

UNIVERSITÀ DEGLI STUDI DEL MOLISE



CORSO DI LAUREA IN INFORMATICA

ScroKING

Ciaramella Giovanni – 158937

Discenza Christian – 160198

Fagnano Stefano – 158985

Russodivito Marco – 158940

Anno Accademico 2017/2018

“Una base di viaggio”

PROGETTO

DI

BASI DI DATI E SISTEMI INFORMATIVI

Prof. Francesco Mercaldo

INDICE:

Agenzia di Viaggi	6
Progettazione concettuale	7
Entità	8
Associazioni	10
Grafo ER	12
Progettazione logica	13
Trasformazioni Preliminari	14
Grafo E-R Trasformato	15
Traduzioni di Entità	16
Traduzioni di Associazioni	18
Documentazione degli schemi	20
Implementazione	23
Inserimento dati	26
Trigger, stored procedure, query	31
Trigger	32
Stored procedure	34
Query – Giovanni Ciaramella	36
Query – Christian Discenza	37
Query – Stefano Fagnano	38
Query – Marco Russodivito	39
Sito web	41
https://www.scrokingviaggi.com	42
Database.php	43

*Non dirmi quanti anni hai,
o quanto sei educato e
colto, dimmi dove hai
viaggiato e che cosa sai.*

~Maometto

Agenzia di Viaggi

Si vuole realizzare la base di dati per la gestione dei viaggi di un'agenzia di viaggi. Ogni tipo di viaggio è identificato dal luogo di partenza e dalla destinazione. Per ogni tipo di viaggio sono noti la durata in giorni e il costo. Ogni **viaggio** è identificato univocamente dalla data di partenza e dal tipo di viaggio. Per i viaggi scontati deve essere memorizzata la percentuale di sconto applicata sul prezzo originale.

Gli spostamenti durante il viaggio vengono effettuati tramite dei mezzi affittati sul posto. Ogni **mezzo** è identificato univocamente da un **codice numerico**. Per ogni mezzo sono noti il costo di affitto e la capienza massima (numero di persone a sedere). Si deve mantenere traccia dei mezzi affittati per ogni viaggio, e della data in cui tali affitti sono stati effettuati. Ogni mezzo può essere affittato più volte, in date differenti, per lo stesso viaggio.

Gli **alberghi** in cui i viaggiatori **soggiornano** sono identificati univocamente da un **codice numerico**. Per ogni albergo sono noti il nome, il numero di posti disponibili, i numeri di telefono dell'albergo. Si vuole memorizzare il numero di stanze prenotate per ogni viaggio in ogni albergo.

I **dipendenti** dell'albergo sono identificati univocamente da un **codice progressivo all'interno dell'albergo**. Per ogni dipendente sono noti il nome, il cognome, la data di nascita. I dipendenti si dividono in due categorie: **fissi e stagionali**. Per i dipendenti fissi sono note la retribuzione annuale e la qualifica assunta nell'organigramma aziendale. Per i dipendenti stagionali sono note la data di inizio e quella di fine della collaborazione e la retribuzione giornaliera. Alcuni dipendenti fissi sono responsabili dell'albergo. Memorizzare tale informazione nella base di dati.

I **clienti** sono identificati univocamente dal loro **codice fiscale**. Per i clienti sono noti nome, cognome, indirizzo e recapito telefonico. Memorizzare i viaggi, **prenotazioni** da ogni cliente e la data in cui la prenotazione viene fatta.

Legenda:

Entità

Identificatori

Associazioni

Attributi

Vincoli



1

PROGETTAZIONE CONCETTUALE

ENTITÀ

NOME ENTITÀ: Viaggio

DESCRIZIONE ENTITÀ: Viaggio organizzato dall'agenzia di viaggi.

Attributo	Descrizione	Tipo	ID
codViaggio	Chiave artificiale progressiva identificativa di un viaggio	intero	si
tipoViaggio	Tipologia del viaggio (es. vacanza, crociata)	stringa	no
luogoPartenza	Luogo di partenza del viaggio	stringa	no
luogoDestinazione	Luogo di destinazione del viaggio	stringa	no
costoViaggio	Costo in euro del viaggio	decimale	no
dataInizioViaggio	Data di inizio del viaggio	data	no
dataFineViaggio	Data di fine del viaggio	data	no

Vincolo1	Tutti gli attributi devono essere NOT NULL
Vincolo2	<i>costoViaggio</i> deve essere maggiore di 0
Vincolo3	<i>dataInizioViaggio</i> deve essere successiva alla data odierna
Vincolo4	<i>dataFineViaggio</i> deve essere successiva a <i>dataInizioViaggio</i>

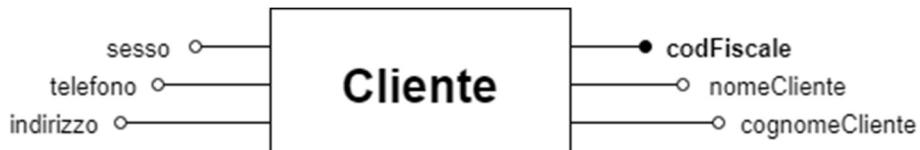


NOME ENTITÀ: Cliente

DESCRIZIONE ENTITÀ: Cliente dell'agenzia di viaggi.

Attributo	Descrizione	Tipo	ID
codFiscale	Codice fiscale identificativo del cliente	stringa	si
nomeCliente	Nome del cliente	stringa	no
cognomeCliente	Cognome del cliente	stringa	no
sesso	Sesso del cliente	carattere	no
telefono	Numero di telefono del cliente	stringa	no
indirizzo	Indirizzo del cliente	stringa	no

Vincolo1	Tutti gli attributi devono essere NOT NULL
Vincolo2	<i>codFiscale</i> deve essere formato da esattamente 16 caratteri
Vincolo3	<i>sesso</i> deve essere 'M' oppure 'F'
Vincolo4	<i>telefono</i> deve essere unico per ogni cliente

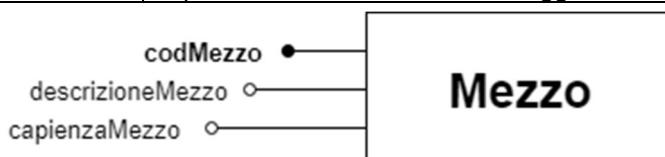


NOME ENTITÀ: Mezzo

DESCRIZIONE ENTITÀ: Mezzo di trasporto utilizzato durante un viaggio.

Attributo	Descrizione	Tipo	ID
codMezzo	Codice progressivo identificativo del mezzo	intero	si
descrizioneMezzo	Informazioni sul mezzo (es. tipologia, modello)	stringa	no
capienzaMezzo	Capienza del mezzo, posti totali disponibili sul mezzo	intero	no

Vincolo1	Tutti gli attributi devono essere NOT NULL
Vincolo2	<i>capienzaMezzo</i> deve essere maggiore strettamente di 0



NOME ENTITÀ: Albergo

DESCRIZIONE ENTITÀ: Albergo in cui soggiornano i clienti.

Attributo	Descrizione	Tipo	ID
codAlbergo	Codice numerico progressivo identificativo dell'albergo	intero	si
denominazione	Denominazione, nome dell'albergo	stringa	no
stanzeDisponibili	Stanze disponibili nell'albergo	intero	no
telefono	Numeri telefonici dell'albergo (es. fisso, fax, cellulare)	stringhe	no

Vincolo1	Tutti gli attributi devono essere NOT NULL
Vincolo2	<i>stanzeDisponibili</i> deve essere maggiore di 0

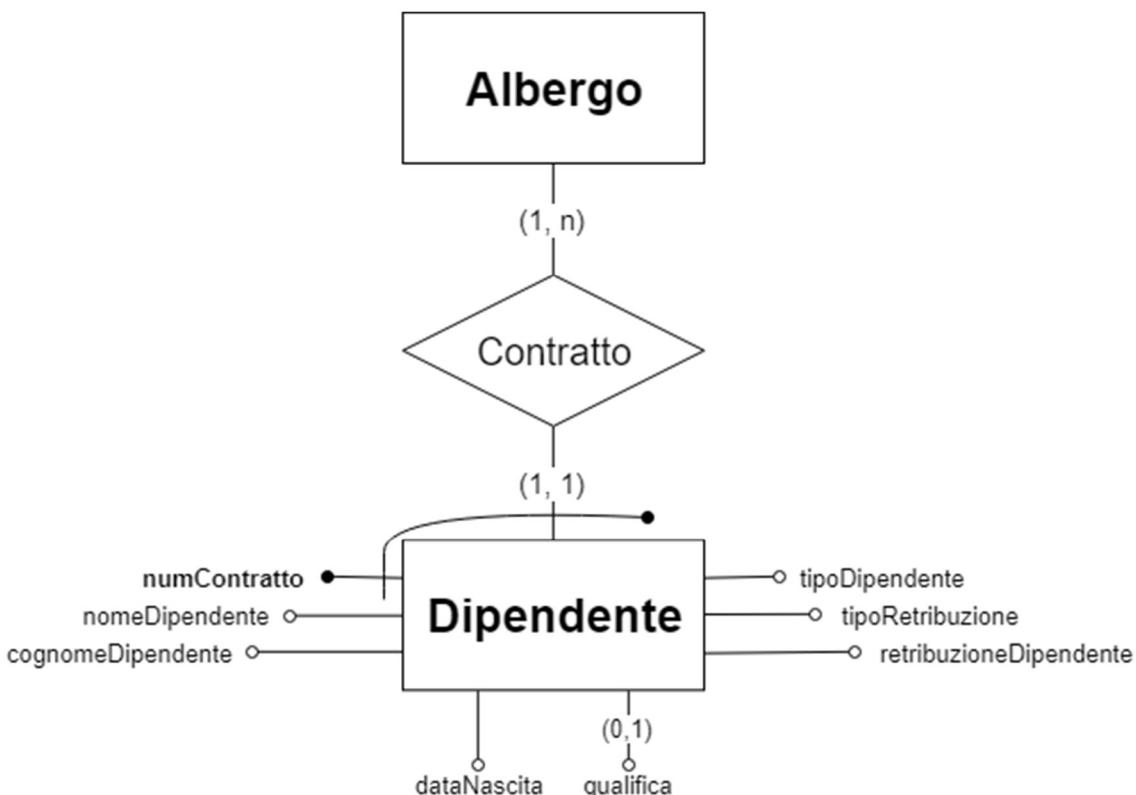


NOME ENTITÀ: Dipendente

DESCRIZIONE ENTITÀ: Dipendente di un albergo.

Attributo	Descrizione	Tipo	ID
numContratto	Codice numerico progressivo interno all'albergo	intero	si
tipoDipendente	Tipo di dipendente dell'albergo	stringa	no
nomeDipendente	Nome del dipendente	stringa	no
cognomeDipendente	Cognome del dipendente	stringa	no
dataNascita	Data di nascita del dipendente	data	no
tipoRetribuzione	Cadenza della retribuzione del dipendente	stringa	no
retribuzioneDipendente	Retribuzione in euro del dipendente	decimale	no
qualifica	Qualifica assunta da dipendenti fissi	stringa	no

Vincolo1	<i>tipoDipendente</i> deve essere "fisso", 'fisso dirigente' oppure 'stagionale'
Vincolo2	<i>tipoRetribuzione</i> deve essere 'giornaliero' oppure 'annuale'
Vincolo3	<i>tipoRetribuzione</i> è 'giornaliero' se <i>tipoDipendente</i> è 'stagionale'
Vincolo4	<i>tipoRetribuzione</i> è 'annuale' se <i>tipoDipendente</i> è 'fisso' o 'fisso dirigente'
Vincolo5	<i>retribuzioneDipendente</i> deve essere strettamente maggiore di 0
Vincolo6	<i>qualifica</i> deve essere NULL se <i>tipoDipendente</i> è 'stagionale'



ASSOCIAZIONI

NOME ASSOCIAZIONE: Prenotazione

ENTITÀ COINVOLTE: Cliente – Viaggio

DESCRIZIONE: Un cliente può effettuare o meno più prenotazioni
Un viaggio può essere prenotato o meno da più clienti

Attributo	Descrizione	Tipo
dataPrenotazione	Data prenotazione effettuata da un cliente	data
pSconto	Percentuale di sconto applicata al cliente	decimale
Vincolo1	<i>dataPrenotazione</i> deve essere precedente a <i>dataInizioViaggio</i>	
Vincolo2	<i>pSconto</i> deve essere maggiore di 0% ma strettamente minore di 100%	



NOME ASSOCIAZIONE: Affitto

ENTITÀ COINVOLTE: Mezzo – Viaggio

DESCRIZIONE: Un mezzo può essere affittato o meno
Si possono affittare o meno più mezzi durante un viaggio

Attributo	Descrizione	Tipo
dataInizioAffitto	Data di inizio dell'affitto	Data
dataFineAffitto	Data di fine dell'affitto	Data
costoAffitto	Costo in euro per l'affitto del mezzo	decimale
Vincolo1	<i>dataInizioAffitto</i> deve essere successiva a <i>dataPartenza</i> .	
Vincolo2	<i>dataFineAffitto</i> deve essere successiva a <i>dataInizioAffitto</i> .	
Vincolo3	<i>costoAffitto</i> deve essere maggiore di zero.	



