Apricot DB

The User Guide

v 0.3

Contents

Preface

The “Apricot DB” is a desktop application, which purpose is to help users with the design, generation, analysis and maintenance of the structures of the complex relational databases.

The main presumable audience of this software are the software developers, analysts, database designers, all sort of the software architects and even some advanced participants of the company’s business processes. They all have been working with the many-tables, non trivial databases and need a tool for the graphic representation of the existing and newly created database- structures.

The database structure is a set of tables, connected with each other by relationships (that’s been making the database “relational”). Tables also can be called “entities”. The relatively simple idea of the entity-relationship diagram (ER-diagram, or ERD) lies in the background of the graphical representation of the database structure.

A list of essential functions of “Apricot DB” includes the following:

* Creation of the database structure in form of ERD;
* Filling in the necessary detailed information for all elements of the ERD and generate the target database scrips;
* Perform the Reverse Engineering from of the physical (existing) structure of the target database into the ERD for further analysis, development and alterations;
* Prepare and publish the technical documentation using clear and descriptive graphical representation of the database structure;
* Maintain the full cycle of the design database “from scratch” process starting from the initial prototyping of the ERD’s and finishing by generations of the final database scripts to create, update or delete physical objects of the database);
* Generation of the Excel Report with the detailed representation of the current EDR information.

No Open Source applications at the moment of writing of this documentation provide such functionality out of the box.

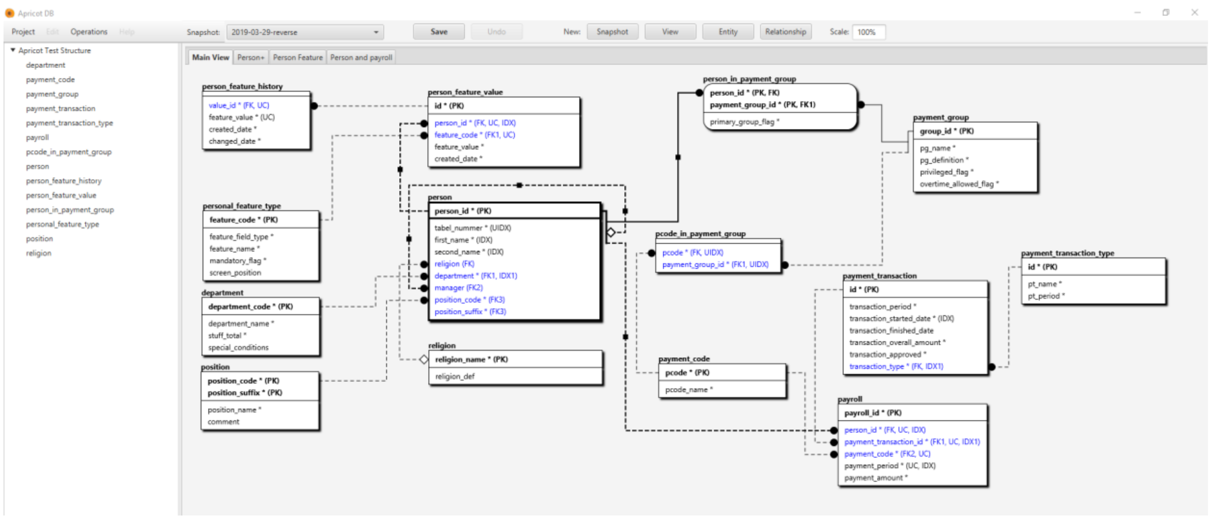
What “Apricot DB” does not do

Even though “Apricot DB” is designed to maintain all major relational databases, it has limitations on working with some physical objects specific for concrete database. “Apricot DB” does not support any programmatic structures on the database side (example: PL/SQL in Oracle, Transact SQL in SQL Server, any sort of database triggers).

The current version of “Apricot DB” does not support the database views (this support, most probable will be included into the further releases).

Introduction

The “Apricot DB” application is similar to the “classic” IDE’s like Eclipse or NetBeans. The top element of the interface is the Project. There might be any number of the Projects created, stored and maintained in “Apricot DB”, but only one Project can be opened at any particular moment of time.



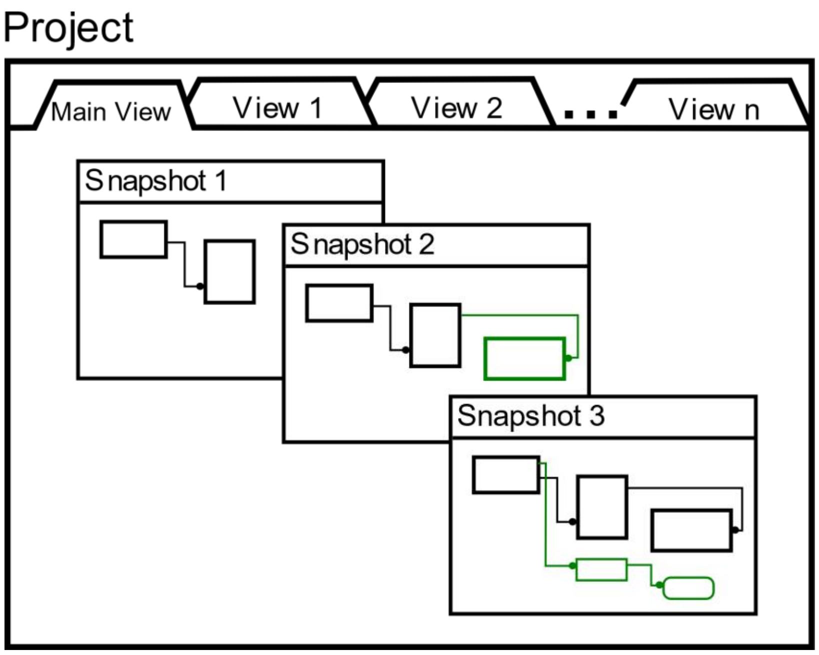
Ideally, one Project is dedicated to one database (or schema, or any other stable and isolated database structure depending on the type of the database). In my example (see the screenshot) the project named “Apricot-Test-Structure” represents the SQL Server database - “APRICOT_TEST”.

The Project contains Snapshots and Views.

Apricot DB Project/Snapshot/View concept

A Project includes Snapshots and Views. Snapshot is a container of the Entities and relationships between them. There might be many Snapshots inside one project. This permits to keep different versions of the same database.

The Views are responsible for the layout of the graphical representation of the database structure. The View can include a subset of the whole set of tables. This makes analysis and documenting of the complex databases more convenient.



Important notice: The Views belong to the whole Project (not to some particular Snapshot). If a new View was created, it will be presented in all Snapshots of the Project. As soon as some change made in allocation of the Entity/Relationships presented in this particular View, these changes will be reflected in all Snapshots. The View in "Apricot DB" can be considered as a "skin", which changes have automatically been applied to all Snapshots. Think about the View as a CSS sheet, which is applied to all pages in the Internet Site. As soon as CSS is changed, the look and feel of all pages will change, even though the internal data represented by the pages stay the same.

The Snapshot

The Snapshot is a main container of tables and relationships between them. Project has to have at least one Snapshot. There might be any number of Snapshots in one Project. The different Snapshots in the Project might have not equal collections of tables/relationships. The concept of the Snapshot might be considered as a primitive version control system, which stores a graphical representation of the database structure at the different moments of time. Technically, the Snapshots can have completely different and not overlapping sets of tables in them. It makes sense, though, to stick with the logically the same database structure in all Snapshots of the Project. It allows to perform efficient comparison between different Snapshots in one Project as well as use an advantage of the concept of View.

The View

The view contains the graphical representation of entities/relationships (see the screenshot above). Do not confuse the "Apricot DB" View with the database views which can been created by CREATE VIEW… DDL command!

There always one mandatory View included in any Project - the Main View. It shows all tables of the chosen Snapshot. The Main View cannot be removed.

For the large database structures, which might include hundreds of tables and relationships between them, it is more convenient to "split" the whole set of the tables of the current Snapshot into the logical subsets: Views.

Physical or Logical

It is common practice to distinguish between the logical and physical ER- models. The logical model represents the higher level of abstraction comparing to the physical one. The physical representation usually contains great details of the physical parameters of the designed database, while the logical ERD abstracts many details specific for the physical representation.

In sake of simplification (and the simplicity is a vital thing!) "Apricot DB" does not distinguish between physical and logical schemas. We do not support the logical ERD- elements such as many-to-many relationships, the explicit cardinality on different sides of the relationship or the logical names of entities.

All relationships in "Apricot DB" are of "one-to-many" type. This is essential feature of the modern relational databases. The "many-to-many" relationship has be resolved using the "association table" pattern (will be touched below).

"Apricot DB" supports the only name for the Entity or the Field Name within the Entity. All the identifications have been considered as "physical" and become the final names of the tables, their fields and constraints in the generated database scripts.

Entities and Tables

According to IDEF1x standard Entities are elements of the ER- diagrams. Entities have been converted into the physical tables of the target database during the DB script generations. We will use terms Entity and Relationships as mutually interchangeable.

The Attributes within Entity of the entity relationship diagram have been converted into the fields of the table of the target database. We assume Attribute and Field terms interchangeable.

How “Apricot DB” stores the Project Data

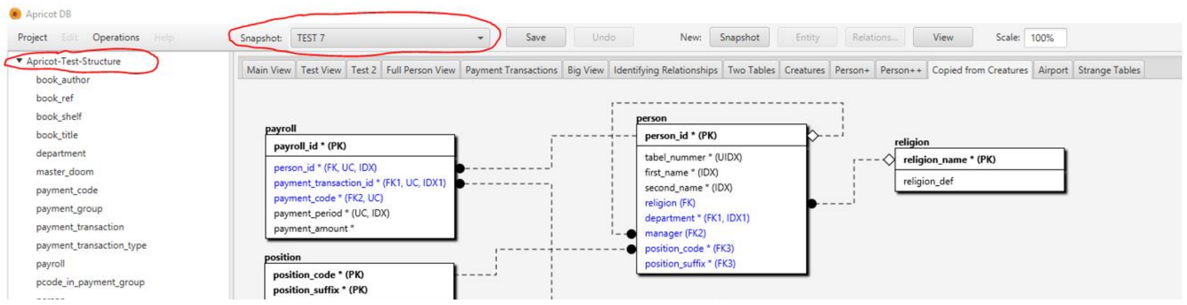
The “Apricot DB” Project is a long living thing. A designer of the database wants to be able to save the results of her work and been able to get back to the design process later. The multiple changes have been done during the design of ERD and the final database structure.

