# How to create your own TS2 simulations

Nicolas Piganeau

December 11, 2008

## Contents

## 1 Introduction

This document explains how to create your own TS2 simulations from scratch. This document is up to date for version 0.2 (beta 1) of TS2

TS2 simulations (.ts2 files) are in fact sqlite3 databases. One file completely defines the simulation, including scenery layout, train information, schedules, etc.

## 2 Support

For support on this document or on how to create your own TS2 simulations, post on our mailing list ts2-devel@lists.sourceforge.net or on the project forum at https://sourceforge.net/forum/forum.php?forum_id=861092

## 3 TS2 simulation database structure

This section describes the structure of the database of a TS2 simulation. The details of each table is given in the following paragraphs

A TS2 simulation is a database with the following tables:

- **options:** Simulation wide options
- **trackitems:** Scenery layout, tracks, signals, points, etc.
- **places:** "Places" where trains will have a schedule (mainly station, but can also be a main junction for example)
- **routes:** List of allowed routes between signals
- **directions:** For each route whether the points should be in normal or reverse position
- **traintypes:** Information about the stock used in the simulation
- **trains:** List of trains
- **services:** List of schedules
- **servicelines:** Description of each stop and time for schedules

## 4 The "options" table

### 4.1 Table description

The options table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| optionkey | VARCHAR(30) | The key of the option. See below for different option keys. |
| optionvalue | VARCHAR(50) | The value of the option for the given key. |

## 4.2 Option keys

The following options exist in TS2. All options should be set in a new simulation. Option keys and values are case sensitive.

- **currentTime:** Current time of the simulation. For a new simulation, it is the time at which the simulation starts. Time should be given as a string in the following format “hh:mm:ss”.
- **timeFactor:** Simulation speed. (e.g. for 5x write “5”)
- **defaultMaxSpeed:** The default maximum speed for trains in the simulation, in metres per second. (e.g. “18.5”)
- **warningSpeed:** The maximum speed authorized for trains with a red signal ahead. (e.g. “8.3”)
- **defaultMinimumStopTime:** The minimum time in seconds that a train stops at a station by default. This is particularly relevant for trains that are late and arrive at a station after the scheduled departure time. (e.g. “90”)

# 5 The “trackitems” table

## 5.1 Table description

The “trackitems” table describes all the track “items”. An item is a piece of scenery. Each item has defined coordinates in the scenery layout and is connected to other items so that the trains can travel from one to another. The coordinates are expressed in pixels. The origin is the top left most corner of the scene. The X-axis is from left to right and the Y-axis is from top to bottom.

In the current version of TS2, the following item types exist (see corresponding paragraphs for more details):

- **Line:** A line is a simple track used to connect other items together. The important parameter of a line is its real length, i.e. the length it would have in real life, since this will determine the time the train takes to travel on it.
- **Signal:** A signal is a basic red - yellow - green signal. The color of the signal is determined by the presence of a train just after this signal or by the color of the next signal on the line.
- **Points:** Points allow trains to change direction.
- **Platform:** A platform in a station. Internally it is a particular type of line, including extra graphics.
- **Bumper:** A bumper terminates a line. Internally it is a “always red” signal.
- **Timer Signal:** This is a special type of signal used at the edge of a scenery, before trains leave the area. This type of signal goes red when a train passes it, and then goes yellow after a first timeout and green after a second timeout.
- **End item:** This is a special item type to which the free edge of tracks going out of the area must be connected.

| **Field Name** | **Type** | **Description** |
|---|---|---|
| tiid | INTEGER | Unique ID. It is also the primary key of the table. Note that sqlite auto-increment the primary key by default. |
| name | VARCHAR(25) | Intelligible name of the item. Must be unique or null. |
| titype | VARCHAR(5) | The type of the item. See below for available types. |
| ptiid | INTEGER | Previous item ID. Most of the time TS2 automatically computes the previous item ID. See below. |
| ntiid | INTEGER | Next item ID. Most of the time TS2 automatically computes the next item ID. See below. |
| rtiid | INTEGER | Next item ID in the reverse direction. Only applicable to points items. |
| conflicttiid | INTEGER | ID of an item that is conflicting with the current item. |
| x<br>y<br>xf<br>yf<br>xr<br>yr<br>xn<br>yn | DOUBLE<br>DOUBLE<br>DOUBLE<br>DOUBLE<br>DOUBLE<br>DOUBLE<br>DOUBLE<br>DOUBLE | Coordinates of the item. See specific items description for more details. |
| reverse | BOOLEAN | Indicates whether the item is in the normal direction (i.e. from left to right) or in the reverse direction (i.e. from right to left). Currently only applicable for signals. |
| reallength | DOUBLE | Real length in metres of the item. |
| maxspeed | DOUBLE | Maximum speed for trains on this item in metres per second. If not set, it defaults to the option "defaultMaxSpeed". Has no effect in the current version of TS2. |
| placecode | VARCHAR(10) | The place code for this item, if it belongs to a place. Mainly applicable for platforms that belong to a station. |
| trackcode | VARCHAR(10) | The track number or track code, if the item belongs to a place. Usually the platform number for platforms in stations. |
| timersw | DOUBLE | Only applicable for timer signal items. Time in minutes for the signal to pass from red to yellow. |
| timerwc | DOUBLE | Only applicable for timer signal items. Time in minutes for the signal to pass from yellow to green. |