NOME ASSOCIAZIONE: Soggiorno
ENTITÀ COINVOLTE: Albergo – Viaggio

DESCRIZIONE: Si può soggiornare o meno in più alberghi
 Un albergo può avere o meno più soggiorni

Attributo	Descrizione	Tipo
dataInizioSoggiorno	Data di inizio del soggiorno	data
dataFineSoggiorno	Data di fine del soggiorno	data
numStanze	Numero di stanze prenotate	intero
Vincolo1	<i>dataInizioSoggiorno</i> deve essere successiva a <i>dataInizioViaggio</i>	
Vincolo2	<i>dataFineSoggiorno</i> deve essere successiva a <i>dataInizioSoggiorno</i>	
Vincolo3	<i>numStanze</i> deve essere minore o uguale <i>stanzeDisponibili</i>	
Vincolo4	<i>numStanze</i> deve essere maggiore strettamente di 0	



NOME ASSOCIAZIONE: Contratto

ENTITÀ COINVOLTE: Albergo – Viaggio

DESCRIZIONE: Ogni albergo stipula più contratti.
 Ogni dipendente deve avere un contratto.

Attributo	Descrizione	Tipo
dataInizioContratto	Progressivo interno all'albergo	int
dataFineContratto	Data di inizio	data
Vincolo1	<i>dataInizioContratto</i> deve essere successiva alla data odierna	
Vincolo2	<i>dataFineContratto</i> deve essere NULL o successiva a <i>dataInizioContratto</i>	

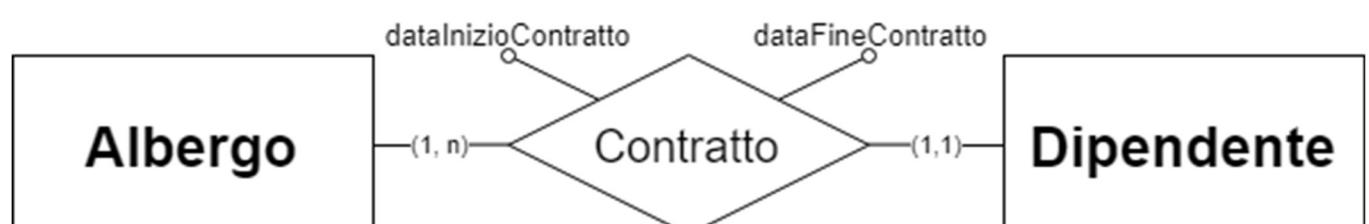
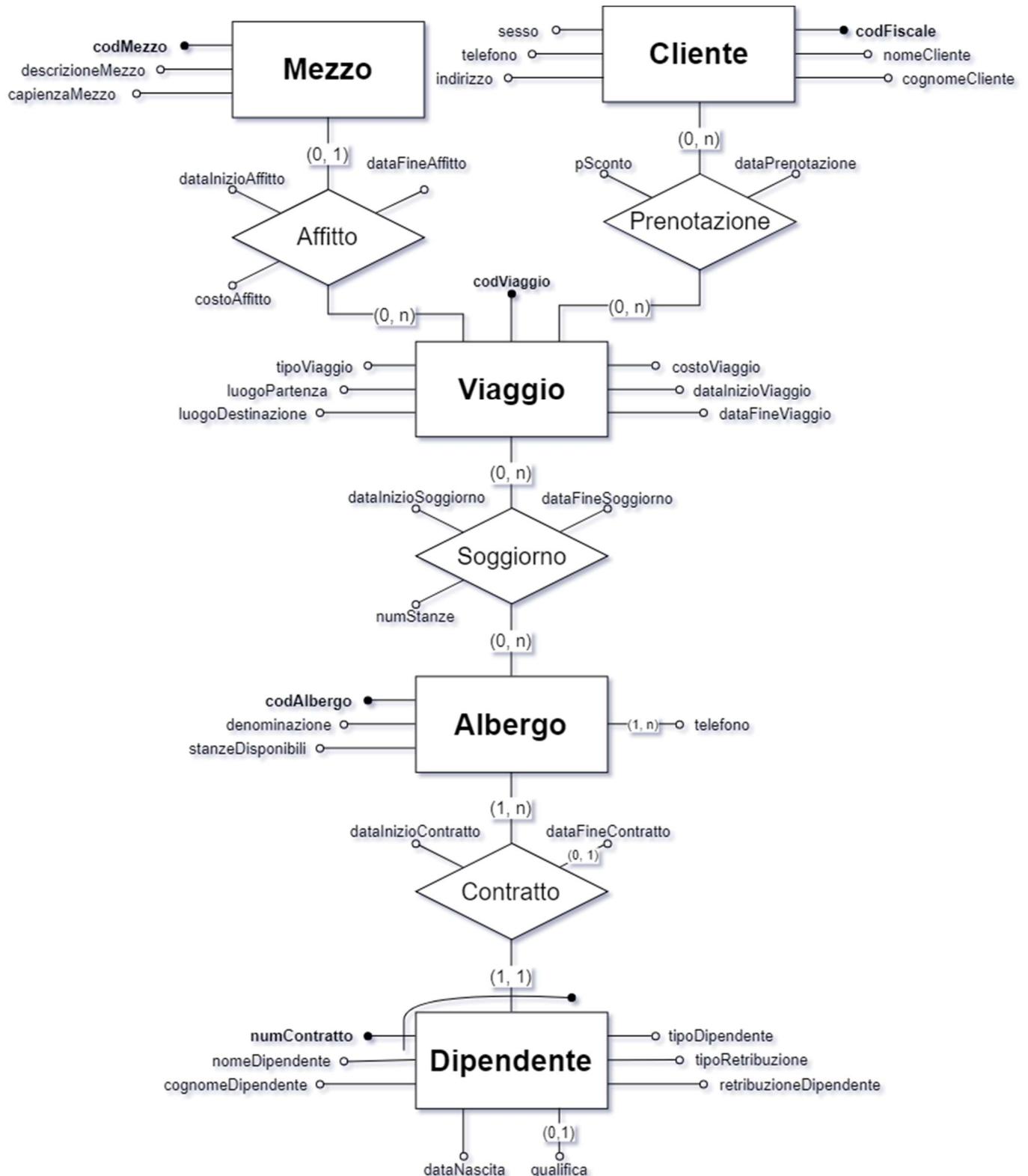


DIAGRAMMA E-R



Customer_id
Firstname
Lastname
Address
Postal_code
Age
Gender
Email
Order_id
Invoice_id

Order

Order_id
Total
Product_id
Customer_id
Date_time
Remark

Product

Product_id
Product_name
Amount
Price
Description
Image
Date_time
Status
Statistic

Invoice

Invoice_id
Customer_id
Order_id
Product_id
Date_time
Status
Total
Remark



2

PROGETTAZIONE LOGICA

TRASFORMAZIONI PRELIMINARI

L'attributo multi-valore *telefono* viene trasformato nella nuova Entità **Telefono** e viene creata l'Associazione uno a molti **Recapito** tra le entità Albergo e Telefono.

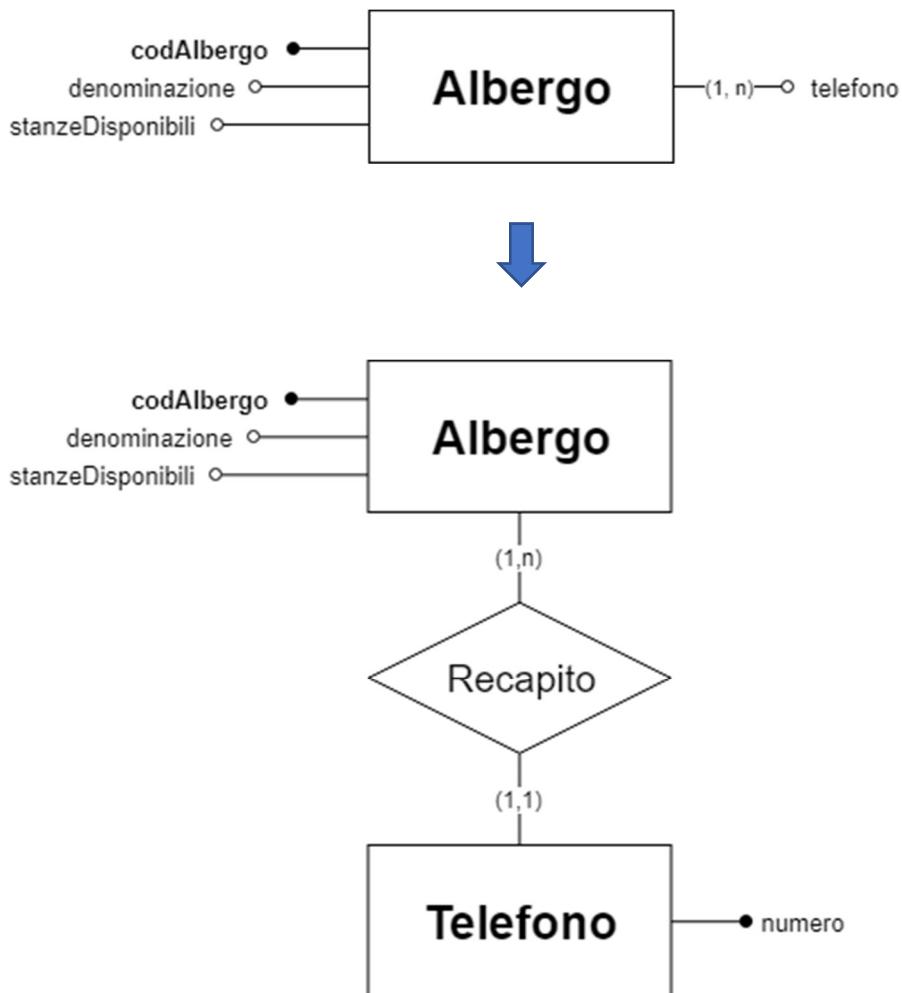
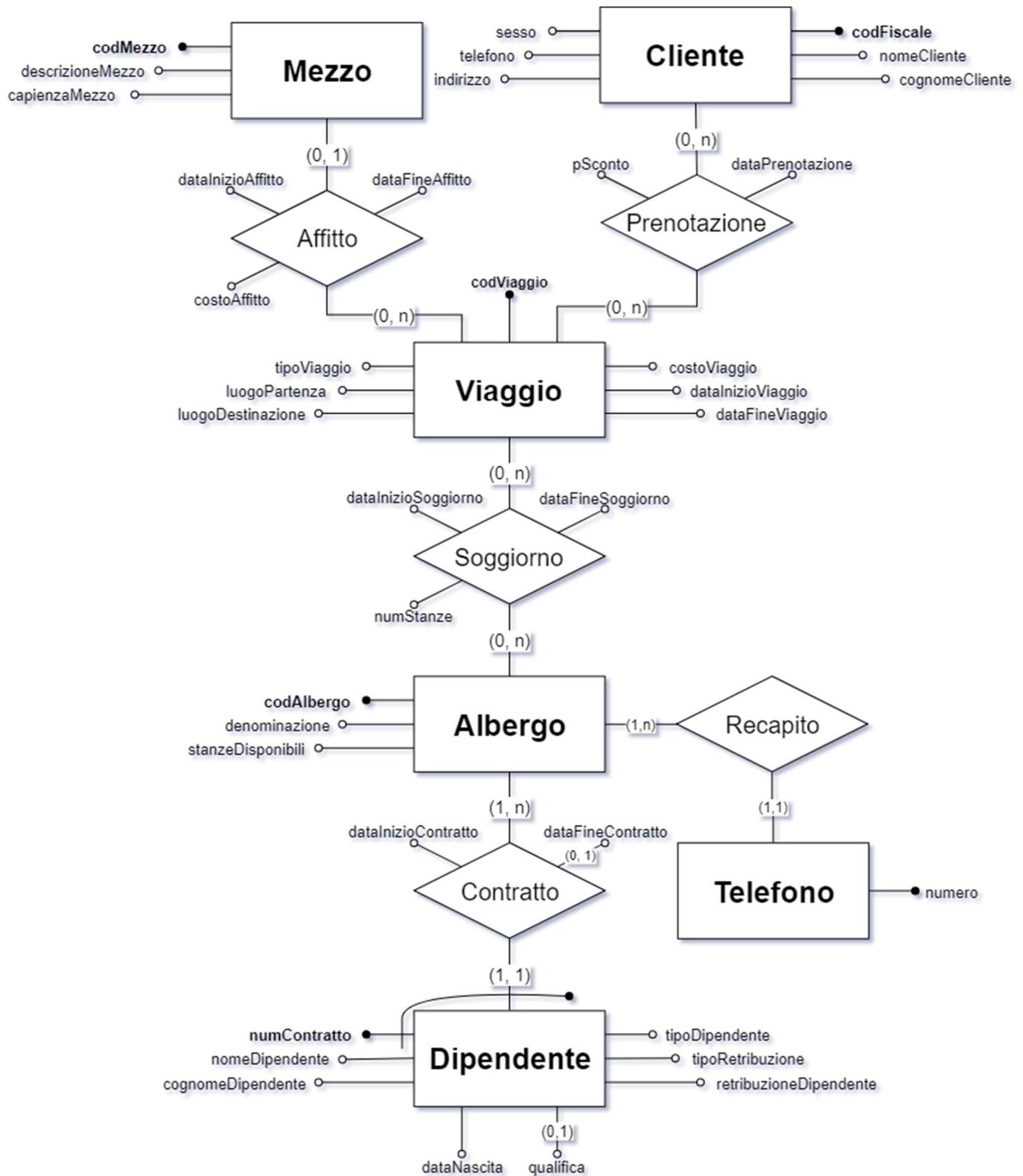


DIAGRAMMA E-R TRASFORMATO



TRADUZIONE DI ENTITÀ

L'Entità **Viaggio** si traduce nella relazione **Viaggi** con gli attributi di **Viaggio**.

