

# **Apricot DB**

## **The Entity Relationship diagramming tool**

### **The User Guide**

v. 2.3 (July 2020)



## Contents

Preface .....	4
The user interface .....	5
The Projects, Snapshots and Views concept.....	6
The Snapshot .....	7
The View.....	7
Physical or Logical .....	7
The Entities and Tables .....	7
How “Apricot DB” stores the Project Data .....	8
Working with the Projects .....	8
The “ApricotDB” Scenarios .....	9
The Reverse Engineering Process .....	9
The ERD- graphical notations in “Apricot DB” .....	10
Entities and Relationships between them .....	11
The Identifying Relationship .....	11
The Non-Identifying Mandatory Relationship .....	12
The Not-Identifying Optional Relationship .....	12
The “Auto”- relationship.....	12
Resolving the “many-to-many” relationship .....	12
The Entity notation .....	13
Working with the Entities .....	14
The Default-, Simplified- and Extended- Entity information .....	16
Working with the Relationships.....	17
Working with the Views.....	19
Working with the Snapshots.....	20
Working with the Projects .....	21
Editing the diagram content .....	23
Selecting and moving the elements of the diagram .....	23
How “Apricot” draws the relationships .....	23
The context menus .....	24
The automatic aligning of the diagram.....	25
The Save and Undo operations .....	26
Selecting Entities on the Diagram.....	26
The Blacklist .....	26
Generation of the Database Scripts.....	28
Retrieving the structure of Database in form of Excel table .....	30



## Preface

“Apricot DB” is a simple but powerful tool for working with the database structures. Based on Entity Relationship paradigm, “Apricot DB” allows to do efficient design, analysis, scripting and documenting of the relational database structures.

“Apricot DB” has been developed for the analysts, database designers, developers, testers and any other roles of IT projects, who have to work with the existing entity/relationship structures as well as to produce the new ones. The modern database can include hundreds and thousands of tables and relationships between them. Very often the structure of the database needs to be analysed, documented and altered.

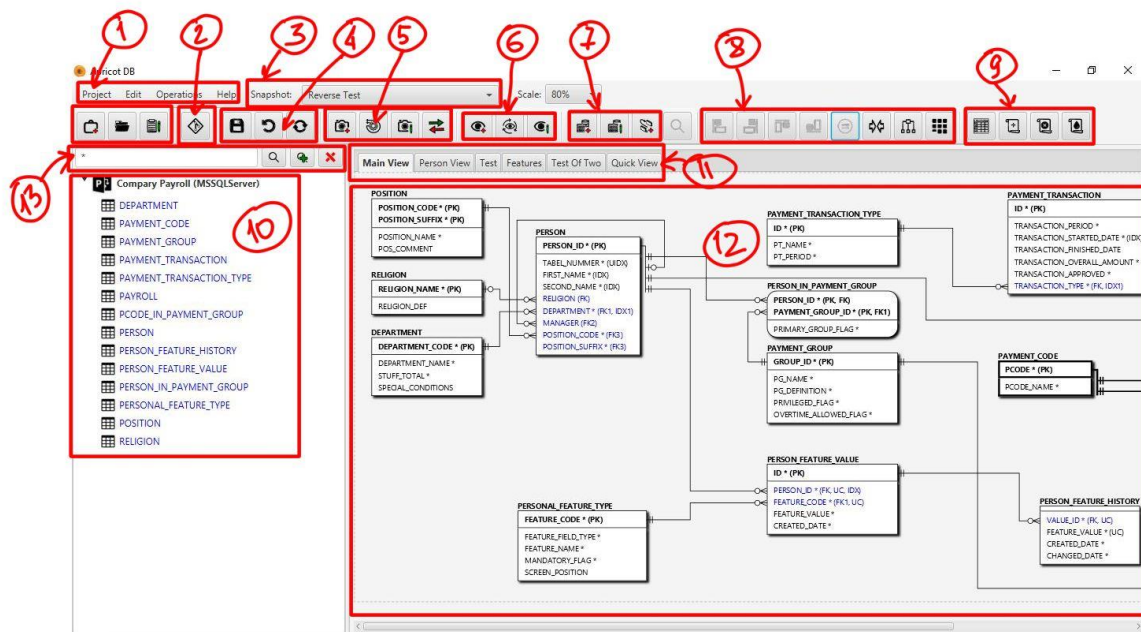
The database structure is a set of tables, connected with each other by the relationships (that makes the database “relational”). The “tables” and “entities” terms are synonyms in this document. The relatively simple idea of the Entity Relationship Diagram lies at the background of the graphical representation of the database structure.

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

- \* Creation and editing the database structure, using the graphical editor;
- \* Storing all database design information produced by the user in a well-structured projects (the ApricotDB Project);
- \* Generate SQL/DDL database scripts for different purposes: create/drop the tables, delete data, compare and modify different versions of the same database;
- \* The Reverse Engineering of the existing database into the ApricotDB Project;
- \* Import/Export the projects from/to the file;
- \* The team work: synchronisation of the Apricot Projects using the remote Git repository;
- \* Generate the report which represents the project information in form of Excel Sheet.

## The user interface

The main screen of “Apricot DB” contains several areas, which allow to access and manipulate efficiently all the controls and graphic content of the tool.



On the Main Screen:

- 1 -> **The Main Menu:** all major operations of the application are accessible via the main menu;
- 2 -> **The Remote Repository:** “Apricot DB” allows to store Projects in a standard Git Repository. The team members can share their work via the Remote Repository;
- 3 -> **The Snapshots drop down** allow to select the concrete Snapshot in the current Project;
- 4 -> **The Save, Undo and Refresh functions** have been used when drawing/editing the diagram content;
- 5 -> **The Snapshot functions:** create a new Snapshot, Reverse Engineer into the current empty Snapshot, Edit the Snapshot information, Compare Snapshots;
- 6 -> **The View functions:** Create a new View, Create a Quick View, Edit the current View Information;
- 7 -> **The Element on the current Diagram:** Create a new Entity; Edit the selected Entity; Create a new Relationship between the selected Entities;
- 8 -> **The Alignment functions:** the selected entities can be all aligned to the left, right, top, bottom. Their sizes can be minimized or set the same. The Entities on the diagram can be automatically aligned for better look;
- 9 -> **The generation** of the Excel report and the DB scripts for Create, Drop of the selected diagram objects and delete data in the selected entities;
- 10 -> **The Project Explorer:** all Entities included into the Snapshot have been presented as an alphabetically sorted list in the Project Explorer;

11 -> **The View Tabs:** All Apricot Views belong to the current Project have been available via the tabs at the top of the diagram;

12 -> **The Diagram Drawing Area:** all the Diagram objects have been shown graphically in this area and might be moved, aligned and edited;

13 -> **The Quick Search** allows to search entities on the Diagram.

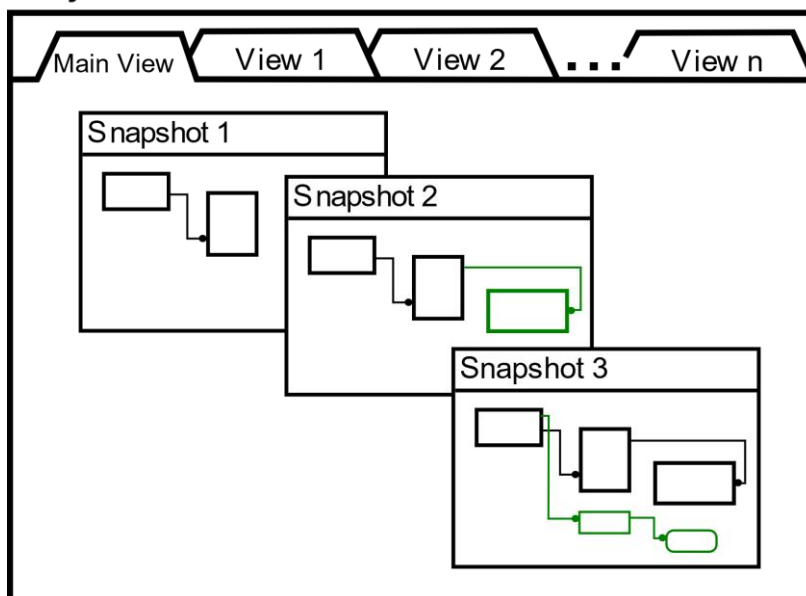
## The Projects, Snapshots and Views concept

A Project is a top of the hierarchy of Apricot objects. Only one Project might be open in the same time. The best approach is to have an individual Project for each database you working with.