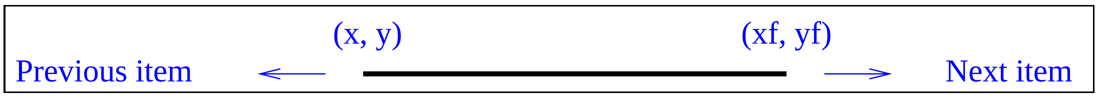
Table 1: "trackitems" table fields

The list of the fields of the "trackitems" table is given in table 1.

Each item has a unique ID : **tiid**. This ID is used for connecting items together. If you do not wish to define an ID for each item, you can use the autoincrement feature of sqlite. The main issue is that you will probably need to set explicitly the ID for some items that need to be connected manually, and these might interfere with the autoincremented IDs. The solution to this issue is to set the ID of the first item you define with a very high value (1000000 for example). Then the autoincrement feature will count from this value and you can define your own IDs below this value.

## 5.2 The "Line" item type

Graphical representation of a line item is fairly simple:



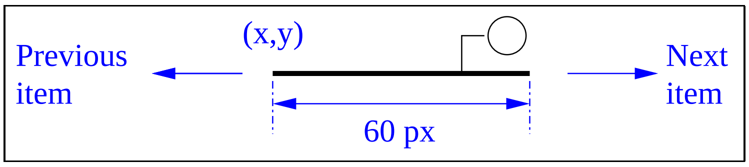
It consists of a line defined by the following:

- Its **titype** which is the string "L".
- Its first end which has the coordinates (**x**, **y**).
- Its second end which has the coordinates (**xf**, **yf**).
- Its real length defined in metres given by **reallength**.
- The maximum speed in metres per second allowed on this line: **maxspeed**. *Not used in the current TS2 version.*
- Its **placecode** and **trackcode** if this line belongs to a place.
- Its "previous item" with ID **ptiid** connected to its first end.
- Its "next item" with ID **ntiid** connected to its second end.
- The ID of an item which conflicts with it: **conflicttiid**. I.e. if a route is set on this item, no route must be set on the conflict item and vice-versa. This is mainly used when two lines crossover, they are declared conflicting one with the other.

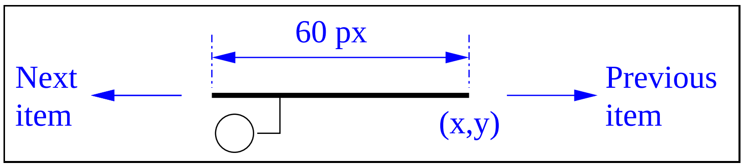
Other fields in the trackitems table are discarded when **titype** is set to "L".

## 5.3 The "signal" item type

The signal item exists in two versions. The normal version, from left to right:



and the reverse version, from right to left:



Signals can only be horizontal. They have a length dedicated to writing the service code of the nearest train just before the representation of the signal itself.

The following fields are relevant for signal items and define them:

- Its **titype** which is “S”.
- Its origin with coordinates (**x**, **y**).
- Its **name**. This is usually a signal number. If it is not set, it defaults to **tiid**. The name is displayed as a tooltip when hovering over the signal.
- Its direction defined by the **reverse** field. If true, the signal is reversed, otherwise it is normal.
- Its “previous item” with ID **ptiid** connected to its origin.
- Its “next item” with ID **ntiid** connected to its other end.