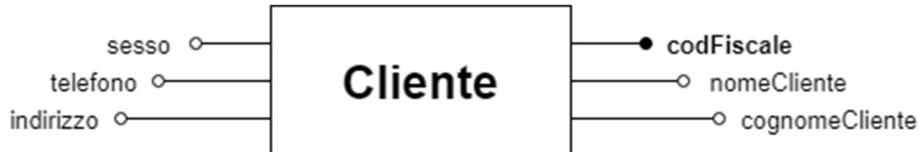
La chiave primaria è l'identificatore dell'Entità **Viaggio**.



Viaggi $\equiv \{\underline{\text{codViaggio}}, \text{tipoViaggio}, \text{luogoPartenza}, \text{luogoDestinazione}, \text{costoViaggio}, \text{dataInizioViaggio}, \text{dataFineViaggio}\}$

L'Entità **Cliente** si traduce nella relazione **Clienti** con gli attributi di **Cliente**.

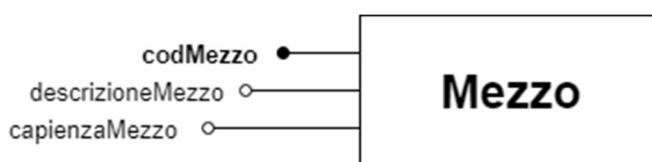
La chiave primaria è l'identificatore dell'Entità **Cliente**.



Clienti $\equiv \{\underline{\text{codFiscale}}, \text{nomeCliente}, \text{cognomeCliente}, \text{sesso}, \text{telefono}, \text{indirizzo}\}$

L'Entità **Mezzo** si traduce nella relazione **Mezzi** con gli attributi di **Mezzo**.

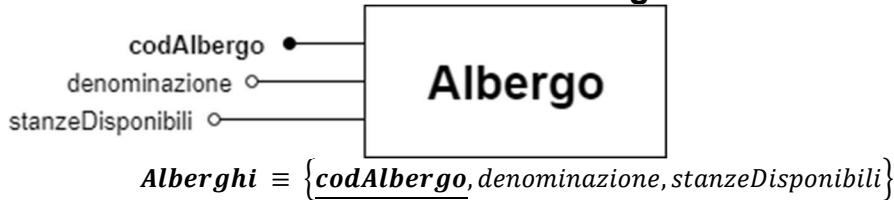
La chiave primaria è l'identificatore dell'Entità **Mezzo**.



Mezzi $\equiv \{\underline{\text{codMezzo}}, \text{descrizioneMezzo}, \text{capienzaMezzo}\}$

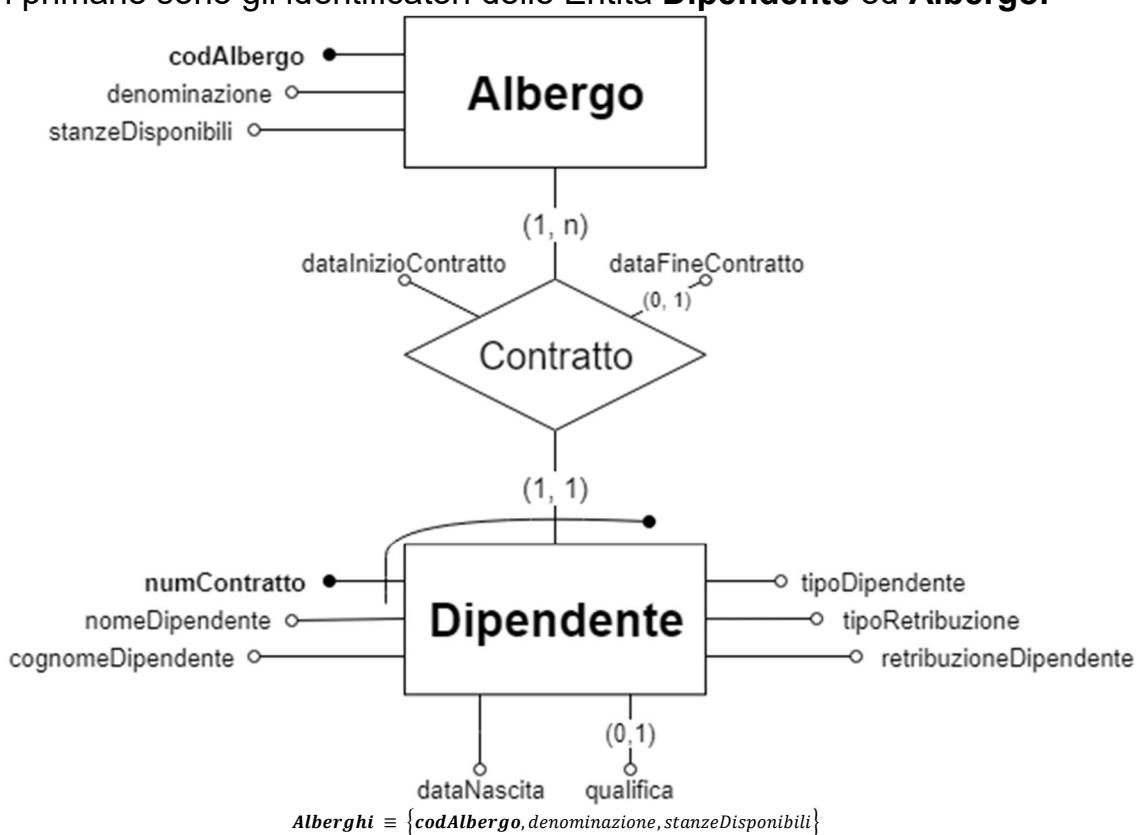
L'Entità **Albergo** si traduce nella relazione **Alberghi** con gli attributi di **Albergo**.

La chiave primaria è l'identificatore dell'Entità **Albergo**.



L'Entità **Dipendenti**, con identificatore esterno, si traduce nella relazione **Dipendenti** con gli attributi di **Albergo** assorbendo l'Associazione **Contratto** ed i suoi attributi.

Le chiavi primarie sono gli identificatori delle Entità **Dipendente** ed **Albergo**.

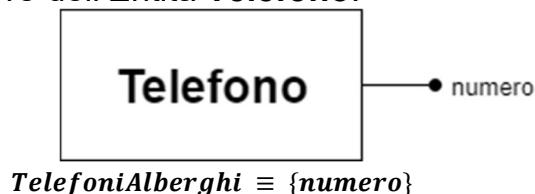


Dipendenti ≡ {numContratto, nomeDipendente, cognomeDipendente, dataNascita, tipoDipendente, tipoRetribuzione, retribuzione, qualifica, dataInizioContratto, dataFineContratto, albergo}

Per l'attributo *albergo* vincolo di referenza esterna all'attributo *codAlbergo* della relazione **Alberghi**.

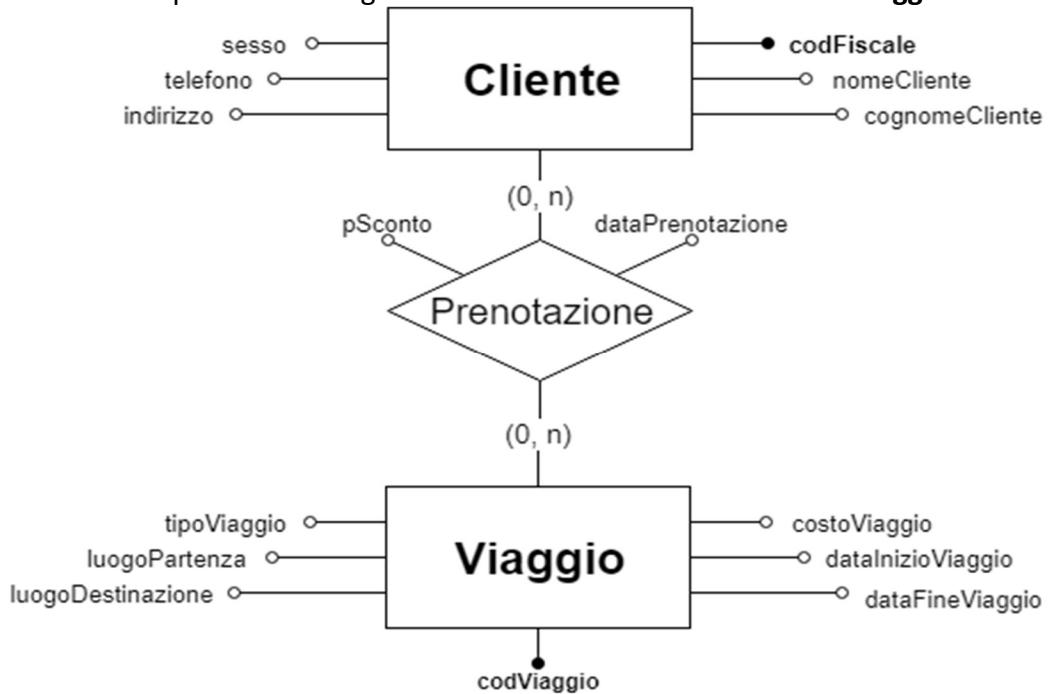
L'Entità **Telefono** si traduce nella relazione **TelefoniAlberghi** con gli attributi di **Telefono**.

La chiave primaria l'identificatore dell'Entità **Telefono**.



TRADUZIONE DI ASSOCIAZIONI

L'Associazione molti a molti **Prenotazione** si traduce nella relazione **Prenotazioni** con gli attributi di **Prenotazione**. Le chiavi primarie sono gli identificatori delle Entità **Cliente** e **Viaggio**.



Clienti ≡ {codFiscale, nomeCliente, cognomeCliente, sesso, telefono, indirizzo}

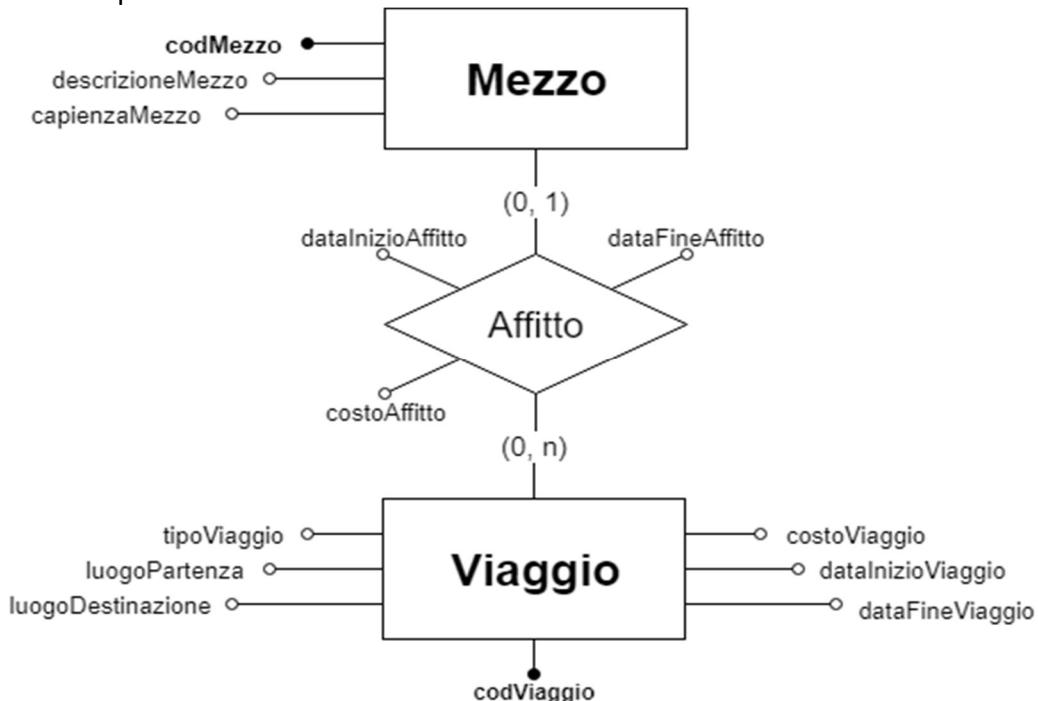
Viaggi ≡ {codViaggio, tipoViaggio, luogoPartenza, luogoDestinazione, costoViaggio, dataInizioViaggio, dataFineViaggio}

Prenotazioni ≡ {cliente, viaggio, pSconto, dataPrenotazione}

Per l'attributo *viaggio* vincolo di referenza esterna all'attributo *codViaggio* della relazione *Viaggi*.