In order to store the Project data, “Apricot DB” uses the lightweight file-based database H1. This database contains the meta-data of all Projects created and saved in the current local environment.

Opening any Project, “Apricot DB” automatically restored the condition maximally close to the latest working session with this particular Project.

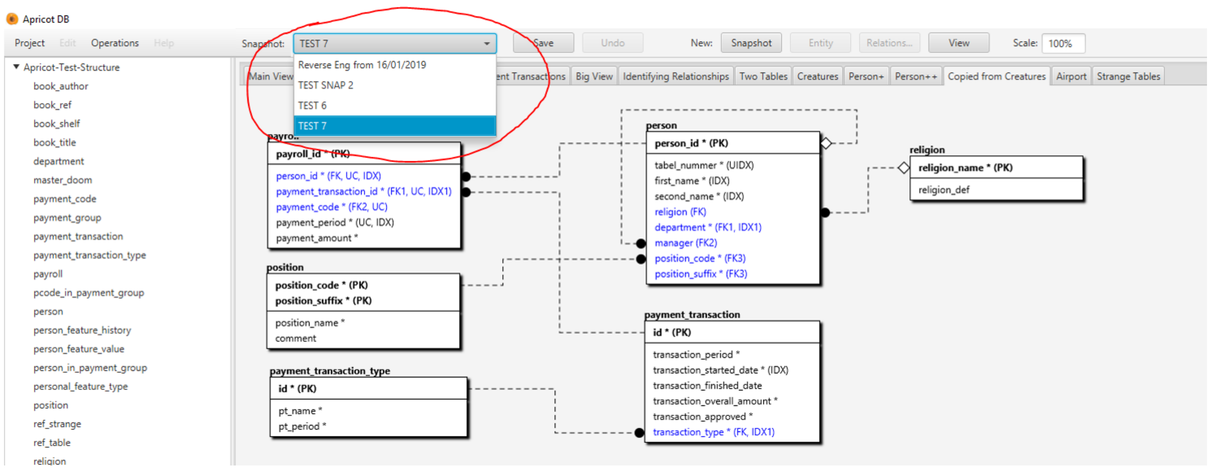
The Elements of the “Apricot DB”- main screen

The current Project and Snapshot: the current Project is shown on the left pane. The current Snapshot, selected inside the current Project is presented by the drop-down list.



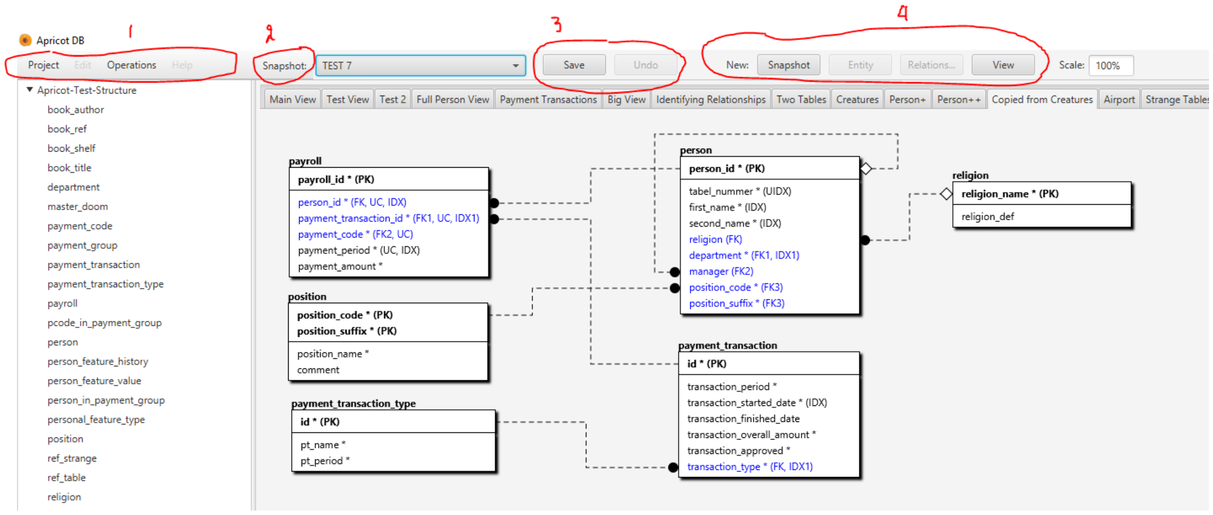
A collection of tables included into the current snapshot is shown under the current Project- name (see the left pane).

To select other snapshot use the drop down list shown below. As soon as Snapshot is selected in the drop down list, it becomes the current one.



The current Project and Snapshot have been stored between different sessions of work with "Apricot DB". This means, that when you run "Apricot DB" application next time, it automatically selects the Project and Snapshot selected in the end of the previous working session.

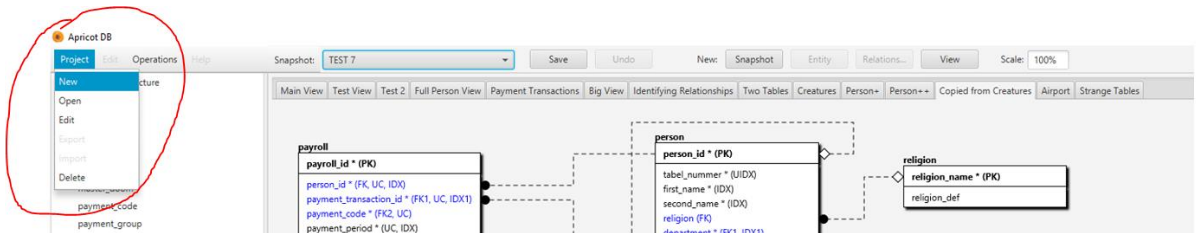
Menu and tool bar



1) the main menu. It permits access to the most important functions of "Apricot DB";
2) menu "Snapshot" – allows edit and delete the currently selected Snapshot;
3) menu "Save/Undo" – saves or undoes the recent changes in the selected view;
4) menu "New" – allows to create new Snapshots, Views, Entities and Relationships (will be discussed later).

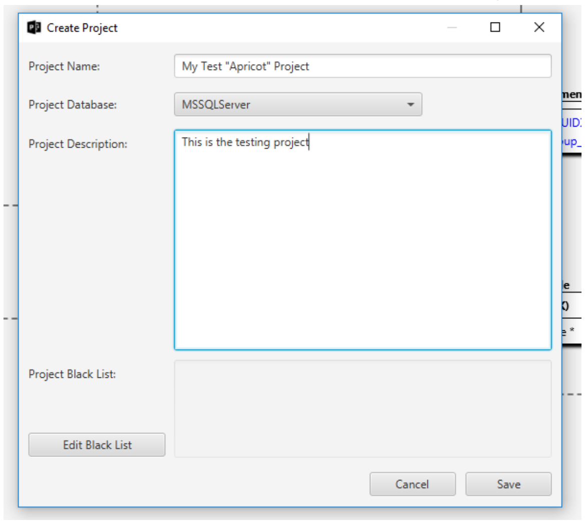
The creation a new Project

The first thing needed to start working with "Apricot DB" is creation of the new Project. To create a new Project, select Project/New in main menu.



The form of creation of the new Project is shown below.

The Project Name is mandatory. On the stage of creation of the new project the type of the target database has to be selected (the field Project Database). The Project Description is optional.



Project Name: My Test "Apricot" Project

Project Database: MSSQLServer

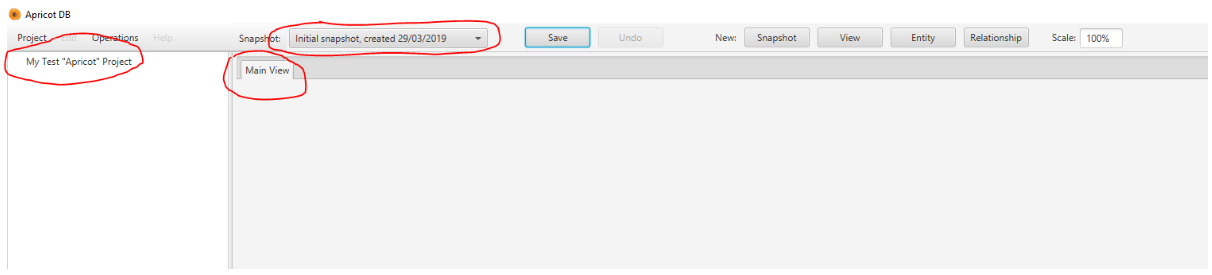
Project Description: This is the testing project

Project Black List:

After the “Save” button was pushed, the new Project was created and presnted in the application.

The new project “My Test “Apricot” Project” does not have any tables yet, therefore the tables list is empty (see the screenshot below).

By default “Apricot DB” created an empty Snapshot – “Initial snapshot created 29/03/2019”. The default name of the initial Snapshot can be changed later. The only view on the screen is the “Main View”. This view is empty since there is no entities in the project yet.

