Università degli Studi di Napoli Federico II

Corso di Laurea in Informatica Insegnamento di Basi di Dati e Sistemi Informativi I Anno accademico 2018/2019

Heaven's Rail v. 1.0

Progettazione e sviluppo di una base di dati relazionale per la gestione di un sistema ferroviario.

Autore:
Giuseppe Romano
Matr: 566/1739

Indice

1. Descrizione Progetto

L'azienda **Heaven's Rail** intende espandere il proprio business in altri paesi della Comunità Europea, in tal contesto l'azienda intende operare sul territorio italiano fornendo servizi di trasporto ad alta velocità.

L'azienda utilizzerá la rete ferroviaria già esistente ma ha necessità di automatizzare diversi aspetti organizzativi, tra cui la gestione delle tratte di percorrenza, degli orari di partenza/arrivo, e dei relativi ritardi effettivi sulle corse effettuate.

Inoltre l'azienda intende fornire agli utenti un sistema di booking online e di consultazione dell'orario di tutte le tratte di percorrenza (gestendo eventuali cambi treno per tratte non dirette).

Lo scopo del presente documento è quello di progettare e realizzare una base dati ed una interfaccia grafica per la realizzazione dei requisiti dell'azienda.

1.1. Raccolta dei requisiti

Si intende progettare e realizzare una base di dati relazionale che possa essere d'ausilio alla gestione della rete ferroviaria.

La base di dati conterrà tutte le informazioni relative al parco treni a disposizione con particolare dettaglio sulla tipologia di ogni treno, nonché il codice identificativo, la sua velocità nominale e inoltre di quante carrozze si compone ogni convoglio.

Inoltre per ogni stazione si vuole catalogare l'indirizzo, eventuali recapiti telefonici, il numero di piattaforme di cui essa è equipaggiata ed inoltre una serie di informazioni di utilitá inerenti i servizi in stazione (come ad esempio: accesso per disabili, servizio ristorante, servizio taxi ed altro…).

Facendo seguito alle richieste del cliente, in seguito viene descritta la progettazione tecnica della base dati al fine di realizzare un sistema che abbia ottime prestazioni in termini di quantitá di dati gestiti e in termini di accessi simultanei al servizio di booking e consultazione degli orari di partenza e arrivo presso le stazioni ferroviarie ed online attraverso un'interfaccia Swing dedicata.

