Existing Transit DB Tables

Database is MySQL. Database name is ‘TrafficTransit’. Hosting server is path2go (128.32.235.223), located in Campus data center.

1. Tables to store static transit data (obtained from GTFS file)

**Table Agency**

Table schema:

CREATE TABLE `Agency` (

`agency\_id` int(10) unsigned NOT NULL,

`agency\_name` varchar(255) NOT NULL,

`agency\_url` varchar(255) NOT NULL,

`agency\_timezone` smallint(6) NOT NULL,

`agency\_lang` varchar(255) NOT NULL,

`agency\_phone` varchar(255) NOT NULL,

`timezone\_name` varchar(45) NOT NULL,

PRIMARY KEY (`agency\_id`),

KEY `agency\_timezone` (`agency\_timezone`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS agency.txt

Note: ‘agency\_id’ is an optional field (string) in GTFS agency.txt, and is the primary key (integer) of the Agency table. The value needs to be uniquely assigned for each transit agency. We can use agency\_id = 10 for VTA, and agency\_id = 11 for Tri Delta Transit.

**Table Routes**

Table schema:

CREATE TABLE `Routes` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`route\_type` int(10) unsigned NOT NULL,

`route\_long\_name` varchar(255) NOT NULL default 'N/A',

`route\_desc` varchar(255) NOT NULL default 'N/A',

`route\_url` varchar(255) NOT NULL default 'N/A',

`route\_color` varchar(255) NOT NULL default 'FFFFFF',

`route\_text\_color` varchar(255) NOT NULL default '000000',

`route\_id` varchar(255) NOT NULL default '000000',

`version` varchar(255) NOT NULL,

PRIMARY KEY USING BTREE (`agency\_id`,`route\_short\_name`,`route\_dir`,`version`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS routes.txt, trips.txt

Note:

* ‘agency\_id’ is the same field as the ‘Agency’ table.
* ‘route\_short\_name’, ‘route\_type’, ‘route\_long\_name’, ‘route\_desc’, ‘route\_url’, ‘route\_color’, ‘route\_text\_color’ and ‘route\_id’ are from GTFS routes.txt.
* ‘route\_dir’ is not included in GTFS routes.txt and is generated from trips.txt. For each ‘route\_id’ in routes.txt finds the distinct ‘direction\_id’ in trips.txt. ‘direction\_id’ is an optional field in trips.txt, please notify us if ‘direction\_id’ is not included (or blank) in trips.txt, we need to add ‘route\_dir’ manually.
* ‘version’ is the md5 string of the GTFS zip file.

**Table Stops**

Table schema:

CREATE TABLE `Stops` (

`agency\_id` int(10) unsigned NOT NULL,

`stop\_id` bigint(20) unsigned NOT NULL,

`stop\_code` varchar(255) NOT NULL default 'N/A',

`stop\_name` varchar(255) NOT NULL,

`stop\_desc` varchar(255) NOT NULL default 'N/A',

`stop\_lat` double NOT NULL,

`stop\_lon` double NOT NULL,

`lat\_lon` point NOT NULL,

`stop\_url` varchar(255) NOT NULL default 'N/A',

`location\_type` int(10) unsigned NOT NULL default '0',

`parent\_station` bigint(20) unsigned NOT NULL default '0',

`wheelchair\_boarding` int(10) unsigned NOT NULL default '0',

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`stop\_id`,`version`),

SPATIAL KEY `lat\_lon` (`lat\_lon`)

) ENGINE=MyISAM DEFAULT CHARSET=latin1

Source: GTFS stops.txt,

Note:

* ‘agency\_id’ is the same field as the ‘Agency’ table
* Check whether ‘stop\_id’ field in GTFS stops.txt is numeric string and convert it to integer.
* ‘stop\_code’, ‘stop\_name’, ‘stop\_desc’, ‘stop\_lat’ ,’ stop\_lon’, ‘stop\_url’, ‘location\_type’, ‘parent\_station’, and ‘wheelchair\_boarding’ are from GTFS stops.txt.
* ‘lat\_lon’ is a spatial column, its value can be set from ‘stop\_lat’ and ‘stop\_lon’ by using MySQL ‘GeomFromWKB’ function. Ignore this field if output to csv file.
* ‘version’ is the md5 string of the GTFS zip file.