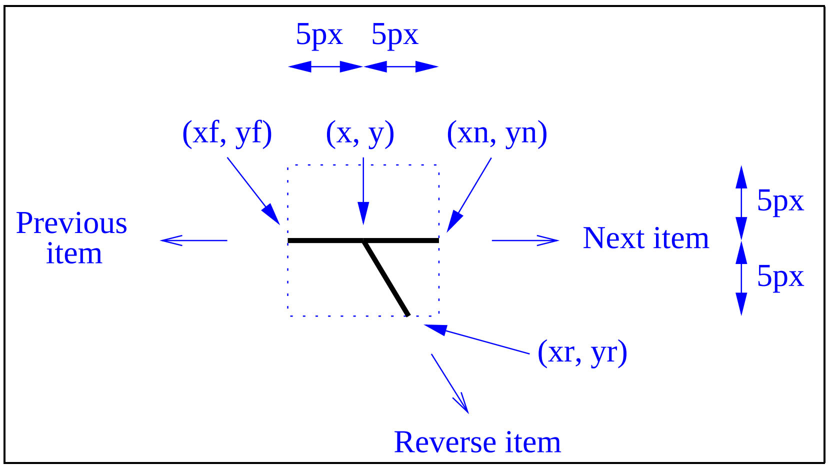
Other fields in the trackitems table are discarded when **titype** is set to “S”.

The signal item is a simple green-yellow-red signal. The color of the signal is determined the following way:

- If there is no route set from this signal, the signal is red.
- If there is a route set from this signal, and there is a train ahead before the next signal on the line, then the signal is red.
- If there is a route set from this signal and no trains ahead before the following signal, the color of the signal is determined after the color of the next signal. If the next signal is red, then this signal will be yellow. Otherwise, this signal will be green.

## 5.4 The “points” item type

A points item is a three-way junction. We call the three ends: common end, normal end and reverse end. Trains can go from common end to normal or reverse ends depending on the state of the points. They cannot go from normal end to reverse end. Usually, the normal end is aligned with the common end and the reverse end is sideways, like on the diagram below, but this is not mandatory in TS2.



Points items are defined by the following fields:

- Its **titype** which is "P".
- Its origin with coordinates (**x**,**y**). Note that the origin is the center of the item, unlike other items.
- Its **name**. This is usually a points number. If it is not set, it defaults to **tiid**. The name is displayed as a tooltip when hovering over the points.
- Its common end defined by relative coordinates (**xf**, **yf**) to the origin. These coordinates must be on a square 10x10 around the origin (see diagram). This means that at least one of **xf** or **yf** is equal to +5 or -5.
- Its normal end defined by relative coordinates (**xn**, **yn**) to the origin. These coordinates must be on a square 10x10 around the origin (see diagram). This means that at least one of **xn** or **yn** is equal to +5 or -5.
- Its reverse end defined by relative coordinates (**xr**, **yr**) to the origin. These coordinates must be on a square 10x10 around the origin (see diagram). This means that at least one of **xr** or **yr** is equal to +5 or -5.
- Its "previous item" with ID **ptiid** connected to its common end.
- Its "next item" with ID **ntiid** connected to its normal end.
- Its "reverse item" with ID **rtiid** connected to its reverse end. **rtiid** must always be specified and is never computed automatically.

Other fields in the trackitems table are discarded when **titype** is set to "P".

Note that items connected to the points items must be connected to the common, normal or reverse ends and not to the origin. For example, if the origin has the coordinates (100, 50), the common end (-5,0), the normal end (5, 0) and the reverse end (5, 5), the previous item must end at (95, 50), the next item at (105, 50) and the reverse item at (105, 55).

## 5.5 The "platform" item type

Platform items are a special type of line items but including the drawing of a colored rectangle to symbolise the platform. This colored rectangle also permits user interaction. The colored rectangle is defined by the coordinates (**xn**, **yn**) of its top left corner and the coordinates (**xr**, **yr**) of its bottom right corner. Thus it can be drawn anywhere on the screen, and not necessarily next to the track of the platform item.

Their **titype** is "LP". See line item type for all other parameters.

## 5.6 The "bumper" item type

Bumpers are "always red" signals. Their representations show a triangle instead of the signal.

Their **titype** is "SB". Parameters for bumper items are the same as signals.

Logically the next item of a bumper item should be an "end" item.

## 5.7 The “timer signal” item type

The timer signal type is a special type of signals that exist only for the purpose of the simulation. They are to be used for the last signal before the edge of the scene where trains leave the area.