Per l'attributo *cliente* vincolo di referenza esterna all'attributo *codFiscale* della relazione *Clienti*.

L'Associazione uno a molti *opzionale* **Affitto** si assorbe nella relazione **Mezzi** aggiungendone gli attributi di **Affitto**. La chiave primaria è l'identificatore dell'Entità **Mezzo**.



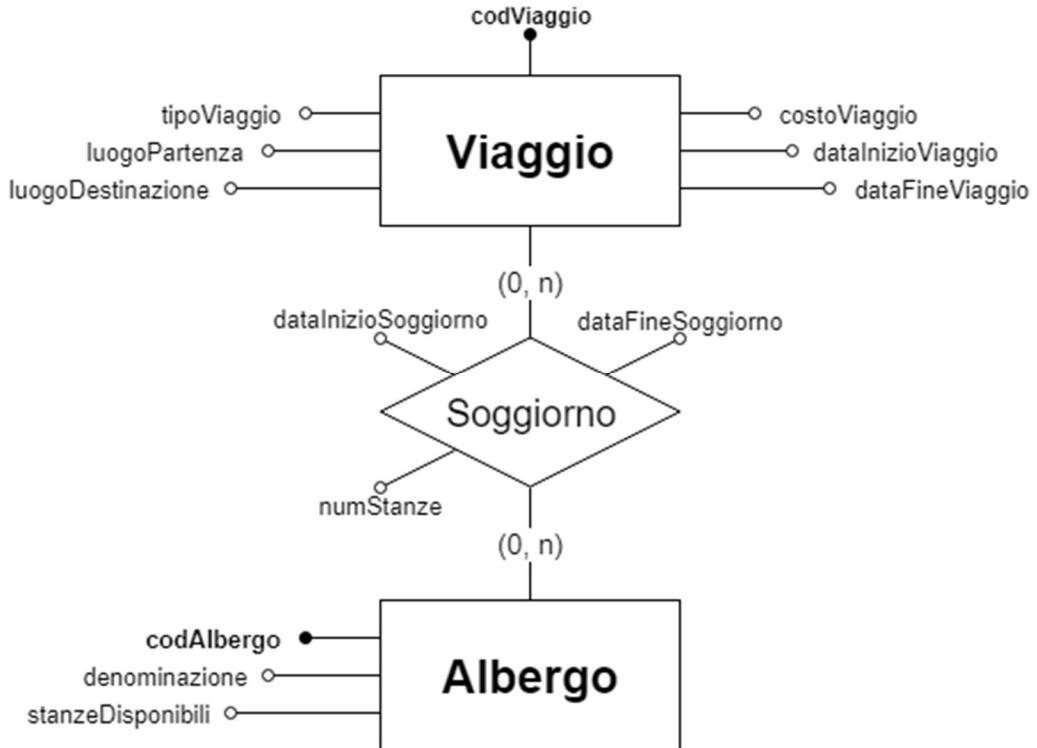
Viaggi ≡ {codViaggio, tipoViaggio, luogoPartenza, luogoDestinazione, costoViaggio, dataInizioViaggio, dataFineViaggio}

Mezzi ≡ {codMezzo, descrizioneMezzo, capienzaMezzo, viaggio, dataInizioAffitto, dataFineAffitto, costAffitto}

Gli attributi *dataInizioAffitto*, *dataFineAffitto*, *costAffitto* e *viaggio* ammettono valori NULL.

Per l'attributo *viaggio* vincolo di referenza esterna all'attributo *codViaggio* della relazione *Viaggi*.

L'Associazione molti a molti **Soggiorno** si traduce nella relazione **Soggiorni** con gli attributi di **Soggiorno**. Le chiavi primarie sono gli identificatori delle Entità **Viaggio** e **Albergo**.



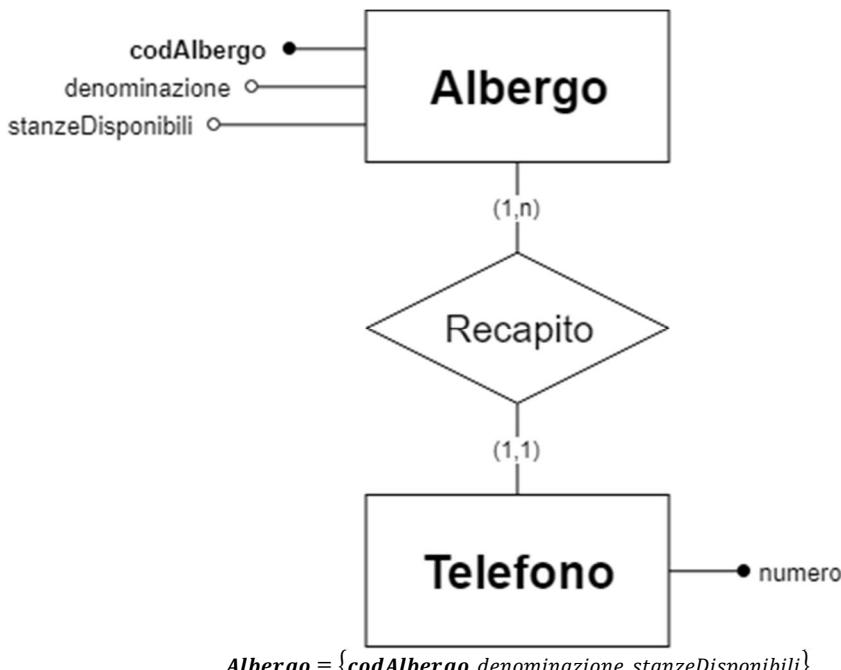
Viaggi $\equiv \{\underline{\text{codViaggio}}, \text{tipoViaggio}, \text{luogoPartenza}, \text{luogoDestinazione}, \text{costoViaggio}, \text{dataInizioViaggio}, \text{dataFineViaggio}\}$
Albergo $\equiv \{\underline{\text{codAlbergo}}, \text{denominazione}, \text{stanzeDisponibili}\}$

Soggiorno $\equiv \{\underline{\text{viaggio}}, \underline{\text{albergo}}, \text{dataInizioSoggiorno}, \text{dataFineSoggiorno}, \text{numStanze}\}$

Per l'attributo *viaggio* vincolo di referenza esterna all'attributo *codViaggio* della relazione **Viaggi**.

Per l'attributo *albergo* vincolo di referenza esterna all'attributo *codAlbergo* della relazione **Alberghi**.

L'Associazione uno a molti **Recapito** si assorbe nella relazione **TelefoniAlberghi** aggiungendone gli attributi di **Albergo**. La chiave primaria è l'identificatore di **Telefono**.



Albergo $\equiv \{\underline{\text{codAlbergo}}, \text{denominazione}, \text{stanzeDisponibili}\}$

TelefoniAlberghi $\equiv \{\underline{\text{numero}}, \underline{\text{albergo}}\}$

Per l'attributo *albergo* vincolo di referenza esterna all'attributo *codAlbergo* della relazione **Alberghi**.

DOCUMENTAZIONE INTEGRATIVA DEGLI SCHEMI

Viaggi ≡ {codViaggio, tipoViaggio, luogoPartenza, luogoDestinazione, costoViaggio, dataInizioViaggio, dataFineViaggio}

Attributo	Tipo	Clausola	Vincoli
<u>codViaggio</u>	INT(10)	PRIMARY KEY	<u>codViaggio</u> > 0
<u>tipoViaggio</u>	VARCHAR(20)	NOT NULL	-
<u>luogoPartenza</u>	VARCHAR(10)	NOT NULL	-
<u>luogoDestinazione</u>	VARCHAR(10)	NOT NULL	-
<u>costoViaggio</u>	DECIMAL(10,2)	NOT NULL	<u>costoViaggio</u> >= 0
<u>dataInizioViaggio</u>	DATE	NOT NULL	<u>dataInizioViaggio</u> > CURRENT_DATE()
<u>dataFineViaggio</u>	DATE	NOT NULL	<u>dataFineViaggio</u> > <u>dataInizioViaggio</u>

Clienti ≡ {codFiscale, nomeCliente, cognomeCliente, sesso, telefono, indirizzo}

Attributo	Tipo	Clausola	Vincoli
<u>codFiscale</u>	VARCHAR(16)	PRIMARY KEY	CHAR_LENGTH(<u>codFiscale</u>) = 16
<u>nomeCliente</u>	VARCHAR(15)	NOT NULL	-
<u>cognomeCliente</u>	VARCHAR(20)	NOT NULL	-
<u>sesso</u>	ENUM('M', 'F')	NOT NULL	-
<u>telefono</u>	VARCHAR(16)	UNIQUE	-
<u>indirizzo</u>	VARCHAR(30)	NOT NULL	-

Prenotazioni ≡ {cliente, viaggio, pSconto, dataPrenotazione}

Attributo	Tipo	Clausola	Vincoli
<u>cliente</u>	CHAR(16)	PRIMARY KEY	-
<u>viaggio</u>	INT(10)	PRIMARY KEY	-
<u>pSconto</u>	DECIMAL(5,2)	NOT NULL	<u>pSconto</u> >= 0 AND <u>pSconto</u> <= 100
<u>dataPrenotazione</u>	DATE	NOT NULL	<u>dataPrenotazione</u> < <u>dataInizioViaggio</u>

Mezzi ≡ {codMezzo, descrizioneMezzo, capienzaMezzo, dataInizioAffitto, viaggio, dataFineAffitto, costoAffitto}

Attributo	Tipo	Clausola	Vincoli
<u>codMezzo</u>	INT(10)	PRIMARY KEY	<u>codMezzo</u> > 0
<u>descrizioneMezzo</u>	VARCHAR(30)	NOT NULL	-
<u>capienzaMezzo</u>	INT(10)	NOT NULL	<u>capienzaMezzo</u> > 0
<u>viaggio</u>	INT(10)	FOREIGN KEY	-
<u>costoAffitto</u>	DECIMAL(10,2)	-	<u>costoAffitto</u> >= 0
<u>dataInizioAffitto</u>	DATE	-	<u>dataInizioAffitto</u> > <u>dataInizioViaggio</u>
<u>dataFineAffitto</u>	DATE	-	<u>dataFineAffitto</u> > <u>dataInizioAffitto</u>

Albergo $\equiv \{ \underline{\text{codAlbergo}}, \text{denominazione}, \text{stanzeDisponibili} \}$

Attributo	Tipo	Clausola	Vincoli
codAlbergo	INT(10)	PRIMARY KEY	codAlbergo > 0
denominazione	VARCHAR (20)	NOT NULL	-
stanzeDisponibili	INT(10)	NOT NULL	stanzeDisponibili > 0

Soggiono $\equiv \{ \underline{\text{viaggio}}, \underline{\text{albergo}}, \text{dataInizioSoggiorno}, \text{dataFineSoggiorno}, \text{numStanze} \}$

Attributo	Tipo	Clausola	Vincoli
codAlbergo	INT(10)	FOREIGN KEY	-
codViaggio	INT(10)	FOREIGN KEY	-
dataInizioSoggiorno	DATE	NOT NULL	dataInizioSoggiorno > dataInizioViaggio
dataFineSoggiorno	DATE	NOT NULL	dataFineSoggiorno > dataInizioSoggiorno
numStanze	INT(10)	NOT NULL	numStanze > 0 AND numStanze <= stanzeDisponibili

TelefoniAlberghi $\equiv \{ \underline{\text{numero}}, \text{albergo} \}$

Attributo	Tipo	Clausola	Vincoli
numero	INT UNSIGNED	PRIMARY KEY	numero > 0
albergo	INT UNSIGNED	FOREIGN KEY	codAlbergo > 0