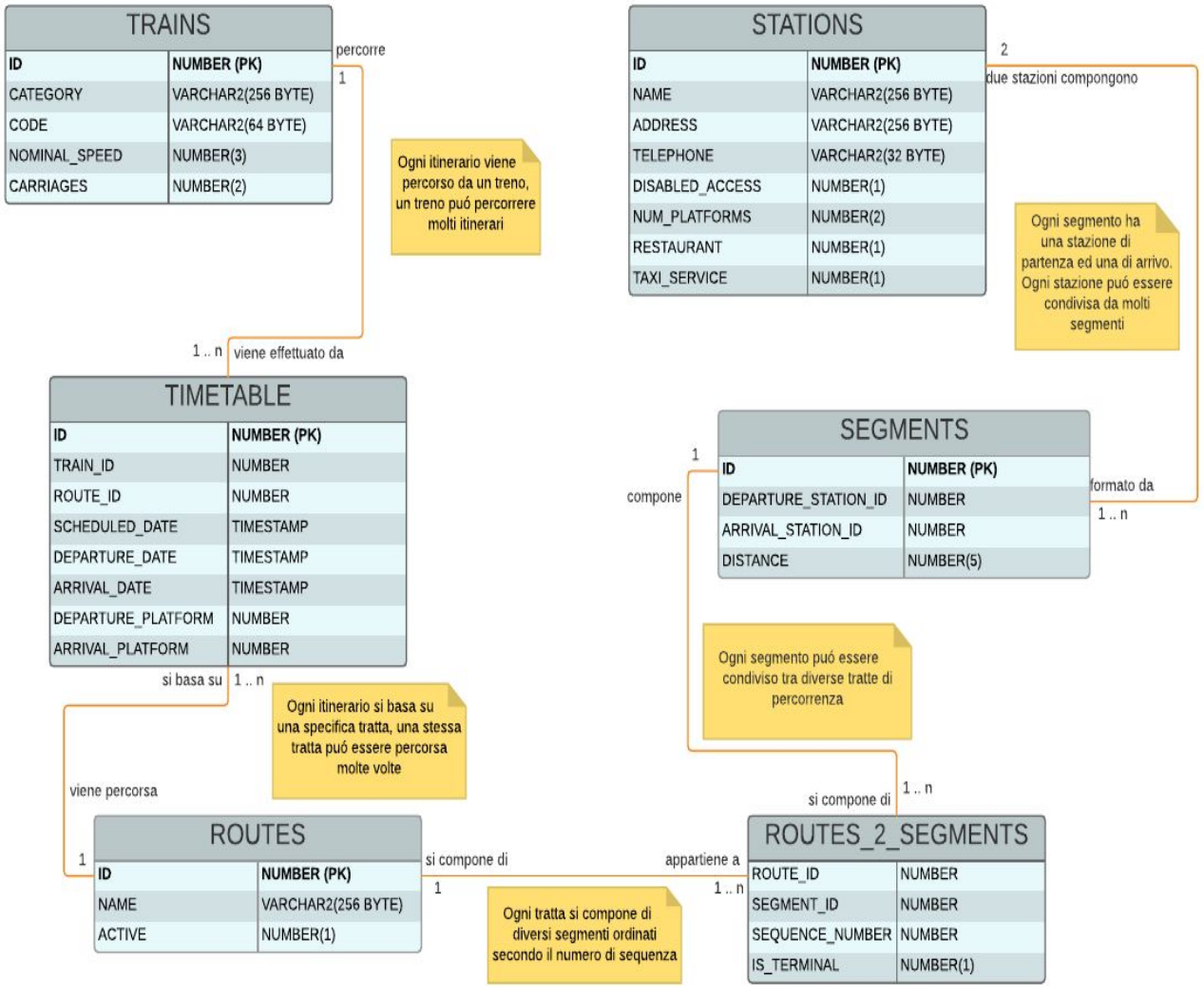
2. Progettazione Concettuale

In questa sezione viene effettuata la definizione dei concetti ad un livello di astrazione più alto. Il livello di astrazione fa si che ci si concentri soltanto sui concetti nel dominio del business senza fare cenno ad alcuna problematica di implementazione tecnologica concreta. Per la definizione dei concetti si fa uso del linguaggio di modellazione UML.

Ciò che si evidenzia in questa sezione sono i concetti del dominio di business (anche conosciuti come entitá) e le relazioni che intercorrono tra di essi.

Per la natura intrinseca del dominio di business, la realizzazione concettuale più adatta al problema è quella basata sulla **teoria dei grafi**. Un grafo è una struttura dati complessa composta da un insieme di vertici (*stazioni*) ed un insieme di archi orientati (*segmenti*) che connettono i diversi vertici. Inoltre una tratta di percorrenza può essere vista come un sottografo in cui gli archi orientati sono un sottoinsieme di tutti i possibili archi.

Class Diagram



2.1. Dizionario delle Classi

Classe	Descrizione	Attributi
TRAINS	Modella tutti i treni a disposizione della compagnia ferroviaria, ogni treno ha una categoria, un codice identificativo, una velocità nominale ed è composto da diverse carrozze.	**ID (number):** Chiave primaria tecnica, identifica univocamente un treno. **CATEGORY (string):** La categoria di ogni treno, essa può assumere valori del tipo: “FrecciaRossa”, “Frecciargento” etc etc. **CODE (string):** Codice identificativo del treno di una spacifica categoria (es: Frecciarossa 9563) **NOMINAL_SPEED (number):** La velocità nominale che il treno è il grado di percorrere in media. Essa è espressa in Km/h. **CARRIAGES (number):** Il numero di carrozze di cui si compone il treno
STATIONS	Modella tutte le stazioni ferroviarie su cui la compagnia opera. Ogni stazione viene qualificata da un nome ed un indirizzo (con eventuali recapiti telefonici), inoltre definisce diversi servizi come accesso per disabili, servizio ristorante, servizio taxi ed altro. NOTA: Secondo la teoria dei grafi, ogni stazione rappresenta un vertice del grafo.	**ID (number):** Chiave primaria tecnica, identifica univocamente una stazione. **NAME (string):** Il nome della stazione ferroviaria (es: Napoli Centrale, Milano Centrale) **ADDRESS (string):** L’indirizzo della stazione. **TELEPHONE (string):** Il recapito telefonico della stazione. **DISABLED_ACCESS (number):** Informazione di servizio per accesso consentito ai disabili. **NUM_PLATFORMS (number):** Il numero di binari disponibile in