**Table Route\_stop\_seq**

Table schema:

CREATE TABLE `Route\_stop\_seq` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`stop\_id` int(10) unsigned NOT NULL,

`seq` int(10) unsigned NOT NULL,

`is\_time\_point` int(10) unsigned NOT NULL Default 0,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`route\_dir`,`pattern\_id`,`seq`,`version`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS routes.txt, trips.txt, stop\_times.txt

Note: A pattern is defined as the sequence of stops of a directional route. One directional route could have multiple patterns. For example, northbound route A consists of 5 stops (stop\_ids 1, 2, 3, 4, 5). Pattern #1 stops (loading/unloading passengers) at all 5 stops; pattern #2 stops at 3 stops (stop\_ids 1, 3 and 5); and pattern #3 stops at 3 stops (stop\_ids 2, 4, and 5). The fields of Route\_stop\_seq table are generated from the combination of GTFS routes.txt, trips.txt and stop\_times.txt. For each ‘route\_id’ in routes.txt, there are multiple trips (‘trip\_id’ and ‘direction\_id’) in trips.txt that are associated with the ‘route\_id’. ‘route\_id’ + ‘direction\_id’ defines a directional route, which is unique associated with ‘route\_short\_name’ + ‘route\_dir’. Each trip on a directional route has a sequence of ‘stop\_id’ (in stop\_times.txt) associated with its ‘trip\_id’. The Route\_stop\_seq table stores the unique sequence of stop\_ids on each directional route.

* ‘agency\_id’, ‘route\_short\_name’ and ‘route\_dir’ are the same fields as the ‘Routes’ table.
* ‘pattern\_id’ (string) is generated from ‘route\_short\_name’, ‘route\_dir’ and patternSeq, and has the format route\_short\_name\_ route\_dir\_patternSeq. patternSeq starts from 1 and increases by step 1. In the example above, the 3 pattern\_ids are A\_0\_1, A\_0\_2 and A\_0\_3 where northbound route\_dir is 0.
* ‘seq’ is the sequence # of the stop\_id within the pattern\_id, starts from 1 and increases by step 1.
* ‘is\_time\_point’ is the ‘timepoint’ field in stop\_times.txt. Set its value to 0 if the ‘timepoint’ field is not included in stop\_times.txt.
* ‘version’ is the md5 string of the GTFS zip file.

**Table RunPattern**

Table schema:

CREATE TABLE `RunPattern` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`start\_date` date NOT NULL,

`end\_date` date NOT NULL,

`service\_id` varchar(255) NOT NULL,

`day` char(7) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`run` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`trip\_headsign` varchar(255) NOT NULL,

`trip\_id` bigint(20) unsigned NOT NULL,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`start\_date`,`day`,`route\_dir`,`run`,`version`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS trips.txt, calendar.txt, stop\_times.txt

Note: When processing ‘Route\_stop\_seq’, each ‘trip\_id’ in GTFS trips.txt has been associated with a ‘pattern\_id’. Combining trips.txt and calendar.txt, each ‘trip\_id’ is served on different day of week (‘service\_id’ in trips.txt and calendar.txt). The ‘RunPattern’ table stores the trip sequence (i.e. ‘run’) on each directional route by service\_id (i.e. ‘day’). Using the northbound route A example above, say this directional route provides 5 trips on weekdays (‘day’ = ‘1111100’) with trip # 1 and 5 on pattern\_id A\_0\_2 and the rest 3 trips on pattern\_id A\_0\_1; provides 2 trips on Saturday (‘day’ = ‘0000010’) with pattern\_id A\_0\_2; and provides 2 trips on Sunday (‘day’ = ‘0000001’) with pattern\_id A\_0\_3. The 9 records are:

10,A,2016-01-01,2016-06-30,xxxx,1111100,0,1,A\_0\_2,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30, xxxx,1111100,0,2,A\_0\_1,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,,xxxx,1111100,0,3,A\_0\_1,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,1111100,0,4,A\_0\_1,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,1111100,0,5,A\_0\_3,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,0000010,0,1,A\_0\_2,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,0000010,0,2,A\_0\_2,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,0000001,0,1,A\_0\_3,xxxx,xxxx,xxxx

10,A,2016-01-01,2016-06-30,xxxx,0000001,0,2,A\_0\_3,xxxx,xxxx,xxxx

* ‘agency\_id’, ‘route\_short\_name’, ‘route\_dir’, ‘pattern\_id’ and ‘version’ are the same fields as the ‘Route\_stop\_seq’ table.
* ‘start\_date’ and ‘end\_date’ are from calendar.txt (convert the format from YYYYMMDD to YYYY-MM-DD).
* ‘service\_id’ from calendar.txt
* ‘day’ is char[7], with char[0] = Monday field in calendar.txt, and char[6] = Sunday field in calendar.txt
* ‘run’ is the sequence # of trips (in trips.txt) that are associated with ‘agency\_id’, ‘route\_short\_name’, ‘route\_dir’ and ‘day’. ‘run’ starts from 1 and increases by step 1.
* ‘trip\_headsign’ and ‘trip\_id’ are from GTFS trips.txt. If ‘trip\_headsign’ field is included in trips.txt and is not blank, use it as ‘trip\_headsign’; otherwise use the first ‘stop\_headsign’ in stop\_times.txt on the ‘trip\_id’.

