UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFROMATICA

INSEGNAMENTO DI BASI DI DATI E SISTEMI OPERATIVI I

ANNO ACCADEMICO 2019/2020

Documentazione

Progettazione di una base di dati relazionale

<Sistema Gestionale per Recensioni Turistiche>

A cura di:

Federico Gargiulo - N86002884

Antonio Garofalo - N86003129

Indice:

1 – Introduzione

Il progetto che abbiamo scelto tra le tracce disponibili è stato quello di un ***Sistema Gestionale per Recensioni Turistiche,*** si chiedeva al gruppo di due studenti di creare una base di dati che venisse gestita da un applicativo java.

I vincoli imposti, sono stati prettamente 2:

-Un utente può unicamente fare una singola recensione per una determinata location.

-Le location devo essere gestite in 3 macro-specializzazioni che sarebbero:

- Alloggio
- Attrazione
- Ristorante

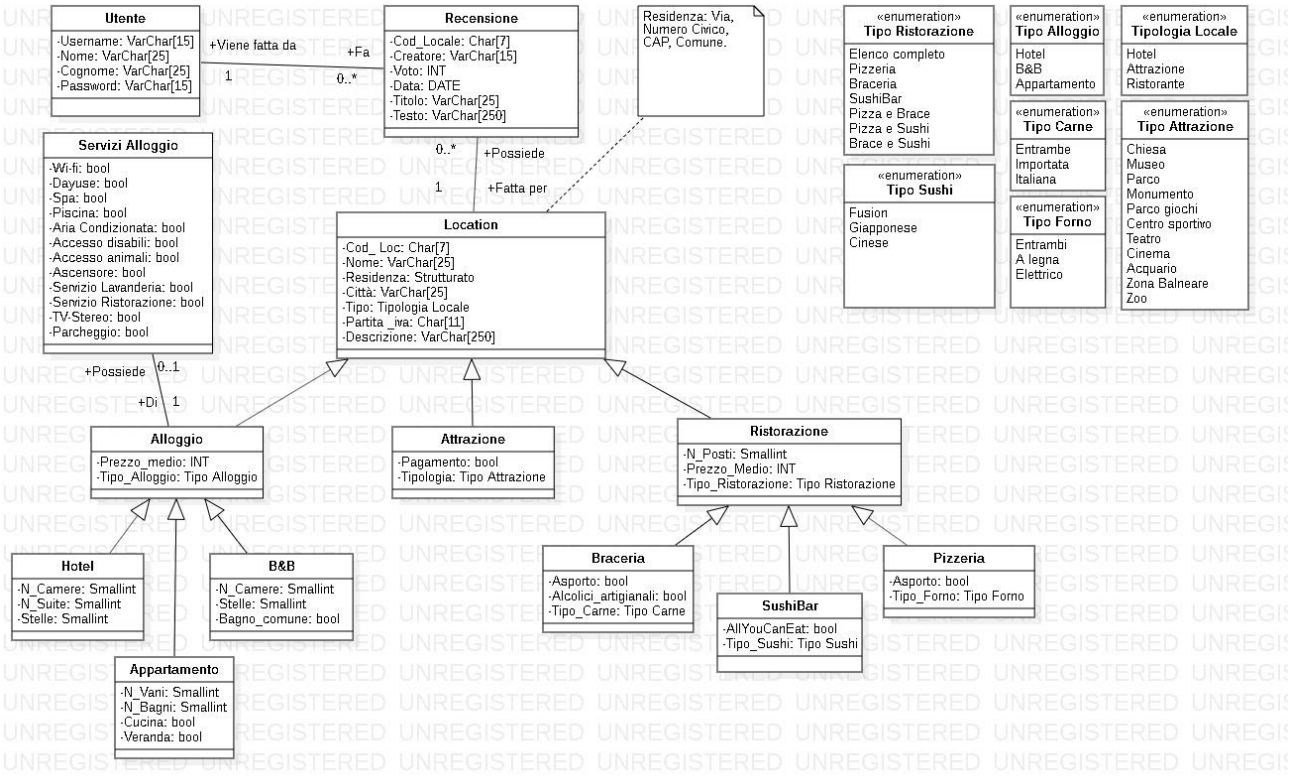
(N.B le specializzazioni devono essere ulteriormente raffinate)

E' stato pensato di fare una base di dati su Oracle (precisamente versione 11g facilmente configurabile) gestendo il tutto con l'applicativo "SQL developer".

La base di dati è stata modellata mediante Class Diagram con successiva revisione e ristrutturazione. Si noterà bene nella definizione di vincoli e tabelle la creazione anche di trigger che, per mancata presenza in Oracle, abbiamo inserito al fine di avere una base di dati dinamica che si aggiornasse automaticamente al cambiamento di specifici parametri e alcuni che velocizzassero gli inserimenti creando parallelamente altri inserimenti a catena da modificare con le informazioni desiderate.

Infine per i gruppi a 3 veniva chiesto un sistema di moderazione, tramite notifiche e controlli sulle recensione fatte prima di essere pubblicate.

2 – Class Diagram (non ristrutturato)



Il class diagram proposto per la soluzione del problema è formato da 13 classi (che diventeranno 14 con la ristrutturazione), 9 di queste classi sono specializzazioni di altre classi, per la precisione le classi sottostanti alla classe “Location” ad esclusione della classe “Servizi alloggio” sono specializzazioni.

Per raffinare meglio le classi si è optato per l’introduzione di vari nuovi tipi che sono sviluppati nelle enumerazioni in alto a destra.

Inoltre per completezza si è aggiunto un attributo strutturato (“Residenza” in “Location”) che verrà analizzato successivamente con la ristrutturazione del class diagram.

3 – Modifiche di ristrutturazione

Con un quadro generale avuto dal class diagram (non ristrutturato) si passa alle prime modifiche fatte in vista della ristrutturazione.

Ovviamente bisognava:

- Eliminare generalizzazioni/specializzazioni
- Eliminare attributi strutturati
- Eliminare eventuali molteplicità (assenti in principio)

Analizzando il precedente class diagram possiamo notare:

- Un attributo strutturato (in “Location)
- 9 specializzazioni e sotto-specializzazioni

La soluzione per il primo problema di ristrutturazione è stata scelta tra le soluzioni studiate durante il corso, ovvero, si è optato per la creazione di un “Entità” o classe totalmente estranea che crea un'unica associazione 1 a 0…* con la classe “Location”, essa conterrà le residenze che esistono associate poi a nessuna o molte località.