A Project includes the Snapshots and Views. The Snapshot is a container for the Entities and Relationships between them. There might be multiple Snapshots inside one project. This allows keeping different versions of the same database structure. It might be, for example, different environments where this database has been deployed – DEV, TST, PROD and so on.

The Views are responsible for the graphical representation of the database structure. The View can include a subset or the whole set of tables in the current Snapshot. This approach allows splitting one massive database structure into several smaller views, which include only the logically close related Entities each.

### Project



Views are associated with the Project, which means that there is not necessity to “re-draw” the ER-diagrams for different Snapshots. You create a view once and then it has been presented in every new Snapshot. 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 has 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, which use this CSS, will change, even though the internal data represented by the pages stay the same.

## The Snapshot

The Snapshot is a main container of the Entities and Relationships between them. The 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 simple version a control system, which stores the database objects for the different moments of time. Technically, the Snapshots can have completely different and not overlapping sets of Entities in them. It makes sense though, to stick with logically the same database structure in all Snapshots of the Project (remember – one Project equals to one database/schema). It allows performing efficiently the comparison between different Snapshots in one Project as well as using an advantage of the 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 be created by CREATE VIEW... DDL command!

There is always one mandatory View included in any Project - “Main View”. It reflects all tables of the chosen Snapshot. The Main View cannot be removed and always contains all tables of the current Snapshot.

The content of the views other that the Main View can be edited. This means that the Entities can be added or removed from the view at any time.

## Physical or Logical

It is a common practice in ERD editors to distinguish between the logical and physical models. The logical model represents the higher level of abstraction comparing to the physical one. The physical representation usually contains the great details about the physical parameters of the database being designed, while the logical ERD omits the internal details specific only for this particular database.

In sake of simplicity “Apricot DB” does not distinguish between physical and logical schemas. We do not support the pure logical ERD- elements such as many-to-many relationships, the explicit cardinality definitions on the different sides of the Relationship or the logical names of Entities in addition to their physical names (the physical name of the Entity is the name of the corresponding table in database).

All the Relationships in “Apricot DB” are of “one-to-many” type.

## The Entities and Tables

The Entities are elements of the ER- diagram. Each Entity corresponds to the table in the target database. In this document I use terms Entity and Table as synonyms.

The Attributes within Entity have been corresponding to the fields of the database table. I use terms “attribute” and “field” as synonyms.

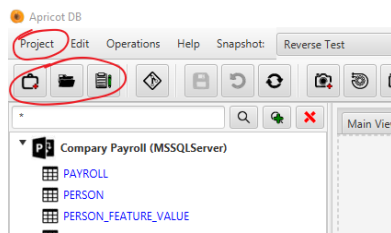
## How “Apricot DB” stores the Project Data

The Projects, Snapshots, Entities, Relationships, Constraints and Attributes have been stored by “Apricot DB” in the special database - the Project Database. The Project Database is a binary file, stored on the local disk at the catalogue ~<user home>\.apricotdb, the file - apricot-project.mv.db

On every start of “Apricot DB” it restores the most recent condition, which took place at the moment, when the application was stopped. The condition includes the current Project and last selected View.

## Working with the Projects

The Projects can be created, edited and deleted in Apricot DB.



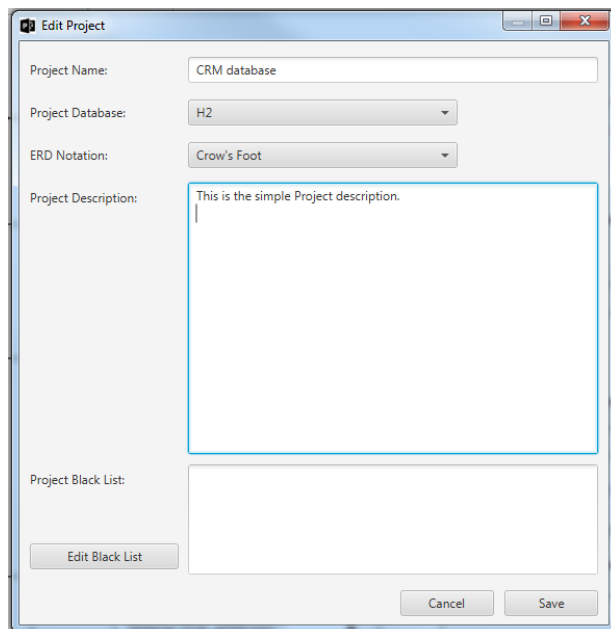
The menu item “Project” or the buttons of the main button bars allow access the functions.

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

The Project has the target database type associated with it (see the drop-down list “Project Database”). The database type might be changed at any time.

The ERD Notation might be selected during the new Project creation or later on.

The Project Description is optional.



By default with every newly created Project, “Apricot DB” creates an empty Snapshot – “Initial snapshot created <on date of creation>”. The default name of the initial Snapshot can be changed later.



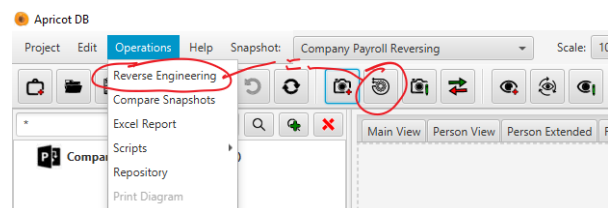
## The “ApricotDB” Scenarios

There are two main scenarios of working with “Apricot DB”:

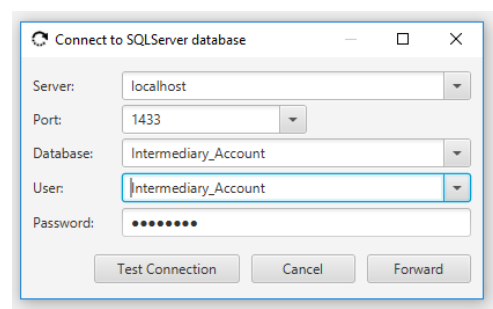
- \* The Entity/Relationships in the current Project to be created from scratch;
- \* The database structure to be retrieved from the existing database. This process called Reverse Engineering.