**Table Schedules**

Table schema:

CREATE TABLE `Schedules` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`start\_date` date NOT NULL,

`end\_date` date NOT NULL,

`day` char(7) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`run` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`seq` int(10) unsigned NOT NULL,

`stop\_id` int(10) unsigned NOT NULL,

`is\_time\_point` int(10) unsigned NOT NULL Default 0,

`pickup\_type` int(10) unsigned NOT NULL,

`dropoff\_type` int(10) unsigned NOT NULL,

`arrival\_time` varchar(10) NOT NULL,

`departure\_time` varchar(10) NOT NULL,

`stop\_headsign` varchar(255) default NULL,

`trip\_id` bigint(20) unsigned NOT NULL,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`start\_date`,`day`,`route\_dir`,`run`,`seq`,`version`),

KEY `start\_date` (`start\_date`),

KEY `route\_short\_name` (`route\_short\_name`),

KEY `run` (`run`),

KEY `day` (`day`),

KEY `stop\_id` (`stop\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS trips.txt, stop\_times.txt

Note: The ‘Schedule’ table stores the arrival and departure times in stop\_times.txt for each trip.

* ‘agency\_id’, ‘route\_short\_name’, ’start\_date’, ’end\_date’, ’day’, ‘route\_dir’, ‘run’, ‘pattern\_id’, ‘trip\_id’ and ‘version’ are the same fields as the ‘RunPattern’ table.
* ‘seq’, ‘stop\_id’ and ‘is\_time\_point’ should be consistent with the ‘Route\_stop\_seq’ table.
* ‘pickup\_type’, ‘dropoff\_type’, ‘stop\_headsign’, ‘arrival\_time’ and ‘departure\_time’ are from stop\_times.txt.

**Table Route\_point\_seq**

Table schema:

CREATE TABLE `Route\_point\_seq` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`shape\_id` varchar(255) NOT NULL,

`point\_id` int(10) unsigned NOT NULL,

`seq` int(10) unsigned NOT NULL,

`length` double NOT NULL,

`heading` double NOT NULL,

`dist` double NOT NULL,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`route\_dir`,`pattern\_id`,`seq`,`version`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS routes.txt, trips.txt, shapes.txt

Note: The ‘Route\_point\_seq’ table is in parallel with the ‘Route\_stop\_seq’ table. While the ‘Route\_stop\_seq’ describes a directional route by a sequence of stops, the ‘Route\_point\_seq’ describes its physical road by a sequence of way-points. For each ‘pattern\_id’ in ‘Route\_stop\_seq’, there are multiple trips associated with it (in trips.txt). Each trip has a ‘shape\_id’ that defines the shape (i.e. road) for the trip. ‘shape\_id’ is referenced from shapes.txt, which defines the way-points of a shape. For trips with the same ‘pattern\_id’, please verify that they have a same ‘shape\_id’, otherwise a same sequence of stops could go on different roads. If that’s the case (one pattern\_id maps to multiple shape\_ids), increasing patternSeq (see note in Table Route\_stop\_seq) such that a pattern\_id unique defines a sequence of stops and a sequence of way-points.

* ‘agency\_id’, ‘route\_short\_name’, ‘route\_dir’, ‘pattern\_id’ and ‘version’ are the same as the Route\_stop\_seq table.
* ‘shape\_id’ is from trips.txt
* ‘point\_id’ is referenced from the Points table below
* ‘seq’ is the sequence # of the way-points defined by the sape\_id (from shape.txt). ‘seq’ starts from 1 and increases by step 1.
* ‘length’, ‘heading’ and ‘dist’ are calculated from the way-point sequence, with length (in meters) be the straight-line distance between consecutive way-points (seq # *i* and *i+1*), heading (in degree) be the direction w.r.t. north from seq # *i* to *i+1*, and dist be the accumulate length counting from the first way-point.