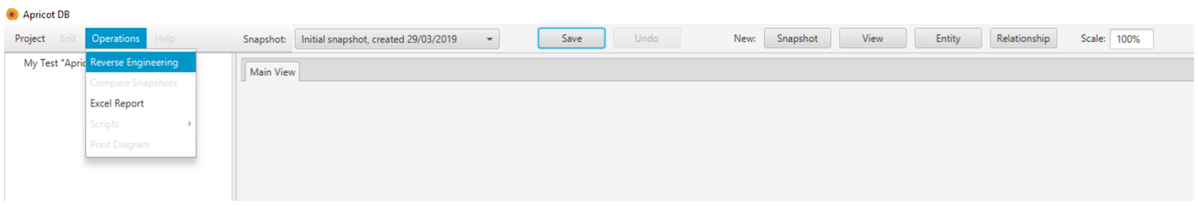


The “ApricotDB” Scenarios

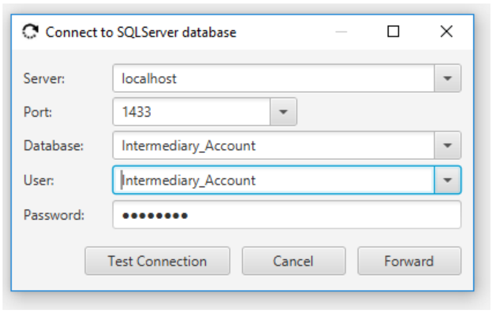
There are two main scenarios of working with “Apricot DB”:
* The Entity/Relationships in the current Project can be created from scratch;
* The database elements can be pulled from the real database. This process called Reverse Engineering.

The Reverse Engineering Process

Let’s assume that we’ve got access to some real existing database, which structure we would like to start working with (analyse, alter, extend).



Using menu Operations/Reverse Engineering, the form of connection to the Database is shown below:


Connect to SQLServer database

Server:	localhost
Port:	1433
Database:	Intermediary_Account
User:	Intermediary_Account
Password:	••••••••

After the successful connection to the target database, Apricot DB scans the database structure and offers a list of the tables found:

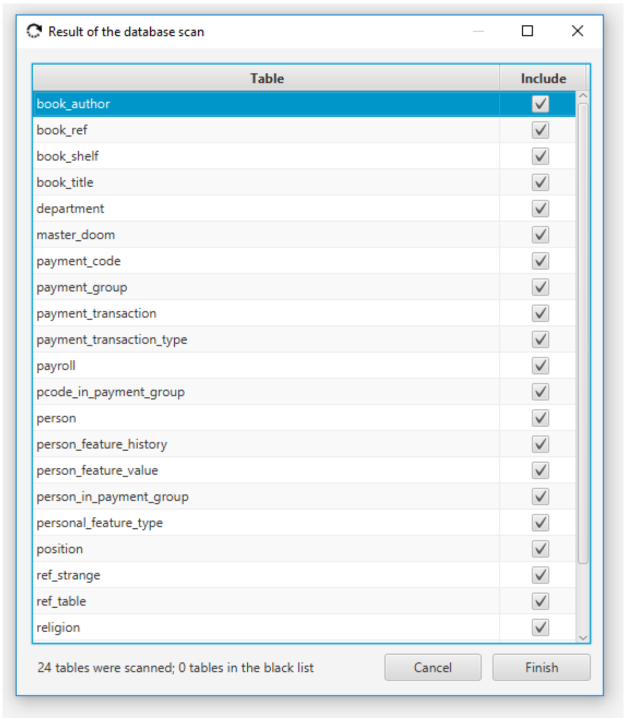

Result of the database scan

Table	Include
book_author	✓
book_ref	✓
book_shelf	✓
book_title	✓
department	✓
master_doom	✓
payment_code	✓
payment_group	✓
payment_transaction	✓
payment_transaction_type	✓
payroll	✓
pcode_in_payment_group	✓
person	✓
person_feature_history	✓
person_feature_value	✓
person_in_payment_group	✓
personal_feature_type	✓
position	✓
ref_strange	✓
ref_table	✓
religion	✓

24 tables were scanned; 0 tables in the black list

The selected tables will be “pulled” into the current Project/Snapshot. Any tables in the list can be excluded from the resulting set on this stage.
Note: Unchecking the tables in the “Include” list, excludes them from the resulting set. This operation has potential risk of violation of the referential integrity of the resulting ER Diagram. In order to prevent that, “Apricot DB” analyses the referential integrity existing in the current database schema. It warns if any integrity violations have been identified on this stage. For example, there is a table “A” and it is a parent of the tables “B” and “C”. If the parent table “A” is excluded from the result, the tables “B” and “C” have to be removed as well. Otherwise the tables “B” and “C” would have the foreign keys with no Primary Key, related to them.

If potential violation of the referential integrity was found, “Apricot DB” warns the user with the message shown below:

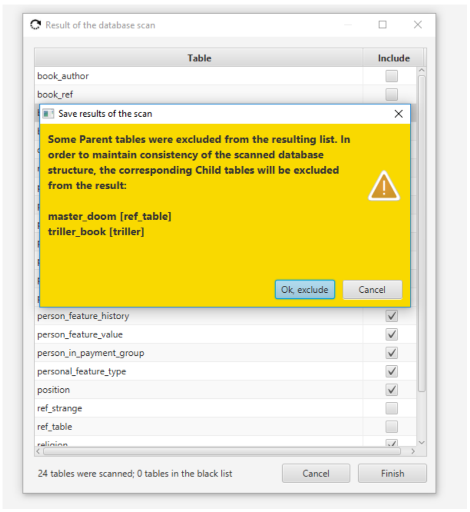

Result of the database scan

Table	Include
book_author	☐
book_ref	☐
person_feature_history	✓
person_feature_value	✓
person_in_payment_group	✓
personal_feature_type	✓
position	✓
ref_strange	☐
ref_table	☐

Save results of the scan

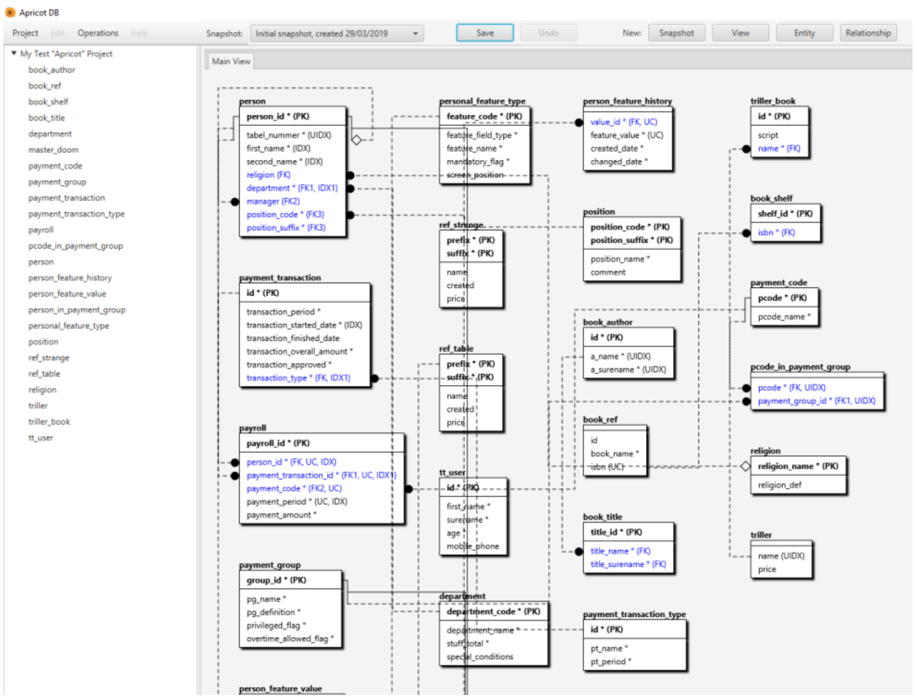
Some Parent tables were excluded from the resulting list. In order to maintain consistency of the scanned database structure, the corresponding Child tables will be excluded from the result:

master_doom [ref_table]
triller_book [triller]

24 tables were scanned; 0 tables in the black list

If the user accepts the suggestion given by the application, the related tables will be removed from the result as well. This helps to keep the database structure consistent at any moment of time.

That’s how the initial structure, red from the database looks like after the Reverse Engineering process:



(note: the current default layout is simplified and will be improved in the forthcoming versions of the app).

The graphical notation of “Apricot DB”

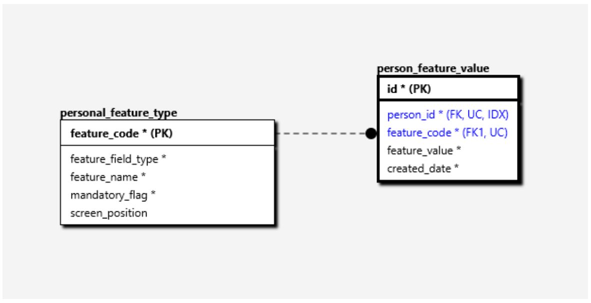
“Apricot DB” uses the standard notation IDEF1x. There are not sufficient simplifications, extensions and alterations, though.

IDEF1x is very intuitive ERD- notation. It is not contradictive to other big desktop systems like “Enterprise Architect”, “ERWin”, “Oracle Designer”, “IBM Rational Rose” and so one.

Entities and Relationships between them

The Entities on the ER- diagram are connected with Relationship (there might be the stand-alone entities, which have been by themselves and do not have any relationships with others).

In the Relationship below, the Entity on the left side called a Parent. The Entity of the right side called a Child. In other popular terminology they would be a Master and a Slave.



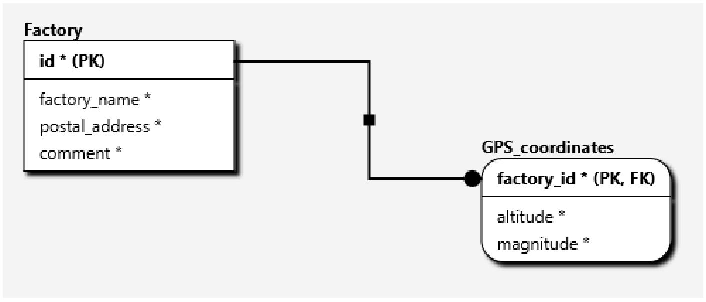
They say that the Primary Key of the Parent is exported to the Child. The corresponding field of the Child Entity called the Foreign Key.

Often the Parent entity acts as a reference or detailed entity. They say that the Child Entity refers to the Parent.

There are 3 types of the relationships between Parent and Child.

