2018.06.15 VIPA AND VANESSA INTEGRATION NOTES

—2018.06.15.ET

Introduction

This document describes the current state of the **VIPA** project now that the **VANESSA** database is up and running and all **VIPA** data are managed alongside non-**VIPA** data in **VANESSA**.

VANESSA now serves as a single repository of **ICAM** data and currently includes the following types of incidents:

** single vehicle*
** vehicle-on-vehicle*
** pedestrian*
** bicyclist*

There is no separate VIPA database. Instead, VANESSA includes all data, including all VIPA-specific data, within its various tables and forms. In the web interface, VIPA-specific fields are marked in *green* and anyone entering VIPA data should pay particular attention to fill in these fields.

There are also many *common* fields that are shared among both VIPA and ICAM cases. *Common* fields that were noted as belonging to both VIPA and ICAM are marked in pale *yellow*. However, as VANESSA now contains hundreds of data elements, it is likely that a number of fields may not be specifically marked as common even though in reality they are. Therefore, *the wisest course of action may be to fill in as many data fields as possible, regardless of the "traditional" distinctions among case types*.

How are VIPA cases identified?

The **incident** table contains a column called **first_crash_tag**. This column contains ICAM identifiers such as "*ICAM-2017-P24*" or "*ICAM-2018-B03*". The presence of a "**P**" or a "**B**" in the identifier tells you that it is a *pedestrian* or *bicyclist* incident, respectively.

Also, in the **person** table, there is a column called **icam_crash_tag** which will contain the same sort of ICAM identifier.

When running SQL SELECT queries, you can use these identifiers to pull out specific types of cases. Additionally, the **entity** table in VANESSA has a **type** column which resolves to *vehicle, bicyclist, pedestrian,* or *other*.

What is the difference between first_crash_tag and icam_crash_tag?

Suppose there is an incident in which four people were injured. The first person injured was assigned and ICAM identifier of "*ICAM-2016-04*", the second "*ICAM-2016-05*", the third "*ICAM-2016-06*", and the fourth "*ICAM-2016-07*".

In **VANESSA**, this is considered a single *incident* with four *patient cases*. The **first_crash_tag** in the **incident** table will contain "*ICAM-2016-04*", while the records for the individuals in the **person** table will have their respective unique **icam_crash_tag**s.

What about the VIPA Web Application?

It is, for the moment, still present at:

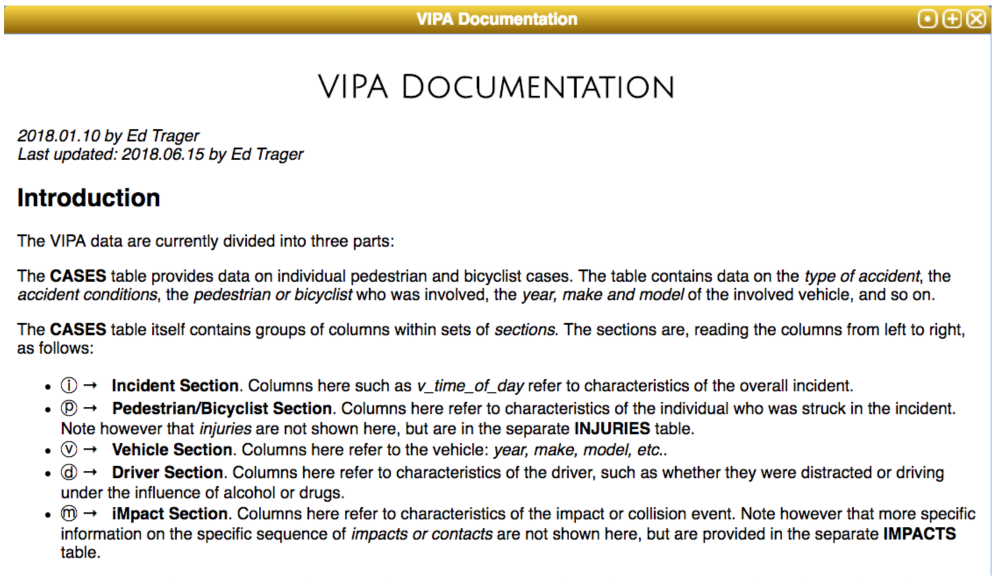
https://madeline.med.umich.edu/vipa/

However please note that this URL will change soon to an ICAM-managed URL.

VIPA Web Application Updates

The VIPA web application has been updated in the following areas:

- The data tables are now snapshots of data views exported out of VANESSA.
- The data tables themselves, although largely similar to those that were present in the old VIPA app, have been changed and updated in various ways reflecting their new VANESSA origin.
- The key identifiers have been changed to *v_incident_id, v_person_id*, and *v_iop_id,* reflecting their new VANESSA origin.
- The INJURY table now includes *iss_max_ais_severity* and *iss_body_region* columns
- The CASES table now includes *month* and *time* columns to provide a sense of when incidents occurred without revealing specific dates.
- The documentation page found on the VIPA web app has been revised and updated to reflect all changes. *(fig. 1)*.
- The VIPA app now handles Unicode data much better than before.
- The data download file creation process now provides a "hint" that Microsoft Excel uses to distinguish Unicode data files from legacy character set encoded files.