Their graphical representation is identical to that of signals. The following parameters differ from signal items:

- Their **titype** is “ST”.
- The time taken for the signal to go from red to yellow is defined in minutes by **timerSW**.
- The time taken for the signal to go from yellow to green is defined in minutes by **timerWC**.

Other parameters are the same as signals.

Unlike normal signals, the color of the signal is determined the following way:

- If a train passed less than **timerSW** minutes ago, the signal is red.
- If a train passed less more than **timerSW** minutes ago but less that **timerSW** + **timerWC**, the signal is yellow.
- Else the signal is green.

## 5.8 The “end” item type

End items are invisible items to which the free ends of other trackitems must be connected to prevent the simulation from crashing TS2.

End items are defined by their **titype** which is “E” and their position (**x**, **y**). They are single point items.

## 5.9 Automatic connection of items

If the previous item ID (ptiid) or the next item ID (ntiid) is not defined, TS2 will try to automatically compute them based on the coordinates of their ends (x, y) and (xf, yf). However, they must be declared in the following cases:

- Items connected to the reverse end of a points must explicitly declare their ptiid or ntiid as being the mentioned points.
- Reciprocally, points must explicitly declare the item connected to their reverse end. Items connected to normal and common ends are computed automatically.
- In case the next item end is not physically on the same coordinates as this items end (typically tracks passing under a bridge), the ntiid and/or ptiid must be declared explicitly for both items.

# 6 The “places” table

A place represents a named position on the scene gathering several tracks (line items). Most of the time this will be a station but can also be a major junction for example. Schedules are made from place to place. The place names are displayed on the scenery layout.

## 6.1 Table description

The places table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| placecode | VARCHAR(10) | A unique code for the place. This is used for reference to this place in line items or services for example. |
| placename | VARCHAR(50) | The complete name of the place. |
| x | DOUBLE | The x coordinate of the label of the place on the scene. |
| y | DOUBLE | The y coordinate of the label of the place on the scene. |

# 7 Routes

Routes are pathes between two signals. Routes are selected in the simulation by clicking on the signal at the beginning of the route and then the signal at the end of the route. If the route is valid, the points are turned in the correct direction to go from the first signal to the second signal, and the state of the first signal is updated.

Routes are defined by two tables: *routes* and *directions*.

## 7.1 The "routes" table

The routes table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| routenum | INTEGER | Unique ID of the route. Table primary key. |
| beginsignal | INTEGER | ID of the signal at the beginning of the route. The signal must exist in the trackitems table (tiid). |
| endsignal | INTEGER | ID of the signal at the end of the route. The signal must exist in the trackitems table (tiid). |

## 7.2 The "directions" table

On a route between two signals, there can be several points. By default, TS2 will consider that all the points are in normal direction to compute the route. The table directions is used to tell TS2 which points are in reverse direction for each route. Note that it is only necessary to declare the points in reverse direction if the route arrives at the common end and must continue through the reverse end of the points. If the route arrives at the reverse end, TS2 automatically continues through the common end.

The directions table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| routenum | INTEGER | Route ID for which we define the direction. |
| tiid | INTEGER | ID of the points to declare in reverse position. The points must exist in the trackitems table (tiid). |
| direction | INTEGER | Set to "1" to declare reverse direction. |

# 8 The "traintypes" table

The traintypes table is used to declare available stock in the simulation.

### 8.1 Table description

The traintypes table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| code | VARCHAR(10) | A unique code used to reference this traintype. |
| description | VARCHAR(200) | Description string of the traintype. |
| maxspeed | DOUBLE | Maximum speed in m/s. |
| stdaccel | DOUBLE | Acceleration in m/s2. |
| stdbraking | DOUBLE | Normal braking speed in m/s2. |
| emergbraking | DOUBLE | Maximum possible braking speed, in case of emergency, in m/s2. |
| tlength | DOUBLE | Stock length in metres. |

## 9 Services

Services are mainly predefined schedules that trains are supposed to follow. They are defined by two tables : *services* and *servicelines*. The services table declares the service and includes one serviceline per place to call at. Service lines are automatically sorted by scheduled arrival times.