Per il secondo problema, nonché quello principale richiesto anche dalla traccia si è optato per eliminare le generalizzazioni/specializzazioni lasciando intatta la struttura ad albero creatasi. Per questo motivo c’è stata la sostituzione completa con associazioni vincolate.

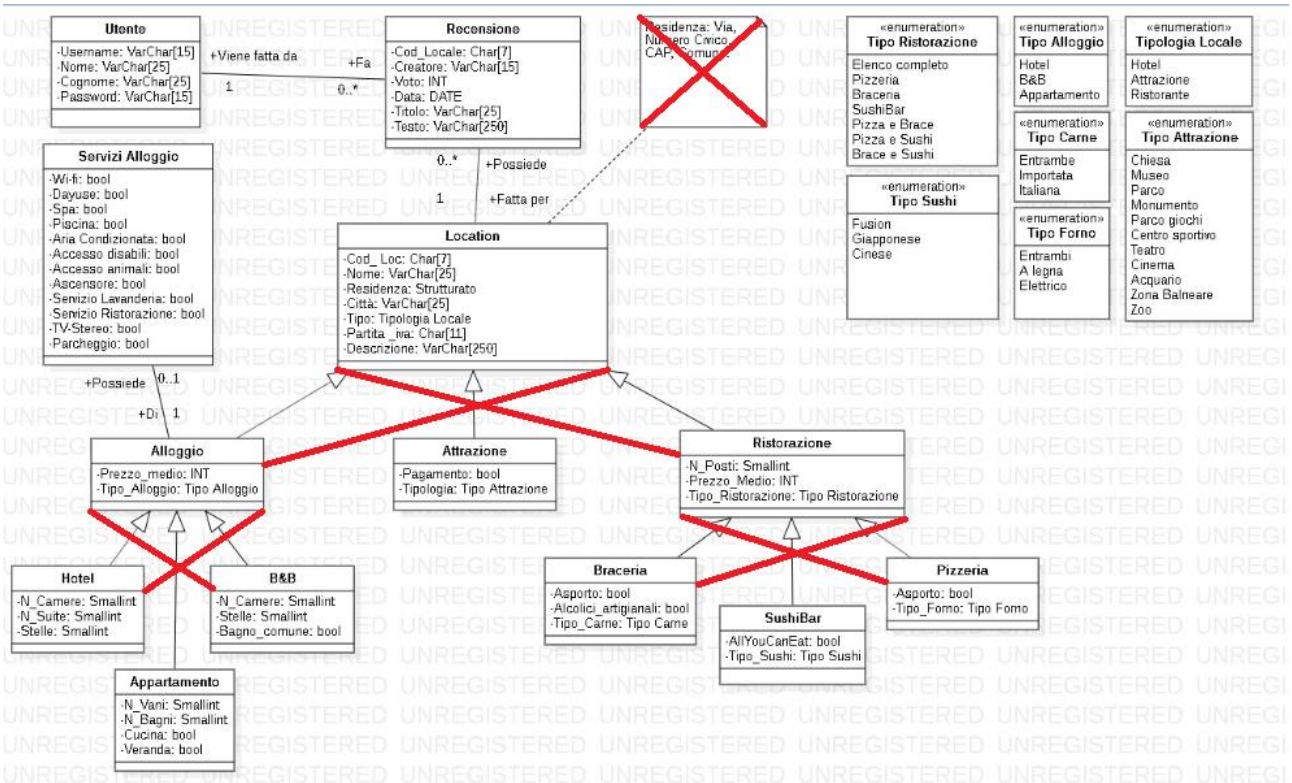
Infatti i vincoli che si sono introdotti (imposti anche dalla traccia) definiscono che la prima specializzazione è una total/disjoint ciò significa che un istanza “Location” può essere SOLO “Alloggio”, “Attrazione” o “Ristorante”.

Le successive specializzazioni invece si sono divise i 3 macro esempi:

- Alloggio: specializzazione partial/disjoint con “Hotel”, “BeB” e “Appartamento”
- Attrazione: per la vastità di tipologie si è pensato di non far avere specializzazioni da “Attrazione”
- Ristorante: specializzazione partial/overlapping con “Pizzeria”, “SushiBar” e “Braceria”

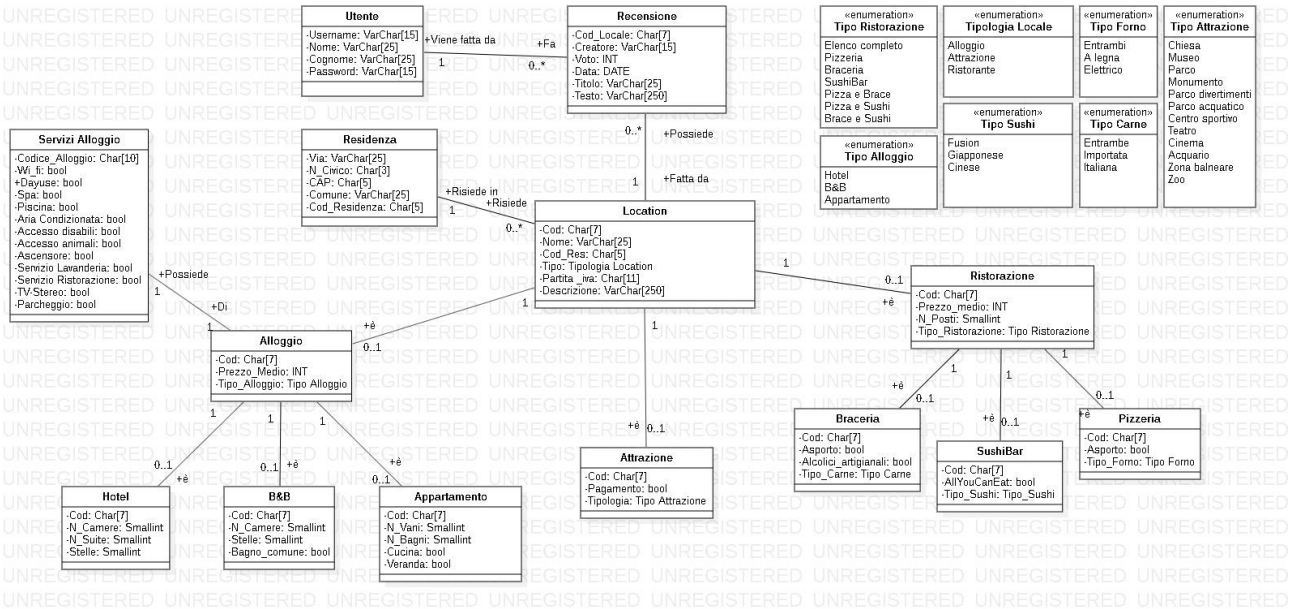
La ristrutturazione completa del class diagram ha portato (come c'era da aspettarsi) alla obbligatoria aggiunta di attributi di *chiave esterna* per istanziare le associazioni che hanno sostituito le specializzazioni

Infine per quanto riguarda la classe creatasi "Residenza", gli attributi che la compongono son i campi strutturati precedentemente definiti, con l'aggiunta di una chiave surrogata per avere una *chiave primaria* a cui associarsi dalla classe "Location".