**Table Points**

Table schema:

CREATE TABLE `Points` (

`agency\_id` int(10) unsigned NOT NULL,

`point\_id` int(10) unsigned NOT NULL,

`point\_lat` double NOT NULL,

`point\_lon` double NOT NULL,

`lat\_lon` point NOT NULL,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`point\_id`,`version`),

SPATIAL KEY `lat\_lon` (`lat\_lon`)

) ENGINE=MyISAM DEFAULT CHARSET=latin1

Source: GTFS shapes.txt

Note: Each ‘shape\_id’ consists of a sequence of way-points defined in shapes.txt (‘shape\_pt\_lat’ and ‘shape\_pt\_lon’). The “Points’ stores the unique way-points in shapes.txt (two way-points are considered as a same point if they have the same values of shape\_pt\_lat and shape\_pt\_lon).

* ‘point\_id’ is the sequence # of the unique way-points (no particular order)
* ‘point\_lat’ and ‘point\_lon’ are the ‘shape\_pt\_lat’ and ‘shape\_pt\_lon’ of the ‘point\_id’.
* ‘lat\_lon’ field can be ignored when output to cvs file. This field can be generated in MySQL, from ‘point\_lat’ and ‘point\_lon’.

**Table Fare**

Table schema:

CREATE TABLE `Fare` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`price` double NOT NULL,

`currency\_type` varchar(255) NOT NULL,

`payment\_method` int(10) unsigned NOT NULL,

`origin\_id` int(11) NOT NULL default '-1',

`destination\_id` int(11) NOT NULL default '-1',

`transfers` int(11) NOT NULL default '-1',

`transfer\_duration` int(10) unsigned NOT NULL default '0',

`fare\_id` varchar(45) NOT NULL default 'Regular',

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`route\_dir`,`pattern\_id`,

`origin\_id`,`destination\_id`, `fare\_id`,`version`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS fare\_rules.txt, fare\_attributes.txt, stops.txt

Note: The ‘Fare’ table stores transit fare between each stop pair (from ‘origin\_id’ to ‘destination\_id’) on each pattern\_id.

* ‘agency\_id’, ‘route\_short\_name’, ‘route\_dir’, ‘pattern\_id’, and ‘version’ are the same as the ‘Route\_stop\_seq’ table.
* For seq in Route\_stop\_seq.seq

‘origin\_id’ = Route\_stop\_seq.stop\_id

Find origin\_zone\_id from stops.txt

For lp in (Route\_stop\_seq.seq > seq)

‘destination\_id’ = Route\_stop\_seq.stop\_id

Find destination\_zone\_id from stops.txt

Fine ‘fair\_id’ in fare\_rules.txt

Referenced to ‘fair\_id’, find ‘price’, ‘currency\_type’, ‘payment\_method’, ‘transfers’ and ‘transfer\_duration’ from fare\_attributes.txt.

**Table Calendar\_dates**

Table schema:

CREATE TABLE `Calendar\_dates` (

`agency\_id` int(10) unsigned NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`special\_date` date NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`run` int(10) unsigned NOT NULL,

`exception\_type` int(10) unsigned NOT NULL,

`day` char(7) NOT NULL,

`service\_id` varchar(255) NOT NULL,

`version` varchar(255) NOT NULL,

PRIMARY KEY (`agency\_id`,`route\_short\_name`,`special\_date`,`run`,`route\_dir`,`day`,`version`),

KEY `special\_date` (`special\_date`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS calendar\_dates.txt

Note: The Calendar\_dates table stores changing of transit service on specified dates (e.g. holidays).

* Use ‘service\_id’ in calendar\_dates.txt to link with affected trips in RunPattern table.
* ‘agency\_id’, ‘route\_short\_name’, ‘route\_dir’, ‘run’, ‘day’, ‘service\_id’ and ‘version’ are the same as ‘RunPattern’ table.
* ‘special\_date’ is the ‘date’ filed in calendar\_dates.txt (converting from YYYYMMDD format to YYYY-MM-DD format).
* ‘exception\_type’ is from calendar\_dates.txt

**Table Transfers**

Table schema:

CREATE TABLE `Transfers` (

`from\_agency\_id` int(11) NOT NULL default '-1',

`from\_id` int(10) unsigned NOT NULL,

`to\_agency\_id` int(11) NOT NULL default '-1',

`to\_id` int(10) unsigned NOT NULL,

`transfer\_type` int(10) unsigned NOT NULL,

`min\_transfer\_time` int(10) unsigned NOT NULL,

`transfer\_dist` int(11) NOT NULL default '0',

PRIMARY KEY (`from\_agency\_id`,`from\_id`,`to\_agency\_id`,`to\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Source: GTFS transfers.txt, Route\_stop\_seq, Stops