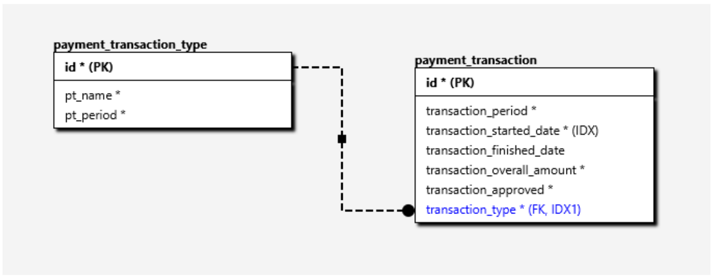
The Identifying Relationship

…takes place when the Parent’s Primary Key is included into the Child’s Primary Key. The Identifying Relationship has been drawn as a solid line:



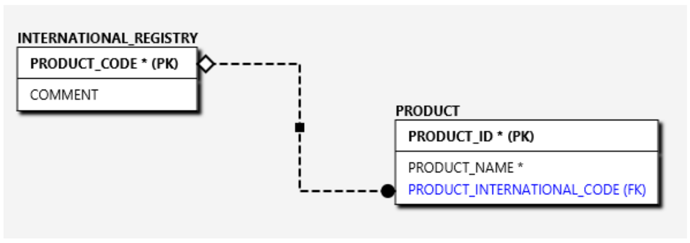
The Non-Identifying Mandatory Relationship

This type of relationship takes place when the Parent’s Primary Key is linked to the mandatory field on the Child- side. The notation for this type of Relationship is shown below:



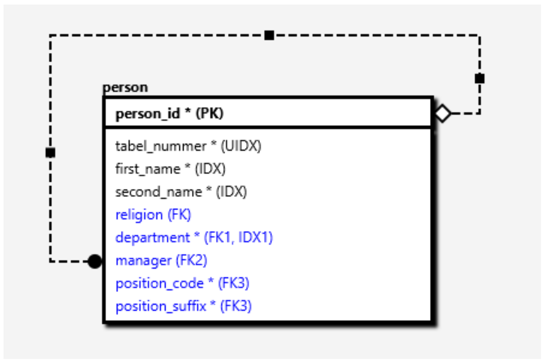
The Not-Identifying Optional Relationship

This type of relationship exists if the Foreign Key field on the Child- side is optional.



The field “PRODUCT_INTERNATIONAL_CODE” in the table “PRODUCT” on the fragment of the diagram above is optional. Therefore, the relationship is Not-Identifying Optional.

The “Auto”- relationship



The Entity can be referring to itself. For example, on the diagram above the entity “person” has the “auto”- relationship between “person” and “manager”. The “auto” relationship is always Non-Identifying. It can be mandatory or optional.

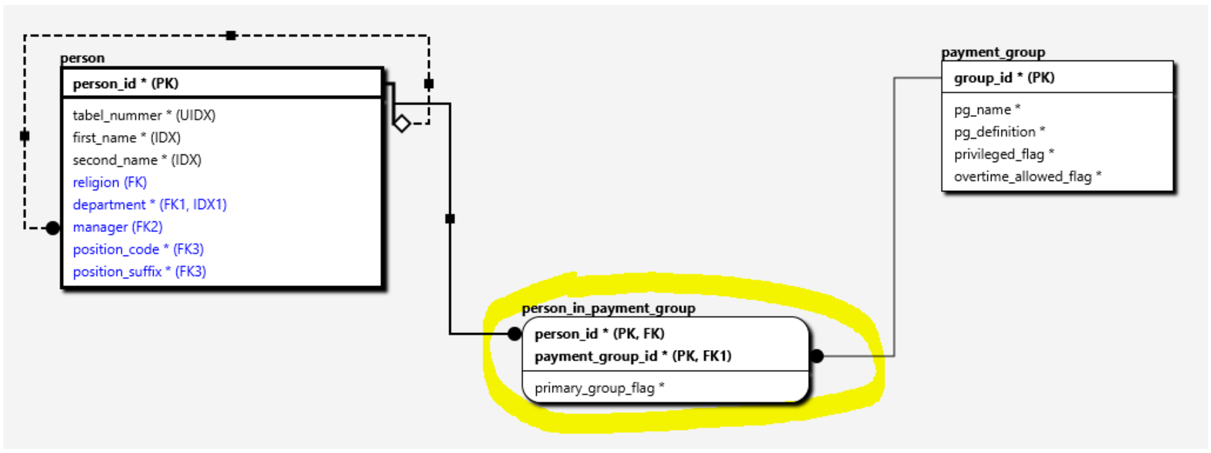
Resolving the “many-to-many” relationship

As it was mentioned earlier, “Apricot DB” does not provide the direct support of the “many-to-many” logical relationship.

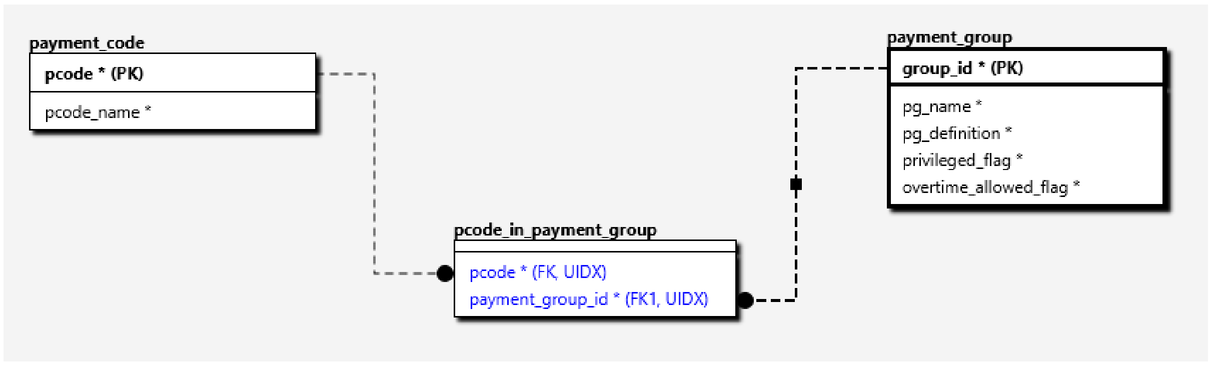
In order to implement the “many-to-many” relationship, an additional entity needs to be introduced, so called “Association Entity”.

The “Association Entity” is shown on the diagram below and marked in yellow.

This “many-to-many” resolution was implemented using the identifying relationships on both sides.

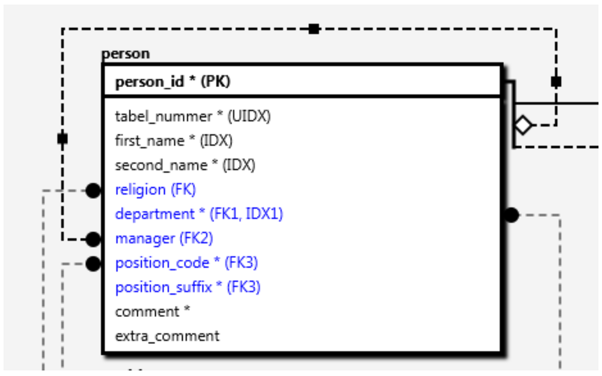


Alternatively, the "many-to-many" relationship can be resolved using the non-identifying relationships, as it is shown below:

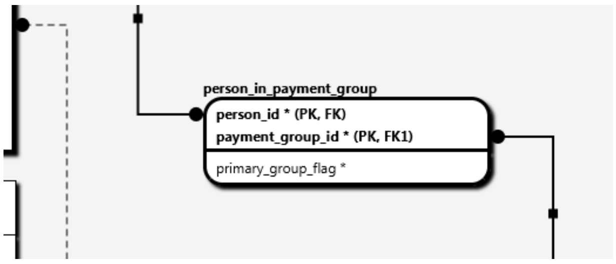


The Entity notation

The entity on the ERD- diagram contains fields. The asterisk symbol means that the table column is mandatory. On opposite, when no asterisk is set, this would be an optional column.



The columns might have the following additional property identifiers:
PK – the Primary Key;
FK – the Foreign Key;
UIDX – the Unique Index;
UC – the Unique Constraint;
IDX – the Non-unique Index.

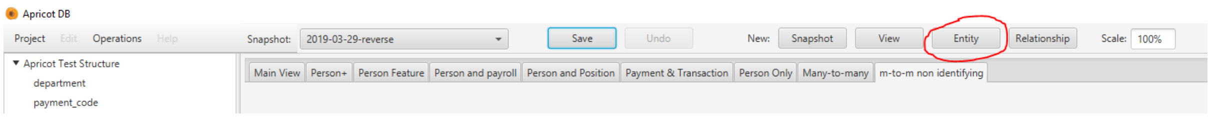


If multiple constraints are applied to the same Entity, they will be marked as 1, 2 and so on. Example: FK1, FK2, FK3. The tip with information about the field type will pop-up, when the mouse is positioned over the field.

The Child- Entities with Identifying relationships has round corners.

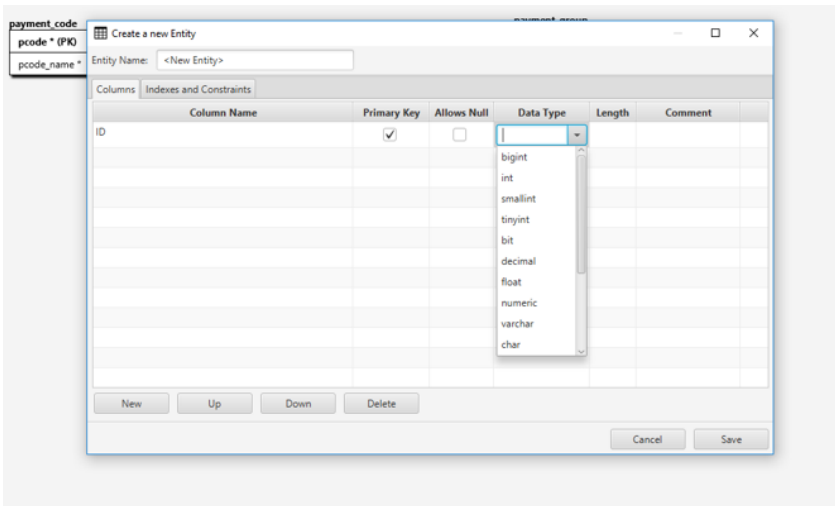
Working with the Entities

The key element of any ERD is an Entity.



