UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Basi di Dati II

Sistema di prenotazioni Fixture

Prof. Michelangelo Ceci

Christopher Piemonte

659723

---

a.a. 2016/2017

1 - Database

AFC United
La AFC United è una squadra di calcio che ospita partite (note anche come Fixtures) contro una squadra avversaria. È richiesto un database per conservare le informazioni a supporto della prenotazione dei posti per assistere a una partita nello stadio del club nel corso di diverse stagioni. Prima dell'inizio di una stagione vengono organizzati gli incontri tra l'AFC United (la squadra di casa) e una squadra avversaria (la squadra in trasferta). Le partite sono guardate dagli spettatori che si sono registrati al club. Questi sono chiamati Passholders. I passholders devono prenotare i posti in anticipo per assistere ad ognuna delle 20 partite casalinghe di quella stagione. Una volta prenotato un posto (e effettuato il pagamento), il titolare del pass è emesso con uno o più biglietti. Questo perchè una Passholder può acquistare uno o più biglietti per i posti per ogni fixture. Per ogni biglietto conosciamo la persona associata, con nome, cognome, data di nascita e luogo di nascita. Un posto particolare può avere un'occupazione limitata. Questo è marcato in base al tipo di posto a sedere (per esempio "riservato al presidente", "riservato a spettatori disabili"). Ogni posto a sedere, a seconda della sua posizione / settore, ha un prezzo diverso, che può anche cambiare partita per partita.

1.1 - Progettazione Concettuale

Analisi delle specifiche

Termine	Riga	Motivo	Diventa
Incontri	4	sinonimo	Partite
Titolare del pass	7	sinonimo	Passholder
Posto a sedere	11, 12	livello di astrazione	Posto
Posizione	12	sinonimo	Anello

Standardizzare e Linearizzare

“È richiesto un database per conservare le informazioni a supporto della prenotazione dei posti per assistere a una partita nello stadio del club nel corso di diverse stagioni.” Diventa “È richiesto un database per conservare le informazioni a supporto della prenotazione dei posti per assistere a una partita nello stadio nel corso di diverse stagioni.”
“Prima dell’inizio di una stagione vengono organizzati gli incontri tra l’AFC United (la squadra di casa) e una squadra avversaria (la squadra in trasferta).” Diventa “Le partite vengono organizzate prima dell’inizio della stagione.“ “Le partite si svolgono tra l’AFC United e una squadra avversaria.”
“Le partite sono guardate dagli spettatori che si sono registrati al club. Questi sono chiamati Passholders.” Diventa “Gli spettatori registrati al club sono chiamati passholders. “
“Una volta prenotato un posto (e effettuato il pagamento), il titolare del pass è emesso con uno o più biglietti. Questo perchè una Passholder può acquistare uno o più biglietti per i posti per ogni fixture.” Diventa “Un passholder può acquistare uno o più biglietti per ogni partita.“

“Un biglietto viene assegnato ad un passholder alla prenotazione di ogni posto.”
“Per ogni biglietto conosciamo la persona associata, con nome, cognome, data di nascita e luogo di nascita.“ Diventa “Per biglietto conosciamo la persona associata, con nome, cognome, data di nascita e luogo di nascita.”
“Un posto particolare può avere un’occupazione limitata. Questo è marcato in base al tipo di posto a sedere (per esempio “riservato al presidente”, “riservato a spettatori disabili”).” Diventa “Un posto può essere di tipo speciale (per esempio “riservato al presidente”, “riservato a spettatori disabili”).”
“Ogni posto a sedere, a seconda della sua posizione / settore, ha un prezzo diverso, che può anche cambiare partita per partita.” Diventa “Un posto ha un prezzo che varia in base all’anello e al settore, che può anche cambiare per partita.”

Raggruppare per Concetti

Generali

- La AFC United è una squadra di calcio che ospita partite contro una squadra avversaria.
- È richiesto un database per conservare le informazioni a supporto della prenotazione dei posti per assistere a una partita nello stadio nel corso di diverse stagioni.

