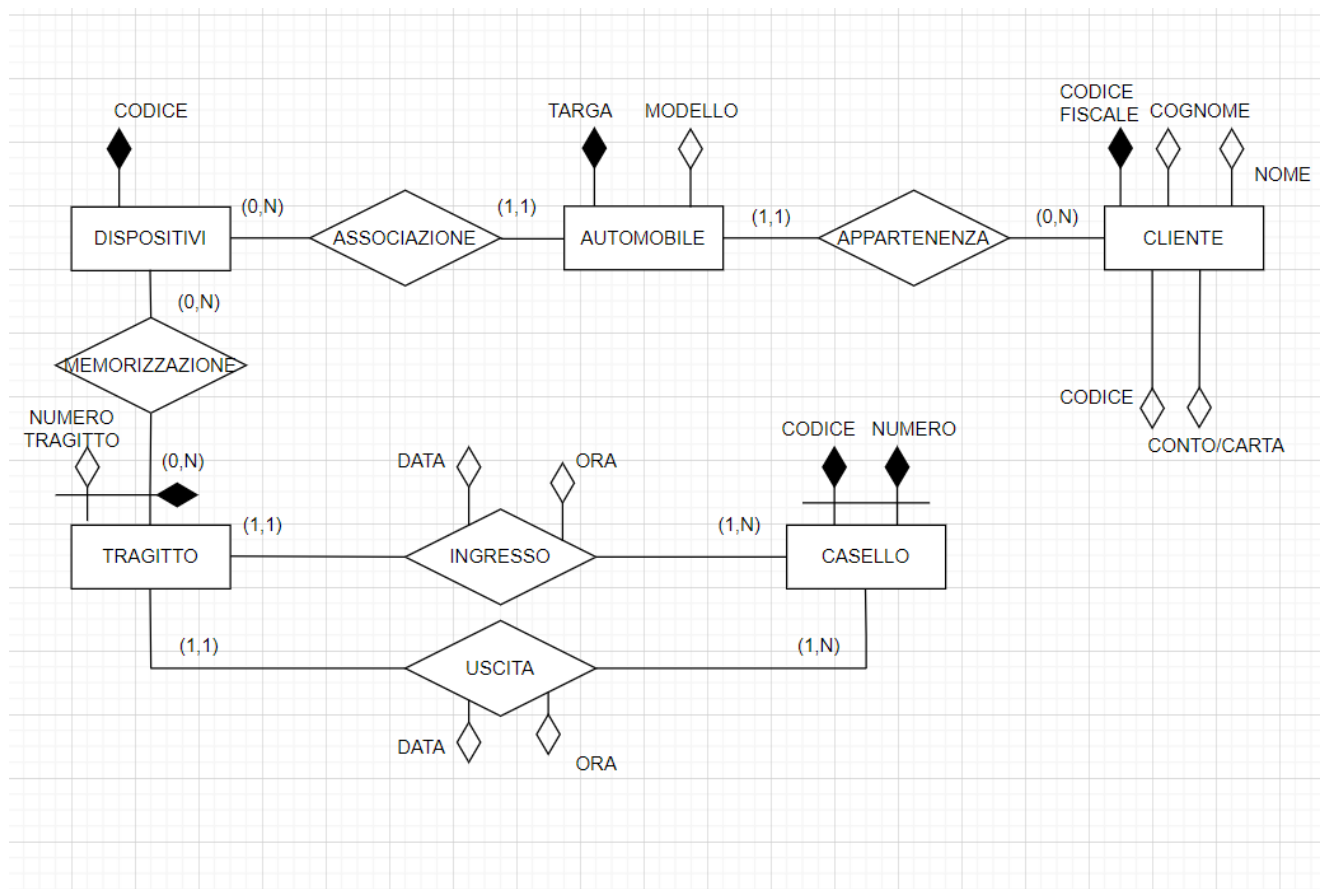


La trasformazione dell'attributo composto viene effettuata sostituendolo con attributi semplici, quanti sono gli attributi componenti.

Per quanto riguarda invece le generalizzazioni, solo a scopo puramente teorico, ci sono diverse scelte da applicare:

1. L'accorpamento del superclasse nelle sottoclassi;
2. Accorpamento delle sottoclassi nel superclasse;
3. Sostituzione della generalizzazione con associazioni (che saranno identificate esternamente dal superclasse);
4. Soluzione mista (delle tre precedenti).

Quindi, banalmente, il nostro modello diventerà:



Abbiamo, quindi, semplificato l'attributo composto in due attributi semplici.

## 3.2 FASE DI TRADUZIONE

Essa si occupa di costruire lo schema logico relazionale, cioè l'insieme di relazioni e vincoli che rappresentano gli stessi concetti che si trovano all'interno dello schema relazionale.

1. Per prima cosa, ogni entità si trasforma in una relazione avente come attributi gli attributi dell'entità e come chiave l'identificatore dell'entità.
2. Per quanto riguarda le traduzioni delle associazioni, esse si differenziano in tre casi diversi in base al grado:
  - L'associazione molti a molti si traduce in una relazione avente lo stesso nome. L'insieme di tali identificatori delle entità costituisce la chiave primaria della relazione, ed ognuno di essi ha un vincolo di integrità referenziale con il riferimento da cui discende.
  - Invece, in quella uno a molti ci sono 2 modalità: se l'associazione "uno" è del tipo (0,1) si consiglia di usare la stessa tipologia dei molti a molti, ma all'interno della relazione, andando a inserire come chiave soltanto quella dove il grado è "uno". Invece, se ci troviamo nel caso del tipo (1,1), si usa un altro metodo, dove non si va a creare la relazione, ma all'interno di quello con cardinalità uno, si va a inserire la chiave di quello con cardinalità molti che fa riferimento ad esso (non va messo come chiave).
  - Infine, in quella uno ad uno, ci sono due risoluzioni: sono molto simili a quelli nel caso uno a molti, ma in questo caso, l'attributo non chiave che fa riferimento (referenziale), avrà un vincolo di UNICITÀ.

Andiamo a vedere nel nostro caso come va a cambiare il modello E-R:

- Per prima cosa le relazioni INGRESSO, USCITA e APPARTENENZA scompaiono, facendo riferimento all'associazione uno a molti, e gli identificatori del lato molti verranno inseriti (opportunitamente ridenominati) al lato 1;
- Tra DISPOSITIVO e TRAGITTO troviamo una associazione del tipo molti a molti, questo vuol dire che verrà inserita nel modello la relazione MEMORIZZAZIONE andando a inserire in essa gli identificatori di entrambi i lati;
- Tra DISPOSITIVO e AUTOMOBILE si fa riferimento all'associazione uno a molti, e anche in questo caso, il lato uno è del tipo (1,1), quindi andremo a trovare la stessa tipologia rispetto al primo caso.

Le relazioni che andremo a realizzare nella base di dati saranno quindi:

- **CLIENTI:** (codice fiscale, cognome, nome, codice, conto/carta)
- **CASELLO:** (codice, numero)
- **AUTOMOBILE:** (targa, modello, cliente: CLIENTI, dispositivo: DISPOSITIVI)
- **DISPOSITIVI:** (codice)
- **MEMORIZZAZIONE:** (dispositivo: DISPOSITIVI, tragitti: TRAGITTO)
- **TRAGITTO:** (numerotragitto, dataI, oraI, dataU, oraU, casello: CASELLO, numero\_casello: CASELLO)

Fatto questo, è possibile andare a controllare se le relazioni individuate sono in 3NF.