		stazione. **RESTAURANT (number):** Informazione di servizio per servizio ristorante in stazione. **TAXI_SERVICE (number):** Informazione di servizio per servizio taxi fuori stazione.
SEGMENTS	Modella i segmenti che connettono due stazioni specifiche. Ogni segmento può essere inteso come la rete ferroviaria fisica che connette due stazioni. NOTA: Secondo la teoria dei grafi, ogni segmento rappresenta un arco orientato tra due vertici	**ID (number):** Chiave primaria tecnica, identifica univocamente un segmento. **DEPARTURE_STATION_ID (number):** Chiave esterna verso la stazione di partenza. **ARRIVAL_STATION_ID (number):** Chiave esterna verso la stazione di arrivo. **DISTANCE (number):** La distanza espressa in Kilometri tra le due stazioni connesse dal segmento.
ROUTES	Modella le tratte di percorrenza su cui la compagnia opera. NOTA: Secondo la teoria dei grafi, ogni tratta può essere visto come un sottografo aciclico in cui il dominio dei vertici sono tutte le stazioni ed il dominio degli archi orientati sono l'insieme dei segmenti per quella specifica tratta.	**ID (number):** Chiave primaria tecnica, identifica univocamente una tratta di percorrenza. **NAME (string):** Il nome mnemonico della tratta di percorrenza, come ad esempio: "Tratta 1 (Napoli Centrale - Milano Centrale) **ACTIVE (number):** Questo flag indica se la tratta di percorrenza è attiva oppure no. Una tratta non è attivo quando i segmenti non sono contigui oppure se vi sono cicli interni. Ogni tratta deve essere orientata in una sola direzione e non può avere ritorni su una stessa stazione.

ROUTES_2_SEGMENTS	Questa entitá modella la relazione many-to-many tra ROUTES e SEGMENTS. In aggiunta modella delle informazioni aggiuntive riguardo il numero di sequenza e il flag se un tale segmento è capolinea oppure no. NOTA: Secondo la teoria dei grafi, questa entitá modella il sottoinsieme degli archi orientati di un sottografo.	**ROUTE_ID (number):** Chiave esterna verso la tratta di percorrenza. **SEGMENT_ID (number):** Chiave esterna verso il segmento che connette due stazioni. **SEQUENCE_NUMBER (number):** Il numero di sequenza secondo il quale i segmenti compongono la tratta di percorrenza. **IS_TERMINAL (number):** Questo flag indica che il segmento è l'ultimo della lista.
TIMETABLE	Modella gli orari di partenza/arrivo degli itinerari definiti dalle tratta di percorrenza.	**ID (number):** Chiave primaria tecnica, identifica univocamente un orario di partenza/arrivo. **TRAIN_ID (number):** Chiave esterna verso il treno. **ROUTE_ID (number):** Chiave esterna verso la tratta di percorrenza. **SCHEDULED_DATE (timestamp):** L'orario di partenza previsto. **DEPARTURE_DATE (timestamp):** L'orario di partenza effettivo. Esso può essere diverso dall'orario previsto in quanto la partenza può effettuarsi in ritardo e/o anticipo. **ARRIVAL_DATE (timestamp):** L'orario di arrivo effettivo. Si noti che l'orario di arrivo previsto viene calcolato come formula considerando la distanza della tratta (Km) e la velocità nominale del treno (km/h).

2.2. Dizionario delle Associazioni

Nome	Descrizione	Entitá coinvolte
TRAINS.[percorre]	Esprime la relazione tra un treno e gli orari di partenza in relazione alla tratta di percorrenza	TRAINS e TIMETABLE
TIMETABLE.[viene effettuato da]	Esprime il concetto che uno o più orari di partenza su tratte di percorrenza diverse possono essere effettuate dallo stesso treno o più treni	TRAINS e TIMETABLE
TIMETABLE.[si basa su]	Esprime il concetto che un orario di partenza fa riferimento ad una specifica tratta di percorrenza	TIMETABLE e ROUTES
ROUTES.[viene percorsa]	Esprime la relazione tra la definizione della tratta e l'effettiva percorrenza in merito agli orari di partenza	TIMETABLE e ROUTES
ROUTES.[si compone di]	Esprime la relazione di composizione della tratta, essa è composta da uno o più segmenti (archi secondo la teoria dei grafi)	ROUTES e ROUTES_2_SEGMENTS
ROUTES_2_SEGMENTS.[appartiene a]	Esprime il concetto di appartenenza di un segmento in funzione della tratta. Si ricordi che ogni segmento può essere condiviso da diverse tratte, questa relazione indica tale segmento in che posizione (sequenza) si trova rispetto alla tratta stessa.	ROUTES_2_SEGMENTS e ROUTES
ROUTES_2_SEGMENTS.[si compone di]	Esprime la relazione di composizione della tratta, essa è composta da uno o più segmenti (archi secondo la teoria dei grafi)	ROUTES_2_SEGMENTS e SEGMENTS
SEGMENTS.[compone]	Esprime il concetto di composizione di una tratta formata da uno o più segmenti	ROUTES_2_SEGMENTS e SEGMENTS
SEGMENTS.[formato da]	Esprime la relazione many-to-two in cui ogni segmento è in duplice relazione con l'entità stazione. Vi è una stazione di partenza ed una stazione di arrivo	SEGMENTS, STATIONS