## The Reverse Engineering Process

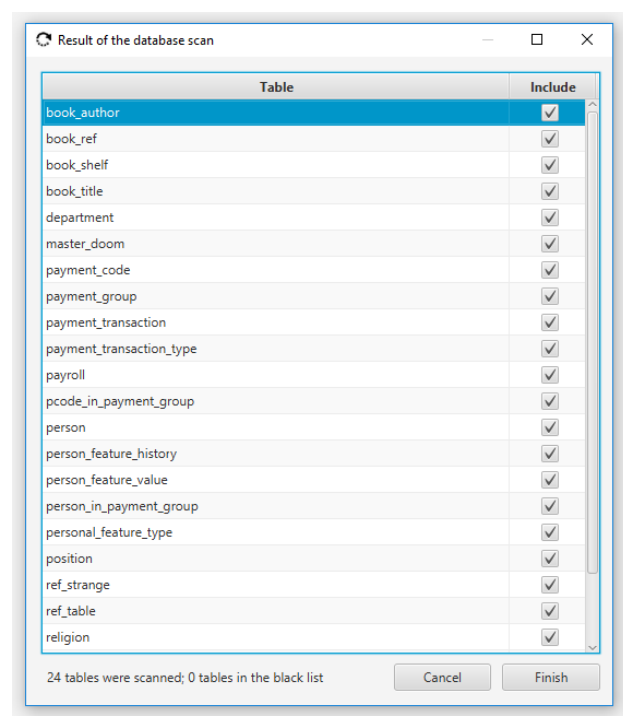
Use menu Operations/Reverse Engineering to start the Reverse Engineering process:



The form of connection to the Database depends on the database type (see the Project parameters above). Below the example of the SQL Server connection form is shown:



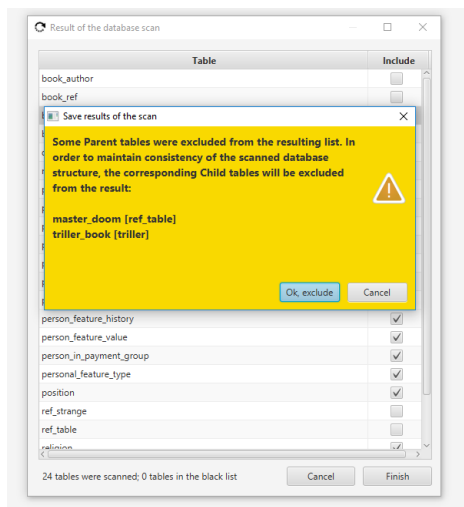
If the connection to the target database was successful, Apricot DB scans the database structure and offers a list of the tables found:



Only selected tables will be added into the current Snapshot. Any tables in the list can be excluded from the resulting set on this stage (uncheck the flag “included”).

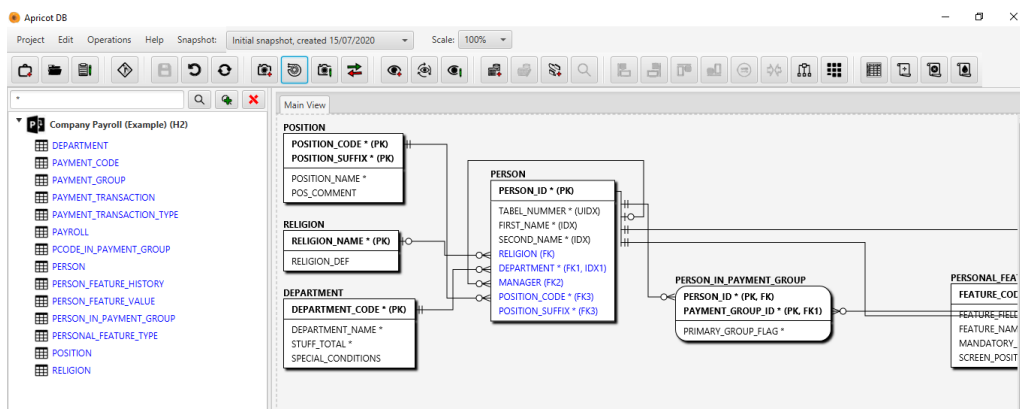
Note: Unchecking the tables, excludes them from the result of the Reverse Engineering process. This operation has potential risk of violation of the referential integrity of the resulting ER Diagram. In order to prevent that, “Apricot DB” automatically analyses the referential integrity existing in the current database schema. The warning is issued if any potential integrity violations were 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 the potential violation of the referential integrity was found, “Apricot DB” issues a warning in the flowing form:



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.

The Main View after the Reverse Engineering process:



## The ERD- graphical notations in “Apricot DB”

“Apricot DB” supports two ERD notations: The “Crow’s Foot” and the standardized notation IDEF1x. The “Crow’s Foot” notation is very popular nowadays. It is a primary notation in Apricot DB.

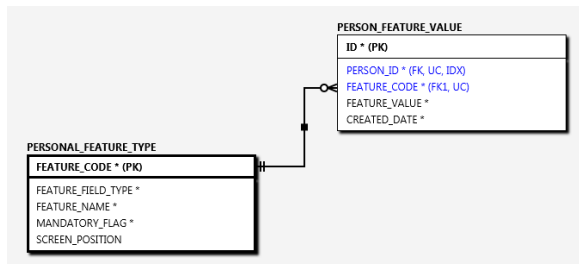
The IDEF1x standard ERD notation is very intuitive, but less popular than the “Crow’s Foot”. In order to switch between notations, use the Edit Project form (menu Project/Edit).

## Entities and Relationships between them

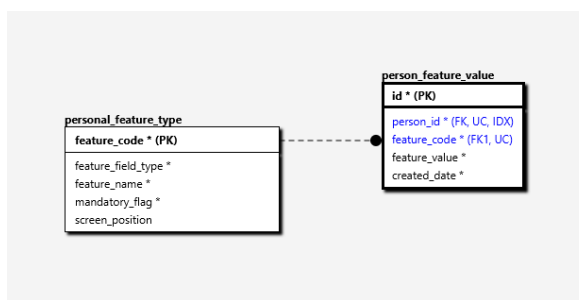
The Entities on the ER- diagram are connected with Relationship (there might be the stand-alone entities, which have been by themselves and do not have any relationships with others).

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

The “Crow’s Foot” notation:



The IDEX1x notation:



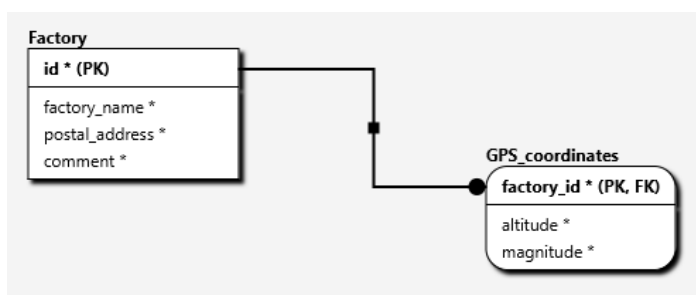
They say that the Primary Key of the Parent is exported to the Child. The corresponding field of the Child Entity called the Foreign Key.

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.

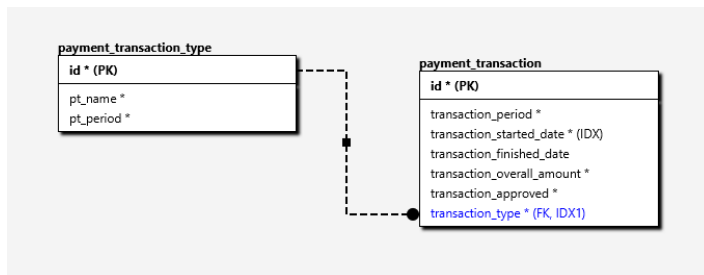
## The Identifying Relationship

...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:



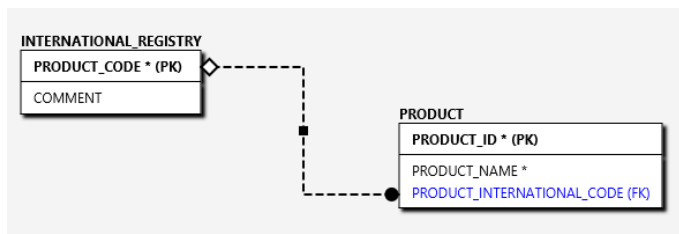
## The Non-Identifying Mandatory Relationship

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:



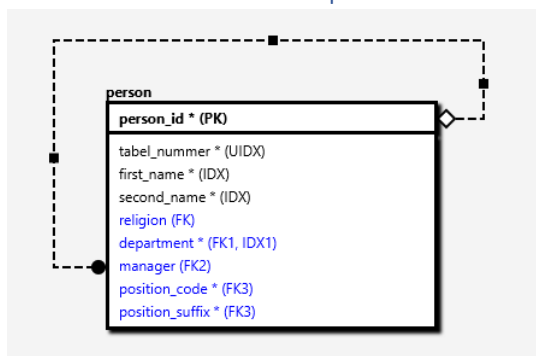
## The Not-Identifying Optional Relationship

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.