Ma cosa vuol dire essere in 3NF?

Dette anche “forme normali”, esse sono state introdotte con l’intento di fornire un criterio di scelta tra i vari schemi relazionali che possano modellare una data realtà di interesse.

Introduciamo molto velocemente le 3 forme normali:

- 1NF: uno schema di relazione  $R(X)$  si dice in prima forma normale se il relativo dominio è atomico;
- 2NF: è in seconda forma normale se, è ovviamente in 1NF e se ogni attributo non primo è in dipendenza funzionale completa da ogni chiave (individuando le varie dipendenze funzionali);
- 3NF: è in terza forma normale se, è in 2NF e se ogni attributo non primo non dipende transitivamente da ogni chiave (individuando le varie dipendenze transitive)

Nel nostro caso possiamo concludere dicendo che siamo in 3NF perché **SONO SODDISFATTI** tutti i requisiti.

## NOTA

*Se non ci fossimo trovati in 2NF oppure in 3NF (o entrambe), è possibile applicare delle decomposizioni, tecnica per cui le informazioni incluse in una relazione vengono suddivise in due o più relazioni che hanno un numero di attributi inferiore a quelli contenuti in quella di partenza.*

## 4.PROGETTAZIONE FISICA

La progettazione fisica di una base dati prevederà la risoluzione delle seguenti fasi:

1. Per quanto riguarda la progettazione fisica del progetto, si è optato di andare a fornire informazioni sul “dimensionamento fisico”, cioè, spiegare quanto occupano (in byte) i tipi dati più diffusi visti a lezione.
2. Il passaggio successivo è quello di andare a creare il Database, con le relative creazioni degli oggetti della base dati. Questo verrà sviluppato tramite creazione di tabelle, creazione di vincoli di integrità referenziale, inserimento di dati all’interno delle varie tabelle create, e per finire, varie query (dove poi nel capitolo dopo verranno illustrati i vari risultati delle singole query).

### 4.1 DIMENSIONAMENTO FISICO

A partire da ogni tabella, è possibile andare ad effettuare una stima dei costi per quanto riguarda l’occupazione di memoria della base di dati. Ad esempio, sappiamo che l’occupazione dei tipi più diffusi è:

- CHAR(x): x byte
- VARCHAR(x): da 0 a x byte
- DATE: 7 byte
- NUMBER(x):  $(x/2) + 2$  byte

Ad esempio, nel nostro caso, il tipo NOME, essendo un CHAR (20), occupa 20 byte.

### NOTA:

*Un’alternativa a **VARCHAR** avrebbe potuto essere **VARCHAR2**: la differenza sta nel fatto che **VARCHAR** supporta la dimensione massima di 8000 B, mentre **VARCHAR2** può arrivare fino a 32000 B. Tuttavia, risulta inutile utilizzare stringhe così grandi nel caso in analisi, dunque si sceglie l’uso di **VARCHAR** per una maggiore velocità di esecuzione.*

Andiamo a visionare delle stime di costi per le tabelle in esame:

## CLIENTI

Nome attributo	Tipo	Dimensione
Codice fiscale	char (16)	16B
Nome	Varchar (20)	20B
Cognome	Varchar (20)	20B
Codice	Int (16)	4B
Conto/carta	Varchar (20)	20B
<b>totale</b>		<b>70B</b>

## CASELLO

Nome attributo	Tipo	Dimensione
Codice	Varchar (20)	20B
Numero	Int (4)	4B
<b>totale</b>		<b>24B</b>

## AUTOMOBILE

Nome attributo	Tipo	Dimensione
Targa	Char (7)	7B
modello	Varchar (20)	20B
<b>totale</b>		<b>27B</b>

## DISPOSITIVI

Nome attributo	Tipo	Dimensione
Codice	Varchar (16)	16B
<b>totale</b>		<b>16B</b>

## TRAGITTO

Nome attributo	Tipo	Dimensione
Numero tragitto	Varchar (20)	20B
dataI	date	7B
oraI	Char (8)	8B
dataU	date	7B
oraU	Char (8)	8B
<b>totale</b>		<b>50B</b>

## TOTALE

<b>Occupazione</b>
<b>187B</b>

## 4.2 CREAZIONE DATABASE

La creazione del database rappresenta per una base di dati un'operazione preliminare alla creazione degli oggetti in essa contenuti.

Per Oracle Database si usa un **modello di database relazione** che consente di archiviare e riprodurre dati utente e dell'azienda sotto forma di **record di dati organizzati**.

Le quantità di dati vengono strutturate in colonne, tabelle e righe e i punti di dati vengono messi in relazione con il supporto degli attributi.

## 4.3 POLITICHE DI SICUREZZA

Come abbiamo studiato, i moderni DBMS mettono a disposizione meccanismi di sicurezza per proteggere i dati da accessi non autorizzati.

Si è optato per la creazione di un DBA (Data Base Administrator), proprietario della base dati, e un ruolo esterno che può effettuare diverse funzioni.

Per creare un utente della base dati, si utilizza la seguente sintassi:

```
CREATE USER username IDENTIFIED BY password;
```

così facendo, nel nostro caso, l'utente potrà essere identificato dalla sua username e autenticato dalla sua password.

```
CREATE USER azienda_autostradale  
IDENTIFIED BY napoli_1926;  
  
-- Admin con tutti i privilegi  
GRANT DBA TO azienda_autostradale;
```

Adesso, infine, è riportato lo script relativo alla generazione del ruolo con la definizione di alcuni privilegi:

```
CREATE ROLE cliente;  
GRANT CONNECT TO cliente;  
  
GRANT INSERT ON azienda_autostradale.AUTOMOBILE TO utente;  
GRANT SELECT ON azienda_autostradale.DISPOSITIVI TO utente;  
GRANT SELECT ON azienda_autostradale.TRAGITTO TO utente;
```

## 4.4 CREAZIONE OGGETTI

Passiamo adesso alla creazione delle varie tabelle, inserimento dei valori, realizzazione di query e così via.

### 4.4.1 CREAZIONE TABELLE

Per la creazione delle tabelle useremo la piattaforma vista a lezione, cioè ORACLE LIVE SQL.

Iniziamo visionando per esteso le tabelle a cui dobbiamo far riferimento:

```
--CLIENTI(codice_fiscale*, cognome, nome, codice, tipo)  
--CASELLO(codice*, numero*)  
--AUTOMOBILE(targa*, modello, cliente**, dispositivo**)  
--DISPOSITIVI(codice*)  
--MEMORIZZAZIONE(dispositivo* **, tragitti* **)  
--TRAGITTO(numero_tragitto*, datai, orai, datau, orau)
```

(tragitto, oltre ai suoi campi, possiede i due riferimenti alla tabella casello)

## NOTA

*Il simbolo \* sta ad indicare che quel particolare attributo è una chiave primaria della tabella in questione.*

*Invece il simbolo \*\* sta ad indicare che quel particolare attributo è una chiave secondaria/esterna che fa riferimento ad un'altra tabella.*

Il passo successivo è quello di andare a creare man mano le singole tabelle.