Le "X" rosse rappresentano ciò che nel class diagram (ristrutturato) non apparirà.

4 – Class diagram (Ristrutturato)



Il class diagram sovrastante rappresenta la ricostruzioni adattabile alla traduzione dei dati logica e relazionale.

Come precedentemente illustrato il class diagram conteneva un attributo strutturato e varie specializzazione che andavano modellate per una più facile comprensione.

Ristrutturazione “Attributi strutturati”:

L’attributo strutturato “Residenza” contenuto nella tabella “Location” è stato prontamente eliminato per rappresentarlo sotto forma di classe a se stante.

La classe creatasi, chiamata appunto “Residenza”, contiene come attributi quelli che sono le informazioni principali di una residenza, introducendo la chiave primaria (per introdurre unicità e associazione ad altre classi), l’attributo è stato chiamato per convenzione “Cod_Residenza” associato con la chiave esterna di “Location” rinominata “Cod_Res”.

L’associazione creatasi tra “Residenza” e “Location” è una associazione 1 a 0…*, questa perché definisce l’ovvietà dove una residenza può avere associate nessuna o molte location mentre una location può avere un'unica residenza associata.

Ristrutturazione “Molteplicità”:

Assenti durante la progettazione del problema.

Ristrutturazione “Generalizzazioni/Specializzazioni”:

La ristrutturazione delle ben 9 specializzazioni della classe “Location” sono state gestite mantenendo intatta la struttura “ad Albero” senza complicazioni di schiacciamento o vincoli che permettessero la creazione di istanze solo in specifici casi.

Sono stati introdotti attributi creati (gestiti con apposite enumerazioni) alcune introdotte per creare una piccola differenza tra e varie specializzazioni altre sono invece attributi “Surrogati” usati come switch per identificare la specializzazione successiva e facilitare il lavoro sull’applicativo JAVA.

Esempio pratico:

Tipo_Location, identifica la tipologia tra Alloggio, Attrazione e Ristorante.

Identificata la tipologia di location abbiamo (a seconda della scelta) un altro attributo (definito nella classe specializzata da location) che identifica il Tipo che successivamente si andrà a specializzarsi (ad eccezione della classe “Attrazione”).

N.B:

Le specializzazioni create sono per la classe “Alloggio” partial/disjoint mentre per la calsse “Ristorante” partial/overlapping, ovviamente sono parziali per il semplice fatto che ci sono altre istanze che possono essere considerate nel problema ma che per semplicità ne vengono rappresentate solo 3.

Disjoint sono le specializzazioni di alloggio dato che per naturalezza un “Hotel” non può essere un “BeB” oppure un “Appartamento

Overlapping dato che un “Ristorante” può essere sia “Pizzeria” che “Braceria” che “SushiBar” (si consideri il viceversa).

L’unica total/disjoint è quella definita dalla traccia ovvero la specializzazione “Alloggio”, “Attrazione” e “Ristorante” dato che rappresentano le tre istanze del problema completo.

5 – Dizionario delle classi

In questa sezione approfondiamo le classi presenti nel class diagram finale, definendo attributi, tipi e che ruolo hanno.

Classi presenti:

- Utente
- Recensione
- Location
- Residenza

Come entità forti, mentre come specializzazioni di “Location” abbiamo:

- Alloggio
- Attrazione
- Ristorante

Raffiniamo con altre classi, per Alloggio:

- Servizi alloggio
- Hotel
- BeB
- Appartamento

Raffiniamo Ristorazione:

- Braceria
- SushiBar
- Pizzeria

Classe - Utente

Username – Chiave primaria che identifica il singolo utente;

Password – Usata per l'accesso nel programma JAVA;

Nome – Stringa di non nulla ;

Cognome – Stringa di raffinazione non nulla;

Classe – Recensione

Cod_Locale – Chiave esterna associata a Cod di "Location";

Creatore - Chiave esterna associata a Username di "Utente";

(Entrambe le chiavi identificano una recensione)

Voto – Intero non nullo, usato per la media nella stored procedure;

Data – Tipo DATE non nullo usato per una cronologia o un determinato ordinamento;

Titolo – Stringa non nulla;

Testo – Stringa che può essere nulla (Gestita con un check);

Classe – Location

Cod – Chiave primaria di identificazione di "Location";

Nome – Stringa non nulla, indica il nome della Location;

Cod_Res – Chiave esterna al Cod_Residenza di "Residenza";

Tipo – Tipo_Location, è una stringa gestita da un check in sulla base di tre valori, Alloggio, Attrazione e Ristorante;

Partita_iva – Stringa obbligatoria ma non necessaria per le attrazioni;

Descrizione – Stringa di raffinamento, non per forza nulla;

Classe – Residenza

Via – Stringa non nulla, indica la via;

N_Civico – Stringa non nulla, indica il civico;

CAP – Stringa non nulla, indica il codice di avviamento postale (CAP);

Comune – Stringa non nulla, indica il comune;

Codice_Residenza – Chiave primaria di identificazione delle residenze;

La classe “Residenza” ha tutti gli attributi non nulli dato che deve essere visualizzata tramite applicativo sottoforma di concatenazione di stringhe ben formattate.

Specializzazioni:

Classe Alloggio

Cod – Chiave esterna associata a Cod di “Location”

Prezzo_medio – Prezzo medio di una notte;

Tipo_Alloggio - Tipo_Alloggio, è una stringa gestita da un check in sulla base di tre valori, Hotel, BeB e Appartamento;

Classe Servizi alloggio (Non è specializzazione ma associata esternamente ad Alloggio)

Cod_Alloggio – Chiave esterna associata alla chiave Cod presente in “Alloggio”, grazie alla definizione di unicità questa chiave può essere associata ad una chiave esterna.

Serie di attributi – Tutti i restanti 12 attributi di “Servizi alloggio” sono definiti booleani ma come tutti i booleani su oracle vengono definiti da due caratteri CHAR che prendono un falore di default (ns = non specificato) e possono avere valore “si” o “no.