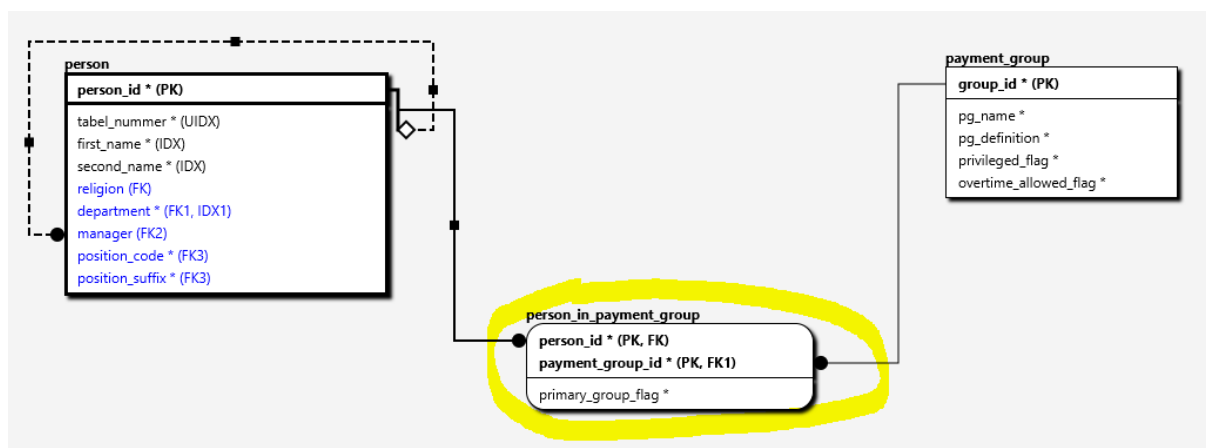
## Resolving the "many-to-many" relationship

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

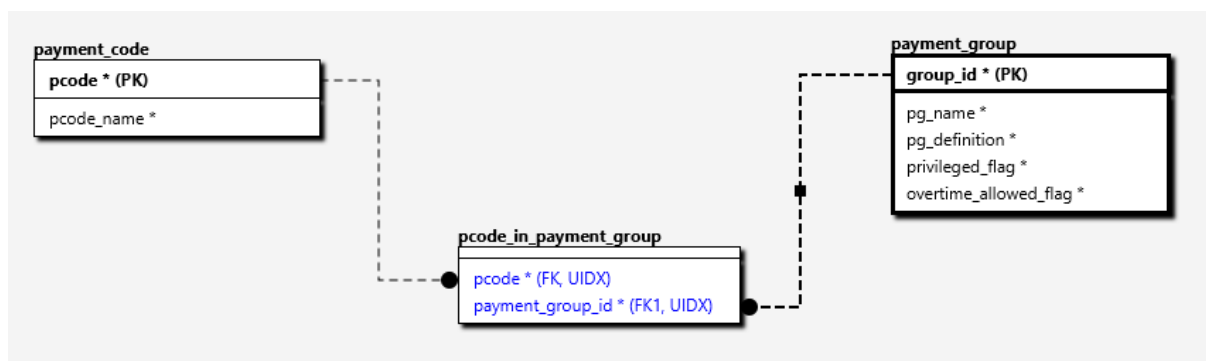
In order to implement the "many-to-many" relationship, an additional entity needs to be introduced, so called "Association Entity".

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.

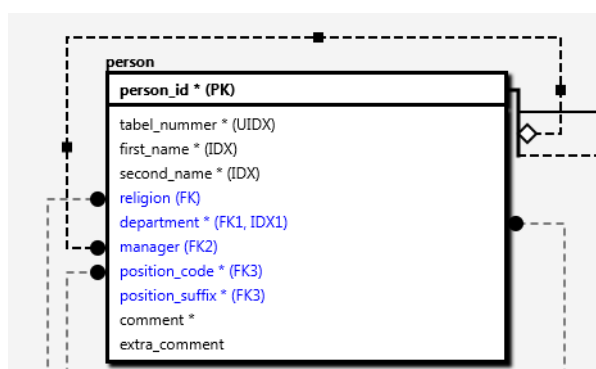


Alternatively, the “many-to-many” relationship can be resolved using the non-identifying relationships, as it is shown below:



## The Entity notation

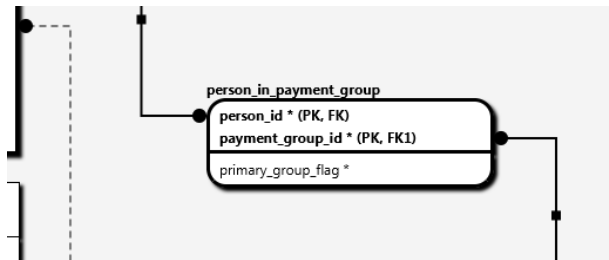
The entity on the ERD- diagram contains fields. The asterisk symbol means that the table column is mandatory. On opposite, when no asterisk is set, this would be an optional column.



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.

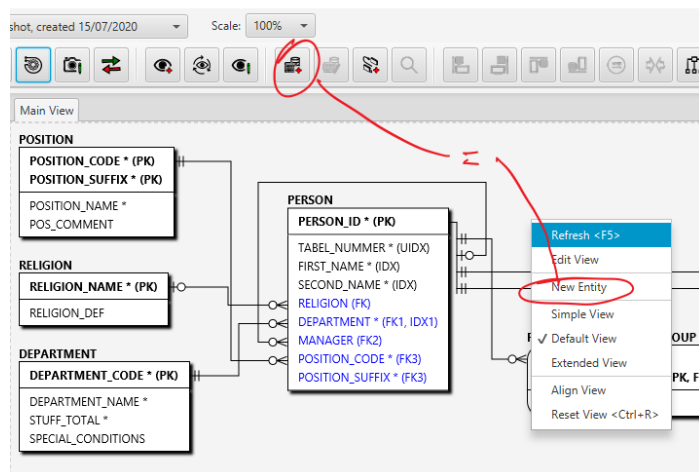


If multiple constraints are applied to the same Entity, they will be marked as 1, 2 and so on. Example: FK1, FK2, FK3. The tip with information about the field type will pop-up, when the mouse is positioned over the field.

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 button on the toolbar or right click anywhere in the diagram:



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

The screenshot shows the 'Create a new Entity' dialog box. The 'Columns' tab is active, displaying a table for defining the columns of the new entity. The table has columns for Column Name, Primary Key, Allows Null, Data Type, Length, and Comment. A dropdown menu is open for the 'Data Type' column, showing options: bigint, int, smallint, tinyint, bit, decimal, float, numeric, varchar, and char.

Column Name	Primary Key	Allows Null	Data Type	Length	Comment
ID	<input checked="" type="checkbox"/>	<input type="checkbox"/>	bigint		
			int		
			smallint		
			tinyint		
			bit		
			decimal		
			float		
			numeric		
			varchar		
			char		

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.

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 is shown below. The fields, included onto the constraint can be added/removed.

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.

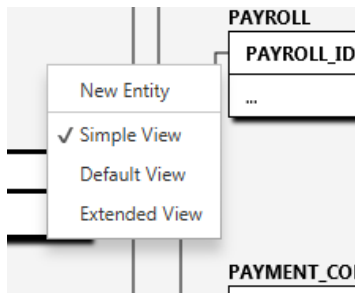
The Entity can be removed at any moment of time. The <DEL> button deleted Entity. If entity being deleted has relationships, these relationships will be removed automatically together with the Entity.