### Tabella clienti:

```
CREATE TABLE CLIENTI(  
    codice_fiscale CHAR(16),  
    cognome VARCHAR(20),  
    nome VARCHAR(20),  
    codice INTEGER,  
    tipo VARCHAR(20),  
    CONSTRAINT PK_CLIENTI PRIMARY KEY (codice_fiscale)  
);
```

Questo è una prima creazione, più precisamente, la creazione della tabella clienti.

Notiamo che grazie all'ultima riga, vado a definire codice fiscale come chiave primaria della tabella.

Andiamo adesso a vedere come si fa a visualizzarla, e quando viene eseguito, qual è il risultato:

```
DESC CLIENTI;
```



E ora vediamo l'output della richiesta:

TABLE CLIENTI		
Column	Null?	Type
CODICE_FISCALE	NOT NULL	CHAR(16)
COGNOME	-	VARCHAR2(20)
NOME	-	VARCHAR2(20)
CODICE	-	NUMBER
TIPO	-	VARCHAR2(20)

Come possiamo notare, codice fiscale ha il campo null sbarrato come NOT NULL, questo perché, per definizione, la chiave primaria è NOT NULL.

Andiamo avanti così con tutte le altre tabelle.

### Tabella casello:

```
CREATE TABLE CASELLI(  
    codice VARCHAR(20),  
    numero INTEGER,  
    CONSTRAINT PK_CASELLO PRIMARY KEY (codice,numero)  
);
```

```
DESC CASELLI;
```

TABLE CASELLO		
Column	Null?	Type
CODICE	NOT NULL	VARCHAR2(20)
NUMERO	NOT NULL	NUMBER

### Tabella dispositivi:

```
CREATE TABLE DISPOSITIVI(  
    codice VARCHAR(16),  
    CONSTRAINT PK_DISPOSITIVI PRIMARY KEY (codice)  
);
```

```
DESC DISPOSITIVI;
```

TABLE DISPOSITIVI

Column	Null?	Type
CODICE	NOT NULL	VARCHAR2(16)

### Tabella automobile:

```
CREATE TABLE AUTOMOBILE(  
    targa CHAR(7),  
    modello VARCHAR(20),  
    cliente CHAR(16),  
    dispositivo VARCHAR(16),  
    CONSTRAINT PK_AUTOMOBILE PRIMARY KEY (targa)  
);
```

```
DESC AUTOMOBILE;
```

TABLE AUTOMOBILE

Column	Null?	Type
TARGA	NOT NULL	CHAR(7)
MODELLO	-	VARCHAR2(20)
CLIENTE	-	CHAR(16)
DISPOSITIVO	-	VARCHAR2(16)

### Tabella memorizzazione:

```
CREATE TABLE MEMORIZZAZIONE(  
    dispositivo VARCHAR(16),  
    tragitti VARCHAR(20),  
    CONSTRAINT PK_MEMORIZZAZIONE PRIMARY KEY (dispositivo)  
);
```

```
DESC MEMORIZZAZIONE;
```

TABLE MEMORIZZAZIONE

Column	Null?	Type
DISPOSITIVO	NOT NULL	VARCHAR2(16)
TRAGITTI	-	VARCHAR2(20)

### Tabella tragitto:

```
CREATE TABLE TRAGITTO(  
    numero_tragitto VARCHAR(20),  
    datai date,  
    orai CHAR(8),  
    datau date,  
    orau CHAR(8),  
    casello VARCHAR(20),  
    numero_casello INTEGER,  
    CONSTRAINT PK_TRAGITTO PRIMARY KEY (numero_tragitto)  
);
```

```
DESC TRAGITTO;
```

TABLE TRAGITTO

Column	Null?	Type
NUMERO_TRAGITTO	NOT NULL	VARCHAR2(20)
DATAI	-	DATE
ORAI	-	CHAR(8)
DATAU	-	DATE
ORAU	-	CHAR(8)
CASELLO	-	VARCHAR2(20)
NUMERO_CASELLO	-	NUMBER

Il passo successivo è quello di andare ad inserire i vari vincoli di integrità referenziale. Conviene fare questo passaggio a parte, per avere anche una visione del progetto più pulita e ordinata.

L'altro motivo, quello principale, è che affinché gli statement di creazione non generino errori è necessario, in presenza di chiavi esterne, creare prima le tabelle referenziate e poi quelle referenzianti. SQL fornisce l'utilizzo del comando DDL ALTER TABLE per modificare le definizioni di relazioni.

```
ALTER TABLE AUTOMOBILE
```

```
ADD CONSTRAINT FK_AUT_CLIENTI FOREIGN KEY (cliente) REFERENCES CLIENTI(codice_fiscale)
ON DELETE CASCADE;
```

--perchè se elimino un cliente voglio eliminare le sue auto

Questo è un primo caso dove l'attributo cliente nella tabella automobile andrà a referenziare, tramite questa modifica sulla tabella, la tabella clienti associando la chiave codice\_fiscale.

```
ALTER TABLE AUTOMOBILE
ADD CONSTRAINT FK_AUT_DISPOSITIVI FOREIGN KEY (dispositivo) REFERENCES DISPOSITIVI(codice)
ON DELETE CASCADE;
--stesso ragionamento
```

Tabella memorizzazione:

```
ALTER TABLE MEMORIZZAZIONE
ADD CONSTRAINT FK_MEM_DISPOSITIVI FOREIGN KEY (dispositivo) REFERENCES DISPOSITIVI(codice)
ON DELETE CASCADE;
ALTER TABLE MEMORIZZAZIONE
ADD CONSTRAINT FK_MEM_TRAGITTO FOREIGN KEY (tragitti) REFERENCES TRAGITTO(numero_tragitto)
ON DELETE CASCADE;
```

Tabella tragitto:

```
ALTER TABLE TRAGITTO
ADD CONSTRAINT FK_TRA_CASELLI FOREIGN KEY (numero_casello,casello) REFERENCES CASELLI(numero,codice)
ON DELETE CASCADE;
```

#### 4.4.2 INSERIMENTO VALORI

Una volta create tutte le tabelle, ovviamente tocca popolarle.

L'operazione di inserimento permette di inserire una nuova tupla all'interno della relazione elencandole i valori.

Ovviamente, essa, può portare dei problemi, come ad esempio delle violazioni dei vincoli. Esse sono:

- Vincolo di dominio: violato se si inseriscono valori non appartenenti al dominio;
- Vincolo di chiave primaria: se il valore inserito è NULL;
- Vincolo di integrità referenziale: riferimento ad una chiave che non esiste nella relazione riferita.

La sintassi SQL per inserire è:

- INSERT INTO nome tabella VALUES (elenco valori)

Se non vengono assegnati tutti i valori, si darà un valore di default che è pari a NULL.

Ovviamente, oltre all'inserimento, esistono altre due modalità, che sono la CANCELLAZIONE e la MODIFICA. Anch'esse, come l'INSERIMENTO, possono portare delle problematiche.