STATIONS.[due stazioni compongono]	Questa relazione indica che una coppia di stazioni diverse tra loro formato un segmento (o arco orientato)	SEGMENTS, STATIONS

2.3. Dizionario dei Vincoli

Nome Vincolo	**Descrizione**
Unique Train	Questo vincolo implementa il controllo di univocità della coppia di valori categoria e codice. Due treni non possono avere la stessa coppia di valori.
Unique Station	Il nome della stazione deve essere univoco.
Unique Segment	Ogni segmento formato da una coppia di stazioni (partenza e arrivo) devono essere diverse tra loro. Tale vincolo impone la proprietà di aciclicità della tratta (sottografo).
Unique Route	Ogni tratta di percorrenza deve avere un nome univoco. Il nome della tratta non è altro che un nome mnemonico che individua la tratta, ad esempio “Tratta 1 (Napoli Centrale - Milano Centrale)”

3. Progettazione Logica

In questa sezione viene tradotto lo schema concettuale in uno schema logico più vicino al “mondo relazionale”. Negli schemi relazionali che seguono le chiavi primarie sono indicate in **grassetto** mentre le chiavi esterne con una sottolineatura.

3.1. Schemi logici

TRAINS(**ID**, CATEGORY, CODE, NOMINAL_SPEED, CARRIAGES)

Vincoli: Nessuno

STATIONS(**ID**, NAME, ADDRESS, TELEPHONE, DISABLED_ACCESS, NUM_PLATFORMS, RESTAURANT, TAXI_SERVICE)

Vincoli:

- Vincolo di unicitá su NAME

SEGMENTS(**ID**, DEPARTURE_STATION_ID, ARRIVAL_STATION_ID, DISTANCE)

Vincoli:

- Vincolo di chiave esterna DEPARTURE_STATION_ID → STATIONS.ID
- Vincolo di chiave esterna ARRIVAL_STATION_ID → STATIONS.ID
- Vincolo di check DEPARTURE_STATION_ID <> ARRIVAL_STATION_ID
- Vincolo di unicitá su DEPARTURE_STATION_ID e ARRIVAL_STATION_ID

ROUTES(**ID**, NAME, ACTIVE)

Vincoli:

- Vincolo di unicitá su NAME

ROUTES_2_SEGMENTS(ROUTE_ID, SEGMENT_ID, SEQUENCE_NUMBER, IS_TERMINAL)

Vincoli:

- Vincolo di chiave esterna ROUTE_ID → ROUTES.ID
- Vincolo di chiave esterna SEGMENT_ID→ SEGMENTS.ID

TIMETABLE(ID, TRAIN_ID, ROUTE_ID, SCHEDULED_DATE, DEPARTURE_DATE, ARRIVAL_DATE, DEPARTURE_PLATFORM, ARRIVAL_PLATFORM)

Vincoli:

- Vincolo di chiave esterna TRAIN_ID → TRAINS.ID
- Vincolo di chiave esterna ROUTE_ID → ROUTES.ID

4. Progettazione Fisica

La base di dati verrà implementata nel sistema per la gestione di basi di dati Oracle XE 11g.

Alcuni dettagli implementativi del progetto saranno modificati al fine di sfruttare al meglio le funzionalità del DBMS Oracle XE 11g. Poiché Oracle non implementa il tipo boolean, questo è stato simulato con un valore NUMBER(1) i cui valori possono essere 0 = FALSE e 1 = TRUE, inoltre sono state implementate alcune funzioni di calcolo ed alcuni triggers per la gestione di vincoli più complessi. Infine sono state anche definite delle viste sui dati al fine di poter meglio visualizzare i dati.

4.1. Definizione Tabelle

In questa sezione vengono elencate tutte le tabelle.