Note: GTFS usually does not include transfer.txt (an optional file). When it is included, transfer.txt may only include few key transfer points. The Transfer table is filled with data stored in Route\_stop\_seq table. The rules for generating entries in this table are:

1. Set a bound on maximum transfer distance (the straight-line distance between from\_stop\_id (i.e. ‘from\_id’) and to\_stop\_id (i.e. ‘to\_id’); and
2. No transfer needed on the same patter\_id

* ‘transfer\_dist’ (in meters) is the straight-line distance between the two stops
* ‘min\_transfer\_time (in seconds) is the minimum walking time between the two stops
* If GTFS includes transfers.txt, use its ‘transfer\_type’ and ‘min\_transfer\_time’; otherwise set ‘transfer\_type’ to 0.
* It is better to querying Google Walk Route to get ‘transfer\_dist’ and ‘mini\_tranfer\_time’ between the two stops.

1. Tables to store dynamic transit data (obtained from GTFS real-time)

Note: Real-time transit location data were obtained from NextBus, and the arrival prediction data were generated with our programs. Not sure what data elements are included in the GTFS real-time data feed, we may need to modify the existing tables or generate new tables to work with GTFS real-time data feed.

**Table gps\_fixes**

Table schema:

CREATE TABLE `gps\_fixes` (

`agency\_id` int(10) unsigned NOT NULL,

`veh\_id` int(11) NOT NULL,

`RecordedDate` date NOT NULL,

`RecordedTime` time NOT NULL,

`UTC\_at\_date` date NOT NULL,

`UTC\_at\_time` time NOT NULL,

`latitude` double NOT NULL,

`longitude` double NOT NULL,

`speed` double NOT NULL,

`course` double NOT NULL,

PRIMARY KEY USING BTREE (`agency\_id`,`veh\_id`,`RecordedDate`,`RecordedTime`),

KEY `RecordedDate` (`RecordedDate`),

KEY `veh\_id` (`veh\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Note:

* ‘agency\_id’ is the same as static transit tables
* ‘RecordedDate’ and ‘RecordedTime’ are timestamp when the entry is inserted into the DB (or received from GTFS real-time feed)
* ‘UTC\_at\_date’ and ‘UTC\_at\_time’ are from real-time data feed, which tells the UTC time when an update of a vehicle location became available
* ‘veh\_id’, ‘latitude, ‘longitude’, ‘speed’, and ‘course’ are from real-time data feeds regarding the details of a vehicle location update.

**Table TransitETA**

Table schema:

CREATE TABLE `TransitETA` (

`agency\_id` int(10) unsigned NOT NULL,

`RecordedDate` date NOT NULL,

`RecordedTime` time NOT NULL,

`veh\_id` int(11) NOT NULL,

`veh\_lat` double NOT NULL,

`veh\_lon` double NOT NULL,

`veh\_speed` double NOT NULL,

`veh\_location\_time` bigint(20) NOT NULL,

`route\_short\_name` varchar(255) NOT NULL,

`route\_dir` int(10) unsigned NOT NULL,

`day` char(7) NOT NULL,

`run` int(10) unsigned NOT NULL,

`pattern\_id` varchar(255) NOT NULL,

`stop\_id` int(10) unsigned NOT NULL,

`seq` int(10) unsigned NOT NULL,

`ETA` time NOT NULL,

PRIMARY KEY USING BTREE (`agency\_id`,`veh\_id`,`RecordedDate`,`RecordedTime`,`route\_short\_name`,`route\_dir`,`day`,`run`,`seq`,`ETA`),

KEY `route\_short\_name` (`route\_short\_name`),

KEY `RecordedDate` (`RecordedDate`),

KEY `stop\_id` (`stop\_id`),

) ENGINE=InnoDB DEFAULT CHARSET=latin1

Note:

* ‘agency\_id’, ‘veh\_id’, ‘veh\_lat’, ‘veh\_lon’, ‘veh\_speed’, and ‘veh\_location\_time’ are the same as ‘gps\_fixes’ table.
* ‘RecordedDate’ and ‘RecordedTime’ are timestamp when the entry is inserted into the DB.
* ‘Route\_short\_name’, ‘route\_dir’, ‘day’, ‘run’ and ‘pattern\_id’ were generated by matching vehicle location with ‘RunPattern’ (check whether GTFS real-time feed includes the ‘trip\_id’ field. If so, can use ‘trip\_id’ to obtain directional route info)
* ‘ETA’ was generated for all downstream stop\_ids within the ‘pattern\_id’ using ‘Route\_stop\_seq’ and ‘Route\_point\_seq’ tables (check whether GTFS real-time feed already contains updated arrival or departure times)