Dipendenti $\equiv \{ \underline{\text{numContratto}}, \text{nomeDipendente}, \text{cognomeDipendente}, \text{dataNascita}, \text{tipoDipendente}, \text{tipoRetribuzione}, \text{retribuzione}, \text{qualifica}, \text{dataInizioContratto}, \text{dataFineContratto}, \underline{\text{albergo}} \}$

Attributo	Tipo	Clausola	Vincoli
numContratto	INT(10)	PRIMARY KEY	numContratto > 0
tipoDipendente	ENUM (string)	NOT NULL	tipoDipendente = 'fisso' 'fisso dirigente' 'stagionale'
nomeDipendente	VARCHAR(20)	NOT NULL	-
cognomeDipendente	VARCHAR(20)	NOT NULL	-
dataNascita	DATE	NOT NULL	-
tipoRetribuzione	ENUM (string)	NOT NULL	tipoRetribuzione = 'giornaliero' 'annuale'
retribuzione	DECIMAL(10,2)	NOT NULL	retribuzioneDipendente > 0
qualifica	VARCHAR(20)	NOT NULL	-
albergo	INT(10)	FOREIGN KEY	codAlbergo > 0
dataInizioContratto	DATE	NOT NULL	-
dataFineContratto	DATE	NOT NULL	dataFineContratto > dataInizioContratto



3

IMPLEMENTAZIONE

CREAZIONE DELLO SCHEMA DEL DATABASE

```
CREATE TABLE Viaggi (
    codViaggio INT(10) UNSIGNED NOT NULL,
    tipoViaggio VARCHAR(30) NOT NULL,
    luogoPartenza VARCHAR(30) NOT NULL,
    luogoDestinazione VARCHAR(30) NOT NULL,
    costoViaggio DECIMAL(10,2) UNSIGNED NOT NULL,
    dataInizioViaggio DATE NOT NULL,
    dataFineViaggio DATE NOT NULL,
    PRIMARY KEY (codViaggio)
);

CREATE TABLE Clienti (
    codFiscale VARCHAR(16) NOT NULL,
    nomeCliente VARCHAR(15) NOT NULL,
    cognomeCliente VARCHAR(20) NOT NULL,
    sesso ENUM('M', 'F') NOT NULL,
    telefono VARCHAR(16) NULL UNIQUE,
    indirizzo VARCHAR(30) NOT NULL,
    PRIMARY KEY (codFiscale)
);

CREATE TABLE Prenotazioni (
    viaggio INT(10) UNSIGNED NOT NULL,
    cliente VARCHAR(16) NOT NULL,
    pSconto DECIMAL(5,2) UNSIGNED NOT NULL,
    dataPrenotazione DATE NOT NULL,
    FOREIGN KEY (viaggio) REFERENCES Viaggi (codViaggio)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (cliente) REFERENCES Clienti (codFiscale)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Mezzi (
    codMezzo INT(10) UNSIGNED NOT NULL,
    descrizioneMezzo VARCHAR(30) NOT NULL,
    capienzaMezzo INT(10) UNSIGNED NOT NULL,
    viaggio INT(10) UNSIGNED DEFAULT NULL,
    costoAffitto DECIMAL(10,2) DEFAULT NULL,
    dataInizioAffitto DATE DEFAULT NULL,
    dataFineAffitto DATE DEFAULT NULL,
    PRIMARY KEY (codMezzo),
    FOREIGN KEY (viaggio) REFERENCES Viaggi (codViaggio)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

```

CREATE TABLE Alberghi (
    codAlbergo INT(10) UNSIGNED NOT NULL,
    denominazione VARCHAR(50) NOT NULL,
    stanzeDisponibili INT(10) UNSIGNED NOT NULL,
    PRIMARY KEY (codAlbergo)
);

CREATE TABLE TelefoniAlberghi (
    numero VARCHAR(16) NOT NULL,
    albergo INT(10) UNSIGNED NOT NULL,
    PRIMARY KEY (numero),
    FOREIGN KEY (albergo) REFERENCES Alberghi (codAlbergo)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Soggiorni (
    viaggio INT(11) UNSIGNED NOT NULL,
    albergo INT(11) UNSIGNED NOT NULL,
    dataInizioSoggiorno DATE NOT NULL,
    dataFineSoggiorno DATE NOT NULL,
    numStanze INT(11) UNSIGNED NOT NULL,
    FOREIGN KEY (albergo) REFERENCES Alberghi (codAlbergo)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (viaggio) REFERENCES Viaggi (codViaggio)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Dipendenti (
    numContratto INT(10) UNSIGNED NOT NULL,
    albergo INT(10) UNSIGNED NOT NULL,
    nomeDipendente VARCHAR(20) NOT NULL,
    cognomeDipendente VARCHAR(20) NOT NULL,
    dataNascita DATE NOT NULL,
    tipoDipendente ENUM('fisso', 'fisso dirigente', 'stagionale') NOT NULL,
    tipoRetribuzione ENUM('annuale', 'giornaliera') NOT NULL,
    retribuzione DECIMAL(10,2) UNSIGNED NOT NULL,
    qualifica VARCHAR(20) DEFAULT NULL,
    dataInizioContratto DATE NOT NULL,
    dataFineContratto DATE NULL DEFAULT NULL,
    PRIMARY KEY (numContratto),
    FOREIGN KEY (albergo) REFERENCES Alberghi (codAlbergo)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

```

INSERIMENTO DATI

4

TABELLA Viaggi

```

CREATE PROCEDURE insertViaggio(
    IN pCodViaggio INT(10),
    IN pTipoViaggio VARCHAR(30),
    IN pLuogoPartenza VARCHAR(30),
    IN pLuogoDestinazione VARCHAR(30),
    IN pCostoViaggio DECIMAL(10,2),
    IN pDataInizioViaggio DATE,
    IN pDataFineViaggio DATE
)
BEGIN
    INSERT INTO Viaggi(codViaggio,tipoViaggio,luogoPartenza,luogoDestinazione,costoViaggio, dataInizioViaggio,dataFineViaggio)
    VALUES(pCodViaggio,pTipoViaggio,pLuogoPartenza,pLuogoDestinazione,pCostoViaggio,pDataInizioViaggio, pDataFineViaggio);
END

CALL insertViaggio(1, 'Scrocco', 'Campobasso', 'Pesche', 0.00, '2019-01-20', '2019-02-20');

```

codViaggio	tipoViaggio	luogoPartenza	luogoDestinazione	costoViaggio	dataInizioViaggio	dataFineViaggio
1	Scrocco	Campobasso	Pesche	0.00	2019-02-20	2019-02-20
2	Crociata	Termini	Genova	400.00	2019-05-10	2019-05-20
3	Crociata	Roma	Gerusalemme	100.00	2019-12-12	2020-12-12
4	Internazionale	Roma	Londra	800.00	2019-06-07	2019-06-17
5	Internazionale	Roma	amsterdam	400.00	2019-02-10	2019-02-12
6	Relax	Campobasso	Castelpetroso	370.00	2019-01-30	2019-02-01
7	Istruzione	Campobasso	Città del Vaticano	100.00	2019-02-10	2019-02-10
8	Istruzione	Campobasso	Roma	150.00	2019-03-04	2019-03-07
9	Organizzato	Campobasso	Milano	250.00	2019-04-06	2019-04-07
10	Organizzato	Roma	Gerusalemme	1,000.00	2019-09-01	2019-09-10
11	Relax	Campobasso	Pescara	100.00	2019-03-08	2019-03-09

TABELLA Clienti

```

CREATE PROCEDURE insertCliente(
    IN pCodFiscale VARCHAR(16),
    IN pNome VARCHAR(15),
    IN pCognome VARCHAR(20),
    IN pSesso ENUM('M', 'F'),
    IN pTelefono VARCHAR(16),
    IN pIndirizzo VARCHAR(30)
)
BEGIN
    INSERT INTO Clienti(codFiscale, nomeCliente, cognomeCliente, sesso, telefono, indirizzo)
    VALUES (pCodFiscale, pNome, pCognome, pSesso, pTelefono, pIndirizzo);
END

CALL insertCliente("CRMGN98H04H926X", "Giovanni", "Ciaramella", "M", "+39 327 840 4153", "c.da Collelongo, 39A");

```

codFiscale	nomeCliente	cognomeCliente	sesso	telefono	indirizzo
CRMGN98H04H926X	Giovanni	Ciaramella	M	+39 327 840 4153	c.da Collelongo, 39A
CTNLCU72A06F839H	Luca	Catone	M	+39 328 900 2356	c.da Colle delle api, 21
DSCCRS98P05B519J	Christian	Disenza	M	+39 349 547 5475	via Monsignor Bologna, 46
FGNSFN98H10B519L	Stefano	Fagnano	M	+39 327 834 9902	via Pirandello, 21
FRNRMN70R07H501E	Francesco	Romanò	M	+39 328 458 3456	Via Geneova, 1
GLTRFRZ98L19B519N	Fabrizio	Galati	M	+39 327 869 5702	via Veneto, 20
GRZRCV97E02F839H	Grazia	Arcavolo	F	+39 329 845 2569	via Puglia, 21
MRCSS70L04F839L	Marco	Espostò	M	+39 339 1258817	via Mazzini, 34B
NNAFNC65M45B519W	Anna	Franco	F	+39 392 540 1222	via Milano, 98
NTNSST79A01F839B	Antonio	Espostò	M	+39 324 128 9120	via Ungaretti, 32
PPPPP98A00B519G	Peppe	N/D	M	+39 335 353 3463	via Roma, 100
RSSLSS68L06L219D	Alessio	Rossi	M	+39 327 285 8681	via Ungaretti, 45
RSSMRA80A01H501U	Maria	Rossi	F	+39 327 650 7000	via Roma, 13
RSSMRC98E10B519G	Marco	Russodivito	M	+39 388 166 3571	via Emilia, 21
VLLLCU94R06H501Z	Luca	Vallo	M	+39 334 567 987	via Stalla, 45

TABELLA Prenotazioni

```

PROCEDURE insertPrenotazione(
    IN pCliente VARCHAR(16),
    IN pViaggio INT(10),
    IN pPSconto DECIMAL(5,2),
    IN pDataPrenotazione INT(10)
)
BEGIN
    INSERT INTO Prenotazioni(cliente, viaggio, pSconto, dataPrenotazione)
    VALUES (pViaggio, pCliente, pPSconto, pDataPrenotazione);
END

CALL insertPrenotazione ("CRMGN98H04H926X", 1, 100.00,"2019-02-19");

```

cliente	viaggio	pSconto	dataPrenotazione
CRMGN98H04H926X	1	100.00	2019-02-19
CRMGN98H04H926X	13	100.00	2019-01-13
CTNLCU72A06F839H	14	3.00	2019-01-13
DSCCRS98P05B519J	9	14.00	2019-01-14
FGNSFN98H10B519L	4	12.00	2019-01-13
FRNRMN70R07H501E	8	0.00	2019-01-14
NTNSST79A01F839B	6	1.50	2019-01-14
PPPPP98A00B519G	2	0.00	2019-01-13
RSSMRC98E10B519G	1	1.00	2019-02-12
RSSMRC98E10B519G	9	10.00	2019-03-30

TABELLA Mezzi

```

CREATE PROCEDURE insertMezzo(
    IN pDescrizioneMezzo VARCHAR(30),
    IN pCapienzaMezzo INT(10)
)
BEGIN
    INSERT INTO Mezzi (descrizioneMezzo, capienzaMezzo)
    VALUES (pDescrizioneMezzo, pCapienzaMezzo);
END

CREATE PROCEDURE affittaMezzo(
    IN pMezzo INT(10),
    IN pViaggio INT(10),
    IN pCostoAffitto DECIMAL(10,2),
    IN pDataInizioAffitto DATE,
    IN pDataFineAffitto DATE
)
BEGIN
    SET @numMezzi = (SELECT COUNT(*) FROM Mezzi WHERE codMezzo = pMezzo);
    SET @numViaggi = (SELECT COUNT(*) FROM Viaggi WHERE codViaggio = pViaggio);

    IF (@numMezzi = 1 AND @numViaggi = 1) THEN
        UPDATE Mezzi
        SET costoAffitto = pCostoAffitto, dataInizioAffitto = pDataInizioAffitto, dataFineAffitto = pDataFineAffitto, viaggio = pViaggio
        WHERE codMezzo = pMezzo;
    ELSE
        SIGNAL SQLSTATE '45000' SET message_text = 'Mezzo non trovato';
    END IF;
END