4.1.1. Tabella TRAINS

```
CREATE TABLE TRAINS (
    ID                  NUMBER PRIMARY KEY,
    CATEGORY            VARCHAR2(256 BYTE) NOT NULL,
    CODE                VARCHAR2(64 BYTE) NOT NULL,
    NOMINAL_SPEED       NUMBER(3),
    CARRIAGES           NUMBER(2)
);
/

COMMENT ON TABLE TRAINS IS 'The table modelling all the trains belonging to the railway company';
COMMENT ON COLUMN TRAINS.ID IS 'The primary key of the table';
COMMENT ON COLUMN TRAINS.CATEGORY IS 'The train''s category like ''Frecciarossa'', ''Frecciargento'', ''Intercity'' etc etc';
COMMENT ON COLUMN TRAINS.CODE IS 'The identification code of the train';
COMMENT ON COLUMN TRAINS.NOMINAL_SPEED IS 'The nominal speeed of the train. It is expressed in kilometers/hour';
```

```
COMMENT ON COLUMN TRAINS.CARRIAGES IS 'The number of carriages attached to
the train';
/
```

4.1.2. Tabella STATIONS

```
CREATE TABLE STATIONS (
    ID                  NUMBER PRIMARY KEY,
    NAME                VARCHAR2(256 BYTE) NOT NULL,
    ADDRESS             VARCHAR2(256 BYTE) NOT NULL,
    TELEPHONE           VARCHAR2(32 BYTE),
    DISABLED_ACCESS     NUMBER(1) DEFAULT 0,
    NUM_PLATFORMS       NUMBER(2),
    RESTAURANT          NUMBER(1) DEFAULT 0,
    TAXI_SERVICE        NUMBER(1) DEFAULT 0,

    CONSTRAINT STATION_NAME_UK
      UNIQUE (NAME)
);
/
COMMENT ON TABLE STATIONS IS 'The table modelling all the stations on which the
railway company performs segments';
COMMENT ON COLUMN STATIONS.ID IS 'The primary key of the table';
COMMENT ON COLUMN STATIONS.NAME IS 'The station''s name';
COMMENT ON COLUMN STATIONS.ADDRESS IS 'The station''s address';
COMMENT ON COLUMN STATIONS.TELEPHONE IS 'The station''s telephone number';
COMMENT ON COLUMN STATIONS.DISABLED_ACCESS IS 'The flag indicating if the
station is compliant to the accessibility rules';
COMMENT ON COLUMN STATIONS.NUM_PLATFORMS IS 'The number of platforms
provided by the station';
COMMENT ON COLUMN STATIONS.RESTAURANT IS 'The flag indicating if the station is
equipped with restaurants';
COMMENT ON COLUMN STATIONS.TAXI_SERVICE IS 'The flag indicating if there is a
taxi station near the station';
```

4.1.3. Tabella SEGMENTS

```
CREATE TABLE SEGMENTS (
    ID                      NUMBER PRIMARY KEY,
    DEPARTURE_STATION_ID    NUMBER NOT NULL,
    ARRIVAL_STATION_ID      NUMBER NOT NULL,
    DISTANCE                NUMBER(5),
```

```
    CONSTRAINT SEGMENTS_2_DEP_STATION
      FOREIGN KEY (DEPARTURE_STATION_ID)
      REFERENCES STATIONS (ID),

    CONSTRAINT SEGMENTS_2_ARR_STATION
      FOREIGN KEY (ARRIVAL_STATION_ID)
      REFERENCES STATIONS (ID),

    CONSTRAINT SEGMENTS_CK_DEP_ARR_ST CHECK
                (DEPARTURE_STATION_ID <> ARRIVAL_STATION_ID),

    CONSTRAINT SEGMENTS_UK UNIQUE (DEPARTURE_STATION_ID,
                                   ARRIVAL_STATION_ID)
);
/
COMMENT ON TABLE SEGMENTS IS 'The table modelling a segment connecting two
single stations in one way (one direction only).';
COMMENT ON COLUMN SEGMENTS.ID IS 'The primary key of the table.';
COMMENT ON COLUMN SEGMENTS.DEPARTURE_STATION_ID IS 'The departure
station id. It is the starting point of the route node.';
COMMENT ON COLUMN SEGMENTS.ARRIVAL_STATION_ID IS 'The arrival station id. It
is the end point of the route node.';
COMMENT ON COLUMN SEGMENTS.DISTANCE IS 'The distance between the two
stations. It is expressed in kilometers.';
/
```

4.1.4. Tabella ROUTES

```
CREATE TABLE ROUTES (
    ID                  NUMBER PRIMARY KEY,
    NAME                VARCHAR2(256 BYTE) NOT NULL,
    ACTIVE              NUMBER(1) DEFAULT 0,

        CONSTRAINT ROUTES_UK UNIQUE (NAME)
);
/
COMMENT ON TABLE ROUTES IS 'The table modelling a route composed by several
segments linked together';
COMMENT ON COLUMN ROUTES.ID IS 'The primary key of the table.';
COMMENT ON COLUMN ROUTES.NAME IS 'The mnemonic name of the route';
COMMENT ON COLUMN ROUTES.ACTIVE IS 'The flag indicating wether the route is well
linked between segments or not';
/
```