In order to create a new Entity, use the item "New Entity" on the toolbar, as shown above.

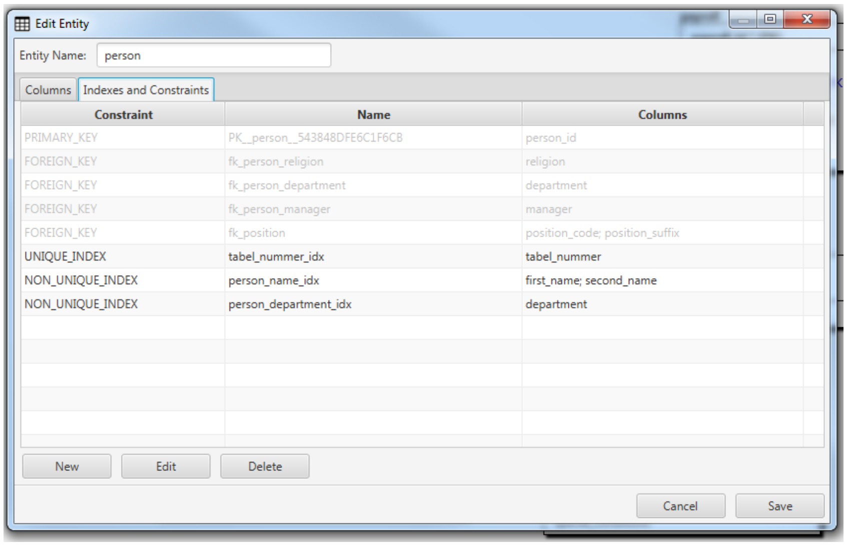
The form of creation/editing of Entity contains two tabs: "Columns" and "Indexes and Constraints".



The names of fields in the table above are self-explanatory. The set of types, presented in the drop-down menu "Data Type" is a database specific. This list includes only most popular types, specific for the target database type. If user needs some type which is not included into the provided list, the new type can be entered manually into the "Data Type" field.
Note: the type of the target database is set for the current Project during the project-create procedure. The Project's database type can be changed at any time.

The sequence of columns in the "Columns" grid will be used for generation of the database create-script. This will be the physical sequence of the columns then. The column can be moved up and down in the sequence.
The "Indexes and Constraints" tab.

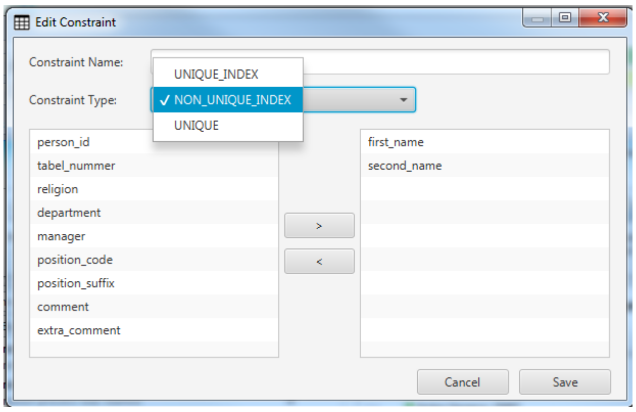
Here user can add/edit the Entity- constraints and indexes.



Entity Name: person

Constraint	Name	Columns
PRIMARY_KEY	PK__person__543848DFE6C1F6CB	person_id
FOREIGN_KEY	fk_person_religion	religion
FOREIGN_KEY	fk_person_department	department
FOREIGN_KEY	fk_person_manager	manager
FOREIGN_KEY	fk_position	position_code; position_suffix
UNIQUE_INDEX	tabel_nummer_idx	tabel_nummer
NON_UNIQUE_INDEX	person_name_idx	first_name; second_name
NON_UNIQUE_INDEX	person_department_idx	department

The create/edit constraint form id shown below. The fields, included onto the constraint can be added/removed.



Constraint Name:

Constraint Type: NON_UNIQUE_INDEX

person_id	first_name
tabel_nummer	second_name
religion	
department	
manager	
position_code	
position_suffix	
comment	
extra_comment	

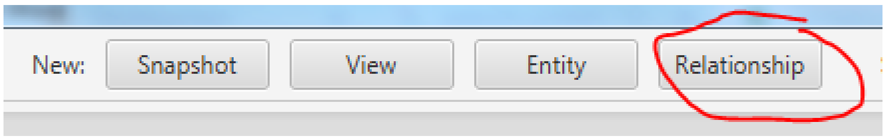
The Primary and Foreign Key constraints are not editable in the “Indexes and Constraints” tab. They have been created automatically, when the Primary Key is described in the “Columns” tab, or the Relationship is established between Entities.

The Entity can be removed at any moment of time. The <DEL> button deleted Entity. If entity being deleted has relationships, these relationships will be removed automatically together with the Entity.

To edit Entity double click of it either in the list of Entities of the left- side, or on the Entity image in ERD.

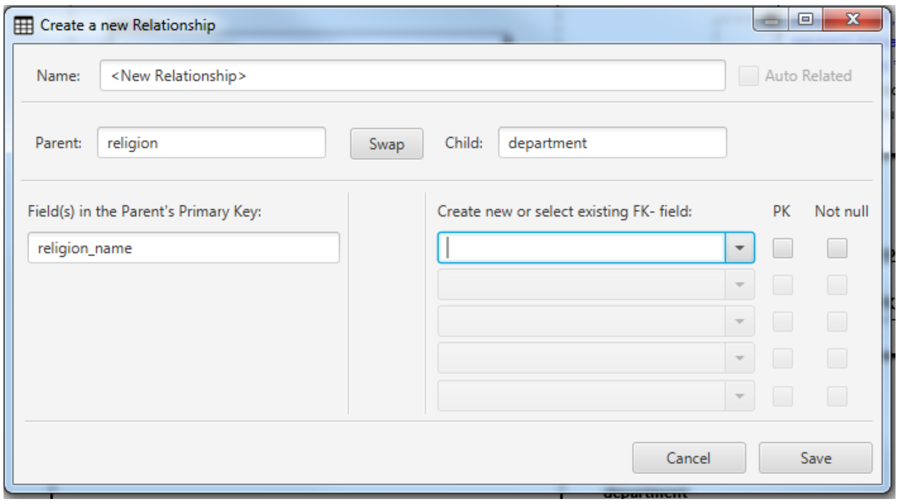
Working with the Relationships

In order to create a new Relationship between Entities, one or two entities have to be selected. Then use “New Relationship” button:



If only one entity was selected, the new relationship will be an “Auto”- Relationship (the Relationship which links the Entity with itself).

The new Relationship form is shown below:


Create a new Relationship

Name: <New Relationship> Auto Related

Parent: religion Swap Child: department

Field(s) in the Parent's Primary Key: religion_name

Create new or select existing FK- field: PK Not null

It is important to choose properly the Parent and Child tables in the pair of Entities. A Parent is always allocated on the left side. A Child is allocated on the right side.

The fields shown on the Parent side are included into the Primary Key and provided by the form automatically. The fields on the Child side have to be explicitly entered.

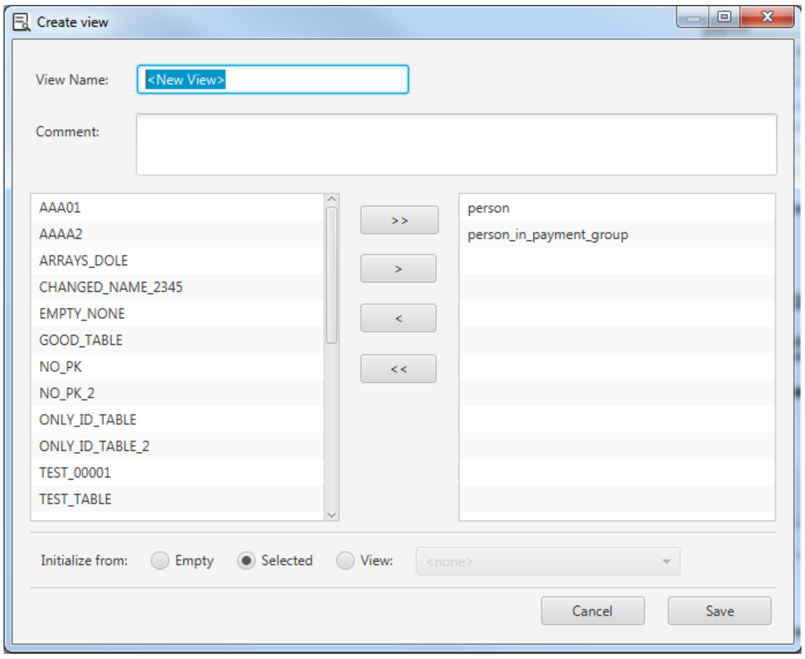
There are two options for the editing of the Child- fields:
* create a new field, linked to the foreign key on the Child- side. In this case, the field name has to be entered into the form. Type of the newly created field is dictated by the corresponding field in the Primary Key of the Parent Entity. If the “PK” option is chosen, the new Relationship will be Identifying. If the new field is set as “Not Null” (and not “PK”) – this will be the Non-Identifying Mandatory Relationship, otherwise - Non-Identifying Optional Relationship. The Primary Key is always not null;
* choose an existing field of the Child- Entity from the drop down list. In this case the chosen field of the Child- Entity will be included into the Foreign Key newly created.

There is no “Edit” functionality for the Relationship. To alter the relationship, it has to be deleted and created from scratch.

Working with the Views

The View is a logical fragment of the main ERD. It includes a subset (or all) tables of the current Snapshot. Usually the tables, included into the View have been logically united.