Andiamo adesso ad inserire dei valori a piacimento all'interno delle tabelle:

### Tabella clienti:

```
INSERT INTO CLIENTI VALUES ('BCCCRL54L07F839K', 'rui', 'mario', 3333, 'conto corrente');
```

In questo primo esempio, come si può vedere, sono stati inseriti dei valori alla tabella clienti, ora con un altro comando andremo a visualizzare il risultato:

```
SELECT * FROM CLIENTI;
```

E il suo risultato sarà:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente

Proseguiamo adesso, inserendo altre righe di valori, per poi alla fine visualizzare il risultato di tutte le tabelle.

```
INSERT INTO CLIENTI VALUES ('MSSLNL87H24Z600D', 'di lorenzo', 'giovanni', 5421, 'conto corrente');  
INSERT INTO CLIENTI VALUES ('RNL CST85B05Z128L', 'meret', 'aex', 1267, 'carta di credito');  
INSERT INTO CLIENTI VALUES ('BLLMNC64P70C745L', 'politano', 'matteo', 1298, 'conto corrente');  
INSERT INTO CLIENTI VALUES ('RMNGPP36H07F209M', 'lobotka', 'stanislav', 5555, 'carta di credito');
```

### Tabella casello:

```
INSERT INTO CASELLO VALUES ('12563', 'NAPOLI EST 1');  
INSERT INTO CASELLO VALUES ('12332', 'MILANO OVEST 3');  
INSERT INTO CASELLO VALUES ('67987', 'FIRENZE SUD 1');
```

### Tabella dispositivo:

```
INSERT INTO DISPOSITIVI VALUES ('764512');  
INSERT INTO DISPOSITIVI VALUES ('986310');  
INSERT INTO DISPOSITIVI VALUES ('093412');  
INSERT INTO DISPOSITIVI VALUES ('888634');
```

### Tabella automobile:

```
INSERT INTO AUTOMOBILE VALUES ('DR743SS', 'mercedes', 'MSSLNL87H24Z600D', '888634' );  
INSERT INTO AUTOMOBILE VALUES ('SS753LO', 'bravo', 'RMNGPP36H07F209M', '986310' );
```

### Tabella tragitti:

```
INSERT INTO TRAGITTO VALUES ('VR555', '02-October-23', '12-30-12', '02-October-23', '13-30-00', '67987', 'FIRENZE SUD 1' );  
INSERT INTO TRAGITTO VALUES ('SS765', '03-March-23', '17-10-00', '03-March-23', '17-30-12', '12563', 'NAPOLI EST 1');  
INSERT INTO TRAGITTO VALUES ('LT821', '02-February-11', '06-10-32', '02-February-11', '12-30-00', '12332', 'MILANO OVEST 3');
```

### Tabella memorizzazione:

```
INSERT INTO MEMORIZZAZIONE VALUES ('764512', 'LT821');  
INSERT INTO MEMORIZZAZIONE VALUES ('986310', 'VR555');
```

### NOTA

*Le eventuali aggiunte di valori all'interno delle varie tabelle non saranno tutte visibili all'interno del progetto, ma verranno aggiunte soltanto per favorire le varie query ed operazioni da fare all'interno del progetto.*

Visioniamo adesso i vari risultati di tutti gli inserimenti effettuati:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente
RNLCST85B05Z128L	meret	aex	1267	carta di credito
BLLMNC64P70C745L	politano	matteo	1298	conto corrente
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito

CODICE	NUMERO
12332	MILANO OVEST 3
12563	NAPOLI EST 1
67987	FIRENZE SUD 1

TARGA	MODELLO	CLIENTE	DISPOSITIVO
DR743SS	mercedes	MSSLNL87H24Z600D	888634
SS753LO	bravo	RMNGPP36H07F209M	986310



CODICE
093412
764512
888634
986310

NUMERO_TRAGITTO	DATAI	ORAI	DATAU	ORAU	CASELLO	NUMERO_CASELLO
VR555	02-OCT-23	12-30-12	02-OCT-23	13-30-00	67987	FIRENZE SUD 1
SS765	03-MAR-23	17-10-00	03-MAR-23	17-30-12	12563	NAPOLI EST 1
LT821	02-FEB-11	06-10-32	02-FEB-11	12-30-00	12332	MILANO OVEST 3

DISPOSITIVO	TRAGITTI
764512	LT821
986310	VR555

#### 4.4.3 QUERY

Prima di andare a parlare di query, si vuole dare una breve definizione su alcuni comandi già usati.

Introduciamo il termine di **OPERAZIONI RELAZIONALI** che sono proprio le operazioni specifiche del modello relazionale.

La prima prende il nome di **PROIEZIONE** ed è il comando che agisce su una decomposizione verticale:

**SELECT [DISTINCT] elenco FROM tabella**

- Ad esempio:

```
select nome,cognome from CLIENTI;
```

Dove il risultato sarà:

NOME	COGNOME
mario	rui
giovanni	di lorenzo
aex	meret
matteo	politano
stanislav	lobotka

L'altra operazione è la **SELEZIONE**, che permette di scegliere tra le tuple, quelle che soddisfano una condizione (in parte già vista):

- Ad esempio:

```
select * from CLIENTI where nome = 'mario';
```

Che restituirà:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente

Ora, possiamo passare alla vera e propria analisi delle query.

Una classica query si ottiene combinando una proiezione con una selezione, ovvero:

**SELECT & FROM % WHERE \$**

dove & è anche detto ELENCO ATTRIBUTI TARGET. Se si usa carattere \*, come già visto, allora si farà su tutti gli attributi della relazione.

## NOTA

*I vari risultati delle query verranno poi visualizzati nell'apposito capitolo, per la precisione, il CAPITOLO 5.*

**Iniziamo** con delle query semplici:

```
--query 1
--si vuole visualizzare il numero del tragitto e il casello dalla tabella
--tragitto dove il numero del casello inizia con la lettera F
select numero_tragitto, casello from TRAGITTO where numero_casello LIKE 'F%';
```

```
--query 2
--si vuole visualizzare quanti clienti hanno come tipo
--impostato il conto corrente
select COUNT(*) AS NUMERO
from CLIENTI
where tipo = 'conto corrente';
```

## NOTA

*COUNT è un particolare tipo che fa riferimento alle funzioni di aggregazione e le varie clausole di raggruppamento, in particolare troviamo anche:*

*SUM: somma;*

*MAX: valore massimo;*

*MIN: valore minimo;*

*AVG: media dei valori.*

*GROUP BY: raggruppa le tuple in sottoinsiemi caratterizzati dallo stesso valore;*