Classe Attrazione

Cod – Chiave esterna associata a Cod di “Location”;

Pagamento – Definisce se l’attrazione è a pagamento;

Tipologia – Tipo_Attrazione è un tipo gestito tramite check in di una serie di stringe che definiscono varie tipologie di attrazioni.

Classe Ristorazione

Cod – Chiave esterna associata a Cod di “Location”;

Prezzo_medio – Prezzo medio di una cena;

N_Posti – Numero di posti a sedere;

-Tipo_Ristorazione – Tipo_Alloggio, è una stringa gestita da un check in sulla base di tre valori, Braceria, SushiBar e Pizzeria insieme a tutte le altre combinazioni a coppie per definire quella che è una specializzazione “Overlapping”;

Sotto-Specializzazioni:

Classe Hotel

Cod – Chiave esterna associata a Cod di “Alloggio” che avendo unicità permette tale associazione;

N_Camere – Numero di camere nell’albergo;

N_Suite – Numero di suite nell’albergo;

Stelle - Indica le stelle di un Hotel/Albergo;

;**Classe BeB**

Cod – Chiave esterna associata a Cod di “Alloggio” che avendo unicità permette tale associazione;

N_Camere – Intero per raffinamento della specializzazione con valore di default “ns”;

Stelle - Intero per raffinamento della specializzazione con valore di default “ns”, indica le stelle di un BeB/Ostello;

Bagno_comune -Booleano usato per raffinare la classe, indica se il BeB ha il bagno in comune;

Classe Appartamento

Cod – Chiave esterna associata a Cod di “Alloggio” che avendo unicità permette tale associazione;

N_Vani – Numero di vani nell’appartamento;

N_Bagni – Numero di bagni nell’appartamento;

Cucina – Boolean che definisce se c’è una cucina;

Veranda – Boolean che definisce se c’è una veranda;

Classe Braceria

Cod – Chiave esterna associata a Cod di “Ristorazione” che avendo unicità permette tale associazione;

Asporto – Boolean che definisce se c’è il servizio di asporto;

Aloclici_artigianali – Boolean che definisce se vengono serviti alcolici della casa;

Tipo_Carne – Stringa definita da un check sul tipo di carne servita (Entrambe, Italiana o Importata);

Classe SushiBar

Cod – Chiave esterna associata a Cod di “Ristorazione” che avendo unicità permette tale associazione;

AllYouCanEat – Boolean che definisce se c’è sevizio AYCE;

Tipo_Sushi – Stringa definita da un check sui valori del tipo di suchi servito (Fusion, Giapponese o Cinese);

Classe Pizzeria

Cod – Chiave esterna associata a Cod di “Ristorazione” che avendo unicità permette tale associazione;

Asporto – Boolean che definisce se c’è il servizio di asporto;

Tipo_Forno – Stringa gestita da un check sui valori del tipo di forno di cottura (Elettri, a legna o Entrambi);

6 – Dizionario delle associazioni

Le associazioni presenti nella soluzione del problema sono precisamente 13 e sono:

- Utente-Recensione
- Recensione-Location
- Location-Residenza
- Location-Alloggio
- Location-Attrazione
- Location-Ristorazione
- Alloggio-Servizi alloggio
- Alloggio-Hotel
- Alloggio-BeB
- Alloggio-Appartamento
- Ristorazione-Braceria
- Ristorazione-SushiBar
- Ristorazione-Pizzeria

Analizziamole tutte:

Utente-Recensione

Tipologia – 1 a Molti (1 – 0…*);

Un utente può fare nessuna o molte recensioni, una recensione è associata ad un unico utente.

Recensione-Location

Tipologia – ,Molti a 1 (0…* - 1);

Una recensione è fatta ad un'unica specifica location, su una location possono essere fatte nessuna o tante recensioni.

Location-Residenza

Tipologia – ,Molti a 1 (0…* - 1);

Una location ha associata un'unica residenza, una residenza avrà da nessuna a molte location.

Location-Alloggio

Tipologia: 1 a 0/1 (1 – 0…1);

Una location può essere o non un alloggio, un alloggio è una location.

Location-Attrazione

Tipologia: 1 a 0/1 (1 – 0…1);

Una location può essere o non un attrazione, un attrazione è una location.

Location-Ristorazione

Tipologia: 1 a 0/1 (1 – 0…1);

Una location può essere o non un ristorante, un ristorante è una location.

Alloggio-Servizi alloggio

Tipologia – 1 a 1 (1 – 1);

Un alloggio ha un servizio alloggio personale, ad un servizio alloggio è associato un alloggio sempre;

Alloggio-Hotel

Tipologia: 1 a 0/1 (1 – 0…1);

Un alloggio può essere o non un Hotel, un Hotel è un alloggio;

Alloggio-BeB

Tipologia: 1 a 0/1 (1 – 0…1);

Un alloggio può essere o non un BeB, un BeB è un alloggio;

Alloggio-Appartamento

Tipologia: 1 a 0/1 (1 – 0…1);

Un alloggio può essere o non un Appartamento, un Appartamento è un alloggio;

Ristorazione-Braceria

Tipologia: 1 a 0/1 (1 – 0…1);

Un ristorante può o non essere braceria, una braceria è un ristorante;

Ristorazione-SushiBar

Tipologia: 1 a 0/1 (1 – 0…1);

Un ristorante può o non essere sushibar, un sushibar è un ristorante;

Ristorazione-Pizzeria

Tipologia: 1 a 0/1 (1 – 0…1);

Un ristorante può o non essere pizzeria, una pizzeria è un ristorante;

Alcune associazioni potrebbero in una determinata istanza non esistere o essere settate a valori di default dato che le specializzazioni appunto definiscono un istanza precisa e non overlapping, per le sotto-specializzazioni invece di “Ristorazione” possono esserci più istanze specializzate al seconda del tipo di ristorazione, ma di questo ne parleremo del dizionario dei vincoli.

7 – Dizionario dei vincoli

In questa ultima parte del dizionario della base di dati vengono elencati ed esposti i vincoli più importanti:

(N.B i vincoli senza nomi sono stati aggiunti durante la creazione delle tabelle quindi hanno un nome sottointeso scelto automaticamente dal DBMS i.e SYSXXX)

I vincoli possono essere:

- Di dominio (Su un attributo di una tabella);
- Di n-upla (Su più attributi di una tabella)
- Intrarelazionali (Unique, Primary key);
- Interrelazionali(Foreign key);

Vincoli di dominio:

Check su i tipi:

I check in definiti nella creazione delle tabelle in SQL sono vincoli di dominio che garantiscono un valore (tra quelli proposti nell'insieme) ad un determinato attributo, per esempio sul Tipologia Location, Tipo alloggio, Tipo ristorazione e così via…

Check sul voto recensione:

Una recensione può avere un voto compreso tra 1 e 5.