There are 3 main options when a new View is being created (see the options in “Initialize From” section of the form):
* Empty – in this case, the Entities, included into the View have to be selected manually. The list on the left side of the form contains all Entities, contained in the current Snapshot;
* Selected – the View will contain the selected entities of the current active View;
* View – the new View will be initialized with all entities of the selected view (the reference view needs to be selected from the drop down list.


Create view

View Name: <New View>

Comment:

AAA01	person
AAAA2	person_in_payment_group
ARRAYS_DOLE	
CHANGED_NAME_2345	
EMPTY_NONE	
GOOD_TABLE	
NO_PK	
NO_PK_2	
ONLY_ID_TABLE	
ONLY_ID_TABLE_2	
TEST_00001	
TEST_TABLE	

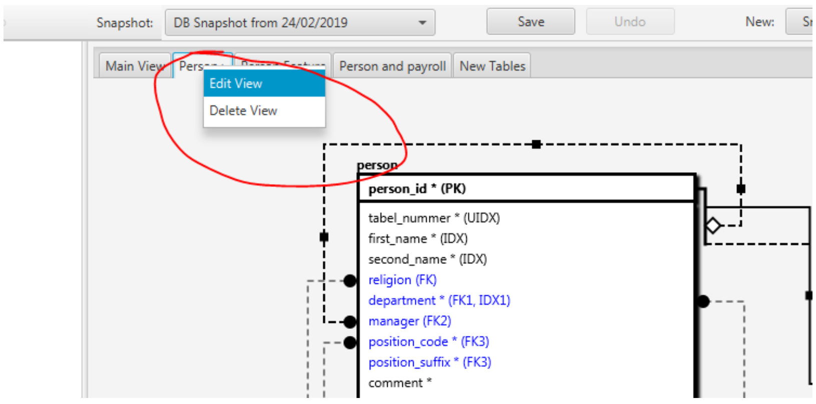
Initialize from: Empty [✓ Selected] View: <none>

As it was already mentioned above, the View belongs to the whole Project – not to the current Snapshot. It means, that any changes in the View reflect in other Snapshots as well. This needs to be taken into account while working with “Apricot DB”.

One View – the “Main View” is not editable and always includes all tables of the current Snapshot.

Any alterations of the Entites/Relationships including the new or removed columns, created or deleted Relationships, made in one View, immediately reflect in all affected Views.

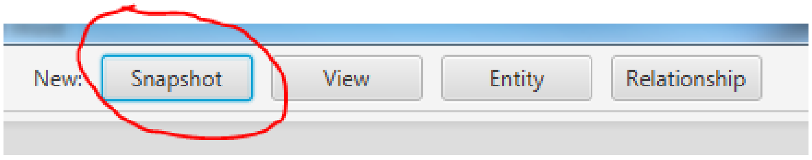
In order to edit or delete the View, use the right button mouse click over the View’s tab. The following context menu is shown below:


Snapshot: DB Snapshot from 24/02/2019

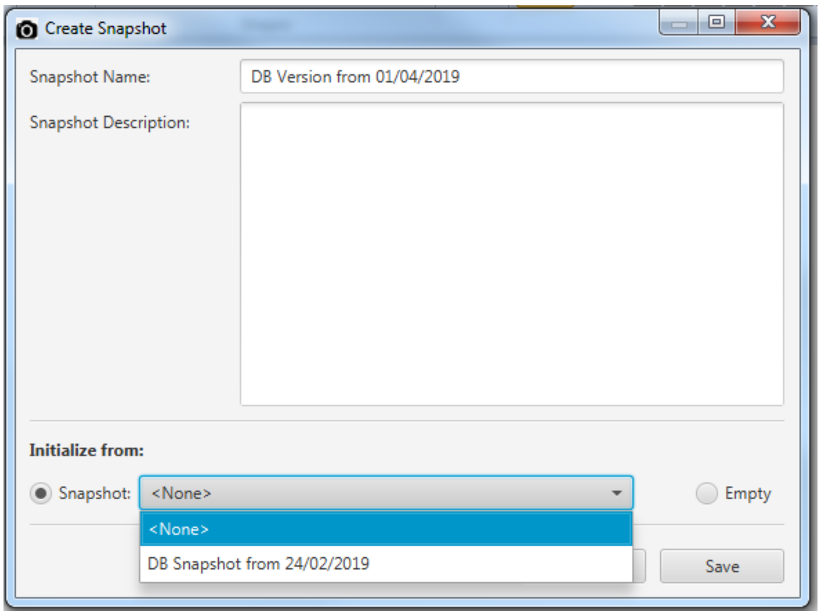
Working with the Snapshots

The Snapshot is a container of the Entities/Relationships. A Project can contain multiple Snapshots. The main idea under the Snapshot is to allow storing the multiple versions of the same logical database. For example, some project needs to support/develop several versions/modifications of the same database. “Apricot DB” allows to generate the “CREATE”- scripts from any Snapshot, as well as compare different snapshots and generate the “difference” DB- scripts, which can be used to align one snapshot (the source) to another (the target Snapshot).

To create a new Snapshot use "New Snapshot" button on the toolbar:



The new Snapshot can be created empty (see "Initialize from" section of the form), or from other Snapshot. The first option is useful when the ERD has to be created from scratch. Another application of the empty Snapshot would be the Reverse Engineering- operation. The Reverse Engineering of the existing DB- structure has to be performed into the empty Snapshot.



Create Snapshot

Snapshot Name: DB Version from 01/04/2019

Snapshot Description:

Initialize from:

[✓ Snapshot:] <None> ◯ Empty

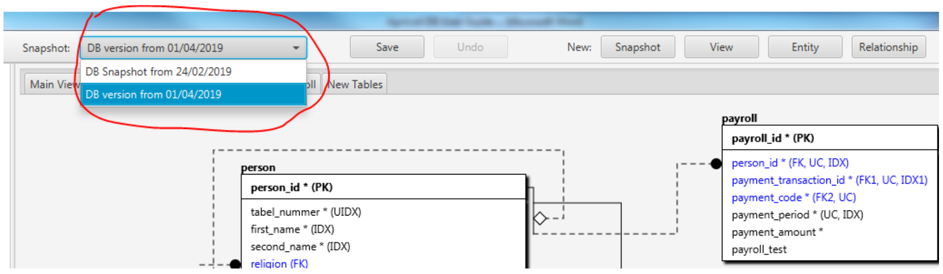
<None>

DB Snapshot from 24/02/2019

Save

Another option of creation of the Snapshot is to create Snapshot from the reference one. This option is useful, when user needs a new version of the database structure and wants to save the original one.

The current Snapshot can be changed at any time, selecting from the drop down list show below:



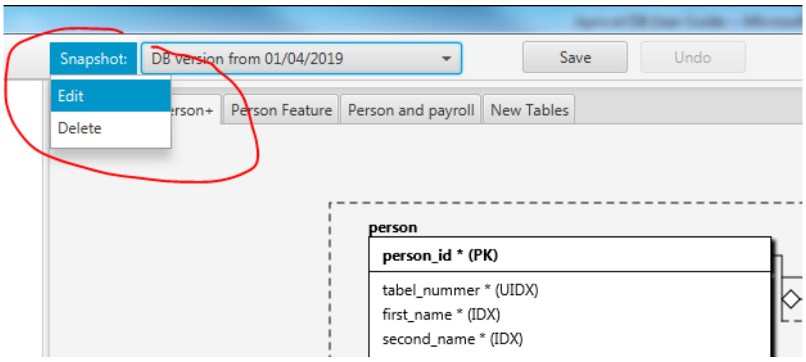
Snapshot: DB version from 01/04/2019

DB Snapshot from 24/02/2019

DB version from 01/04/2019

When the Snapshot is chosen it automatically becomes the default one. Next time, when the application is started, the latest selected Snapshot has been active and ready for work.

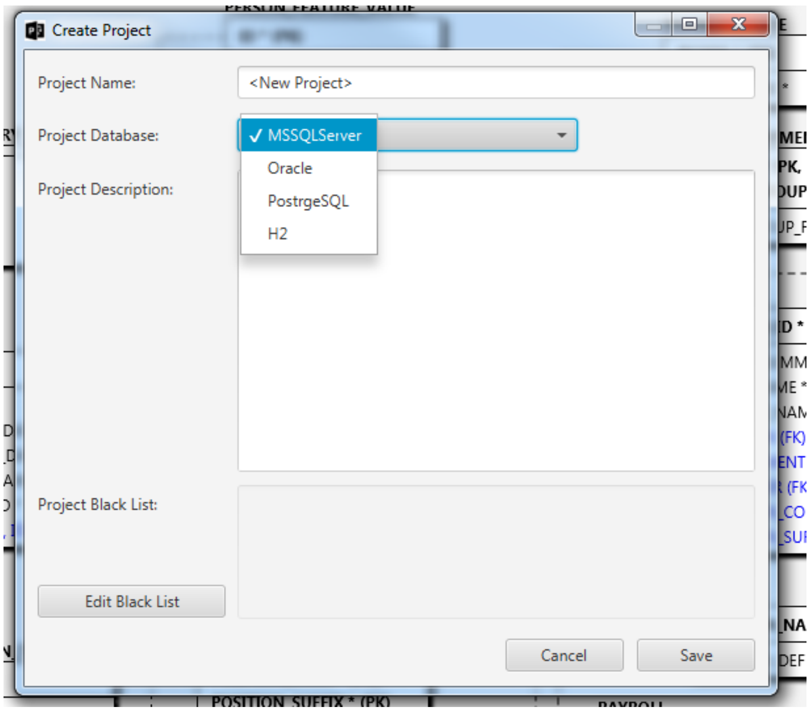
In order to edit the Snapshot name or description or delete Snapshot use the menu "Snapshot:" as shown below:



Working with the Projects

The Project is a top of the "Apricot DB" hierarchy. It is recommended to create a dedicated Project for the logically consistent database structure. For example, the user- schema in Oracle can be considered as such a database structure. Another example is the Database in SQL Server.