*HAVING: specifica la condizione logica che devono rispettare i sottoinsiemi precedentemente creati dalla group by.*

```
--query 3
--si vuole visualizzare una stampa ordinata in senso crescente
--sulla tabella CASELLO rispetto al numero
select codice, numero
from CASELLO
ORDER BY numero;
```

```

--query 4
--si vuole visualizzare, tramite l'utilizzo di una join, l'insieme
--delle informazioni presenti in due tabelle differenti
select *
from CLIENTI C INNER JOIN AUTOMOBILE A ON C.codice_fiscale = A.cliente;

--query 5
--si vuole visualizzare, tramite l'utilizzo di una join,
--il codice e la targa dove il dispositivo è pari ad un
--certo valore
select D.codice, A.targa
from DISPOSITIVI D INNER JOIN AUTOMOBILE A ON D.codice = A.dispositivo
where A.dispositivo = '986310';

--query 6
--visualizziamo adesso una doppia join
select D.codice, M.tragitti, T.casello, T.numero_casello
from (dispositivi D INNER JOIN memorizzazione M ON D.codice=M.dispositivo)
INNER JOIN tragitto T ON T.numero_tragitto=M.tragitti;

--query 7
--vediamo adesso che è possibile usare altri due tipi di JOIN, per
--l'esattezza la LEFT JOIN e la RIGHT JOIN
--(i campi che verranno considerati che non hanno associazione diventeranno NULL '-')
select *
FROM CLIENTI C LEFT JOIN AUTOMOBILE A ON C.codice_fiscale = A.cliente
WHERE C.tipo='conto corrente';

--query 8
--stampare una tabella dove la data di ingresso al casello è dal primo aprile in poi
select T.numero_tragitto, T.datai, T.orai, T.datau, T.orau, T.casello, T.numero_casello, D.codice
FROM (TRAGITTO T INNER JOIN MEMORIZZAZIONE M ON T.numero_tragitto = M.tragitti)
INNER JOIN DISPOSITIVI D ON M.dispositivo=D.codice
where T.datai > '01-april-23';

```

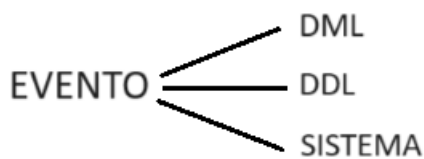
#### 4.4.4 TRIGGER E STORED PROCEDURE

Il ruolo principale dell'utilizzo dei trigger è la capacità di saper reagire in maniera automatica ad eventuali eventi.

Essi si basano su tre concetti importanti:

1. Evento: ha il compito di accedere al trigger;
2. Condizione: deve essere verificata affinché il trigger parta;
3. Azione: viene eseguita dal trigger.

Poiché il paradigma si basa su questi concetti, viene chiamato E-C-A.



Dove l'istruzione DML comprende: INSERT, UPDATE e DELETE su tabelle.

L'istruzione DDL comprende: CREATE, ALTER e DROP.

Gli eventi di SISTEMA sulla base dati come, ad esempio, SERVER ERROR e LOGON.

Lo standard SQL offre la possibilità di definire numerose tipologie sui trigger, tra cui:

- Granularità: indicazione sul numero di eventi dove avviene l'evento che può essere o del tipo row\_level oppure statement\_level;
- Modalità di esecuzione: specifica in che momento deve essere eseguito, tipicamente si tratta di un'azione IMMEDIATA che può essere before o after.

I trigger per funzionare hanno bisogno di una transition table, che è semplicemente l'insieme delle righe della tabella detta target. Una transition variable può essere del tipo NEW o OLD: esse non possono essere applicate a tutte le istruzioni DML. Per l'esattezza:

- NEW → INSERT;
- OLD → DELETE;
- Entrambi → UPDATE.

Andiamo adesso a vedere i trigger e le stored procedure realizzate: iniziamo con la classica sintassi, per capire lo scheletro da adottare.

```
CREATE OR REPLACE TRIGGER nometrigger
{BEFORE|AFTER} evento_dml_attivante
ON tabella_target
[WHEN condizione] -- -> se non c'è, è sempre vera
[FOR EACH {STATEMENT|ROW}] -- -> se non c'è, di default è STATEMENT
corpo_trigger
```

Dove notiamo la modalità di attivazione, che indica se l'attivazione avviene prima o dopo. Invece la clausola FOR EACH impone o che sia attivato ad ogni riga, oppure che sia attivato una sola volta.

Iniziamo adesso con vari esempi di trigger (anche in questo caso i risultati verranno visionati nell'apposito blocco del capitolo 5).

- Trigger 1

```
--nome e cognome vengono inseriti in maiuscolo
CREATE OR REPLACE TRIGGER cliente_maiuscolo
BEFORE INSERT ON CLIENTI
FOR EACH ROW
BEGIN
    :new.nome := UPPER(:new.nome);
    :new.cognome := UPPER(:new.cognome);
END;
```

- Trigger 2

Questo ha un nuovo livello di difficoltà, perché va creata un'altra tabella per poi visionare l'elemento eliminato (vedremo tutto nel capitolo 5)

```
-- quando cancello uno o più clienti, li salvo in una tabella di backup
CREATE OR REPLACE TRIGGER salva_cliente
AFTER DELETE
ON CLIENTI
FOR EACH ROW
BEGIN
    INSERT INTO CLIENTI_BIS
    VALUES(:OLD.codice_fiscale, :OLD.nome);
END;
```

- Trigger 3

```
1  -- se la data di ingresso avviene dopo il 2023, segnala un errore
2  CREATE OR REPLACE TRIGGER controllo_data
3  BEFORE INSERT
4  ON TRAGITTO
5  FOR EACH ROW
6  DECLARE
7      datai char(8);
8      errore EXCEPTION;
9  BEGIN
10     if datai > '31-December-23'
11     then raise errore;
12     END IF;
13 EXCEPTION
14 when errore then raise_application_error(-20003, 'data non corretta!');
15 END;
```

Trigger created.

- Trigger 4

```
-- se il nome del cliente è VICTOR, inserirlo anche nella tabella CLIENTI_BIS
CREATE OR REPLACE TRIGGER controllo_nome
BEFORE INSERT
ON CLIENTI
FOR EACH ROW
BEGIN
    if :new.nome = 'victor'
    THEN INSERT INTO CLIENTI_BIS
    VALUES(:new.codice_fiscale, :new.nome);
    END IF;
END;
```

- Stored Procedure 1

```
-- Realizzare una stored procedure che permette di inserire le informazioni sul CASELLO

CREATE OR REPLACE PROCEDURE INSERISCI_CASELLO(
    In_Codice IN CASELLO.codice%TYPE,
    In_Numero IN CASELLO.numero%TYPE
)
AS
BEGIN
    BEGIN
        INSERT INTO CASELLO VALUES (In_Codice, In_Numero);
        COMMIT;
    END;
END INSERISCI_CASELLO;
```



- Stored Procedure 2

In questo esempio, andremo ad inserire valori casuali sui codici, dove troveremo anche una stampa a video.

```
-- Realizzare una stored procedure che permette di inserire informazione sui clienti
-- con inserimento multiplo su più tabelle
-- N.B. il codice del cliente sarà generato in maniera randomica
-- Stampa a video del codice del cliente

CREATE OR REPLACE PROCEDURE INSERISCI_CLIENTE(
    In_codicefiscale IN CLIENTI.codice_fiscale%TYPE,
    In_cognome IN CLIENTI.cognome%TYPE,
    In_nome IN CLIENTI.nome%TYPE,
    In_tipo IN CLIENTI.tipo%TYPE
)
AS
BEGIN
    DECLARE
        cod_rand CLIENTI.codice%TYPE;
    BEGIN
        SELECT dbms_random.value(0000,9999) num INTO cod_rand FROM dual;
        DBMS_OUTPUT.PUT_LINE('Codice: '||cod_rand); --STAMPA A VIDEO
        INSERT INTO CLIENTI VALUES (In_codicefiscale, In_cognome, In_nome, cod_rand, In_tipo);
        COMMIT;
    END;
END INSERISCI_CLIENTE;
```

## 5. VISUALIZZAZIONI

### 5.1 QUERY

In questo capitolo andremo a visualizzare, per una questione di pulizia, i risultati delle query già create.