### 9.1 The "services" table

The services table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| servicecode | VARCHAR(10) | A unique code used to reference this service. |
| description | VARCHAR(200) | Description string of the service. |
| nextservice | VARCHAR(10) | Reference to the servicecode of the next service to assign automatically. |

### 9.2 The "servicelines" table

The servicelines table has the following fields:

| Field Name | Type | Description |
|---|---|---|
| servicecode | VARCHAR(10) | Reference to the service this line belongs to. |
| placecode | VARCHAR(10) | Reference to a place of the places table. |
| scheduledarrivaltime | TIME | Scheduled arrival time at the place defined above. |
| scheduleddeparturetime | TIME | Scheduled departure time at the place defined above. |
| trackcode | VARCHAR(10) | Track reference where to stop at the defined place. |
| stop | BOOLEAN | Set to true if the train to which this service is assigned must stop at the defined place. |

## 10 The "trains" table

A train is a stock running on a track at a certain speed and to which is assigned a service.

### 10.1 Table description

The servicelines table has the following fields:

| **Field Name** | **Type** | **Description** |
|---|---|---|
| servicecode | VARCHAR(10) | The service code that is assigned to this train. |
| traintype | VARCHAR(10) | The stock of this train. |
| speed | DOUBLE | Current speed of the train, or speed that it will have when it appears. |
| accel | DOUBLE | Current acceleration of the train. |
| tiid | INTEGER | ID of the track item the train is currently on. |
| previoustiid | INTEGER | ID of the previous track item the train was. Used to know the direction of the train. |
| posonti | DOUBLE | Position of the train head on the track item in metres. |
| appeartime | TIME | Time at which the train will appear on the scene. |

## 11 TS2 crashes when loading simulation

There is hardly any checks that the simulation loaded in TS2 is valid. If the simulation is invalid, TS2 will, most of the time, crash with a "Segmentation fault (SIGSEV)".

To avoid this, check particularly:

- References to other tables should exist in the target table. For instance, the values of the "traintype" field of the "trains" table should refer to existing values in the "code" field of the "traintypes" table.
- All ends of all items should be connected to another item. At the edge of the scenery, they should be connected to "end" items. This is also true for bumpers items. Check stderr (Console output) to know which items are not connected.
- Routes defined between two signals with the directions defined must exists. Check stderr to know which routes could not be computed.

## 12 Example

This is the SQL code to fill in the database of the "drain" demo that is shipped with TS2. Signals and points numbering is not the real numbering.

### 12.1 Database structure

Here is the database structure declaration. You can copy/paste it in your own simulation.

```
DROP TABLE IF EXISTS trackitems;
CREATE TABLE trackitems (
        tiid INTEGER PRIMARY KEY,
        name VARCHAR(25) UNIQUE,
        titype VARCHAR(5),
        ptiid INTEGER,
        ntiid INTEGER,
        rtiid INTEGER,
        conflicttiid INTEGER,
        x DOUBLE,
        y DOUBLE,
        xf DOUBLE,
        yf DOUBLE,
        xr DOUBLE,
        yr DOUBLE,
        xn DOUBLE,
        yn DOUBLE,
        reverse BOOLEAN,
        reallength DOUBLE,
        maxspeed DOUBLE,
        placecode VARCHAR(10),
        trackcode VARCHAR(10),
        timersw DOUBLE,
        timerwc DOUBLE);

DROP TABLE IF EXISTS places;
CREATE TABLE places (
        placecode VARCHAR(10),
        placename VARCHAR(50),
        x DOUBLE,
        y DOUBLE);

DROP TABLE IF EXISTS routes;
CREATE TABLE routes (
        routenum INTEGER PRIMARY KEY,
        beginsignal INTEGER,
        endsignal INTEGER);

DROP TABLE IF EXISTS directions;
CREATE TABLE directions (
        routenum INTEGER,
        tiid INTEGER,
        direction INTEGER);

DROP TABLE IF EXISTS traintypes;
CREATE TABLE traintypes (
        code VARCHAR(10),
        description VARCHAR(200),
        maxspeed DOUBLE,
        stdaccel DOUBLE,
        stdbraking DOUBLE,
        emergbraking DOUBLE,
        tlength DOUBLE);

DROP TABLE IF EXISTS trains;
CREATE TABLE trains (
        servicecode VARCHAR(10),
        traintype VARCHAR(10),
        speed DOUBLE,
        accel DOUBLE,
        tiid INTEGER,
        previoustiid INTEGER,
        posonti DOUBLE,
        appeartime TIME);

DROP TABLE IF EXISTS services;
CREATE TABLE services (
        servicecode VARCHAR(10),
        description VARCHAR(200),
        nextservice VARCHAR(10));

DROP TABLE IF EXISTS servicelines;
CREATE TABLE servicelines (
        servicecode VARCHAR(10),
        placecode VARCHAR(10),
        scheduledarrivaltime TIME,
        scheduleddeparturetime TIME,
        trackcode VARCHAR(10),
        stop BOOLEAN);

DROP TABLE IF EXISTS options;
CREATE TABLE options (
        optionKey VARCHAR(30),
        optionValue VARCHAR(50));
```