A new Project can be created using the menu item: Project/New.


Create Project

Project Name: <New Project>

Project Database: ✓ MSSQLServer

Project Description:

Project Black List:

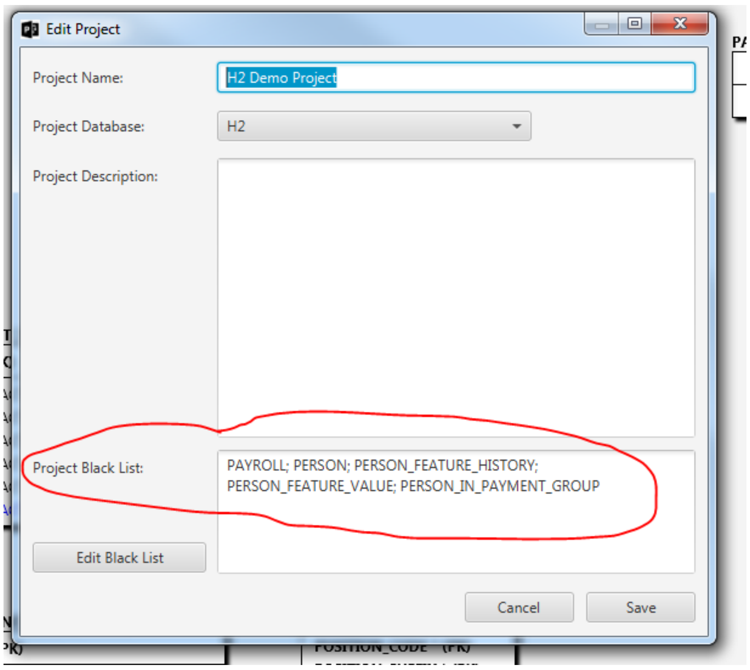
The Project should have a unique name. The Project Database has to be chosen from the drop down list. The Project Database is important for the Reverse Engineering. Different database types have different connection options and algorithms of scanning of the source database structures.

The Project Description is mandatory and can contain the Project- related comments in the free format.

All fields in the Project form can be edited after the Project is created, if needed.

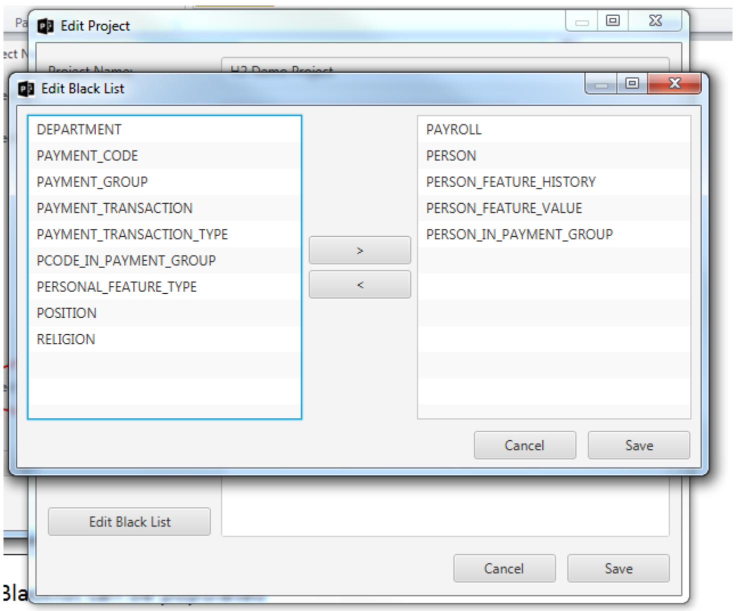
The Blacklist

If database schema, which is being scanned, contains the tables which have to be excluded from the resulting entity set, the Blacklist can be used.


Edit Project

Project Name:	H2 Demo Project
Project Database:	H2
Project Description:	
Project Black List:	PAYROLL; PERSON; PERSON_FEATURE_HISTORY; PERSON_FEATURE_VALUE; PERSON_IN_PAYMENT_GROUP

The Blacklist can be populated in two ways: explicitly via the form of editing of the Project (use the "Edit Black List" button). The entities, included into the Blacklist have to be "moved" to the right-side as shown on the screen below:


Edit Black List

DEPARTMENT	PAYROLL
PAYMENT_CODE	PERSON
PAYMENT_GROUP	PERSON_FEATURE_HISTORY
PAYMENT_TRANSACTION	PERSON_FEATURE_VALUE
PAYMENT_TRANSACTION_TYPE	PERSON_IN_PAYMENT_GROUP
PCODE_IN_PAYMENT_GROUP	
PERSONAL_FEATURE_TYPE	
POSITION	
RELIGION	

The Blacklist can be edited in this form.

The second option for the population of the Blacklist is the following: if, during the Reverse Engineering process, some entities were excluded from the resulting list, they would be automatically included into the Blacklist. As it was mentioned above, the Blacklist can be edited.

If some entities were included into the Blacklist, in all the subsequent Reverse Engineering procedures these entities will be unchecked as shown on the screenshot below:

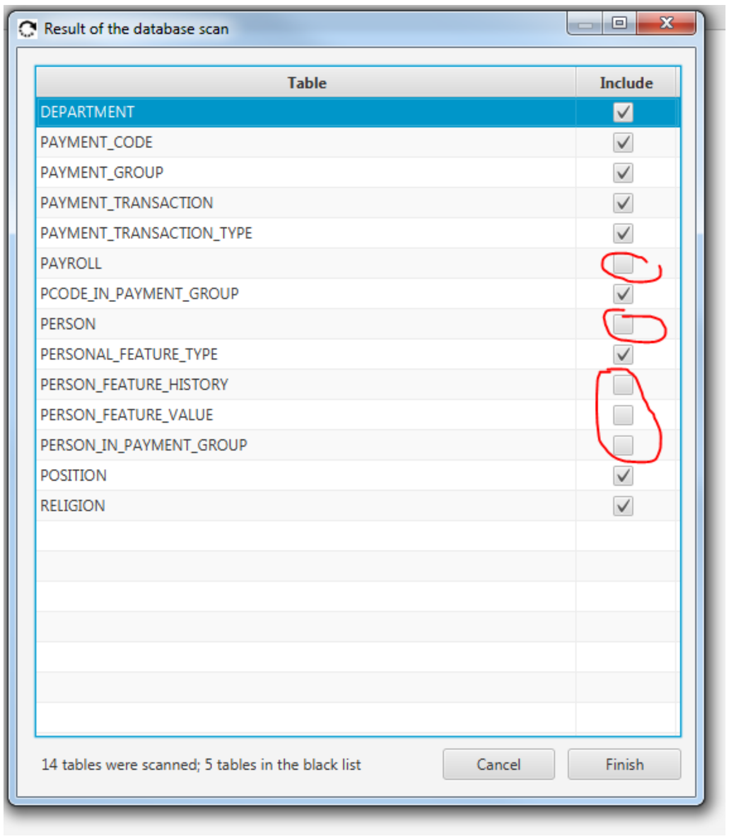


Table	Include
DEPARTMENT	✓
PAYMENT_CODE	✓
PAYMENT_GROUP	✓
PAYMENT_TRANSACTION	✓
PAYMENT_TRANSACTION_TYPE	✓
PAYROLL	☐
PCODE_IN_PAYMENT_GROUP	✓
PERSON	☐
PERSONAL_FEATURE_TYPE	✓
PERSON_FEATURE_HISTORY	☐
PERSON_FEATURE_VALUE	☐
PERSON_IN_PAYMENT_GROUP	☐
POSITION	✓
RELIGION	✓

14 tables were scanned; 5 tables in the black list

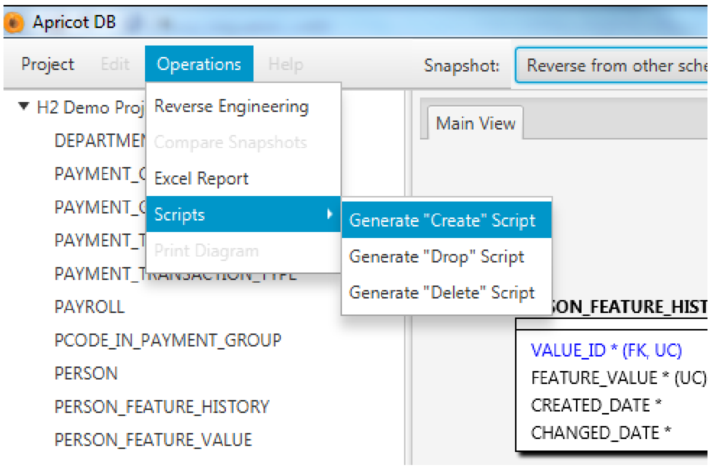
This means that “Apricot DB” scans the whole list of the tables in the source database, but uses the Blacklist and “suggests” excluding the previously excluded tables from the result set. User can check the unchecked entities in the form above, and they will be included into the Reverse Engineering result.

Generation of the Database Scripts

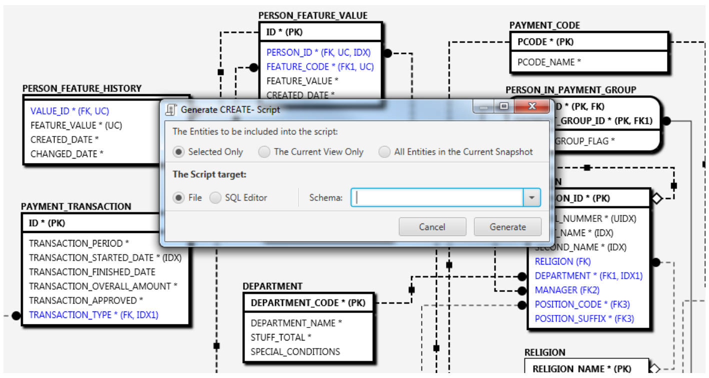
There are 3 types of database scripts which can be generated by “Apricot DB”:

1) Create Script – a script which purpose is to create all objects, presented on the current ER-diagram;
2) Drop Script – script for consistent dropping of the chosen database objects;
3) Delete Script – script which allows deleting data from the chosen tables (entities).