- Query 1:

NUMERO_TRAGITTO	CASELLO
VR555	67987

- Query 2:

NUMERO
3

- Query 3:

CODICE	NUMERO
99762	FIRENZE SUD 1
67987	FIRENZE SUD 1
12332	MILANO OVEST 3
12563	NAPOLI EST 1

- Query 4:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO	TARGA	MODELLO	CLIENTE	DISPOSITIVO
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente	DR743SS	mercedes	MSSLNL87H24Z600D	888634
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito	SS753LO	bravo	RMNGPP36H07F209M	986310

- Query 5:

CODICE	TARGA
986310	SS753LO

- Query 6:

CODICE	TRAGITTI	CASELLO	NUMERO_CASELLO
764512	LT821	12332	MILANO OVEST 3
986310	VR555	67987	FIRENZE SUD 1

- Query 7:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO	TARGA	MODELLO	CLIENTE	DISPOSITIVO
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente	DR743SS	mercedes	MSSLNL87H24Z600D	888634
BLLMNC64P70C745L	politano	matteo	1298	conto corrente	-	-	-	-
BCCCR54L07F839K	rui	mario	3333	conto corrente	-	-	-	-

- Query 8:

NUMERO_TRAGITTO	DATAI	ORAI	DATAU	ORAU	CASELLO	NUMERO_CASELLO	CODICE
VR555	02-OCT-23	12-30-12	02-OCT-23	13-30-00	67987	FIRENZE SUD 1	986310

## 5.2 TRIGGER

- Trigger 1

```
INSERT INTO CLIENTI VALUES ('BDDRRRL54L07F839K', 'raspadori', 'giacomo', 4376, 'conto corrente');  
SELECT * FROM clienti;
```

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente
RNLCST85B05Z128L	meret	aex	1267	carta di credito
BLLMNC64P70C745L	politano	matteo	1298	conto corrente
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito
BDDRRRL54L07F839K	RASPADORI	GIACOMO	4376	conto corrente

- Trigger 2

Creazione nuova tabella:

```
CREATE TABLE CLIENTI_BIS(  
    codice_bis CHAR(16),  
    nome_bis VARCHAR(20),  
    CONSTRAINT PK_CLIENTI_BIS PRIMARY KEY (codice_bis)  
);
```

Elimino un elemento, e visioniamo entrambe le tabelle:

```
DELETE FROM CLIENTI WHERE nome = 'GIACOMO';  
SELECT * FROM clienti_bis;  
SELECT * FROM clienti;
```

CODICE_BIS	NOME_BIS
BDDRRL54L07F839K	GIACOMO

Download CSV

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente
RNLCST85B05Z128L	meret	aex	1267	carta di credito
BLLMNC64P70C745L	politano	matteo	1298	conto corrente
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito

Download CSV

Come vediamo, dalla tabella principale viene eliminato, e viene inserito nella tabella bis.

- Trigger 3

```

16
17 INSERT INTO TRAGITTO VALUES ('VT786', '05-September-24', '10-20-12', '05-September-24', '13-30-00', '67987', 'FIRENZE SUD 1' );
18
19 SELECT * FROM tragitto;
20

```

ORA-04098: trigger 'SQL\_MZVPNHSBEOZTFLQAEIXNLCUDD.CONTROLLO\_ORARIO' is invalid and failed re-validation

Infatti, provando ad inserire una tupla, con la data di ingresso nell'anno 2024, si va contro alla dichiarazione di trigger, quindi non verrà inserita (come si può notare con la scritta in rosso in basso).

- Trigger 4

```
INSERT INTO CLIENTI VALUES ('HTYLOL54L07F845Y', 'osimhen', 'victor', 5312, 'conto corrente');
```

```
SELECT * FROM CLIENTI_BIS;
```

```
SELECT * FROM CLIENTI;
```

Adesso visioniamo entrambe le tabelle per vedere l'inserimento:

CODICE_BIS	NOME_BIS
HTYLOL54L07F845Y	victor

Download CSV

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCRL54L07F839K	rui	mario	3333	conto corrente
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente
RNLCST85B05Z128L	meret	aex	1267	carta di credito
BLLMNC64P70C745L	politano	matteo	1298	conto corrente
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito
HTYLOL54L07F845Y	osimhen	victor	5312	conto corrente

Download CSV

## 5.3 STORED PROCEDURE

- Stored Procedure 1

Iniziamo inserendo i valori:

```
EXEC INSERISCI_CASELLO('37912', 'NAPOLI EST 1');
```

E adesso visioniamo il risultato:

```
16
17 SELECT * FROM casello;
18
19
```

CODICE	NUMERO
12332	MILANO OVEST 3
12563	NAPOLI EST 1
37912	NAPOLI EST 1
67987	FIRENZE SUD 1
99762	FIRENZE SUD 1

- Stored Procedure 2

```
--inserisco il CLIENTE e, NOTA BENE, non inserisco il codice perchè
-- è generato automaticamente
EXEC INSERISCI_CLIENTE('CDRTWC64W21C7900', 'zieliński', 'piotr ', 'conto corrente');
```

Stampa a video:

- Esecuzione EXEX:

```
Statement processed.
Codice: 7637
```

- Esecuzione SELECT:

CODICE_FISCALE	COGNOME	NOME	CODICE	TIPO
BCCCRL54L07F839K	rui	mario	3333	conto corrente
MSSLNL87H24Z600D	di lorenzo	giovanni	5421	conto corrente
RNLCST85B05Z128L	meret	aex	1267	carta di credito
BLLMNC64P70C745L	politano	matteo	1298	conto corrente
RMNGPP36H07F209M	lobotka	stanislav	5555	carta di credito
HTYLOL54L07F845Y	osimhen	victor	5312	conto corrente
CDRTWC64W21C7900	zieliński	piotr	7637	conto corrente

## 6. INDICI

Per avere accesso alle informazioni all'interno di un file in modo più efficiente, si può far riferimento agli indici di accesso: strutture dati che permettono di organizzare in modo opportuno i record al fine di rendere efficiente il recupero tramite una chiave.

Quindi, tramite essi, andremo a velocizzare l'accesso ai dati.

Un data entry, è il record memorizzato in un file indice, quindi l'indice è una collezione di data entry.

Prima di andare a eseguire un esempio di indice su ORACLE, spieghiamo velocemente i tre tipi che esistono:

- 1) indice primario: indice costruito su una chiave, e ne esiste UNO SOLO all'interno di ogni file;
- 2) indice secondario: si tratta di un indice costruito su una chiave non primaria di una relazione;



3) indice clustering: costruito su un campo non chiave, di modo che ad ogni valore corrispondono più record.