Partita

- Le partite vengono organizzate prima dell’inizio della stagione.
- Le partite si svolgono tra l’AFC United e una squadra avversaria.

Passholder

- Gli spettatori registrati al club sono chiamati passholders.

- I passholders devono prenotare i posti in anticipo prima di ognuna delle 20 partite della stagione.
- Un passholder può acquistare uno o più biglietti per ogni partita.

Biglietto

- Per biglietto conosciamo la persona associata, con nome, cognome, data di nascita e luogo di nascita.
- Un biglietto viene assegnato ad un passholder alla prenotazione di ogni posto.

Posto

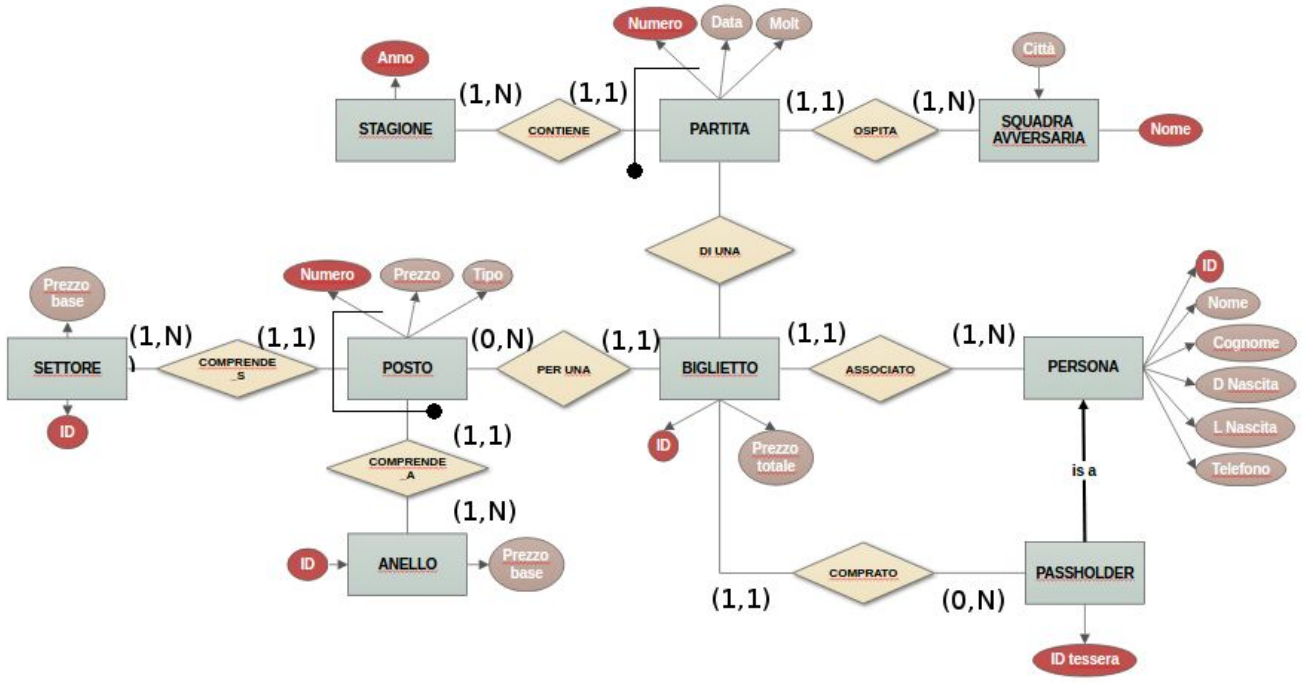
- Un posto può essere di tipo speciale (per esempio “riservato al presidente”, “riservato a spettatori disabili”).
- Un posto ha un prezzo che varia in base all’anello e al settore, che può anche cambiare per partita.