VIPA Documentation

2018.01.10 by Ed Trager
Last updated: 2018.06.15 by Ed Trager

Introduction

The VIPA data are currently divided into three parts:

The **CASES** table provides data on individual pedestrian and bicyclist cases. The table contains data on the *type of accident*, the *accident conditions*, the *pedestrian or bicyclist* who was involved, the *year, make and model* of the involved vehicle, and so on.

The **CASES** table itself contains groups of columns within sets of *sections*. The sections are, reading the columns from left to right, as follows:

- ⓘ → **Incident Section**. Columns here such as *v_time_of_day* refer to characteristics of the overall incident.
- ⓟ → **Pedestrian/Bicyclist Section**. Columns here refer to characteristics of the individual who was struck in the incident. Note however that *injuries* are not shown here, but are in the separate **INJURIES** table.
- ⓥ → **Vehicle Section**. Columns here refer to the vehicle: *year, make, model, etc.*.
- ⓓ → **Driver Section**. Columns here refer to characteristics of the driver, such as whether they were distracted or driving under the influence of alcohol or drugs.
- ⓜ → **iMpact Section**. Columns here refer to characteristics of the impact or collision event. Note however that more specific information on the specific sequence of *impacts or contacts* are not shown here, but are provided in the separate **IMPACTS** table.

Fig. 1. The VIPA web app documentation has been revised and updated.

Data export from VANESSA and import to VIPA web App process

As the VIPA data are now part-and-parcel of VANESSA, the process of obtaining the data for display in the VIPA web application has changed. Previously, VIPA data were imported from a combination of spreadsheets produced by Kristen and SQL queries off the old ICAM database. The process was laborious and not streamlined.

Now the data are in VANESSA and the process is uniform and streamlined. The following paragraphs document the new process for exporting the data from VANESSA, preparing the data, and then importing the data for use by the VIPA web application.

Note that for historical reasons, the VIPA web app was written to talk to a MySQL database, not a Postgres database. This reality introduces a few extra steps into the process. In addition, it is necessary to preserve Unicode-encoded data. This requires paying careful attention to a few extra commands and command-line parameters, especially when transferring data across unrelated or possibly unknown server environments. The process is documented in the following paragraphs:

First, I have created the following three views in the `vanessa` database:

```
------------+------------------------------------+----------+-----------
   database | view name                          | type     | owned by
------------+------------------------------------+----------+-----------
    vanessa | sys_view_vipa_main                 | view     | mluser
    vanessa | sys_view_vipa_injury               | view     | mluser
    vanessa | sys_view_vipa_contact              | view     | mluser
------------+------------------------------------+----------+-----------
```

The SQL script to create or re-create these views is called `VIPA_reconstruction.sql` and is saved in the **ICAM Github** repository under my **Axalotl** project directory.

These views contain exactly the same columns and data as seen in the new VIPA web app and there should be no ePHI in these views.

These views are live views, while the data consumed by the web app are snapshots taken from the views at some discreet point in time.

Because views cannot be exported in the way that tables are exported, the export process requires generating discardable “snapshot” tables as follows:

① Create “snapshot” tables from the views using this SQL pattern:

```
create table <snapshot_table_name> as (select * from <sys_view_name>);
```

This results in the following tables:

```
------------+------------------------------------+----------+-----------
   database | snapshot table name                | type     | owned by
------------+------------------------------------+----------+-----------
    vanessa | vipa_contact_snapshot              | table    | mluser
    vanessa | vipa_injury_snapshot               | table    | mluser
    vanessa | vipa_main_snapshot                 | table    | mluser
------------+------------------------------------+----------+-----------
```

② These snapshots can then be exported to SQL files using the following `pg_dump` command pattern:

```
pg_dump -h pgicam-prod-db.umms.med.umich.edu -U mluser -d sahmdb
-t vanessa.<snapshot_table_name> --format p --inserts > <output_file_name>.sql
```

③ As the resulting **SQL** files retain certain **Postgres**-specific features, a minimal amount of manual editing is required to remove **Postgres**-specific features. Also, if the files will be imported into a **MySQL** based system (*as is currently done, but this may change in the future*), then it will likewise be necessary to add a few **MySQL**-specific features. In particular, the following MySQL-specific command should precede all other commands:

```
set names "utf8";
```

Also, the column name "`condition`" cannot be used as this is a reserved word in **MySQL**, so this can be changed to "`condition_`". Failure to do so will result in an error when you attempt the import.

④ Once the files have been prepared, they can be imported. For MySQL, the command pattern looks like this:

```
mysql -u <username> -p --default-character-set=utf8 <snapshot_file_name>.sql
```

That's it!

*** END OF DOCUMENT ***