Iniziamo con degli esempi:

❖ Creazione indice:

```
-- creazione indice sulla tabella automobili
-- se si vogliono eliminare i doppioni, si usa [UNIQUE]
CREATE INDEX INDICE_AUTOMOBILI
ON AUTOMOBILE (targa, modello, cliente, dispositivo );
```

❖ Come rinominare un indice:

```
-- rinominare indice
ALTER INDEX INDICE_AUTOMOBILI
RENAME TO INDICE_AUTOMOBILI_NEW;
```

❖ Eliminazione indice:

```
-- eliminare un indice
DROP INDEX INDICE_AUTOMOBILI_NEW;
```

L'utilizzo degli indici è utile quando si fa utilizzo di tabelle con grandi dimensioni per velocizzare le analisi sulle query. Questo non è il nostro caso, quindi è visto unicamente a scopo illustrativo.

```
SELECT targa, modello
FROM AUTOMOBILE
WHERE targa = 'DR743SS';
```

TARGA	MODELLO
DR743SS	mercedes

## 7. SITO WEB / INTERFACCIA UTENTE

In questo capitolo finale, andremo a dare una breve introduzione all'interfaccia utente.

Il sito relativo al dispositivo di telepedaggio “PartenoPass” è stato realizzato tramite un linguaggio di markup ossia HTML, e attraverso un linguaggio che permette di avere il controllo completo sulla presentazione di pagine web ossia CSS (Cascading Style Sheets).


Andiamo adesso ad illustrare, tramite codice, il design della pagina iniziale del nostro sito.

“Index.html”

```
1 <html><head>
2   <style type="text/css" id="operaUserStyle"></style>
3   <title>PartenoPass</title>
4 </head>
5 <body style="
6   background-color: #44abde;
7 ">
8   <div class="header_titolo">
9     <div class="header_logo" style="position: relative; bottom: 38px;">
10      <a aria-current="page" class="active">
11        
12      </a>
13    </div>
14  </div>
15  <link rel="stylesheet" href="style.css">
16
17
18
19
20
21  <div class="text" style="
22    margin-top: 4px;
23 ">
24    <p>Il dispositivo di telepedaggio che unisce la passione per il calcio e la cultura della magnifica città di Napoli.
25    Questo non è solo un telepedaggio, ma un'autentica esperienza che si fonde con l'entusiasmo e la fierezza dei tifosi del SSC Napoli.
26    </p><h2>Benvenuto in "PartenoPass"</h2>
27    Con "PartenoPass", viaggiate per le strade della vostra città con stile e passione, portando con voi l'amore per il calcio e il vostro team del cuore.<p></p>
28
29    <h3>Vantaggi del Telepass</h3>
30    <ul>
31      <li><li>Design Personalizzato: Il cuore del PartenoPass risiede nella sua estetica, pensata per riflettere l'anima vibrante e appassionata della squadra di calcio SSC Napoli. Ogni dettaglio del dispositivo è
32      <li>Connettività Azzurra: PartenoPass si collega senza problemi con le reti di pedaggio, assicurandovi una veloce e fluida transizione attraverso i caselli. La sua connettività avanzata è tanto affidabile
33      <li>Suoni Azzurri: Ogni volta che attraversate un casello, il dispositivo emette una melodia ispirata agli inni del SSC Napoli. Vi sentirete come se steste entrando in uno stadio, pronto per sostenere la
34      <li>Sconti e Vantaggi Esclusivi: Acquistando il PartenoPass, diventate automaticamente membri di un club esclusivo. Godete di sconti speciali nei negozi locali, ristoranti e attività culturali della città
35    </ul>
36
37    <h3>Scopri i nostri dispositivi</h3>
38    <p>Abbiamo una vasta gamma di dispositivi di telepedaggio progettati per soddisfare le tue esigenze di viaggio. Scegli il tuo preferito e inizia a godere dei vantaggi del telepedaggio oggi stesso!</p>
39
40    <div>
41      
43
44    </div>
45
46    <p>Per ulteriori informazioni o per effettuare un ordine, contattaci oggi stesso!</p>
47
48    <nav>
49      <a href="index.html">Home</a>
50      <a href="Info.html">Info</a>
51      <a class="Registra" href="Registrati.php">Registrati</a>
52    </nav>
53  </div>
54
55
56
57 </body></html>
```

Nella prima parte, troviamo una descrizione dettagliata sulle caratteristiche del dispositivo, mentre, nella barra di navigazione in basso troveremo:

- Home: vieni reindirizzato alla pagina corrente;
- Info: è una pagina di informazioni relativi ai creatori del sito web;
- Registrati: è la pagina di registrazione da parte di un cliente che vuole usufruire del dispositivo di telepedaggio.



Il dispositivo di telepedaggio che unisce la passione per il calcio e la cultura della magnifica città di Napoli. Questo non è solo un telepedaggio, ma un'autentica esperienza che si fonde con l'entusiasmo e la fiera dei tifosi del SSC Napoli.

### Benvenuto in "PartenoPass"


Con "PartenoPass", viaggiate per le strade della vostra città con stile e passione, portando con voi l'amore per il calcio e il vostro team del cuore.

#### Vantaggi del Telepass

- Design Personalizzato: Il cuore del PartenoPass risiede nella sua estetica, pensata per riflettere l'anima vibrante e appassionata della squadra di calcio SSC Napoli. Ogni dettaglio del dispositivo è un omaggio al glorioso patrimonio del club e alla bellezza di Napoli stessa.
- Connettività Azzurra: PartenoPass si collega senza problemi con le reti di pedaggio, assicurandovi una veloce e fluida transizione attraverso i caselli. La sua connettività avanzata tanto affidabile quanto il passaggio di un assist perfetto sul campo da parte dei vostri giocatori preferiti.
- Suoni Azzurri: Ogni volta che attraversate un casello, il dispositivo emette una melodia ispirata agli inni del SSC Napoli. Vi sentirete come se steste entrando in uno stadio, pronto per sostenere la vostra squadra del cuore.
- Sconti e Vantaggi Esclusivi: Acquistando il PartenoPass, diventate automaticamente membri di un club esclusivo. Godete di sconti speciali nei negozi locali, ristoranti e attività culturali della città. Mostrate il vostro spirito azzurro e vivete Napoli a pieno!

#### Scopri i nostri dispositivi

Abbiamo una vasta gamma di dispositivi di telepedaggio progettati per soddisfare le tue esigenze di viaggio. Scegli il tuo preferito e inizia a godere dei vantaggi del telepedaggio oggi stesso!



Per ulteriori informazioni o per effettuare un ordine, contattaci oggi stesso!

[Home](#) [Info](#) [Registrati](#)

Una volta cliccato su INFO, apparirà la seguente schermata:

### Gli autori di PartenoPass

Siamo entusiasti di presentarvi i cuori pulsanti dietro a questa avventura emozionante: Pietro e Matteo, due veri e propri tifosi sfegatati del Napoli! PartenoPass è nato dalla loro passione smisurata per la città, la squadra e il desiderio di rendere il viaggio attraverso il telepedaggio un'esperienza unica, proprio come tifare per la SSC Napoli.