4.1.5. Tabella ROUTES_2_SEGMENTS

```
CREATE TABLE ROUTES_2_SEGMENTS (
    ROUTE_ID                NUMBER NOT NULL,
    SEGMENT_ID              NUMBER NOT NULL,
    SEQUENCE_NUMBER         NUMBER(3),
    IS_TERMINAL             NUMBER(1) DEFAULT 0,

    CONSTRAINT FK_TO_ROUTES
      FOREIGN KEY (ROUTE_ID)
      REFERENCES ROUTES (ID),

    CONSTRAINT FK_TO_SEGMENTS
      FOREIGN KEY (SEGMENT_ID)
      REFERENCES SEGMENTS (ID),

    CONSTRAINT ROUTES_2_SEGMENTS_UK UNIQUE (ROUTE_ID, SEGMENT_ID)
);
/
COMMENT ON TABLE ROUTES_2_SEGMENTS IS 'The table modelling the many-to-many
relationship between the segments and the routes';
COMMENT ON COLUMN ROUTES_2_SEGMENTS.ROUTE_ID IS 'The foreign key pointing
to route segments';
COMMENT ON COLUMN ROUTES_2_SEGMENTS.SEGMENT_ID IS 'The foreign key
pointing to routes';
COMMENT ON COLUMN ROUTES_2_SEGMENTS.SEQUENCE_NUMBER IS 'The
sequence indicating the order of the segments which compose the route';
COMMENT ON COLUMN ROUTES_2_SEGMENTS.IS_TERMINAL IS 'The flag indicating
that the route end here. In other word, the segment is the last of the route';
/
```

4.1.6. Tabella TIMETABLE

```
CREATE TABLE TIMETABLE (
    ID                      NUMBER PRIMARY KEY,
    TRAIN_ID                NUMBER NOT NULL,
    ROUTE_ID                NUMBER NOT NULL,
    SCHEDULED_DATE          TIMESTAMP NOT NULL,
    DEPARTURE_DATE          TIMESTAMP,
    ARRIVAL_DATE            TIMESTAMP,
    DEPARTURE_PLATFORM      NUMBER(2),
    ARRIVAL_PLATFORM        NUMBER(2),

    CONSTRAINT TIMETABLE_2_TRAINS
```

```
    FOREIGN KEY (TRAIN_ID)
    REFERENCES TRAINS (ID),

  CONSTRAINT TIMETABLE_2_ROUTES
    FOREIGN KEY (ROUTE_ID)
    REFERENCES ROUTES (ID)
);
/
COMMENT ON TABLE TIMETABLE IS 'The table modelling all the scheduled routes at
specific date/time';
COMMENT ON COLUMN TIMETABLE.ID IS 'The primary key of the table';
COMMENT ON COLUMN TIMETABLE.TRAIN_ID IS 'The foreign key pointing to trains';
COMMENT ON COLUMN TIMETABLE.ROUTE_ID IS 'The foreign key pointing to routes';
COMMENT ON COLUMN TIMETABLE.SCHEDULED_DATE IS 'The scheduled departure
date/time';
COMMENT ON COLUMN TIMETABLE.DEPARTURE_DATE IS 'The real departure
date/time';
COMMENT ON COLUMN TIMETABLE.ARRIVAL_DATE IS 'The real arrival date/time';
COMMENT ON COLUMN TIMETABLE.DEPARTURE_PLATFORM IS 'The departure
platform number';
COMMENT ON COLUMN TIMETABLE.ARRIVAL_PLATFORM IS 'The arrival platform
number';
/
```

4.2. Definizione Viste

In questa sezione vengono elencate tutte le viste.