CALL insertMezzo("ciarashuttle", 2);
CALL affittaMezzo(2, 4, 99.00, "2019-06-07", "2019-06-17");

```

codMezzo	descrizioneMezzo	capienzaMezzo	viaggio	costoAffitto	dataInizioAffitto	dataFineAffitto
1	ciarashuttle	2	(NULL)	(NULL)	(NULL)	(NULL)
2	boeing-747	524	4	99.00	2019-06-07	2019-06-17
3	costa-minchia	3,000	13	300.00	2019-08-10	2019-08-24
4	veliero	470	3	100.00	2019-12-12	2020-12-12
5	pullman	55	8	999.99	2019-03-04	2019-03-07
6	lancia DELTA	5	(NULL)	(NULL)	(NULL)	(NULL)
7	pullman	55	7	350.00	2019-02-10	2019-02-10
8	pullman	55	9	650.00	2019-04-06	2019-04-06
9	boeing-747	524	5	90.00	2019-02-10	2019-02-12
10	minivan	9	11	190.00	2019-03-08	2019-03-09
11	doblò DICAPOBIANCO	5	(NULL)	(NULL)	(NULL)	(NULL)
12	taxi	4	13	300.00	2019-01-01	2019-01-02
13	boeing-747	524	10	120.00	2019-09-01	2019-09-10
14	alfa MITO	4	(NULL)	(NULL)	(NULL)	(NULL)
15	VW POLO	5	(NULL)	(NULL)	(NULL)	(NULL)
16	VW GOLF	5	(NULL)	(NULL)	(NULL)	(NULL)
17	audi a3 SPORTBACK	5	(NULL)	(NULL)	(NULL)	(NULL)

TABELLA Alberghi E TelefoniAlberghi

```

CREATE PROCEDURE insertAlbergo(
    IN pCodAlbergo INT(10),
    IN pDenominazione VARCHAR(50),
    IN pStanzeDisponibili INT(10)
)
BEGIN
    INSERT INTO Alberghi (codAlbergo, denominazione, stanzeDisponibili)
    VALUES (pCodAlbergo, pDenominazione, pStanzeDisponibili);
END

PROCEDURE insertTelefono(
    IN pNumero VARCHAR(16),
    IN pAlbergo INT(10)
)
BEGIN
    INSERT INTO TelefoniAlberghi(numero, albergo) VALUES (pNumero, pAlbergo);
END

CALL InsertAlbergo(7,"La Pecora - Alghero", 69);
CALL insertTelefono("+39 0875 705246", 86);

```

codAlbergo	denominazione	stanzeDisponibili
7	La Pecora - Alghero	69
18	BestWestern - Amsterdam	174
33	Rinascimento - Campobasso	30
54	Centrum Palace - Campobasso	50
67	Meridiano - Termoli	20
86	Mistral - Termoli	30
94	Hotel Bristol Palace - Genova	23
347	La Fonte del Astore - Castelpetroso	35
356	Hilton Times Square - New York	70
463	Corus Hotel Hyde Park - Londra	346
564	i31 Hotel - Berlino	72
578	Grand Hotel Europa - Isernia	40
579	La Costa - Cagliari	89
648	The Sukhothai - Shanghai	100
689	Villa Ba Moshava - Gerusalemme	84

numero	albergo
+39 0874 481455	33
+39 0874 484931	33
+39 0874 413341	54
+39 0874 413342	54
+39 0875 702696	67
+39 0875 705946	67
+39 0875 705220	86
+39 0875 705246	86

TABELLA Soggiorni

```

PROCEDURE insertSoggiorno(
    IN pViaggio INT(10),
    IN pAlbergo INT(10),
    IN pDataInizioSoggiorno DATE,
    IN pDataFineSoggiorno DATE,
    IN pNumStanze INT(10)
)
BEGIN
    INSERT INTO Soggiorni(viaggio, albergo, dataInizioSoggiorno, dataFineSoggiorno, numStanze)
    VALUES (pViaggio, pAlbergo, pDataInizioSoggiorno, pDataFineSoggiorno, pNumStanze);
END

CALL insertSoggiorno(2, 7, "2019-08-10", "2019-08-24", 3);

```

viaggio	albergo	dataInizioSoggiorno	dataFineSoggiorno	numStanze
2	7	2019-08-10	2019-08-24	3
2	7	2019-08-10	2019-08-24	6
4	463	2019-06-07	2019-06-17	1
5	18	2019-02-10	2019-02-12	2
6	347	2019-01-30	2019-02-01	1
11	2.335	2019-03-08	2019-03-09	3
8	75.647	2019-03-04	2019-03-07	20
3	579	2019-07-01	2019-07-01	2
2	7	2019-08-10	2019-08-24	1
4	463	2019-06-07	2019-06-17	3

TABELLA Dipendenti

```

CREATE PROCEDURE insertDipendenteFisso(
    IN pAlbergo INT(10),
    IN pNome VARCHAR(20),
    IN pCognome VARCHAR(20),
    IN pDataNascita DATE,
    IN pRetribuzione DECIMAL(10,2),
    IN pQualifica VARCHAR(50),
    IN pDataInizioContratto DATE
)
BEGIN
    SET @nCodContratto = (SELECT MAX(numContratto)+1 FROM Dipendenti WHERE albergo = pAlbergo);
    INSERT INTO Dipendenti(numContratto, albergo, nomeDipendente, cognomeDipendente, dataNascita, tipoDipendente,
                           tipoRetribuzione, retribuzione, qualifica, dataInizioContratto)
    VALUES (@nCodContratto,pAlbergo,pNome,pCognome,pDataNascita, 'fisso', 'annuale',pRetribuzione, pQualifica, pDataInizioContratto);
END

CREATE PROCEDURE insertDipendenteFissoDirigente(
    IN pAlbergo INT(10),
    IN pNome VARCHAR(20),
    IN pCognome VARCHAR(20),
    IN pDataNascita DATE,
    IN pRetribuzione DECIMAL(10,2),
    IN pQualifica VARCHAR(50),
    IN pDataInizioContratto DATE
)
BEGIN
    SET @nCodContratto = (SELECT MAX(numContratto)+1 FROM Dipendenti WHERE albergo = pAlbergo);
    INSERT INTO Dipendenti(numContratto, albergo, nomeDipendente, cognomeDipendente, dataNascita, tipoDipendente,
                           tipoRetribuzione, retribuzione, qualifica, dataInizioContratto)
    VALUES (@nCodContratto,pAlbergo,pNome,pCognome,pDataNascita, 'fisso dirigente','annuale',pRetribuzione, pQualifica, pDataInizioContratto);
END

CREATE PROCEDURE insertDipendenteStagionale(
    IN pAlbergo INT(10),
    IN pNome VARCHAR(20),
    IN pCognome VARCHAR(20),
    IN pDataNascita DATE,
    IN pRetribuzione DECIMAL(10,2),
    IN pDataInizioContratto DATE,
    IN pDataFineContratto DATE
)
BEGIN
    SET @nCodContratto = (SELECT MAX(numContratto)+1 FROM Dipendenti WHERE albergo = pAlbergo);
    INSERT INTO Dipendenti(numContratto, albergo, nomeDipendente, cognomeDipendente, dataNascita, tipoDipendente,
                           tipoRetribuzione, retribuzione, dataInizioContratto, dataFineContratto)
    VALUES (@nCodContratto,pAlbergo,pNome,pCognome,pDataNascita, 'stagionale', 'giornaliera',pRetribuzione,pDataInizioContratto,pDataFineContratto);
END

CALL insertDipendeteFisso(33, "Carmine", "Abate", "1978-06-27", 1400.20, "lavapiatti", "1990-12-12");
CALL insertDipendeteFissoDirigente(33, "Cristina", "Negrone", "1967-11-07", 2700.50, "Direttore albergo", "1995-10-30");
CALL insertDipendenteStagionale(33, "Moira", "Rodd", "1963-11-24", 50.00, "2018-10-02", "2019-03-11");

```

numContratto	albergo	nomeDipendente	cognomeDipendente	dataNascita	tipoDipendente	tipoRetribuzione	retribuzione	qualifica	dataInizioContratto	dataFineContratto
1	33	Carmine	Abate	1978-06-27	fisso	annuale	1,400.20	lavapiatti	1990-12-12	(NULL)
2	33	Federico	Distante	1991-06-17	fisso	annuale	1,450.36	dj	2005-09-01	(NULL)
3	33	Cristina	Negrone	1967-11-07	fisso dirigente	annuale	2,700.50	Direttore albergo	1995-10-30	(NULL)
4	33	Maria	Nicolino	1968-02-07	fisso dirigente	annuale	2,400.40	Direttore ristorante	1996-05-14	(NULL)
5	33	Domenico	Nunziata	1968-06-04	fisso dirigente	annuale	2,500.34	Direttore amministrativo	1996-05-15	(NULL)
6	33	Roberta	Pacileo	1968-10-10	fisso dirigente	annuale	1,900.78	Capo ufficio contabile	1996-08-27	(NULL)
7	33	Moira	Rodd	1963-11-24	stagionale	giornaliera	50.00	(NULL)	2018-10-02	2019-03-11
8	33	Tana	Cushing	1959-03-21	stagionale	giornaliera	50.00	(NULL)	2019-01-22	2019-12-10
1	54	Michele	Aiello	1979-01-10	fisso	annuale	1,400.20	lavapiatti	1991-02-18	(NULL)
2	54	Valerio	Donnarumma	1993-02-01	fisso	annuale	1,450.36	dj	2007-02-14	(NULL)
3	54	Armando	Pagliara	1968-10-31	fisso dirigente	annuale	2,700.50	Direttore albergo	1996-11-20	(NULL)
4	54	Diego	Pesce	1969-01-28	fisso dirigente	annuale	2,400.40	Direttore ristorante	1997-04-30	(NULL)
5	54	Alessandro	Russo	1969-09-03	fisso dirigente	annuale	2,500.34	Direttore amministrativo	1998-12-17	(NULL)



TRIGGER, STORED PROCEDURE E QUERY

TRIGGER

Trigger Viaggi

```

CREATE TRIGGER Trigger_Viaggi BEFORE INSERT OR UPDATE
  ON Viaggi FOR EACH ROW
BEGIN
  IF ( NEW.codViaggio <= 0 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire un codice viaggio positivo';
  END IF;

  IF ( NEW.dataInizioViaggio < CURRENT_DATE() OR NEW.dataFineViaggio < NEW.dataInizioViaggio ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Date viaggio errate';
  END IF;

  IF ( NEW.costoViaggio < 0 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire un prezzo positivo';
  END IF;
END

```

Trigger Clienti

```

CREATE TRIGGER Trigger_Clienti BEFORE INSERT OR UPDATE
  ON Clienti FOR EACH ROW
BEGIN
  IF ( CHAR_LENGTH(NEW.codFiscale) <> 16) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Codice Fiscale errato';
  END IF;
END

```

Trigger Prenotazioni

```

CREATE TRIGGER Trigger_Prenotazioni BEFORE INSERT OR UPDATE
  ON Prenotazioni FOR EACH ROW
BEGIN
  SET @dataViaggio = (SELECT dataInizioViaggio FROM Viaggi WHERE codViaggio = NEW.viaggio);

  IF ( @dataViaggio < new.dataPrenotazione ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Prenotazione non effettuabile successivamente alla data del viaggio';
  END IF;

  IF ( NEW.pSconto < 0 OR NEW.pSconto > 100 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Sconto errato';
  END IF;
END

```

Trigger Mezzi

```

CREATE TRIGGER Trigger_Mezzi BEFORE INSERT OR UPDATE
  ON Mezzi FOR EACH ROW
BEGIN
  IF (NEW.capienzaMezzo <= 0) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire capienza positiva';
  END IF;

  IF ( NEW.dataInizioAffitto < CURRENT_DATE() OR NEW.dataFineAffitto < NEW.dataInizioAffitto ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Date affitto errate';
  END IF;

  IF ( NEW.costoAffitto < 0 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire prezzo positivo';
  END IF;
END

```

Trigger Alberghi

```
CREATE TRIGGER Trigger_Alberghi BEFORE INSERT OR UPDATE
  ON Alberghi FOR EACH ROW
BEGIN
  IF ( NEW.codAlbergo <= 0 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire un codice albergo positivo';
  END IF;

  IF ( NEW.stanzeDisponibili < 0 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire un numero di stanze positivo';
  END IF;
END
```

Trigger TelefoniAlberghi

```
CREATE TRIGGER Trigger_TelefoniAlberghi BEFORE INSERT OR UPDATE
  ON TelefoniAlberghi FOR EACH ROW
BEGIN
  IF ( CHAR_LENGTH(NEW.numero) >16 ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire un numero massimo 16 cifre';
  END IF;
END
```

Trigger Soggiorni

```
CREATE TRIGGER Trigger_Soggiorni BEFORE INSERT OR UPDATE
  ON Soggiorni FOR EACH ROW
BEGIN
  SET @disponibili = (SELECT stanzeDisponibili FROM Alberghi WHERE codAlbergo=NEW.albergo);