Glossario dei Termini

Termine	Descrizione	Tipo	Collegamenti
Stagione	Una serie di partite.	E	Partita
Partita	Evento tra la AFC United ed una Squadra avversaria.	E	Stagione, Biglietto
Squadra avversaria	Gioca una Partita.	E	Partita
Passholder	Spettatore registrato al club. Può comprare Biglietti.	E	Biglietto
Biglietto	Diritto di u posto ad una partita.	E	Passholder, Partita,Posto
Settore	Divisione verticale degli spalti.	E	Posto
Anello	Divisione orizzontale degli spalti.	E	Posto

Posto	Posto a sedere	E	Settore, Anello, Biglietto

Schema E-R



Vincoli e Business Rules

- Il prezzo del posto è calcolato sommando quelli dell'anello e del settore di cui fa parte.
- Il prezzo del biglietto è calcolato moltiplicando il prezzo del posto per il moltiplicatore della partita.

1.2 - Progettazione Logica

Tavola dei Volumi

Assunzioni fatte:

- 10 anni,
- caso pessimo si vendono tutti i biglietti(10,000) per tutte le partite(20) di tutte le stagioni(10)
- 4 anelli
- ci sono 20 partite, 30 settimane all'anno in cui il sistema è attivo, ovvero 210 giorni
- 100 passholder nuovi al giorno sono: 210 * 100 = 21,000 passholder all'anno, per 10 anni = 210,000
- un passholder compra i biglietti in media per un'altra persona

Tavola dei Volumi		
Concetto	**Tipo**	**Volume**

Stagione	E	10
Contiene	R	200
Partita	E	200
Ospita	R	200
Squadra avversaria	E	20
Di una	R	2,000,000
Settore	E	10
Comprende_S	R	10,000
Anello	E	4
Comprende_A	R	10,000
Posto	E	10,000
Per un	R	2,000,000
Biglietto	E	2,000,000
Associato	R	2,000,000
Persona	E	420,000
Comprato	R	2,000,000
Passholder	E	210,000

Tavola delle Operazioni

OP1 ➡ Registrazione di un nuovo biglietto (1100 al giorno)

OP2 ➡ Registrazione di un nuovo passholder (100 al giorno)

OP3 ➡ Stampa la mappa dello stadio per una data partita e settore, con tutte le informazioni su ogni spettatore (500 volte a settimana)

OP4 ➡ Stampa le informazioni sul numero di spettatori per partita e per settore per tutta la stagione (500 volta all'anno)

Tavola delle Operazioni		
Operazione	**Tipo**	**Frequenza**
OP1	**I**	1100 / giorno

OP2	**I**	100 / giorno
OP3	**I**	500 / sett
OP4	**B**	500 / anno

Tavola degli Accessi

Tavola degli Accessi OP1			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Biglietto	E	1	W
Per un	R	1	L
Posto	E	1	L
Di una	R	1	L
Partita	E	1	L
Associato	R	1	L
Persona	E	1	L
Comprato	R	1	L
Passholder	E	1	L

Tavola degli Accessi OP2			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Persona	E	1	W
Passholder	E	1	W

Tavola degli Accessi OP3			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Partita	E	1	R
Di una	R	10,000	R
Biglietto	E	10,000	R

Associato	R	10,000	R
Persona	E	10,000	R
Per un	R	10,000	R
Posto	E	10,000	R
Comprende S	R	10,000	R
Settore	E	10	R

Tavola degli Accessi OP4			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Stagione	E	1	R
Contiene	R	20	R
Partita	E	20	R
Di una	R	200,000	R
Biglietto	E	200,000	R
Per un	R	200,000	R
Posto	E	10,000	R
Comprende S	R	10,000	R
Settore	E	10	R

Ristrutturazione

Si è scelto di collassare verso l'alto l'unica generalizzazione, in quanto la relazione coinvolta con l'entità padre riguardava anche l'entità figlia, no esprimibile in contesto relazionale. Un campo *tipo* è stato aggiunto per indicare una persona registrata al club (Passholder).