Check sul prezzo medio di Alloggio e Ristorante:

Un prezzo medio deve essere maggiore di 0.

Vincoli di n-upla:

Assenti.

Vincoli intrarelazionali:

Username definisce unicamente un Utente (Primary key);

Recensione è definita come una classe entità ma che fa da ponte tra due classi (Utente e Location), una recensione ha associati Un utente e Una location.

Un utente può fare unicamente una recensione per una singola location, non ne può fare di più, questo vincolo è gestito tramite l'aggiunta di un vincolo di unicità che comprende i 2 attributi chiave di recensione (Cod_Locale e Creatore).

Una location è definita da un codice identificativo (Cod) definito come primary key.

Una residenza è definita da un codice identificativo (Cod_Residenza) definito come primary key.

Tutti gli identificativi (Cod) nelle specializzazioni e sotto-specializzazioni di location ereditano unicità e quindi una referenza (Foreign key) verso il l'attributo identificativo di location (Cod);

Vincoli interrelazionali:

Una location può essere specializzata solo in Alloggio, Attrazione o Ristorazione, ciò significa che per una singola location esisterà un'unica istanza che potrà essere scelta tra Alloggio, Attrazione e Ristorante, le altre saranno nulle.

Un alloggio ha sempre associato un Servizi alloggio (Associazione 1 a 1 OBBLIGATORIA).

Un alloggio si specializza solo in Hotel, BeB o Appartamento (Disjoint).

Un ristorante si può specializzare in Braceria, SushiBar e Pizzeria o anche una combinazione tra di loro, esempio, se un ristorante è di tipo Braceria che Pizzeria, ci saranno 2 istanze (Una in braceria e una in Pizzeria che referenzieranno lo stesso ristorante).

Tutti i vincoli oltre ad essere gestiti tramite dichiarazioni SQL e quindi eccezioni di Integrità, sono in parte gestiti dai trigger SQL (scritti nel capitolo 9 – SQL) che garantiscono la presenza nel database di istanze sempre aggiornate.

8 – Schema logico/relazionale

Lo schema relazionale per il problema rappresenterà tutte le relazioni con classi, chiavi primarie, chiavi esterne senza ordine ne molteplicità:

N.B i domini non specificati sono null o comunque non rilevanti.

Utente(Nome, Cognome, Username, Password);

Numero attributi:4

Chiavi primarie:1

Chiavi esterne:0

Recensione(Cod_Locale, Creatore, Voto, Data, Titolo, Testo);

Numero attributi:6

Chiavi primarie:0

Chiavi esterne:2

Dom(Voto)={1…5}

Dom(Data)={Giorno=1…31, Mese=1..12, Anno=1900…2100}

Location(Cod, Nome, Cod_Res, Tipo, Partita_IVA, Descrizione);

Numero attributi:6

Chiavi primarie:1

Chiavi esterne:1

Dom(Tipo)={Alloggio, Attrazione, Ristorante}

Residenza(Via, N_Civico, CAP, Comune, Cod_Residenza);

Numero attributi:5

Chiavi primarie:1

Chiavi esterne:0

Alloggio(Cod, Prezzo_medio, Tipo_Alloggio);

Numero attributi:3

Chiavi primarie:0

Chiavi esterne:1

Dom(Tipo_Alloggio)={Hotel, BeB, Appartamento}

Servizi Alloggio(Codice_Alloggio, Wi_Fi, Dayuse, Spa, Piscina, Aria Condizionata, Accesso disabili, Accesso animali, Ascensore, Servizio Lavanderia, Servizio Ristorazione, TV_Stereo, Parcheggio);

Numero attributi:13

Chiavi primarie:0

Chiavi esterne:1

(I domini degli attributi al di fuori di “Codice_Alloggio” sono boolean)

Hotel(Cod, N_Camere, N_Suite, Stelle);

Numero attributi:4

Chiavi primarie:0

Chiavi esterne:1

Dom(Stelle)={1…5}

BeB(Cod, N_Camere, Stelle, Bagno_comune);

Numero attributi:4

Chiavi primarie:0

Chiavi esterne:1

Dom(Stelle)={1…5}

Dom(Bagno_comune)={si, no, ns}

Appartamento(Cod, N_Vani, N_Bagni, Cucina, Veranda);

Numero attributi:5

Chiavi primarie:0

Chiavi esterne:1

Dom(Cucina)={si, no, ns}

Dom(Veranda)={si, no, ns}

Attrazione(Cod, Pagamento, Tipologia);

Numero attributi:3

Chiavi primarie:0

Chiavi esterne:1

Dom(Pagamento)={si, no, ns}

Dom(Tipologia)={Chiesa, Museo, Parco, Monumento, Parco divertimenti, Parco acquatico, Centro sportivo, Teatro, Cinema, Acquario, Zona balneare, Zoo}

Ristorazione(Cod, Prezzo_medio, N_Posti, Tipo_Ristorazione);

Numero attributi:4

Chiavi primarie:0

Chiavi esterne:1

Dom(Tipo_Ristorazione)={Elenco completo, Braceria, SushiBar, Pizzeria, Pizza e Brace, Pizza e Sushi, Brace e Sushi}

Braceria(Cod, Asporto, Alcolici_artigianali, Tipo_Carne);

Numero attributi:4

Chiavi primarie:0

Chiavi esterne:1

Dom(Asporto)={si, no, ns}

Dom(Alcolici_artigianali)={si, no, ns}

Dom(Tipo_Carne)={Entrambe, Importata, Italiana}

SushiBar(Cod, AllYouCanEat, Tipo_Sushi);

Numero attributi:3

Chiavi primarie:0

Chiavi esterne:1

Dom(AllYouCanEat)={si, no, ns}

Dom(Tipo_Sushi)={Fusion, Giapponse, Cinese}

Pizzeria(Cod, Asporto, Tipo_Forno);

Numero attributi:3

Chiavi primarie:0

Chiavi esterne:1

Dom(Asporto)={si, no, ns}

Dom(Tipo_Forno)={Entrambi, Elettrico, A legna}

9 – SQL

A seguire il codice SQL usato per creare lo scheletro del database, a fondo del capitolo ci sarà una breve spiegazione e visione delle sequenze create e dei trigger usati.