  IF ( @disponibili < new.numStanze ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Numero di stanze richieste non disponibili.';
  END IF;

  IF ( NEW.dataInizioSoggiorno < CURRENT_DATE() OR NEW.dataFineSoggiorno < NEW.dataInizioSoggiorno) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: date soggiorno errate';
  END IF;
END
```

Trigger Dipendenti

```
CREATE TRIGGER Trigger_Dipendenti BEFORE INSERT OR UPDATE
  ON Dipendenti FOR EACH ROW
BEGIN
  IF ( NEW.retribuzione < 0) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: Inserire retribuzione positiva.';
  END IF;

  IF ( NEW.dataInizioContratto < NEW.dataFineContratto ) THEN
    SIGNAL SQLSTATE '45000'
    SET message_text = 'Errore Inserimento: date contratto errate.';
  END IF;
END
```

STORED PROCEDURE

getViaggi()

```
CREATE PROCEDURE getViaggi()
BEGIN
    SELECT * FROM Viaggi;
END
```

getClienti()

```
CREATE PROCEDURE getClienti()
BEGIN
    SELECT * FROM Clienti;
END
```

getPrenotazioni()

```
CREATE PROCEDURE getPrenotazioni()
BEGIN
    SELECT * FROM Prenotazioni;
END
```

getMezzi()

```
CREATE PROCEDURE getMezzi()
BEGIN
    SELECT * FROM Mezzi;
END
```

getMezzi()

```
CREATE PROCEDURE getMezzi()
BEGIN
    SELECT * FROM Mezzi;
END
```

getAlberghi()

```
CREATE PROCEDURE getAlberghi()
BEGIN
    SELECT * FROM Alberghi;
END
```

getTelefoniAlbergo()

```
CREATE PROCEDURE getTelefoniAlbergo()
BEGIN
    SELECT * FROM TelefoniAlbergo;
END
```

getSoggiorni()

```
CREATE PROCEDURE getSoggiorni()
BEGIN
    SELECT * FROM Soggiorni;
END
```

getDipendenti()

```
CREATE PROCEDURE getSoggiorni()
BEGIN
    SELECT * FROM Dipendentii;
END
```

deleteViaggio()

```
CREATE PROCEDURE delteViaggio (IN pCodViaggio INT(10))
BEGIN
    DELETE FROM Viaggi WHERE Viaggi.codViaggio = pCodViaggio;
END
```

deleteCliente()

```
CREATE PROCEDURE delteCliente (IN pCodFiscale VARCHAR(50))
BEGIN
    DELETE FROM Clienti WHERE Clienti.codFiscale = pCodFiscale;
END
```

deleteMezzo()

```
CREATE PROCEDURE delteMezzo (IN pCodMezzo INT(10))
BEGIN
    DELETE FROM Mezzi WHERE Mezzi.codMezzo = pCodMezzo;
END
```

deleteAlbergo()

```
CREATE PROCEDURE delteAlbergo (IN pCodAlbergo INT(10))
BEGIN
    DELETE FROM Alberghi WHERE Alberghi.codAlbergo = pCodAlbergo;
END
```

Query Giovanni Ciaramella – 158937

```
SELECT * FROM Mezzi AS M  
WHERE costoAffitto >= '99';
```

$\sigma_{costoAffitto>99}(Mezzi)$

Vengono visualizzati tutti i mezzi con costo superiore a 99 euro

```
SELECT * FROM Clienti  
WHERE nomeCliente LIKE 'Luc%' AND sesso = 'M';
```

$\sigma_{nomeCliente=Luc}(Clienti) \cap \sigma_{sesso=M}(Clienti)$

Vengono visualizzati i clienti dove il loro nome inizia per Luc e di sesso maschile

```
SELECT codFiscale AS codicefiscale, nomeCliente AS nomecliente FROM Clienti;
```

$\pi_{codFiscale \leftarrow codicefiscale, nomeCliente \leftarrow nomecliente}(Alberghi)$

Vengono visualizzati i codici fiscali e i nomi di tutti i clienti

```
SELECT * FROM Clienti WHERE nomeCliente LIKE '%ano%'
```

UNION

```
SELECT * FROM Clienti WHERE sesso = 'F';
```

$\sigma_{nomeCliente=%ano%}(Clienti) \cup \sigma_{sesso=F}(Clienti)$

Vengono visualizzati i clienti che nel nome hanno "ano" e vengono uniti alle clienti di sesso femminile

```
SELECT Alberghi.codAlbergo, Dipendenti.nomeDipendente, Dipendenti.cognomeDipendente FROM Alberghi  
CROSS JOIN Dipendenti  
ORDER BY Alberghi.codAlbergo;
```

$\pi_{codAlbergo}(Alberghi) \times \pi_{nomeDipendente, cognomeDipendente}(Dipendenti)$

Vengono visualizzati nome e cognome di tutti i dipendenti di tutti gli alberghi ordinati per codice albergo.

```
SELECT tipoViaggio AS TV, costoViaggio AS CV FROM Viaggi  
ORDER BY dataInizioViaggio;
```

$\pi_{tipoViaggio \leftarrow TV, costoViaggio \leftarrow CV}(Viaggi)$

Rinomino gli attributi nella proiezioni della relazione Viaggi

```
SELECT nomeDipendente, cognomeDipendente, qualifica FROM Dipendenti  
WHERE tipoDipendente LIKE '%fisso%';
```

$\sigma_{tipoDipendente=%fisso%}(\pi_{nomeDipendente, cognomeDipendente, qualifica}(Dipendenti))$

Vengono visualizzate le informazioni dei dipendenti fissi e fissi dirigenti

```
SELECT * FROM Alberghi
```

WHERE denominazione IN

```
(SELECT denominazione FROM Alberghi  
WHERE stanzeDisponibili = '69');
```

Vengono visualizzate le informazioni dei dipendenti fissi e fissi dirigenti

```
SELECT nomeDipendente, cognomeDipendente, retribuzione FROM Dipendenti  
WHERE retribuzione IN
```

```
(SELECT retribuzione FROM Alberghi  
WHERE retribuzione < 1000);
```

Vengono visualizzate le informazioni dei dipendenti la cui retribuzione è minore di 1000

```
SELECT cognomeDipendente FROM Dipendenti
```

WHERE retribuzione IN

```
(SELECT MAX(retribuzione) FROM Dipendenti);
```

Vengono visualizzati i dipendenti che hanno massima retribuzione

Query Christian Discenza – 160198

```
SELECT M.codMezzo, M.descrizioneMezzo, M.capienzaMezzo FROM Mezzi AS M;
```

$$M = \pi_{codMezzo, descrizioneMezzo, capienzaMezzo}(Mezzi)$$

Vengono visualizzati il codice, la descrizione e la capienza dei mezzi (proiezione)

```
SELECT M.codMezzo, M.descrizioneMezzo, M.capienzaMezzo, M.costoAffitto FROM Mezzi AS M  
WHERE M.costoAffitto >= '1000';
```

$$M = \sigma_{costoAffitto \geq 1000} (\pi_{codMezzo, descrizioneMezzo, capienzaMezzo, costoAffitto}(Mezzi))$$

Vengono visualizzati il codice, la descrizione, la capienza dei mezzi il cui affitto è maggiore di 1000

```
SELECT * FROM Clienti AS C  
WHERE C.sesso = 'M';
```

$$C = \sigma_{sesso=M}(Clienti)$$

Vengono visualizzati i clienti maschi

```
SELECT *  
FROM Viaggi AS V  
WHERE V.luogoPartenza = 'Roma';
```

$$V = \sigma_{luogoPartenza=Roma}(Viaggi)$$

Vengono visualizzati i viaggi in partenza da Roma

```
SELECT S.viaggio AS CODICE_VIAGGIO, A.denominazione AS ALBERGO, S.numStanze AS  
NUMERO_STANZE_PRENOTATE, P.cliente AS CLIENTE FROM Soggiorni AS S  
JOIN Alberghi AS A ON S.albergo = A.codAlbergo  
JOIN Prenotazioni AS P ON S.viaggio = P.viaggio;  
WHERE P.cliente = _variabilecodicefisclepassatadaphp_;
```

$$S = \pi_{CODICE_VIAGGIO \leftarrow viaggio, COGNOME \leftarrow tipoViaggio, NUMERO_STANZE_PRENOTATE \leftarrow numStanze}(Soggiorni)$$

$$A = \pi_{ALBERGO \leftarrow denominazione}(Alberghi) \quad P = \pi_{CLIENTE \leftarrow cliente}(Prenotazioni)$$

$$\sigma_{cliente=?} (S \bowtie_{albergo=codAlbergo} A \bowtie_{viaggio=viaggio} P)$$

Vengono visualizzati codice viaggio, nome albergo e le stanze prenotate ed il cliente che ha prenotato

```
SELECT D.nomeDipendente AS NOME, D.cognomeDipendente AS COGNOME, D.qualifica AS  
QUALIFICA, A.codAlbergo AS ALBERGO FROM Dipendenti AS D  
JOIN Alberghi AS A ON D.albergo = A.codAlbergo  
WHERE A.codAlbergo = _codicealbergopassatodaPHP_;
```

$$D = \pi_{NOME \leftarrow codViaggio, COGNOME \leftarrow tipoViaggio, QUALIFICA \leftarrow qualifica}(Dipendenti) \quad A = \pi_{ALBERGO \leftarrow codAlbergo}(Alberghi)$$

$$\sigma_{codAlbergo=?} (D \bowtie_{albergo=codAlbergo} A)$$

Viene visualizzato nome, cognome, qualifica ricoperta ed il codice dell'albergo dei dipendenti

```
SELECT V.codViaggio AS VIAGGIO, V.tipoViaggio AS TIPOLOGIA  
FROM Viaggi AS V  
WHERE V.luogoPartenza = _luogopartenzapassatodaPHP_;
```

$$\sigma_{luogoPartenza=?} (\pi_{VIAGGIO \leftarrow codViaggio, TIPOLOGIA \leftarrow tipoViaggio}(Viaggi))$$

Viene visualizzato il codice del viaggio e la tipologia di viaggio

```
SELECT * FROM Prenotazioni
```

```
WHERE viaggio IN
```

```
(SELECT codViaggio FROM Viaggi  
WHERE tipoViaggio='Scrocco');
```

Visualizza le prenotazioni dei viaggi relativo al tipo Viaggio "scrocco"

```
SELECT * FROM Mezzi
```

```
WHERE viaggio IN
```

```
(SELECT codViaggio FROM Viaggi  
WHERE tipoViaggio='Relax');
```

visualizza tutti i dati della tabella Mezzi dove il codice del viaggio corrisponde al codice della tabella viaggi relativo al tipo Viaggio "relax"

```
SELECT * FROM Dipendenti AS D
```

```
WHERE D.retribuzione > 300 AND D.albergo IN
```

```
(SELECT codAlbergo FROM Alberghi  
WHERE Alberghi.denominazione='Hilton Times Square - New York');
```

Vengono visualizzati tutti i dati relativi alla tabella dipendenti, dove la retribuzione è maggiore di 300 e il codice dell'albergo corrisponde a Hilton Times Square - New York

Query Stefano Fagnano – 158985

```
SELECT MIN(Alberghi.stanzeDisponibili) FROM Alberghi  
WHERE Alberghi.denominazione like 'a%';
```

$$\sigma_{denominazione} = a\% \left(\pi_{min(stanzeDisponibili)}(Mezzi) \right)$$

Visualizza il numero minimo di stanze disponibili negli alberghi il cui nome inizia con la A

```
SELECT Viaggi.codViaggio AS 'Codice Viaggio', Mezzi.codMezzo AS 'Codice Mezzo'  
FROM Viaggi CROSS JOIN Mezzi;
```

$$\pi_{codViaggio \leftarrow Codice Viaggio}(Viaggi) \times \pi_{codMezzo \leftarrow Codice Mezzo}(Mezzi)$$

Vengono visualizzate tutte le possibili combinazioni di viaggio e mezzo

```
SELECT count(*) as 'Mezzi disponibili' FROM Mezzi  
WHERE Mezzi.capienzaMezzo >= 5;
```

$$\sigma_{capienzaMezzo \geq 5} \left(\pi_{count(*) \leftarrow Mezzi disponibili}(Mezzi) \right)$$

Conta i mezzi che hanno capienza massima maggiore o uguale a 5.

```
SELECT AVG(Mezzi.costoAffitto) FROM Mezzi;
```

$$\pi_{AVG(costoAffitto)}(Mezzi)$$

Visualizza la media del costo di affitto dei mezzi.