### Pietro - Il Cuore Azzurro:

Pietro, anima e guida di PartenoPass, è un vero napoletano nella carne e nell'anima. Cresciuto tra le vie pittoresche di Napoli, ha imparato ad amare la sua città sin da giovane. Il suo legame con la SSC Napoli è profondo e inossidabile, trasmettendo la passione per il calcio attraverso ogni fibra del suo essere. La creazione di PartenoPass è stata la sua risposta alla domanda: "Come possiamo rendere il telepedaggio un'esperienza che rispecchi la nostra Napoli?"

### Matteo - L'Architetto della Passione:

Matteo, il co-creatore di PartenoPass, è un vero esperto nel suo campo. Con una mente creativa e una passione per la tecnologia, ha contribuito a plasmare PartenoPass in un'opera d'arte digitale. La sua dedizione allo sviluppo di soluzioni innovative è il motore che ha reso possibile l'esistenza di questo telepedaggio unico. La sua ambizione è far sì che PartenoPass non sia solo un mezzo di pagamento, ma una celebrazione della cultura e dell'amore per il calcio partenopeo.

Insieme, Pietro e Matteo hanno unito le loro forze per creare PartenoPass, il telepedaggio che non solo semplifica i tuoi viaggi, ma anche celebra la bellezza e l'anima di Napoli e della SSC Napoli. Siamo certi che con PartenoPass, ogni viaggio diventerà un inno alla passione, all'amicizia e all'amore per il calcio!

*Forza Napoli!*

Home

Nella pagina di registrazione, l'estetica è sempre stata realizzata tramite HTML e CSS, ma la parte di implementazione delle tuple composte dai campi è stata realizzata tramite il linguaggio di programmazione PHP e la piattaforma software multiplatforma XAMPP, costituita dal server web Apache necessario per ospitare il sito web su internet (in questo caso utilizzato al livello locale) e per comodità il DBMS MySql che è incorporato all'interno di XAMPP.

## “Registrati.php”

```
1 <!DOCTYPE html>
2 <html lang="it">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Menu di Registrazione</title>
7   <link rel="stylesheet" href="style.css">
8   <style type="text/css" id="operaUserStyle"></style>
9 </head>
10
11 <body>
12
13   <div class="text">
14     <h2>Fase di registrazione:</h2>
15
16     <form action="registrazione.php" method="post">
17
18       <label for="codicefiscale">Codice Fiscale <span style="color: red;">*</span></label>
19       <input type="text" id="codicefiscale" name="codicefiscale" required="">
20
21       <label for="nome">Nome <span style="color: red;">*</span></label>
22       <input type="text" id="nome" name="nome" required="">
23
24       <label for="cognome">Cognome <span style="color: red;">*</span></label>
25       <input type="text" id="cognome" name="cognome" required="">
26
27       <label for="conto">Conto <span style="color: red;">*</span></label>
28       <input type="text" id="conto" name="conto" required="" placeholder="xxxx xxxx xxxx" maxlength="19" style="width: 21%;">
29
30       <label for="cvv">CVV <span style="color: red;">*</span></label>
31       <input type="text" id="cvv" name="cvv" required="" placeholder="xxx" maxlength="3" style="width: 6%;">
32
33       <input type="submit" class="Registra" value="Fine registrazione">
34     </form>
35
36     <nav>
37       <a href="index.html">Home</a>
38     </nav>
39   </div>
40
41 </body>
42 </html>
```

Quindi, cliccando sul tasto **Registrati** si andrà avanti per inserire i valori riguardanti il cliente con il relativo conto corrente o carta di credito, ovviamente troviamo campi **Obbligatorî**, nel senso che, se si prova a non inserirli, non si potrà procedere alla registrazione.

### Fase di registrazione:

Codice Fiscale \*

Nome \*

Cognome \*

Conto \*

CVV \*

Fine registrazione

Home

In questo caso viene utilizzato il linguaggio PHP per effettuare una connessione locale al database costruito e la creazione della tupla, associando i campi compilati in “Registrati.php” ai campi della tabella clienti costruita nel database “PartenoPass”.

“Registrazione.php”

```
1 <?php
2 // Connessione al database
3 $servername = "localhost";
4 $username = "root";
5 $password = "";
6 $dbname = "partenopass";
7
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Verifica della connessione
11 if ($conn->connect_error) {
12     die("Connessione fallita: " . $conn->connect_error);
13 }
14
15 // Recupero dati dal modulo HTML
16 $codicefiscale = $_POST['codicefiscale'];
17 $nome = $_POST['nome'];
18 $cognome = $_POST['cognome'];
19 $conto = $_POST['conto'];
20 $cvv = $_POST['cvv'];
21
22 // Inserimento dei dati nel database
23 $sql = "INSERT INTO clienti (codice_fiscale, nome, cognome, codice, tipo) VALUES ('$codicefiscale', '$nome', '$cognome', '$cvv', '$conto')";
24
25 if ($conn->query($sql) === TRUE) {
26     //echo "Registrazione avvenuta con successo";
27     header('Location: Registrati3.html');
28 } else {
29     echo "Errore durante la registrazione: " . $conn->error;
30 }
31
32 // Chiusura della connessione
33 $conn->close();
34 ?>
```

Nel caso in cui la registrazione dovesse essere avvenuta con successo, il cliente viene reindirizzato alla pagina di fine registrazione.

## “Registrati3.html”

```
1 <html><head>
2   <meta charset="UTF-8">
3   <meta name="viewport" content="width=device-width, initial-scale=1.0">
4   <title>Menu di Registrazione</title>
5   <style>
6   </style>
7   <link rel="stylesheet" href="style.css"><style type="text/css" id="operaUserStyle"></style></head>
8
9
10
11
12 <body>
13
14   <div class="text">
15     <h2>Grazie per esserti iscritto!</h2>
16
17     <p>Benvenuto nella nostra community dedicata ai tifosi del SSC Napoli e ai cittadini di Napoli.</p><div>
18     <p style="
19">Ora fai parte della nostra famiglia! Sarai sempre aggiornato sulle ultime novità e offerte relative al nostro dispositivo di telepass a tema Napoli.</p>
20     <p>Forza Napoli!</p>
21
22     <nav>
23       <a href="index.html">Home</a>
24     </nav>
25   </div>
26
27
28
29
30 </body></html>
```

Infine, cliccando su ***Fine registrazione***, verrà mostrato un messaggio a video informando l'utente che l'iscrizione è avvenuta con successo.

### Grazie per esserti iscritto!

Benvenuto nella nostra community dedicata ai tifosi del SSC Napoli e ai cittadini di Napoli.

Ora fai parte della nostra famiglia! Sarai sempre aggiornato sulle ultime novità e offerte relative al nostro dispositivo di telepass a tema Napoli.

Forza Napoli!

Home

Siamo giunti alla fine del nostro progetto, dove sono state introdotte varie tematiche mostrando esempi, cenni di teoria e programmazione.

Prodotto da: CONTE PIETRO && BENEDETTO MATTEO.