4.2.1. Vista BOOKING_VIEW

Questa vista effettua una join tra TIMETABLE, ROUTES e SEGMENTS allo scopo di ottenere una vista normalizzata sui dati inseriti.

```
CREATE OR REPLACE VIEW BOOKING_VIEW AS
SELECT DISTINCT
    tt.ID,
    sg.DEPARTURE_STATION_ID,
    sg.ARRIVAL_STATION_ID,
    sg.DISTANCE,
    COMPUTE_DEPARTURE_DATE(tt.ID, sg.DEPARTURE_STATION_ID)
DEPARTURE_DATE,
    COMPUTE_ARRIVAL_DATE(tt.ID, sg.ARRIVAL_STATION_ID) ARRIVAL_DATE,
    tt.DEPARTURE_PLATFORM,
    tt.ARRIVAL_PLATFORM,
    rs.ROUTE_ID,
```

```
        rs.IS_TERMINAL,
        tt.TRAIN_ID,
        rs.SEQUENCE_NUMBER
        FROM ROUTES_2_SEGMENTS rs, SEGMENTS sg, TIMETABLE tt
            WHERE rs.SEGMENT_ID = sg.ID
                AND tt.ROUTE_ID = rs.ROUTE_ID
       ORDER BY rs.ROUTE_ID, rs.SEQUENCE_NUMBER, DEPARTURE_DATE ASC
/
```

4.3. Definizione Funzioni

In questa sezione vengono elencate le funzioni di supporto per la base dati.

4.3.1. Funzione CHECK_ROUTE_LINKING

Questa procedura verifica se i segmenti che compongono una tratta sono contigui, in altre parole verifica che tutti i segmenti ordinati siano effettivamente parte di una sequenza continua. Inoltre verifica la presenza di cicli all'interno della stessa tratta. In entrambi i casi imposta la tratta con ACTIVE = false (cioè non attiva).

```
CREATE OR REPLACE PROCEDURE CHECK_ROUTE_LINKING(routeId IN NUMBER)
IS
        -- variable declarations
        reachedStationId NUMBER;
        firstRow INTEGER;

        TYPE varray_number IS VARRAY(100) OF NUMBER (3);

        reachedStations varray_number := varray_number();
        counter integer := 0;

        ret NUMBER := 0;
BEGIN

        firstRow := 1;

        FOR rec IN (
        SELECT sg.* FROM ROUTES_2_SEGMENTS rs, SEGMENTS sg
        WHERE rs.SEGMENT_ID = sg.ID
                AND rs.ROUTE_ID = routeId
                ORDER BY rs.SEQUENCE_NUMBER ASC)
        LOOP
        counter := counter + 1;

        IF firstRow = 1 THEN
        reachedStations.EXTEND;
```

```
        reachedStations(counter) := rec.DEPARTURE_STATION_ID;
        counter := counter + 1;

        reachedStationId := rec.DEPARTURE_STATION_ID;

        END IF;

        FOR l_row IN 1 .. reachedStations.COUNT
        LOOP
                IF rec.ARRIVAL_STATION_ID = reachedStations(l_row) THEN
                        ret := 1;
                END IF;
        END LOOP;


        --If the link between the two segments isn't correct than the function returns 1
        IF reachedStationId <> rec.DEPARTURE_STATION_ID THEN
                ret := 1;
        END IF;

        EXIT WHEN ret = 1;

        reachedStationId := rec.ARRIVAL_STATION_ID;
        firstRow := 0;

        reachedStations.EXTEND;
        reachedStations(counter) := reachedstationid;


        END LOOP;

        --Set the active flag to be true
        UPDATE ROUTES
                SET ACTIVE = 1
                WHERE ID = routeId;

        --If the link between the two segments isn't correct than the route will be
                 marked an not active
        IF ret = 1 THEN
                UPDATE ROUTES
                        SET ACTIVE = 0 --false
                        WHERE ID = routeId;
        END IF;

        COMMIT;
END;
```

```
/
```

4.3.2. Funzione COMPUTE_DEPARTURE_DATE

Questa funzione calcola la data e l'ora di partenza di una specifica stazione (anche intermedia nella tratta). La data viene calcolata a partire dalla data di partenza sommando il tempo di percorrenza di ogni segmento fino a giungere a quella stazione.

```
CREATE OR REPLACE FUNCTION COMPUTE_DEPARTURE_DATE(timeTableId IN
NUMBER, departureStationId IN NUMBER)
RETURN TIMESTAMP
IS
   timetableRec TIMETABLE%ROWTYPE;
   trainSpeed NUMBER;

   tmpDate TIMESTAMP;
BEGIN
   --Get the timetable record
   SELECT * INTO timetableRec FROM TIMETABLE
      WHERE ID = timeTableId;



   --Get the nominal speed of the train for that timetable record
   SELECT t.NOMINAL_SPEED INTO trainSpeed FROM TRAINS t
      WHERE ID = timetableRec.TRAIN_ID;

   --Get all the segments of the route for that timetable record
   tmpDate := timetableRec.SCHEDULED_DATE;
   FOR segmentRec IN (
      SELECT sg.* FROM ROUTES_2_SEGMENTS rs, SEGMENTS sg
         WHERE rs.SEGMENT_ID = sg.ID
            AND rs.ROUTE_ID = timetableRec.ROUTE_ID
            AND rs.SEQUENCE_NUMBER < (SELECT rs2.SEQUENCE_NUMBER FROM
ROUTES_2_SEGMENTS rs2, SEGMENTS sg2
                                 WHERE rs2.ROUTE_ID = rs.ROUTE_ID AND
rs2.SEGMENT_ID = sg2.ID AND sg2.DEPARTURE_STATION_ID = departureStationId)
            ORDER BY rs.SEQUENCE_NUMBER ASC)
   LOOP
      --This formula adds the time needed for the that distance with that train speed
      tmpDate := tmpDate + (segmentRec.DISTANCE / trainSpeed)/24;

   END LOOP;

   RETURN tmpDate;
END;
```