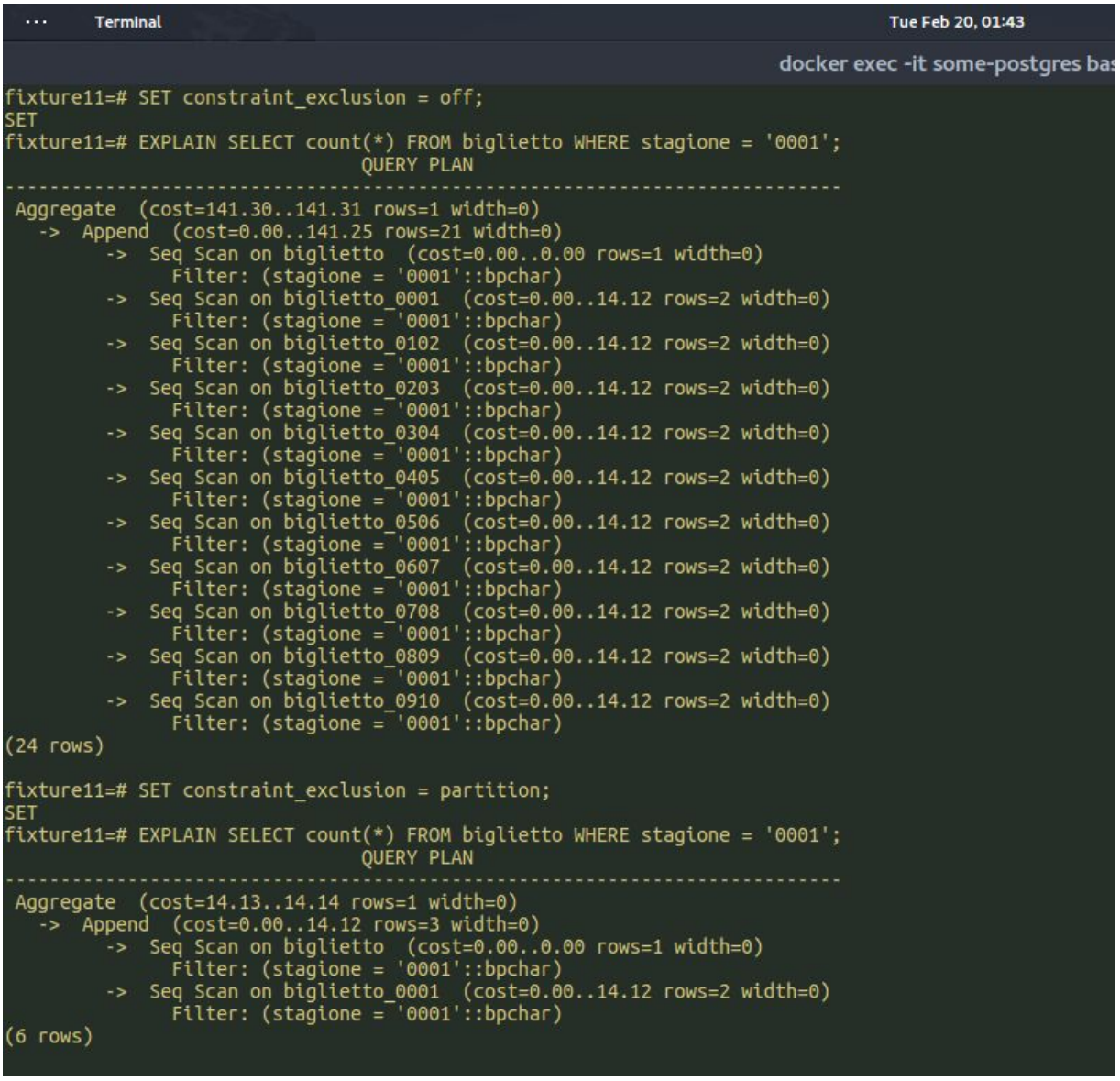
Schema Object-Relational

1.3 - Progettazione Fisica

Partitioning

È stato effettuato un partizionamento sulla tabella Biglietti, inquanto quella più utilizzata dalle operazioni ed in quanto presenta i volumi più grandi. Il criterio di partizionamento scelto è stato quello di dividere la tabella Biglietto principale in tante tabelle quante le stagioni, infatti queste risultano la granularità di dettaglio più bassa nelle operazioni.