## 12.2 Options

All the options below have to be defined. You can adjust the value to your need.

```
INSERT INTO options VALUES ("timeFactor", "5");
INSERT INTO options VALUES ("currentTime","06:00:00");
INSERT INTO options VALUES ("warningSpeed", "8.3");
INSERT INTO options VALUES ("defaultMaxSpeed", "18");
INSERT INTO options VALUES ("defaultMinimumStopTime", "30");
```

## 12.3 Track items

### 12.3.1 Bank platform 7

This platform consists in an end item, a bumper item in reverse, a platform of 80 metres and a signal.

```
INSERT INTO trackitems (tiid, tiType, x, y)
        VALUES (1000000, "E", 0, 100);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (71, "71", "SB", 60, 100, 1);
INSERT INTO trackitems (tiid, tiType, x, y, xf, yf, realLength, xn, yn, xr, yr,
                                                        placecode, trackcode)
        VALUES (7, "LP", 60, 100, 130, 100, 80, 10, 105, 129, 125, "BNK", "7");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (72, "72", "S", 130, 100, 0);
```

### 12.3.2 Bank platform 8

Similar to platform 7.

```
INSERT INTO trackitems (tiid, tiType, x, y)
        VALUES (601, "E", 0, 150);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (81, "81", "SB", 60, 150, 1);
INSERT INTO trackitems (tiid, tiType, x, y, xf, yf, realLength, xn, yn, xr, yr,
                                                                placecode, trackcode)
        VALUES (8, "LP", 60, 150, 130, 150, 80, 10, 125, 129, 145, "BNK", "8");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (82, "82", "S", 130, 150, 0);
```

### 12.3.3 Bank crossover

4 points items at each corner and connected together by line items. Take particular note to:

- The explicit declaration of rtiid of points items.
- The explicit declaration of pttid and ntiid of items 201 and 202 which are connected to points reverse ends on both sides.
- The use of conflicttiid on item 201 to prevent the validation of routes on item 202 at the same time.

```
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (511, "P", 201, 195, 100, -5, 0, 5, 0, 5, 5);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (521, "P", 202, 195, 150, -5, 0, 5, 0, 5, -5);
INSERT INTO trackitems (tiid, tiType, ptiid, ntiid, x, y, xf, yf, conflicttiid)
        VALUES (201, "L", 511, 522, 200, 105, 240, 145, 202);
INSERT INTO trackitems (tiid, tiType, ptiid, ntiid, x, y, xf, yf)
        VALUES (202, "L", 521, 512, 200, 145, 240, 105);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 200, 100, 240, 100);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 200, 150, 240, 150);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (512, "P", 202, 245, 100, 5, 0, -5, 0, -5, 5);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (522, "P", 201, 245, 150, 5, 0, -5, 0, -5, -5);
```

### 12.3.4 Bank ->Waterloo line

Two signals connected by line items.

```
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 250, 100, 450, 100, 700);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (73, "73", "S", 450, 100, 0);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 510, 100, 700, 100, 700);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (74, "74", "S", 700, 100, 0);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 760, 100, 950, 100, 700);
```

### 12.3.5 Waterloo ->Bank line

Three signals connected by line items.

```
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (83, "83", "S", 310, 150, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 310, 150, 450, 150, 700);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (84, "84", "S", 510, 150, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 510, 150, 700, 150, 700);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (85, "85", "S", 760, 150, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 760, 150, 890, 150, 700);
```

### 12.3.6 Waterloo platform 26

Platform + signal.

```
INSERT INTO trackitems (tiType, x, y, xf, yf, realLength, xn, yn, xr, yr,
                                                        placecode, trackcode)
        VALUES ("LP", 950, 100, 1020, 100, 80, 950, 80, 1019, 95, "WTL", "26");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (75, "75", "S", 1020, 100, 0);
```

### 12.3.7 Waterloo platform 25

Platform and two signals one on each side.

```
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (86, "86", "S", 950, 150, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, realLength, xn, yn, xr, yr,
                                                          placecode, trackcode)
        VALUES ("LP", 950, 150, 1020, 150, 80, 950, 155, 1019, 170, "WTL", "25");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (87, "87", "S", 1020, 150, 0);
```