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

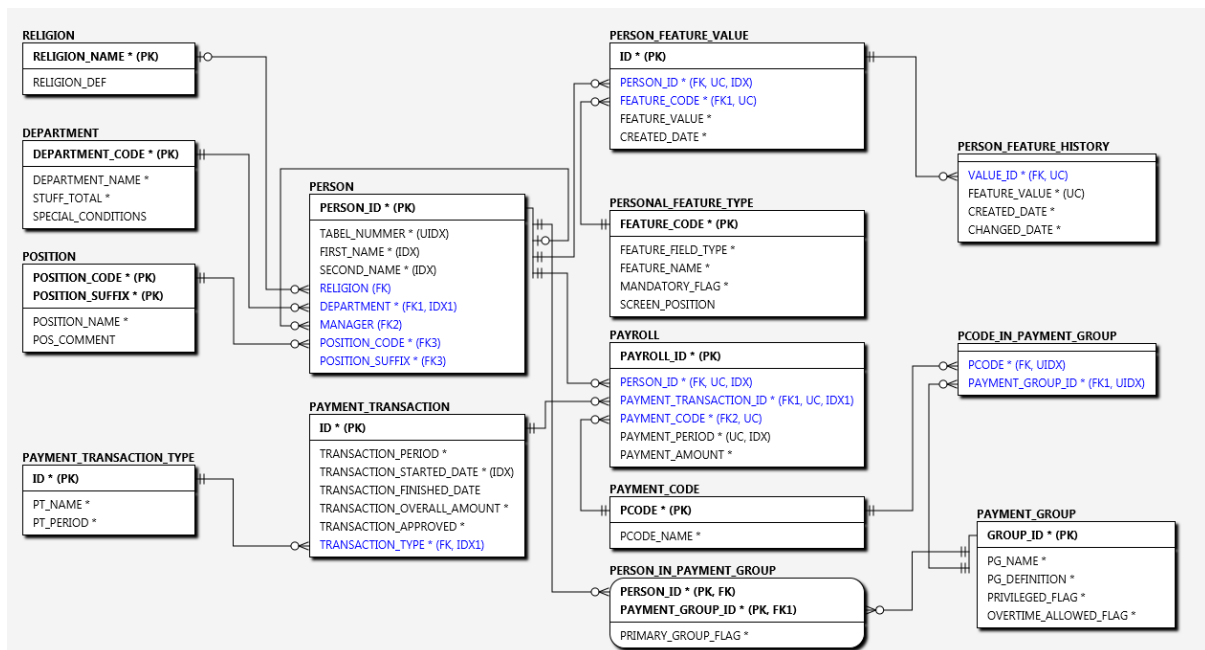
## The Default-, Simplified- and Extended- Entity information

The diagram might be operatively reconfigured depending on the level of required details.

In order to switch between different level of Entity information use the content menu anywhere in the drawing field of the diagram (the right mouse button click):

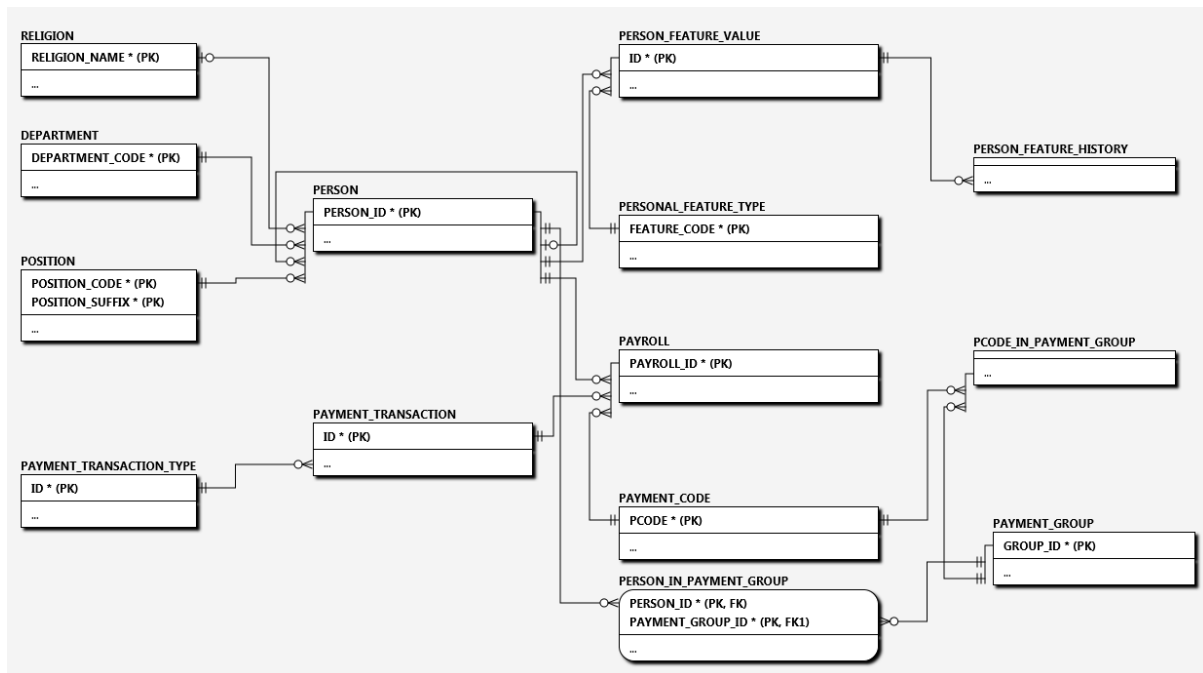


The “default” level of details presents the fields of entities without types:

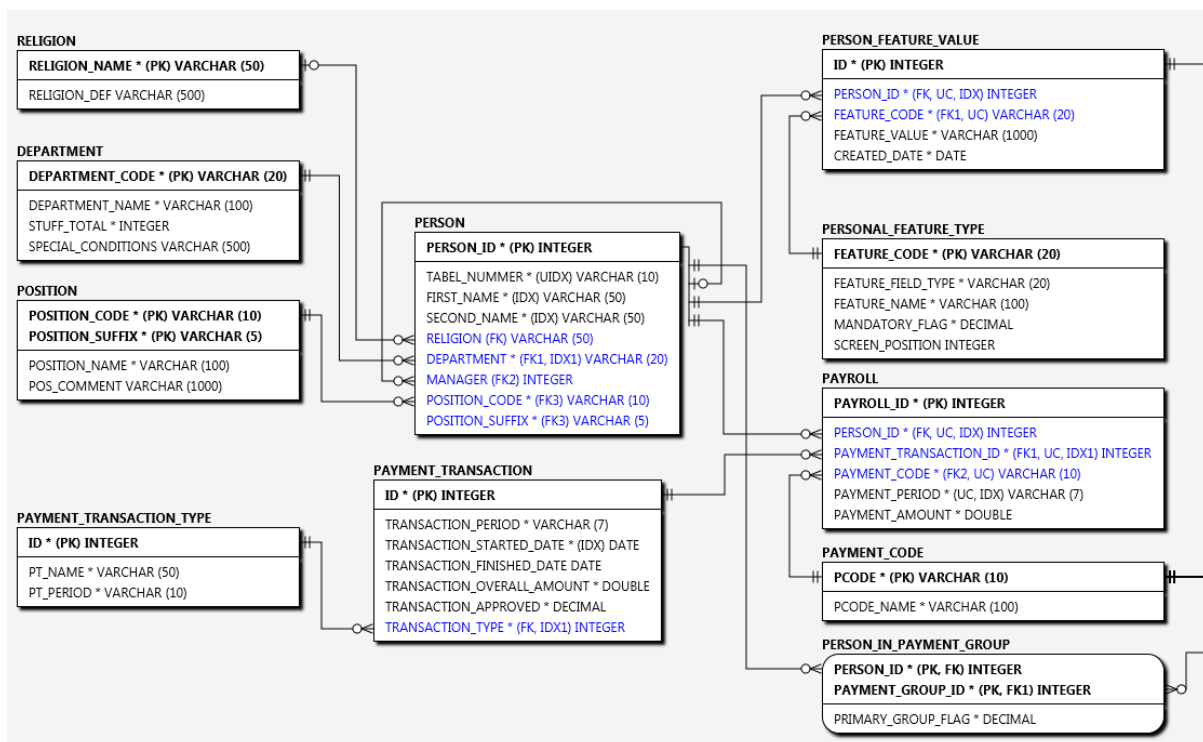


The “simplified” view contains only the primary key attributes, omitting the non-key attributes:



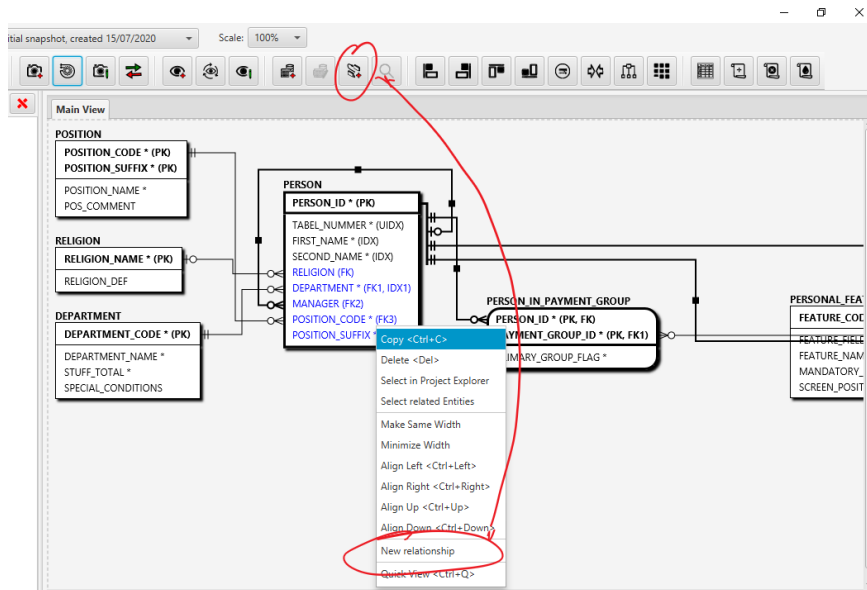


The “extended” view reflects the attributes type information:



## 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, or use the context menu in the drawing area as shown below.



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:

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.

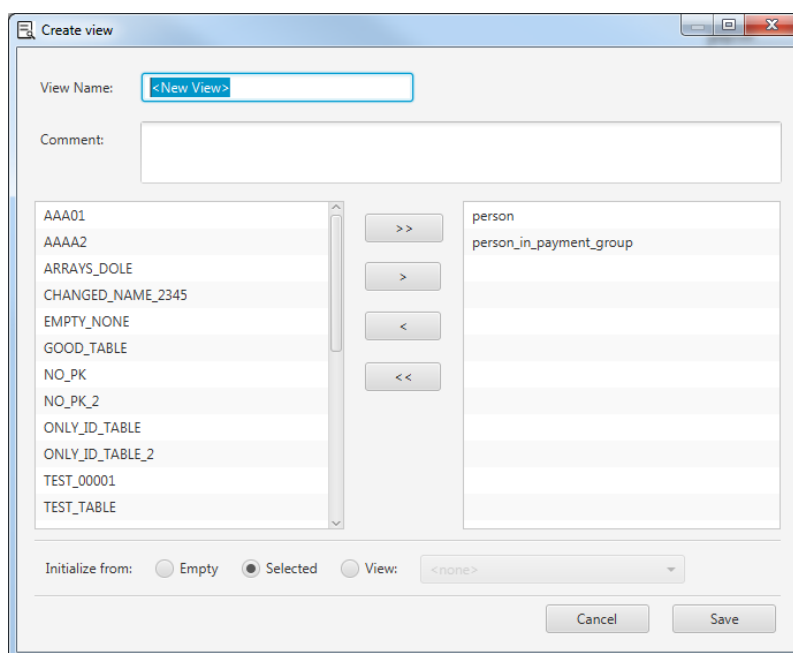
There is no “Edit” functionality for the Relationship. To alter the relationship, it has to be deleted and created from scratch.

## 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).

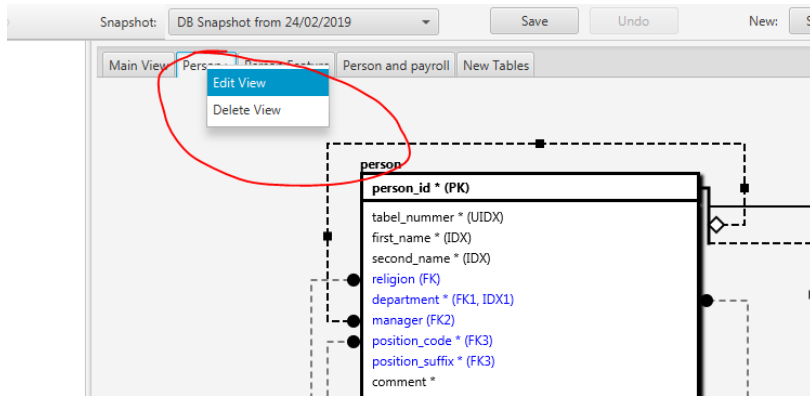


As it was already mentioned above, the View belongs to the whole Project – not to the current Snapshot. It means, that any changes in the View reflect in other Snapshots as well. This needs to be taken into account while working with “Apricot DB”.

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.

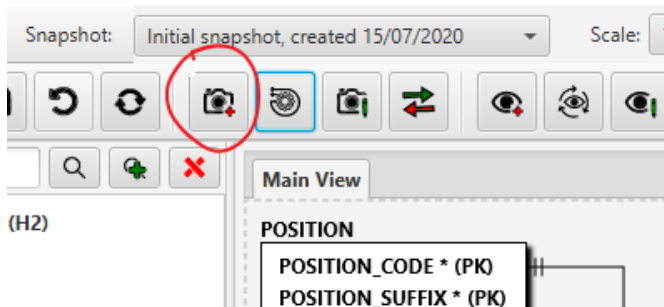
In order to edit or delete the View, use the right button mouse click over the View’s tab. The following context menu is shown below:



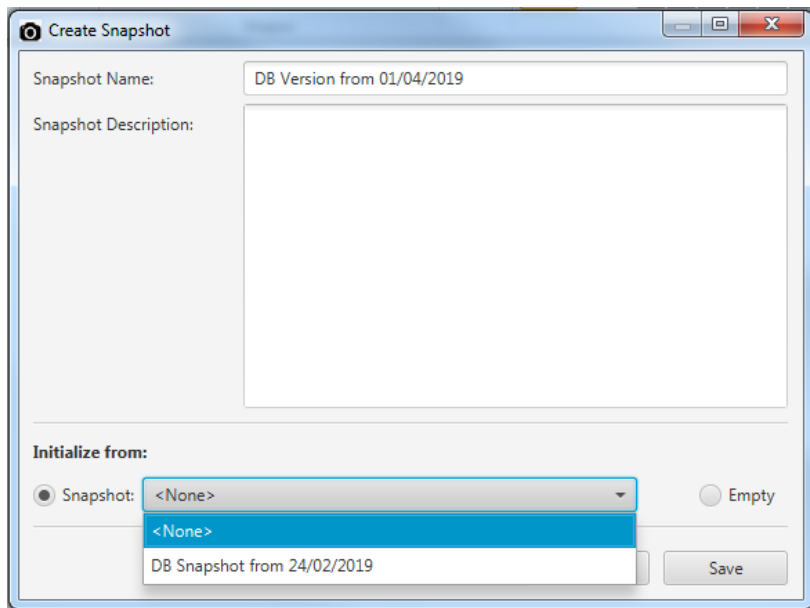
## Working with the Snapshots

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:

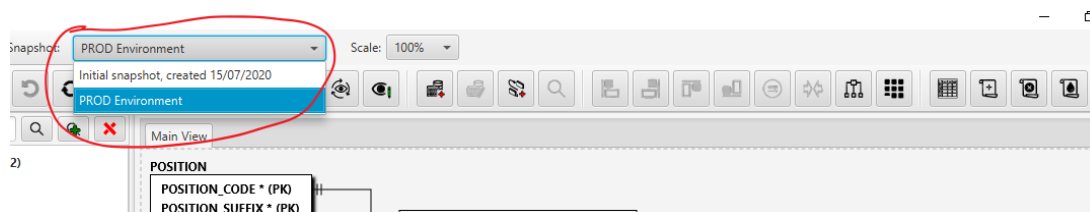


The new Snapshot can be created empty (see “Initialize from” section of the form), or from another Snapshot. The first option is useful when the ERD has to be created from scratch. Another application of the empty Snapshot would be the Reverse Engineering- operation. The Reverse Engineering of the existing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



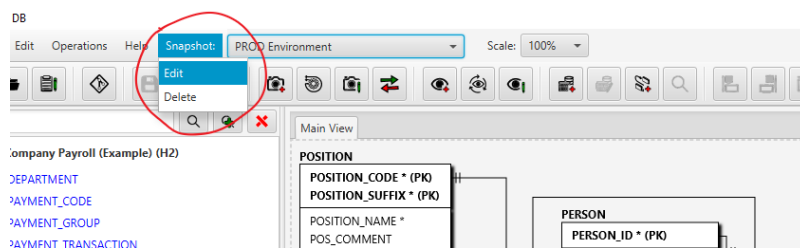
Another option of creation of the Snapshot is to create Snapshot from the reference one. This option is useful, when user needs a new version of the database structure and wants to save the original one.

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



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.

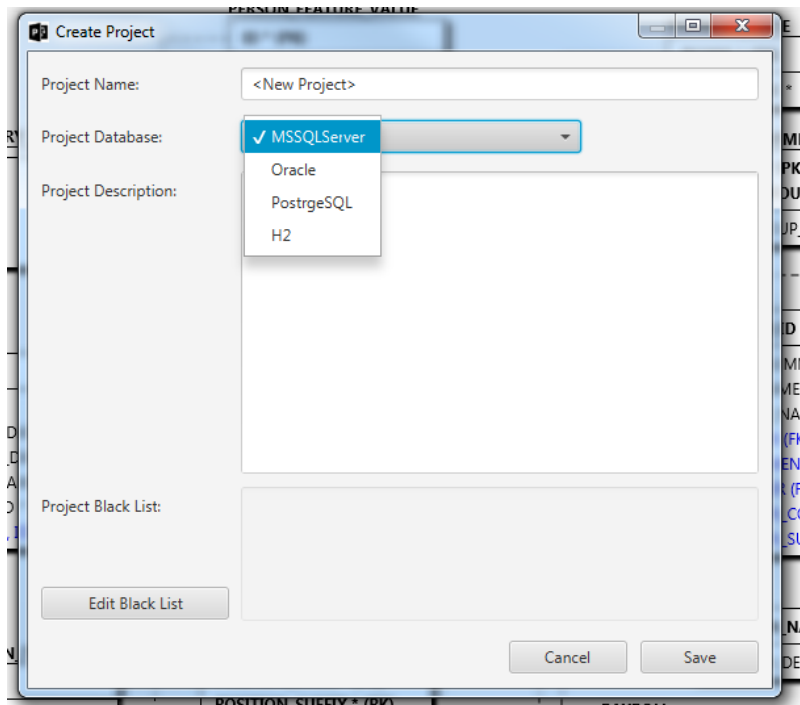
In order to edit the Snapshot name or description or delete Snapshot use the menu "Snapshot:" as shown below:



## 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.



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.

## Editing the diagram content

“Apricot DB” has a convenient graphical editor which allows to precisely positioning the entities and relationships between them.

### Selecting and moving the elements of the diagram

An Entity on the diagram can be selected and moved. If it required to select several entities there are two ways to do that. The selected entities and relationships have been drawn bold.

- Select multiple entities holding the <Ctrl> button.
- Select multiple entities using “lasso”: click and hold the left mouse button. Move the mouse. All entities within the square drawn (inside the “lasso”) will be selected.

In order to un-select all entities click outside of any entities on the diagram.

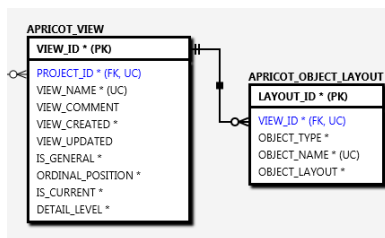
The selected entity can be moved holding the left mouse button.

In order to move more than one selected entity simultaneously, use the <Ctrl> button.

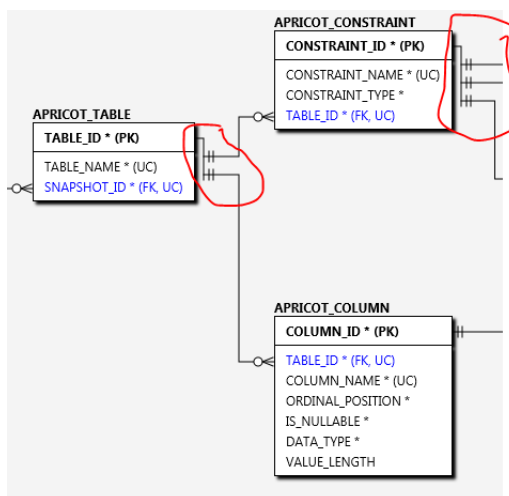
### How “Apricot” draws the relationships

The “Apricot DB” uses “semiautomatic” way of drawing of relationships between entities.

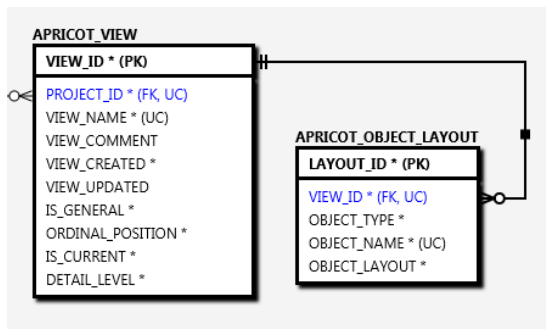
The relationship between two entities always starts at the Primary Key of the parent entity and ends at the Foreign Key field of the child entity:



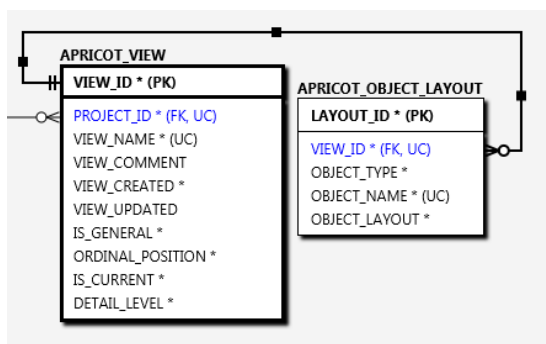
The only exclusion is when the parent entity has more than one child. In this case “Apricot” automatically draws a “stack” of relationships how it is shown on the picture below:



Depending on the mutual allocation of the parent and child entities, the relationship might change its shape. For example, when the entities are located too close, “Apricot” automatically optimizes the shape:



or:

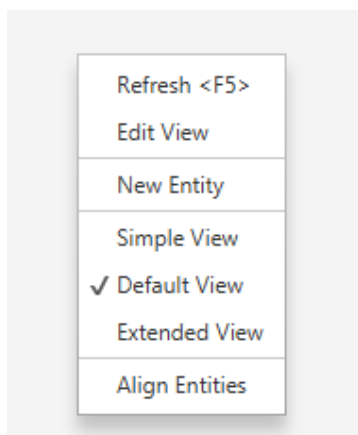


There is a limited opportunity to edit relationship appearance. Being selected, a relationship has one or more rulers (a small black square). The ruler can be moved perpendicularly to the line of relationship which it is drawn above.