Attraverso dei trigger ed alla ereditarietà della tabella principale, ognuna delle sottotabelle è stata creata con dei vincoli ***CHECK*** in modo da riconoscere su quale partizione indirizza reindirizzare la query rivolta alla tabella principale.




```
fixture11=# SET constraint_exclusion = off;
SET
fixture11=# EXPLAIN SELECT count(*) FROM biglietto WHERE stagione = '0001';
                                QUERY PLAN
--------------------------------------------------------------------------
 Aggregate  (cost=141.30..141.31 rows=1 width=0)
   ->  Append  (cost=0.00..141.25 rows=21 width=0)
         ->  Seq Scan on biglietto  (cost=0.00..0.00 rows=1 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0001  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0102  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0203  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0304  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0405  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0506  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0607  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0708  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0809  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0910  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
(24 rows)

fixture11=# SET constraint_exclusion = partition;
SET
fixture11=# EXPLAIN SELECT count(*) FROM biglietto WHERE stagione = '0001';
                                QUERY PLAN
--------------------------------------------------------------------------
 Aggregate  (cost=14.13..14.14 rows=1 width=0)
   ->  Append  (cost=0.00..14.12 rows=3 width=0)
         ->  Seq Scan on biglietto  (cost=0.00..0.00 rows=1 width=0)
               Filter: (stagione = '0001'::bpchar)
         ->  Seq Scan on biglietto_0001  (cost=0.00..14.12 rows=2 width=0)
               Filter: (stagione = '0001'::bpchar)
(6 rows)
```

Trigger

Descrizione

Verifica che il compratore di un biglietto è una persona di tipo PASSHOLDER. Questo trigger è scaturito dalla eliminazione della generalizzazione e delle relazione coinvolte.

```
/* trigger che controlla che chi compra è un passholder */
CREATE OR REPLACE FUNCTION check_compratore_passholder()
RETURNS TRIGGER AS $$
DECLARE
    tipo_compratore persona.tipo%TYPE;
BEGIN
    SELECT tipo INTO tipo_compratore
    FROM persona
    WHERE NEW.compratore = persona.cf;
    IF tipo_compratore <> 'PASSHOLDER' THEN
        RAISE NOTICE 'check_compratore_passholder il compratore non è un passholder';
        RAISE EXCEPTION foreign_key_violation;
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_compratore_passholder_trigger
    BEFORE INSERT ON biglietto
    FOR EACH ROW
    EXECUTE PROCEDURE check_compratore_passholder();
```

Descrizione

Controlla che il posto per una determinata partita non sia già occupato.

```
/* controlla se il posto per quella partita è occpato */
CREATE OR REPLACE FUNCTION posto_occupato()
RETURNS TRIGGER AS $$
DECLARE
    exists INTEGER;   /* partita.molt%type */
BEGIN
    SELECT COUNT(*)  INTO exists
    FROM biglietto as b
    WHERE b.partita = NEW.partita AND b.stagione = NEW.stagione
          AND b.posto = NEW.posto AND b.settore = NEW.settore AND b.anello = NEW.anello;

    IF exists <> 1 THEN
        RAISE NOTICE 'posto occupato';
        RAISE EXCEPTION 'posto occupato --> %', NEW.posto;
    END IF;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER posto_occupato_trigger
    BEFORE INSERT ON biglietto
    FOR EACH ROW
    EXECUTE PROCEDURE posto_occupato();
```