### 12.3.8 Depot track 26

From waterloo platform 26 to the next points (513).

```
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1080, 100, 1100, 100);
INSERT INTO trackitems (tiid, tiType, ntiid, x, y, xf, yf)
        VALUES (203, "L", 513, 1100, 100, 1120, 120);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (513, "P", 203, 1125, 125, 5, 0, -5, 5, -5, -5);
```

### 12.3.9 Depot track 25

From waterloo platform 25 to points 513 (track 26), including points 523 to depot track 3 & 4.

```
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1080, 150, 1090, 150);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (523, "P", 204, 1095, 150, -5, 0, 5, 0, 5, 5);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1100, 150, 1120, 130);
```

### 12.3.10 Depot track 3

From points 523 to track 3 bumper and end item, including points 531 to depot track 4.

```
INSERT INTO trackitems (tiid, tiType, ptiid, x, y, xf, yf)
        VALUES (204, "L", 523, 1100, 155, 1120, 175);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1120, 175, 1125, 175);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (531, "P", 207, 1130, 175, -5, 0, 5, 0, 5, 5);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
        VALUES ("L", 1135, 175, 1155, 175, 70);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (31, "31", "S", 1215, 175, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1215, 175, 1260, 175);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (32, "32", "SB", 1260, 175, 0);
INSERT INTO trackitems (tiType, x, y)
        VALUES ("E", 1320, 175);
```

### 12.3.11 Depot track 4

From points 531 to track 4 bumper and end item.

```
INSERT INTO trackitems (tiid, tiType, ptiid, x, y, xf, yf)
        VALUES (207, "L", 531, 1135, 180, 1155, 200);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (41, "41", "S", 1215, 200, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength)
```

```
        VALUES ("L", 1215, 200, 1260, 200, 70);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (42, "42", "SB", 1260, 200, 0);
INSERT INTO trackitems (tiType, x, y)
        VALUES ("E", 1320, 200);
```

### 12.3.12 Depot track 5

From points 513 to track 5 bumper and end item, including points 551 to depot track 6.

```
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1130, 125, 1145, 125);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (551, "P", 205, 1150, 125, -5, 0, 5, 0, 5, -5);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (51, "51", "S", 1215, 125, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, placecode, trackcode)
        VALUES ("L", 1215, 125, 1300, 125, "DPT", "5");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (52, "52", "SB", 1300, 125, 0);
INSERT INTO trackitems (tiType, x, y)
        VALUES ("E", 1360, 125);
```

### 12.3.13 Depot track 6

From points 551 to track 6 bumper and end item, including points 561 to depot track 7.

```
INSERT INTO trackitems (tiid, tiType, ptiid, x, y, xf, yf)
        VALUES (205, "L", 551, 1155, 120, 1175, 100);
INSERT INTO trackitems (tiType, x, y, xf, yf)
        VALUES ("L", 1175, 100, 1180, 100);
INSERT INTO trackitems (tiid, tiType, rtiid, x, y, xf, yf, xn, yn, xr, yr)
        VALUES (561, "P", 206, 1185, 100, -5, 0, 5, 0, 5, -5);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (61, "61", "S", 1250, 100, 1);
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength, placecode, trackcode)
        VALUES ("L", 1250, 100, 1300, 100, 80, "DPT", "6");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (62, "62", "SB", 1300, 100, 0);
INSERT INTO trackitems (tiType, x, y)
        VALUES ("E", 1360, 100);
```

### 12.3.14 Depot track 7

From points 561 to track 7 end item. Unlike the real depot track 7 which ends with a bumper, we have put here a timer signal and exit of the area as an example.

```
INSERT INTO trackitems (tiid, tiType, ptiid, x, y, xf, yf)
        VALUES (206, "L", 561, 1190, 95, 1210, 75);
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse)
        VALUES (76, "76", "S", 1270, 75, 1);
```

```
INSERT INTO trackitems (tiType, x, y, xf, yf, reallength, placecode, trackcode)
        VALUES ("L", 1270, 75, 1300, 75, 80, "DPT", "7");
INSERT INTO trackitems (tiid, name, tiType, x, y, reverse, timersw, timerwc)
        VALUES (77, "77", "ST", 1300, 75, 0, 2.0, 1.0);
INSERT INTO trackitems (tiType, x, y)
        VALUES ("E", 1360, 75);
```