```
(SELECT* FROM Prenotazioni WHERE cliente = 'FGNSFN98H10B519L')  
UNION
```

```
(SELECT* FROM Prenotazioni WHERE cliente = 'FRNRMN70R07H501E');
```

$$\sigma_{cliente=FGMSF98H10B519L}(Prenotazioni) \cup \sigma_{cliente=FRNRMN70R07H501}(Prenotazioni)$$

Vengono visualizzate le prenotazioni dei clienti con codice fiscale 'FRNRMN70R07H501E' e 'FGNSFN98H10B519L'

```
SELECT Mezzi.descrizioneMezzo, Viaggi.luogoPartenza, Viaggi.luogoDestinazione FROM Viaggi  
JOIN Mezzi ON Viaggi.codViaggio = Mezzi.viaggio  
ORDER BY Mezzi.descrizioneMezzo;
```

$$\pi_{luogoPartenza, luogoDestinazione \leftarrow codViaggio = viaggio} \pi_{descrizioneMezzo}(Mezzi)$$

Visualizza i mezzi e i luoghi di partenza e di destinazione di un viaggio ordinandoli in base al mezzo.

```
SELECT * FROM Dipendenti WHERE tipoDipendente = 'fisso' AND dataInizioContratto LIKE '%-%-%';
```

$$\sigma_{dataInizioContratto = \%-\%- \%}(Dipendenti) \cap \sigma_{tipoDipendente = fisso}(Dipendenti)$$

Visualizza i dipendenti fissi assunti in una determinata data

```
SELECT TelefoniAlberghi.numero AS 'Numero Albergo'
```

```
FROM TelefoniAlberghi
```

```
WHERE TelefoniAlberghi.albergo IN
```

```
    (SELECT Alberghi.codAlbergo FROM Alberghi  
     INNER JOIN Soggiorni ON Soggiorni.albergo = Alberghi.codAlbergo  
     INNER JOIN Viaggi ON Soggiorni.viaggio = Viaggi.codViaggio  
     INNER JOIN Mezzi ON Mezzi.viaggio  
     WHERE Mezzi.costoAffitto > 30 );
```

Restituisce i recapiti degli alberghi visitati dai clienti utilizzanti mezzi di capienza maggiore 30.

```
SELECT Soggiorni.dataInizioSoggiorno AS 'CHECK-IN Albergo' FROM Soggiorni  
WHERE Soggiorni.viaggio IN
```

```
    (SELECT Viaggi.codViaggio FROM Viaggi
```

```
     WHERE Viaggi.luogoPartenza = 'Campobasso' AND Viaggi.luogoDestinazione = 'Alghero');
```

Visualizza le date del check-in nell'albergo e della prenotazione del mezzo

```
SELECT codMezzo AS 'Codice Mezzo', descrizioneMezzo AS 'Descrizione Mezzo',  
capienzaMezzo AS 'Capienza Mezzo', costoAffitto AS 'Costo Affitto', dataInizioAffitto  
AS 'Data Inizio Affitto', dataFineAffitto AS 'Data Fine Affitto'
```

```
FROM Mezzi
```

```
WHERE viaggio IN
```

```
    (SELECT viaggio FROM Prenotazioni
```

```
     WHERE cliente IN
```

```
        (SELECT codFiscale FROM Clienti
```

```
         WHERE nomeCliente = 'nome cliente'));
```

Tutti i mezzi utilizzati da un cliente in un viaggio

Query Marco Russodivito – 158940

```
SELECT * FROM Alberghi  
WHERE stanzeDisponibili > 50;
```

$\sigma_{stanzeDisponibili>50}(Alberghi)$

Vengono visualizzati gli alberghi con almeno 50 stanzeDisponibili (selezione)

```
SELECT denominazione AS nome, stanzeDisponibili AS stanze FROM Alberghi;
```

$\pi_{nome \leftarrow denominazione, stanze \leftarrow StanzeDisponibili}(Alberghi)$

Vengono visualizzati i nomi e le stanze di tutti gli alberghi (proiezione e ridenominazione)

```
SELECT Viaggi.codViaggio AS viaggio, Clienti.codFiscale AS cliente  
FROM Viaggi CROSS JOIN Clienti;
```

$\pi_{codViaggio \leftarrow viaggio}(Viaggi) \times \pi_{codFiscale \leftarrow cliente}(Clienti)$

Vengono visualizzate tutte le possibili combinazioni di viaggio-cliente
(proiezione e ridenominazione, prodotto cartesiano)

```
(SELECT * FROM Dipendenti WHERE albergo = '33')  
UNION  
(SELECT * FROM Dipendenti WHERE albergo = '94');
```

$\sigma_{albergo=33}(Dipendenti) \cup \sigma_{albergo=94}(Dipendenti)$

Vengono visualizzati i dipendenti degli alberghi 33 e 94 (selezione e ridenominazione, unione)

```
SELECT * FROM Viaggi  
WHERE luogoPartenza = <> 9;
```

$Prenotazioni - \sigma_{viaggio=9}(Prenotazioni)$

Vengono visualizzate le prenotazioni non del viaggio 9 (Differenza)

```
SELECT * FROM Viaggi  
WHERE luogoPartenza = 'Campobasso' AND luogoDestinazione = 'Pesche';
```

$\sigma_{luogoPartenza=Campobasso}(Viaggi) \cap \sigma_{luogoDestinazione=Pesche}(Viaggi)$

Vengono visualizzati i viaggi che partono da campobasso e arrivano a pesche (intersezione)

```
SELECT Alberghi.denominazione, Dipendenti.cognomeDipendente  
FROM Dipendenti  
LEFT JOIN Alberghi ON Alberghi.codAlbergo = Dipendenti.albergo  
ORDER BY Alberghi.denominazione;
```

$\pi_{cognome}(Dipendenti) \bowtie_{left codAlbergo=albergo} \pi_{denominazione}(Alberghi)$

Vengono visualizzati i dipendenti e il loro albergo di appartenenza (left outer join)

```
SELECT * FROM Viaggi  
WHERE codViaggio IN  
(SELECT viaggio FROM Prenotazioni  
WHERE cliente = 'CRMGN98H04H926X') AND codViaggio IN  
(SELECT viaggio FROM Mezzi  
WHERE descrizioneMezzo = 'Navetta Unimol');
```

Vengono visualizzati i viaggi prenotati da cliente CRMGN98H04H926X che sfruttano la navetta Unimol

```
SELECT numero FROM TelefoniAlberghi  
WHERE TelefoniAlberghi.albergo IN  
(SELECT Alberghi.codAlbergo FROM Alberghi  
INNER JOIN Soggiorni ON Soggiorni.albergo = Alberghi.codAlbergo  
WHERE Soggiorni.viaggio = 4);  
Vengono visualizzati tutti i numeri di telefono degli alberghi relativi al viaggio 4
```

```
SELECT Dipendenti.nomeDipendente, Dipendenti.cognomeDipendente FROM Dipendenti  
WHERE Dipendenti.tipoDipendente = 'stagionale' AND Dipendenti.albergo IN  
(SELECT Alberghi.codAlbergo FROM Alberghi  
INNER JOIN Soggiorni ON Soggiorni.albergo = Alberghi.codAlbergo  
INNER JOIN Viaggi ON Soggiorni.viaggio = Viaggi.codViaggio  
INNER JOIN Prenotazioni ON Viaggi.codViaggio = Prenotazioni.viaggio  
WHERE Prenotazioni.cliente = 'RSSMRC98E10B519G');
```

Vengono visualizzati tutti i dipendenti stagionali degli alberghi visistati dal cliente RSSMRC98E10B519G



6

SITO WEB

The screenshot shows the homepage of the ScroKING travel website. At the top, there's a navigation bar with links for Home, Viaggi, Clienti, Mezzi, Hotel, About, and Contact. Below the navigation is a large banner with a teal gradient background. The banner features the text "Tutti meritano una vacanza. Ora tocca a te!" and "VIAGGIA CON NOI". It also includes a subtext "Trova le offerte migliori nelle nostre agenzie e prenota a prezzi bassissimi." Below the banner, there's a section titled "Le tre destinazioni più gettonate:" showing three images: the Colosseum in Rome, the Palace of Westminster in London, and the New York City skyline at night. To the right of these images is a photograph of a winding road through a green, hilly landscape. At the bottom of the page, there's another navigation bar with links for Home, Viaggi, Clienti, Mezzi, Hotel, About, Contact, and a copyright notice "© Copyright all rights reserved".

The screenshot shows the "About us" page of the ScroKING website. The page has a dark header with the ScroKING logo and navigation links for Home, Viaggi, Clienti, Mezzi, Hotel, About, and Contact. Below the header, there's a section titled "About us" with a horizontal line underneath. On the left, there's a graphic featuring the word "ScroKING" in yellow, "M S C C" in white, and four small profile pictures of people. Below this graphic is the word "Relationship". To the right of the graphic, there's a paragraph of text: "Siamo un gruppo di amici universitari che si sono divertiti a creare questo sito web quasi per gioco per poi arrivarci nel progetto dell'esame di base di dati. La nostra unione non deriva da anni ed anni di amicizia, il gruppo si è formato circa un anno fa, ma dalla passione per l'informatica e la stessa voglia di vivere e scherzare." Below this paragraph is another section titled "About the project" with a horizontal line underneath. At the very bottom of the page, there's a footer with a dark background, containing the text "Il progetto denominato "Scroking" prende il nome dalla grande professionalità di un componente del gruppo di riuscire a non pagare, nel", the ScroKING logo, and a copyright notice "© Copyright all rights reserved".

```

<?php

class Database extends PDO {
    //database credentials
    const HOST = "localhost";
    const DB_NAME = "scroKING";
    const USERNAME = "marco";
    const PASSWORD = "marcodb";

    /**
     * Database constructor.
     */
    public function __construct() {
        try {
            parent::__construct("mysql:host=" . Database::HOST . ";dbname=" . Database::DB_NAME,
Database::USERNAME, Database::PASSWORD);
            $this->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
        } catch(PDOException $exception){
            throw $exception;
        }
    }

    /**
     * @param $query : the query to execute
     * @param $params : the parameters to bind
     * @return array|bool : an array with the result of the query or FALSE if no result
     * @throws Exception : in case something went wrong
     */
    public function runSelectQuery($query, $params) {
        try {
            $executedStatement = $this->executeQuery($query, $params);

            if($executedStatement->rowCount() != 0) {
                if($executedStatement->rowCount() == 1) {
                    return $executedStatement->fetch(self::FETCH_ASSOC);
                } else { //fetch all result and push in an array
                    $resultSet = array();
                    while ($row = $executedStatement->fetch(self::FETCH_ASSOC)) {
                        array_push($resultSet, $row);
                    }
                    return $resultSet; //return the result set
                }
            } else {
                return false;
            }
        } catch ( Exception $e) {
            throw new Exception($e->getMessage());
        }
    }

    /**
     * @param $query : the query to execute
     * @param $params : the parameters to bind
     * @return bool : true if insert, false otherwise
     * @throws Exception : in case something went wrong
     */
    public function runInsertQuery($query, $params) {
        try {
            $executedStatement = $this->executeQuery($query, $params);
            return ($executedStatement->rowCount() == 1);
        } catch ( Exception $e) {
            throw new Exception($e->getMessage());
        }
    }
}

```

```

    /**
     * @param $query : the query to execute
     * @param $params : the parameters to bind
     * @return bool : true if update, false otherwise
     * @throws Exception : in case something went wrong
     */
    public function runUpdateQuery($query, $params) {
        try {
            $executedStatement = $this->executeQuery($query, $params);
            return ($executedStatement->rowCount() == 1);
        } catch (Exception $e) {
            throw new Exception($e->getMessage());
        }
    }

    /**
     * @param $query : the query to execute
     * @param $params : the parameters to bind
     * @return bool : true if delete, false otherwise
     * @throws Exception : in case something went wrong
     */
    public function runDeleteQuery($query, $params) {
        try {
            $executedStatement = $this->executeQuery($query, $params);
            return ($executedStatement->rowCount() == 1);
        } catch (Exception $e) {
            throw new Exception($e->getMessage());
        }
    }

    /**
     * @param $query : the query to execute
     * @param $params : the parameters to bind
     * @return PDOStatement : statement executed
     * @throws Exception : in case something went wrong
     */
    private function executeQuery($query, $params) {
        try {
            $stmtn = $this->prepare($query); //prepare the query
            if($stmtn) {
                if(isset($params)) { //sanitaries all prams in case any
                    filter_var_array($params, FILTER_SANITIZE_FULL_SPECIAL_CHARS);
                }
                if( $stmtn->execute($params)) { //execute the query with the sanitised params
                    return $stmtn;
                } else {
                    throw new Exception("The database server cannot execute the statement");
                }
            } else {
                throw new Exception("The database server cannot prepare the statement");
            }
        } catch (Exception $e) {
            throw new Exception($e->getMessage());
        }
    }

    /**
     * Close the connection to database
     * @param $connection : pointer of the DataBase's connection to close
     */
    static function closeConnection(&$connection) {
        $connection = null;
    }
}
?>

```