Descrizione

All'inserimento di una nuova Stagione viene creata la relativa partizione della tabella Biglietto con sintassi Biglietto_<stagione>.

```
/* trigger che crea automaticamente la partizione di biglietti all'inserimento di una nuova stagione */
CREATE OR REPLACE FUNCTION create_partition_table()
RETURNS TRIGGER AS $$
BEGIN
    EXECUTE format('CREATE TABLE IF NOT EXISTS biglietto_' || NEW.anno || '(
                        CHECK ( stagione = ''' || NEW.anno || ''' )
                    ) INHERITS (biglietto);');
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER create_partition_biglietto_trigger
    AFTER INSERT ON stagione
    FOR EACH ROW
    EXECUTE PROCEDURE create_partition_table();
```

Descrizione

Se il prezzo del biglietto non è specificato, questo viene calcolato coma da specifiche.

```
/* trigger prezzo */
CREATE OR REPLACE FUNCTION calcola_prezzo()
RETURNS TRIGGER AS $$
DECLARE
    prezzo_posto NUMERIC(5,2);    /* partita.molt%type */
    molt_partita NUMERIC(3,2);
BEGIN
    SELECT prezzo INTO prezzo_posto
    FROM posto
    WHERE posto.numero = NEW.posto AND posto.settore = NEW.settore;

    SELECT molt INTO molt_partita
    FROM partita
    WHERE partita.stagione = NEW.stagione AND partita.numero = NEW.partita;

    NEW.prezzo_totale := prezzo_posto * molt_partita;
    RAISE NOTICE 'i want to print calcola_prezzo % and %', NEW.prezzo_totale, molt_partita;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER calcola_prezzo_trigger
    BEFORE INSERT ON biglietto
    FOR EACH ROW
    WHEN (NEW.prezzo_totale IS NULL OR NEW.prezzo_totale = 0.00)
    EXECUTE PROCEDURE calcola_prezzo();
```

Descrizione
Allo stesso modo, il prezzo del posto, se non specificato altrimenti, viene calcolato come da specifiche.

```
/* trigger che calcolaprezzo posto che se è null(default) lo calcola da anello e settore*/
CREATE OR REPLACE FUNCTION prezzo_posto()
RETURNS TRIGGER AS $$
DECLARE
    prezzo_settore NUMERIC(5,2);
    prezzo_anello  NUMERIC(5,2);
BEGIN
    SELECT prezzo_base INTO prezzo_settore
    FROM settore
    WHERE settore.id = NEW.settore;

    SELECT prezzo_base INTO prezzo_anello
    FROM anello
    WHERE anello.id = NEW.anello;

    NEW.prezzo := prezzo_settore + prezzo_anello;
    RETURN NEW;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER prezzo_posto_trigger
    BEFORE INSERT ON posto
    FOR EACH ROW
    WHEN (NEW.prezzo IS NULL OR NEW.prezzo = 0.00)
    EXECUTE PROCEDURE prezzo_posto();
```

2 - Sistema

Una applicazione client/server è stata creata per interagire con il database e per il popolamneto automatico delle tuple.

2.1 - Server

Per l'Object-Relational mapping è stato utilizzato **Hibernate**, mentre l'infrastruttura dell'applicazione è stata quella utilizzata da **Spring Framework** (in particolare Spring Boot), ovvero:

- **Repository**: layer di accesso ai dati, utilizza direttamente le Hibernate entity.
- **Service**: implementa la logica di business.
- **Controller**: implementa le Servlet API, ricevendo le chiamate dal Servlet Container.

2.2 - Client

Il client è stato creato il framework Javascript AngularJS, che permette di realizzare Single Page Application. Ha l'obiettivo di semplificare lo sviluppo e il test di questa tipologia di applicazioni fornendo un framework lato client con architettura MVC (Model View Controller). Il framework AngularJS lavora leggendo prima la pagina HTML, che ha incapsulati degli attributi specifici, che vengono interpretati

come delle direttive (comandi) per legare le parti di ingresso e uscita della pagina al modello che è rappresentato da variabili standard JavaScript.

Tutto l'elaborato prodotto è disponibile all'indirizzo: https://github.com/chrisPiemonte/Fixture.git

Per l'utilizzo: https://github.com/chrisPiemonte/Fixture/blob/master/README.md