## 12.4 Places

Three places are defined in this simulation, Bank, Waterloo and Depot.

```
INSERT INTO places VALUES ("BNK", "BANK", 60, 50);
INSERT INTO places VALUES ("WTL", "WATERLOO", 950, 50);
INSERT INTO places VALUES ("DPT", "DEPOT", 1250, 50);
```

## 12.5 Routes

### 12.5.1 Bank ->Waterloo routes

```
INSERT INTO routes VALUES (1, 72, 73);
INSERT INTO routes VALUES (101, 82, 73);
    INSERT INTO directions VALUES (101, 521, 1);
INSERT INTO routes VALUES (2, 73, 74);
INSERT INTO routes VALUES (3, 74, 75);
```

### 12.5.2 Waterloo ->Bank routes

```
INSERT INTO routes VALUES (51, 86, 85);
INSERT INTO routes VALUES (52, 85, 84);
INSERT INTO routes VALUES (53, 84, 83);
INSERT INTO routes VALUES (54, 83, 81);
INSERT INTO routes VALUES (102, 83, 71);
    INSERT INTO directions VALUES (102, 522, 1);
```

### 12.5.3 Depot routes

```
INSERT INTO routes VALUES (201, 75, 77);
    INSERT INTO directions VALUES (201, 551, 1);
    INSERT INTO directions VALUES (201, 561, 1);
INSERT INTO routes VALUES (202, 75, 62);
    INSERT INTO directions VALUES (202, 551, 1);
INSERT INTO routes VALUES (203, 75, 52);
INSERT INTO routes VALUES (204, 87, 77);
    INSERT INTO directions VALUES (204, 551, 1);
    INSERT INTO directions VALUES (204, 561, 1);
INSERT INTO routes VALUES (205, 87, 62);
    INSERT INTO directions VALUES (205, 551, 1);
INSERT INTO routes VALUES (206, 87, 52);
INSERT INTO routes VALUES (207, 76, 86);
INSERT INTO routes VALUES (208, 61, 86);
INSERT INTO routes VALUES (209, 51, 86);
INSERT INTO routes VALUES (210, 87, 32);
    INSERT INTO directions VALUES (210, 523, 1);
INSERT INTO routes VALUES (211, 87, 42);
    INSERT INTO directions VALUES (211, 523, 1);
    INSERT INTO directions VALUES (211, 531, 1);
INSERT INTO routes VALUES (212, 31, 86);
INSERT INTO routes VALUES (213, 41, 86);
```

## 12.6 Train Types

```
INSERT INTO traintypes VALUES ("UT", "Underground train", 25, 1.0, 1.0, 2.5, 70);
```

## 12.7 Services

```
INSERT INTO services VALUES ("BW01", "First BANK->WATERLOO service", "WB01");
    INSERT INTO servicelines VALUES ("BW01", "BNK", "06:00:00", "06:00:30", "7", 1);
    INSERT INTO servicelines VALUES ("BW01", "WTL", "06:03:00", "06:04:30", "25", 1);
    INSERT INTO servicelines VALUES ("BW01", "DPT", "06:05:30", "06:06:00", "5", 1);
INSERT INTO services VALUES ("BW02", "Second BANK->WATERLOO service", "");
    INSERT INTO servicelines VALUES ("BW02", "BNK", "06:00:00", "06:05:00", "8", 1);
    INSERT INTO servicelines VALUES ("BW02", "WTL", "06:08:30", "06:09:00", "26", 1);
INSERT INTO services VALUES ("BW03", "Third BANK->WATERLOO service", "");
    INSERT INTO servicelines VALUES ("BW03", "BNK", "06:00:00", "06:10:30", "7", 1);
    INSERT INTO servicelines VALUES ("BW03", "WTL", "06:02:30", "06:13:30", "25", 1);
    INSERT INTO servicelines VALUES ("BW03", "DPT", "06:04:30", "06:15:00", "5", 1);
INSERT INTO services VALUES ("WB01", "First WATERLOO->BANK service", "");
    INSERT INTO servicelines VALUES ("WB01", "WTL", "06:06:30", "06:08:00", "26", 1);
    INSERT INTO servicelines VALUES ("WB01", "BNK", "06:10:30", "06:11:00", "7", 1);
```

## 12.8 Trains

```
INSERT INTO trains VALUES ("BW01", "UT", 0.0, 0.0, 7, 71, 79, "06:00:00");
INSERT INTO trains VALUES ("BW02", "UT", 0.0, 0.0, 8, 81, 79, "06:00:00");
```