The Generate Scripts functionality is available through the main menu:



The script generation form is shown below (this form is common for Create, Drop and Delete types of scripts):



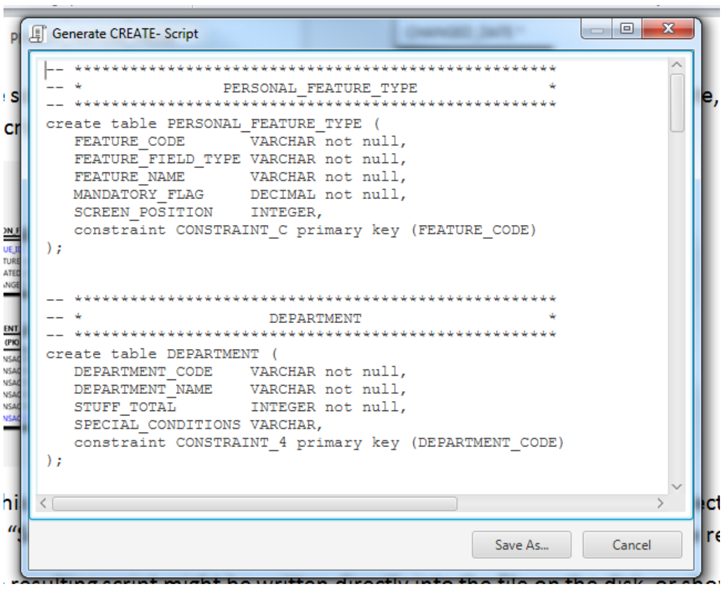
The Entities to be included into the script:
[✓ Selected Only] The Current View Only All Entities in the Current Snapshot

The Script target:
[✓ File] SQL Editor Schema:

In this form a set of parameters can be changed. If some Entities were selected in the current view the “Selected Only” option for the entities, which will be included into the resulting script.

“Schema” is an optional parameter. If schema is filled in, it will be included as a prefix for the table names.

The resulting script might be written directly into the file on the disk, or shown in the simple SQL Editor:



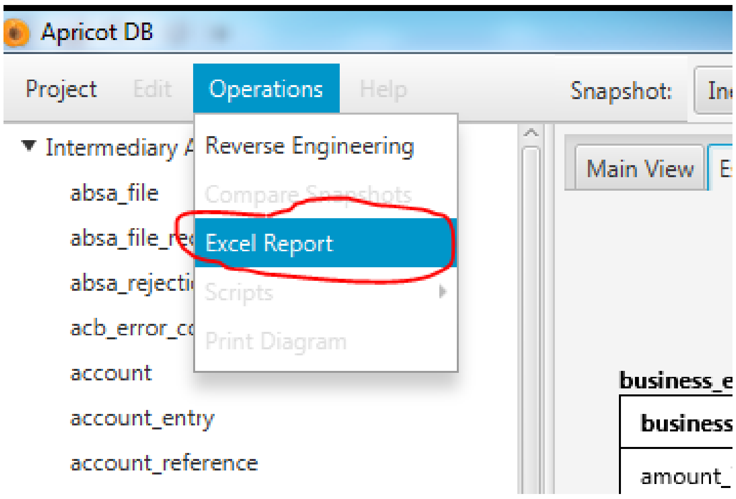
```
-- ****************************************************
-- *               PERSONAL_FEATURE_TYPE              *
-- ****************************************************
create table PERSONAL_FEATURE_TYPE (
    FEATURE_CODE        VARCHAR not null,
    FEATURE_FIELD_TYPE VARCHAR not null,
    FEATURE_NAME        VARCHAR not null,
    MANDATORY_FLAG      DECIMAL not null,
    SCREEN_POSITION     INTEGER,
    constraint CONSTRAINT_C primary key (FEATURE_CODE)
);


-- ****************************************************
-- *                    DEPARTMENT                    *
-- ****************************************************
create table DEPARTMENT (
    DEPARTMENT_CODE     VARCHAR not null,
    DEPARTMENT_NAME     VARCHAR not null,
    STUFF_TOTAL         INTEGER not null,
    SPECIAL_CONDITIONS VARCHAR,
    constraint CONSTRAINT_4 primary key (DEPARTMENT_CODE)
);
```

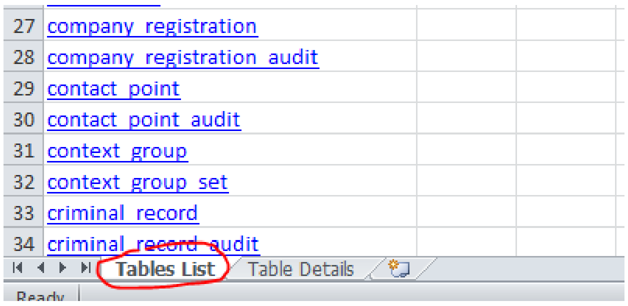
To run the resulting script on the target Database, use your favourite data access and administration tool (DB Visualizer?).

Retrieving the structure of Database in form of Excel table

There is a special feature, provided by “Apricot DB”. The Entities/Relationships in the current Snapshot can be requested in the form of an Excel sheet. Use “Operations/Excel Report” to generate this report:

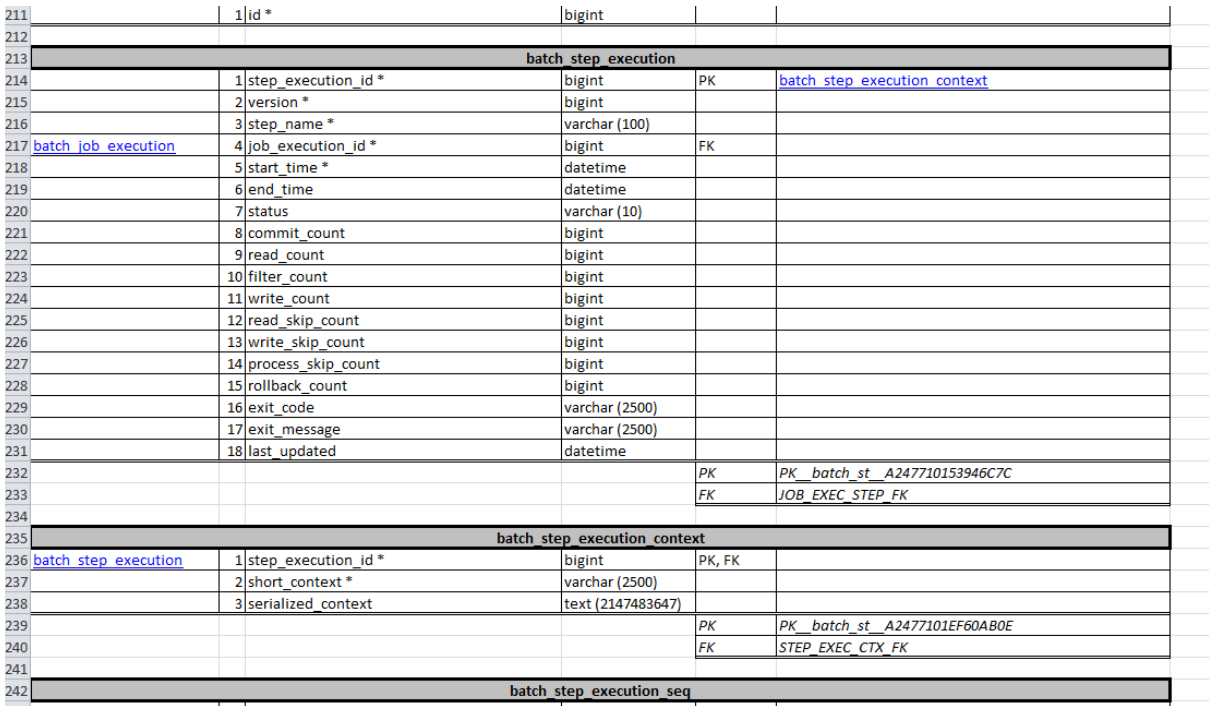


The resulting Report contains two tabs. The “Tables List” represents the whole list of Entities in the current Snapshot sorted alphabetically.



27	company_registration
28	company_registration_audit
29	contact_point
30	contact_point_audit
31	context_group
32	context_group_set
33	criminal_record
34	criminal_record_audit

These table names have been generated as the hyperlinks which lead to the second tab of the Excel sheet: “Table Details”:



211		1	id *	bigint		
212						
213	**batch_step_execution**					
214		1	step_execution_id *	bigint	PK	batch_step_execution_context
215		2	version *	bigint		
216		3	step_name *	varchar (100)		
217	batch_job_execution	4	job_execution_id *	bigint	FK	
218		5	start_time *	datetime		
219		6	end_time	datetime		
220		7	status	varchar (10)		
221		8	commit_count	bigint		
222		9	read_count	bigint		
223		10	filter_count	bigint		
224		11	write_count	bigint		
225		12	read_skip_count	bigint		
226		13	write_skip_count	bigint		
227		14	process_skip_count	bigint		
228		15	rollback_count	bigint		
229		16	exit_code	varchar (2500)		
230		17	exit_message	varchar (2500)		
231		18	last_updated	datetime		
232					*PK*	*PK__batch_st__A247710153946C7C*
233					*FK*	*JOB_EXEC_STEP_FK*
234						
235	**batch_step_execution_context**					
236	batch_step_execution	1	step_execution_id *	bigint	PK, FK	
237		2	short_context *	varchar (2500)		
238		3	serialized_context	text (2147483647)		
239					*PK*	*PK__batch_st__A2477101EF60AB0E*
240					*FK*	*STEP_EXEC_CTX_FK*
241						
242	**batch_step_execution_seq**					

The “Table Details” tab contains extended information for each table in the list. It includes names, types, nullable feature and so on (see the screenshot).

Tables which participate in the Relationships there will be hyperlinks leading to the related tables. The hyperlink on the left side stands for the Parent of this table. Hyperlinks on the right side point to the tables, for which the focused table acts as a Parent.