(N.B è consigliato eseguire per i trigger, uno script alla volta e non tutto insieme dato che oracle non riconosce il susseguirsi di trigger credendo che ne sia uno solo)

--Creazione tabelle

create table utente(username varchar(15) primary key, nome varchar(25) not null, cognome varchar(25) not null, password varchar(15) not null);

create table recensione(cod_locale char(7) not null, creatore varchar(15) not null, voto int not null, check(voto between 1 AND 5), data DATE default CURRENT_TIMESTAMP, titolo varchar(25) not null, testo varchar(250) default 'Non esiste testo');

create table location(cod char(7) primary key, nome varchar(25) not null, cod_res char(5) not null, tipo_location varchar(25) not null, check (tipo_location='Alloggio' OR tipo_location='Attrazione' OR tipo_location='Ristorante'), partita_iva char(11) not null, descrizione varchar(100) default 'Nessuna Descrizione');

create table residenza(via varchar(25) not null, n_civico varchar(5) not null, cap char(5) not null, comune varchar(25) not null, cod_residenza char(5) primary key);

create table alloggio(cod char(7) primary key, prezzo_medio int, check(prezzo_medio>=0), tipo_alloggio varchar(25) default 'Non Specificato', check(tipo_alloggio='Hotel' OR tipo_alloggio='BeB' OR tipo_alloggio='Appartamento' OR tipo_alloggio='Non Specificato'));

create table servizi_alloggio(cod char(7) not null, wi_fi char(2) default 'ns', dayuse char(2) default 'ns', spa char(2) default 'ns', piscina char(2) default 'ns', aria_condizionata char(2) default 'ns', accesso_disabili char(2) default

'ns', accesso_animali char(2) default 'ns', ascensore char(2) default 'ns', servizio_lavanderia char(2) default 'ns', servizio_ristorazione char(2) default 'ns', tv_stereo char(2) default 'ns', parcheggio char(2) default 'ns');

create table attrazione(cod char(7) not null, pagamento char(2) default 'ns', tipo_attrazione varchar(25));

create table ristorazione(cod char(7) primary key, prezzo_medio int, check(prezzo_medio>=0), n_posti smallint, tipo_ristorazione varchar(25) default 'Non Specificato', check(tipo_ristorazione='Elenco completo' OR tipo_ristorazione='Pizzeria' OR tipo_ristorazione='Braceria' OR tipo_ristorazione='SushiBar' OR tipo_ristorazione='Pizza e Brace' OR tipo_ristorazione='Pizza e Sushi' OR tipo_ristorazione='Brace e Sushi' OR tipo_ristorazione='Non Specificato'));

create table hotel(cod char(7) not null, n_camere smallint, n_suite smallint, stelle smallint);

create table beb(cod char(7) not null, n_camere smallint, stelle smallint, bagno_comune char(2) default 'ns', check(bagno_comune='si' OR bagno_comune='no' OR bagno_comune='ns'));

create table appartamento(cod char (7) not null, n_vani smallint, n_bagni smallint, cucina char(2) default 'ns', check(cucina='si' OR cucina='no' OR cucina='ns'), veranda char(2) default 'ns', check(veranda='si' OR veranda='no' OR veranda='ns'));

create table braceria(cod char(7) not null, asporto char(2) default 'ns', alcolici_artigianali char(2) default 'ns', tipo_carne varchar(20) default 'Non Specificato', check(tipo_carne='Entrambe' OR tipo_carne='Importata' OR tipo_carne='Italiana' OR tipo_carne='Non Specificato'));

create table sushibar(cod char(7) not null, allyoucaneat char(2) default 'ns', tipo_sushi varchar(20) default 'Non Specificato', check(tipo_sushi='Fusion' OR tipo_sushi='Giapponese' OR tipo_sushi='Cinese' OR tipo_sushi='Non Specificato'));

create table pizzeria(cod char(7) not null, asporto char(2) default 'ns', tipo_forno varchar(20) default 'Non Specificato', check(tipo_forno='Entrambi'

OR tipo_forno='A legna' OR tipo_forno='Elettrico' OR tipo_forno='Non Specificato'));

--Unique

alter table recensione add constraint uc_recensione unique(creatore,cod_locale);

alter table attrazione add constraint uc_cod_att unique(cod);

alter table servizi_alloggio add constraint uc_cod_sall unique(cod);

alter table hotel add constraint uc_cod_hotel unique(cod);

alter table beb add constraint uc_cod_beb unique(cod);

alter table appartamento add constraint uc_cod_app unique(cod);

alter table braceria add constraint uc_cod_brac unique(cod);

alter table sushibar add constraint uc_cod_sushi unique(cod);

alter table pizzeria add constraint uc_cod_pizza unique(cod);

--Foreign key

alter table recensione add constraint fk_recensione_creatore foreign key (creatore) references utente(username) ON DELETE CASCADE;

alter table recensione add constraint fk_recensione_cod_locale foreign key (cod_locale) references location(cod) ON DELETE CASCADE;

alter table location add constraint fk_residenza_cod_residenza foreign key (cod_res) references residenza(cod_residenza);

alter table alloggio add constraint fk_alloggio foreign key (cod) references location(cod) ON DELETE CASCADE;

alter table attrazione add constraint fk_attrazione foreign key (cod) references location(cod) ON DELETE CASCADE;

alter table ristorazione add constraint fk_ristorazione foreign key (cod) references location(cod) ON DELETE CASCADE;

alter table servizi_alloggio add constraint fk_servizi_alloggio foreign key (cod) references alloggio(cod) ON DELETE CASCADE;

alter table attrazione add constraint tipo_attrazione check(tipo_attrazione='Chiesa' OR tipo_attrazione='Museo' OR tipo_attrazione='Parco' OR tipo_attrazione='Monumento' OR tipo_attrazione='Parco Giochi' OR tipo_attrazione='Centro Sportivo' OR tipo_attrazione='Teatro' OR tipo_attrazione='Cinema' OR tipo_attrazione='Acquario' OR tipo_attrazione='Zona Balneare' OR tipo_attrazione='Zoo');

alter table hotel add constraint fk_hotel foreign key (cod) references alloggio(cod) ON DELETE CASCADE;

alter table beb add constraint fk_beb foreign key (cod) references alloggio(cod) ON DELETE CASCADE;

alter table appartamento add constraint fk_appartamento foreign key (cod) references alloggio(cod) ON DELETE CASCADE;

alter table braceria add constraint fk_braceria foreign key (cod) references ristorazione(cod) ON DELETE CASCADE;

alter table sushibar add constraint fk_sushibar foreign key (cod) references ristorazione(cod) ON DELETE CASCADE;

alter table pizzeria add constraint fk_pizzeria foreign key (cod) references ristorazione(cod) ON DELETE CASCADE;

L’ordine di lettura dello script SQL parte con la creazione delle tabelle, alcuni vincoli (Check di dominio più che altro) sono stati scritti durante la definizione delle tabelle e non aggiunti dopo con un “ALTER TABLE”.