4.3.3. Funzione COMPUTE_ARRIVAL_DATE

Questa funzione calcola la data e l'ora di arrivo ad una specifica stazione (anche intermedia nella tratta). La data viene calcolata a partire dalla data di partenza sommando il tempo di percorrenza di ogni segmento fino a giungere a quella stazione.

```
CREATE OR REPLACE FUNCTION COMPUTE_ARRIVAL_DATE(timeTableId IN
NUMBER, arrivalStationId IN NUMBER)
RETURN TIMESTAMP
IS
   timetableRec TIMETABLE%ROWTYPE;
   trainSpeed NUMBER;

   tmpDate TIMESTAMP;
BEGIN
   --Get the timetable record
   SELECT * INTO timetableRec FROM TIMETABLE
      WHERE ID = timeTableId;



   --Get the nominal speed of the train for that timetable record
   SELECT t.NOMINAL_SPEED INTO trainSpeed FROM TRAINS t
      WHERE ID = timetableRec.TRAIN_ID;

   --Get all the segments of the route for that timetable record
   tmpDate := timetableRec.SCHEDULED_DATE;
   FOR segmentRec IN (
      SELECT sg.* FROM ROUTES_2_SEGMENTS rs, SEGMENTS sg
         WHERE rs.SEGMENT_ID = sg.ID
            AND rs.ROUTE_ID = timetableRec.ROUTE_ID
            AND rs.SEQUENCE_NUMBER <= (SELECT rs2.SEQUENCE_NUMBER FROM
ROUTES_2_SEGMENTS rs2, SEGMENTS sg2
                                 WHERE rs2.ROUTE_ID = rs.ROUTE_ID AND
rs2.SEGMENT_ID = sg2.ID AND sg2.ARRIVAL_STATION_ID = arrivalStationId)
            ORDER BY rs.SEQUENCE_NUMBER ASC)
   LOOP
      --This formula adds the time needed for the that distance with that train speed
      tmpDate := tmpDate + (segmentRec.DISTANCE / trainSpeed)/24;

   END LOOP;

   RETURN tmpDate;
END;
/
```

4.3.4. Funzione COMPUTE_DISTANCE

Questa funzione calcola la distanza tra 2 stazioni nel contesto di una data tratta di percorrenza.

```
CREATE OR REPLACE FUNCTION COMPUTE_DISTANCE(routeId IN NUMBER,
departureStationId IN NUMBER, arrivalStationId IN NUMBER)
RETURN NUMBER
IS
   totalDistance NUMBER;
BEGIN

   SELECT SUM(sg.DISTANCE) INTO totalDistance
      FROM ROUTES_2_SEGMENTS rs, SEGMENTS sg
         WHERE rs.SEGMENT_ID = sg.ID
            AND rs.ROUTE_ID = routeId
            AND rs.SEQUENCE_NUMBER >= (SELECT rs2.SEQUENCE_NUMBER FROM
ROUTES_2_SEGMENTS rs2, SEGMENTS sg2
                                 WHERE rs2.ROUTE_ID = rs.ROUTE_ID AND
rs2.SEGMENT_ID = sg2.ID AND sg2.DEPARTURE_STATION_ID = departureStationId)
            AND rs.SEQUENCE_NUMBER <= (SELECT rs2.SEQUENCE_NUMBER FROM
ROUTES_2_SEGMENTS rs2, SEGMENTS sg2
                                 WHERE rs2.ROUTE_ID = rs.ROUTE_ID AND
rs2.SEGMENT_ID = sg2.ID AND sg2.ARRIVAL_STATION_ID = arrivalStationId)
            ORDER BY rs.SEQUENCE_NUMBER ASC;

   RETURN totalDistance;
END;
```