Susseguono poi i vincoli “Interrelazionali” di unicità di alcuni attributi dato che daranno (almeno alcuni nelle specializzazioni) sia chiavi esterne che chiavi primarie di se stessi.

Infine sono presenti script per creare ed associare tutte le classi tra di loro mediante l'utilizzo di chiavi esterne referenziate ad opportune chiavi primarie.

A seguire Sequenze e trigger:

--SEQUENCE

Create sequence Cod_Location

start with 1000

increment by 1

minvalue 1000

maxvalue 9999

nocache

nocycle;

--

Create sequence Cod_Residenza

start with 1000

increment by 1

minvalue 1000

maxvalue 9999

nocache

nocycle;

--STORED PROCEDURE

CREATE PROCEDURE getMediaVoto (cod in VARCHAR2, media out float) as

BEGIN

select AVG(voto) into media from recensione where recensione.cod_locale=cod;

END;

```
--TRIGGER SEQ
create or replace TRIGGER Seq_Cod_Location
before insert on location
for each row

declare
prefisso char(3):='LOC';
codice char(4):='';
begin
codice:=COD_LOCATION.nextval;
:new.cod:=prefisso||codice;
END;
--
create or replace TRIGGER Seq_Cod_Residenza
before insert on residenza
for each row

declare
prefisso char(1):='R';
codice char(4):='';
begin
codice:=COD_RESIDENZA.nextval;
:new.cod_residenza:=prefisso||codice;
END;
```

Sopra sono riportate 2 sequenze utilizzate per un assegnazione automatica mediate prefisso di codici “Residenza” e codici “Location” (riferendoci a Cod_Residenza in Residenza e Cod in Location).

I prefissi sono:

- LOCXXXX per il codice location;
- RXXXX per il codice residenza;

Le 2 sequenze sono gestite tramite trigger di INSERT chiamati appositamente Seq_Cod_Residenza e Seq_Cod_Location.

Infine si può notare una STORED PROCEDURE utilizzata nell'applicativo Java per un facile richiamo alla media voti delle recensioni di una determinata location.

Infine ci sono vari trigger che come preannunciato nell'introduzione alcuni sono stati aggiunti per ovviare alla mancata presenza di ON UPDATE CASCADE in Oracle.

Altri trigger come:

- insert_new_location;
- update_tipo_location;
- update_tipo_alloggio;
- update_tipo_ristorazione;

Sono trigger di aggiornamento automatico che settano i campi di specializzazioni successive a default o comunemente a null che a loro volta saranno gestite da altri trigger di update.

```
--TRIGGER
create trigger update_utente
  after update of username on utente
  for each row
begin
   update recensione r
   set r.creatore=:new.username
   where r.creatore=:old.username;
END;
```

```
create trigger update_cod_residenza
  after update of cod_residenza on residenza
  for each row
begin
   update location l
   set l.cod_res=:new.cod_residenza
   where l.cod_res=:old.cod_residenza;
END;

create trigger insert_new_location
  after insert on location
  for each row
begin
IF :new.tipo_location='Alloggio' THEN
          insert into alloggio(cod) values (:new.cod);
          insert into servizi_alloggio(cod) values (:new.cod);
ELSIF :new.tipo_location='Ristorante' THEN
          insert into ristorazione(cod) values (:new.cod);
ELSIF :new.tipo_location='Attrazione' THEN
          insert into attrazione(cod) values (:new.cod);
END IF;
END;

create trigger update_tipo_location
  after update of tipo_location on location
  for each row
begin
IF :old.tipo_location='Alloggio' AND :old.tipo_location<>:new.tipo_location THEN
          delete from alloggio where alloggio.cod=:old.cod;
          delete from servizi_alloggio where servizi_alloggio.cod=:old.cod;
```

```
ELSIF :old.tipo_location='Ristorazione' AND :old.tipo_location<>:new.tipo_location THEN
        delete from ristorazione where ristorazione.cod=:old.cod;
ELSIF :old.tipo_location='Attrazione' AND :old.tipo_location<>:new.tipo_location THEN
        delete from attrazione where attrazione.cod=:old.cod;
END IF;
IF :new.tipo_location='Alloggio' AND  :old.tipo_location<>:new.tipo_location THEN
        insert into alloggio(cod) values (:old.cod);
        insert into servizi_alloggio(cod) values (:old.cod);
ELSIF  :new.tipo_location='Ristorazione' AND :old.tipo_location<>:new.tipo_location THEN
        insert into ristorazione(cod) values (:old.cod);
ELSIF  :new.tipo_location='Attrazione' AND  :old.tipo_location<>:new.tipo_location THEN
        insert into attrazione(cod) values (:old.cod);
END IF;
END;
```

create trigger update_tipo_alloggio

```
 after update of tipo_alloggio on alloggio
 for each row
begin
IF :old.tipo_alloggio='Hotel' AND :old.tipo_alloggio<>:new.tipo_alloggio THEN
        delete from hotel where hotel.cod=:old.cod;
ELSIF :old.tipo_alloggio='BeB' AND :old.tipo_alloggio<>:new.tipo_alloggio THEN
        delete from beb where beb.cod=:old.cod;
ELSIF :old.tipo_alloggio='Appartamento' AND :old.tipo_alloggio<>:new.tipo_alloggio THEN
        delete from appartamento where appartamento.cod=:old.cod;
END IF;
IF :new.tipo_alloggio='Hotel' AND  :old.tipo_alloggio<>:new.tipo_alloggio THEN
        insert into hotel(cod) values (:old.cod);
ELSIF  :new.tipo_alloggio='BeB' AND :old.tipo_alloggio<>:new.tipo_alloggio THEN
        insert into beb(cod) values (:old.cod);
```

```
ELSIF  :new.tipo_alloggio='Appartamento' AND  :old.tipo_alloggio<>:new.tipo_alloggio THEN
        insert into appartamento(cod) values (:old.cod);
END IF;
END;
```

create trigger update_tipo_ristorazione

```
 after update of tipo_ristorazione on ristorazione
 for each row
begin
IF :old.tipo_ristorazione='Elenco completo' AND :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   delete from braceria where braceria.cod=:old.cod;
   delete from sushibar where sushibar.cod=:old.cod;
   delete from pizzeria where pizzeria.cod=:old.cod;
ELSIF :old.tipo_ristorazione='Braceria' AND :old.tipo_ristorazione<>:new.tipo_ristorazione THEN
    delete from braceria where braceria.cod=:old.cod;
ELSIF :old.tipo_ristorazione='SushiBar' AND :old.tipo_ristorazione<>:new.tipo_ristorazione THEN
   delete from sushibar where sushibar.cod=:old.cod;
ELSIF :old.tipo_ristorazione='Pizzeria' AND :old.tipo_ristorazione<>:new.tipo_ristorazione THEN
   delete from pizzeria where pizzeria.cod=:old.cod;
ELSIF :old.tipo_ristorazione='Pizza e Brace' AND :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   delete from pizzeria where pizzeria.cod=:old.cod;
   delete from braceria where braceria.cod=:old.cod;
ELSIF :old.tipo_ristorazione='Pizza e Sushi' AND :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   delete from pizzeria where pizzeria.cod=:old.cod;
   delete from sushibar where sushibar.cod=:old.cod;
```

```
ELSIF :old.tipo_ristorazione='Brace e Sushi' AND :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   delete from braceria where braceria.cod=:old.cod;
   delete from sushibar where sushibar.cod=:old.cod;
END IF;
IF :new.tipo_ristorazione='Elenco completo' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   insert into braceria(cod) values (:new.cod);
   insert into sushibar(cod) values (:new.cod);
   insert into pizzeria(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='Braceria' AND :old.tipo_ristorazione<>:new.tipo_ristorazione THEN
   insert into braceria(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='SushiBar' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   insert into sushibar(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='Pizzeria' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione THEN
   insert into pizzeria(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='Pizza e Brace' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   insert into pizzeria(cod) values (:new.cod);
   insert into braceria(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='Pizza e Sushi' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   insert into pizzeria(cod) values (:new.cod);
   insert into sushibar(cod) values (:new.cod);
ELSIF  :new.tipo_ristorazione='Brace e Sushi' AND  :old.tipo_ristorazione<>:new.tipo_ristorazione
THEN
   insert into braceria(cod) values (:new.cod);
   insert into sushibar(cod) values (:new.cod);
END IF;
END;
```

```
create trigger update_cod_locale
  after update of cod on location
  for each row
declare
tipo VARCHAR2(25):='';

begin
  update recensione r
  set r.cod_locale=:new.cod
  where r.cod_locale=:old.cod;

  tipo:=:old.tipo_location;

IF tipo='Attrazione'  THEN
    update attrazione a
    set a.cod=:new.cod
    where a.cod=:old.cod;
ELSIF tipo='Alloggio' THEN
    update alloggio al
    set al.cod=:new.cod
    where al.cod=:old.cod;
ELSIF tipo='Ristorazione' THEN
   update ristorazione ri
   set ri.cod=:new.cod
   where ri.cod=:old.cod;
END IF;
END;
```

```
create trigger update_cod_alloggio
  after update of cod on alloggio
  for each row
declare
tipo VARCHAR2(25):='';

begin
  tipo:=:old.tipo_alloggio;

IF tipo='Hotel'  THEN
    update hotel h
    set h.cod=:new.cod
    where h.cod=:old.cod;
ELSIF tipo='BeB' THEN
    update beb bb
    set bb.cod=:new.cod
    where bb.cod=:old.cod;
ELSIF tipo='Appartamento' THEN
   update appartamento ap
   set ap.cod=:new.cod
   where ap.cod=:old.cod;
END IF;
END;
```

```
create trigger update_cod_ristorazione
  after update of cod on ristorazione
  for each row
declare
tipo VARCHAR2(25):='';

begin
  tipo:=:old.tipo_ristorazione;

IF tipo='Elenco completo'  THEN
    update braceria b
    set b.cod=:new.cod
    where b.cod=:old.cod;

    update sushibar sh
    set sh.cod=:new.cod
    where sh.cod=:old.cod;

    update pizzeria p
    set p.cod=:new.cod
    where p.cod=:old.cod;

ELSIF tipo='Braceria' THEN
    update braceria b
    set b.cod=:new.cod
    where b.cod=:old.cod;
ELSIF tipo='SushiBar' THEN
    update sushibar sh
    set sh.cod=:new.cod
    where sh.cod=:old.cod;
```

```
ELSIF tipo='Pizzeria' THEN
    update pizzeria p
    set p.cod=:new.cod
    where p.cod=:old.cod;
ELSIF tipo='Pizza e Brace' THEN
    update pizzeria p
    set p.cod=:new.cod
    where p.cod=:old.cod;

    update braceria b
    set b.cod=:new.cod
    where b.cod=:old.cod;
ELSIF tipo='Pizza e Sushi' THEN
    update pizzeria p
    set p.cod=:new.cod
    where p.cod=:old.cod;

    update sushibar sh
    set sh.cod=:new.cod
    where sh.cod=:old.cod;
ELSIF tipo='Brace e Sushi' THEN
    update braceria b
    set b.cod=:new.cod
    where b.cod=:old.cod;

    update sushibar sh
    set sh.cod=:new.cod
    where sh.cod=:old.cod;
END IF;
END;
```

```
create trigger check_dates
  before insert or update on recensione
  for each row

begin
  IF(:new.data >=SYSDATE) then
    :new.data:=SYSDATE;
  END IF;
END;

create trigger update_sevizi_alloggio
  after update of cod on alloggio
  for each row

begin
   update servizi_alloggio sa
   set sa.cod=:new.cod
   where sa.cod=:old.cod;
END;
```

10 - Crediti e conclusioni

In conclusione a questo progetto di Basi di Dati per l'anno accademico 2019/2020 possiamo concludere definendo gli strumenti utilizzati durante la progettazione in concomitanza allo sviluppo dell'applicativo Java per il progetto di OO (Object Orientation).

Utilità a i fini della progettazione:

- Guida ufficiale OracleDBMS per la definizione della sintassi PLSQL;
- StarUML – software per la creazione dei class diagrms;
- Oracle DB 11g Express edition come database locale;
- SQLdeveloper per la macro-gestione del database per i dati e metadati;

Strumenti per lo sviluppo del software Java accuratamente riportati nella documentazione di OO.

Fine del documento