## The context menus

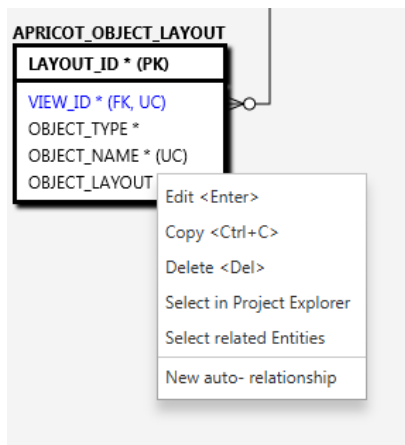
Many elements on the “Apricot”- diagram have a context menus. Below are the examples of the context menus for different elements of the diagram.

The diagram general context menu (called anywhere in clean field of the diagram):

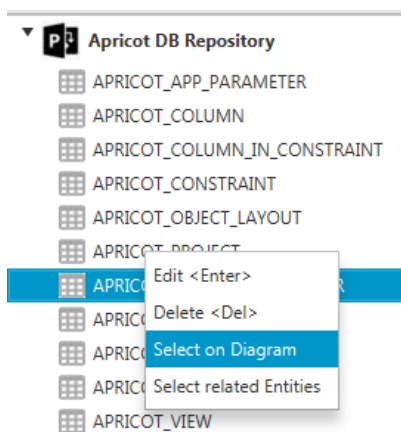




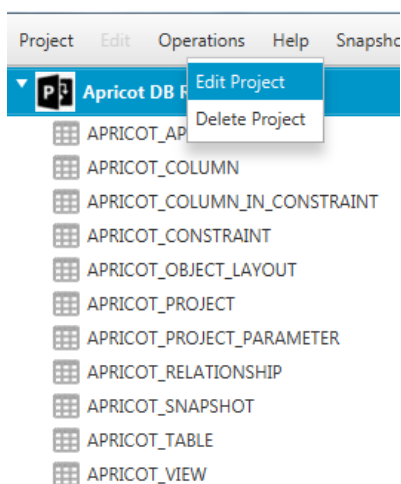
The entity context menu:



The project explorer entity context menu:

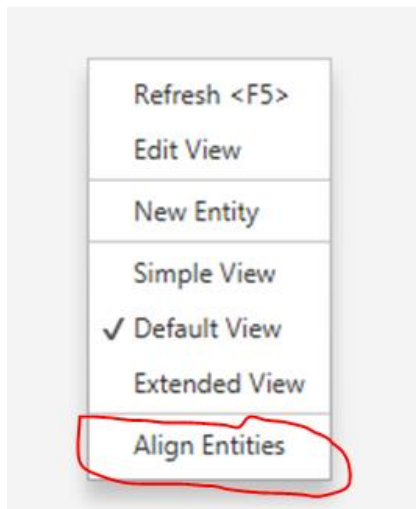


The project context menu in the project explorer:



## The automatic aligning of the diagram

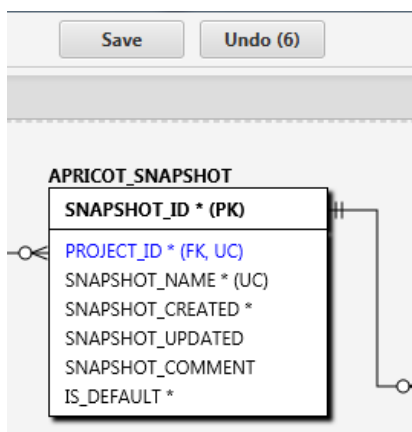
The diagram context menu contains the “Align Entities” item. Calling this function runs the automatic realignment of all elements on the current diagram.



### The Save and Undo operations

Any changes in the allocation of the elements of the diagram might be saved or undone. The “Save”-button becomes bold if there is not saved changes.

The “Undo” diagram stores up to 10 changes on the diagram. The counter on the right side of the “Undo” button contains a deep of the current undo- buffer.



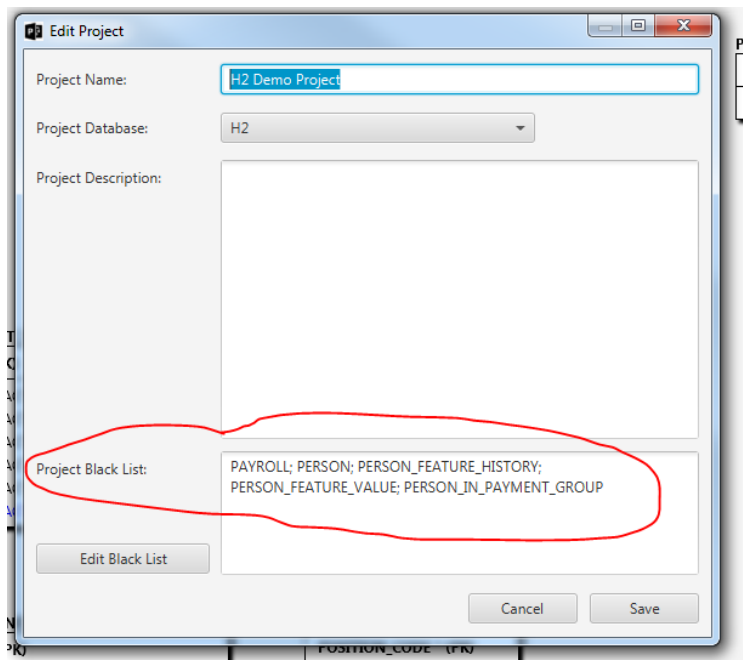
It is recommended to use the “Save” button as often as possible. It allows to avoid any losses in the diagram changes.

### Selecting Entities on the Diagram

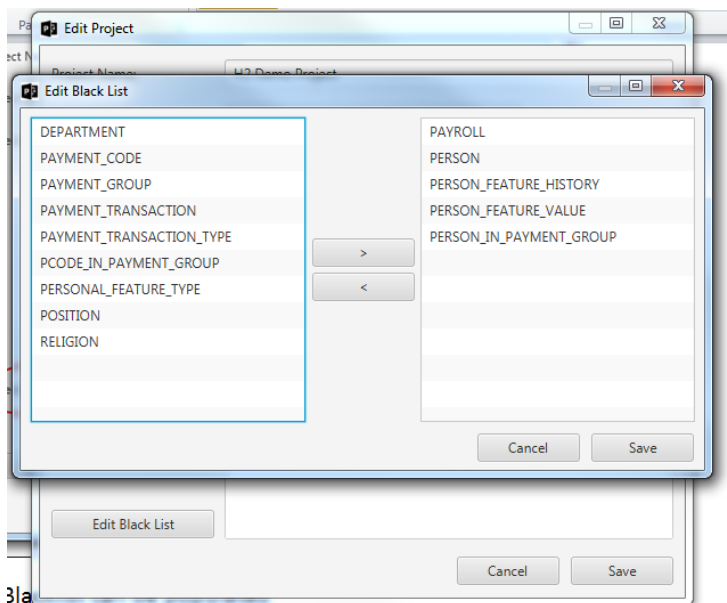
Any Entity in Apricot DB can be selected by left mouse click. The multiple Entities can be selected with the <Ctrl> button plus left mouse button. To move the selected multiple Entities, use <Ctrl> and drag with the left mouse pushed.

### 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.



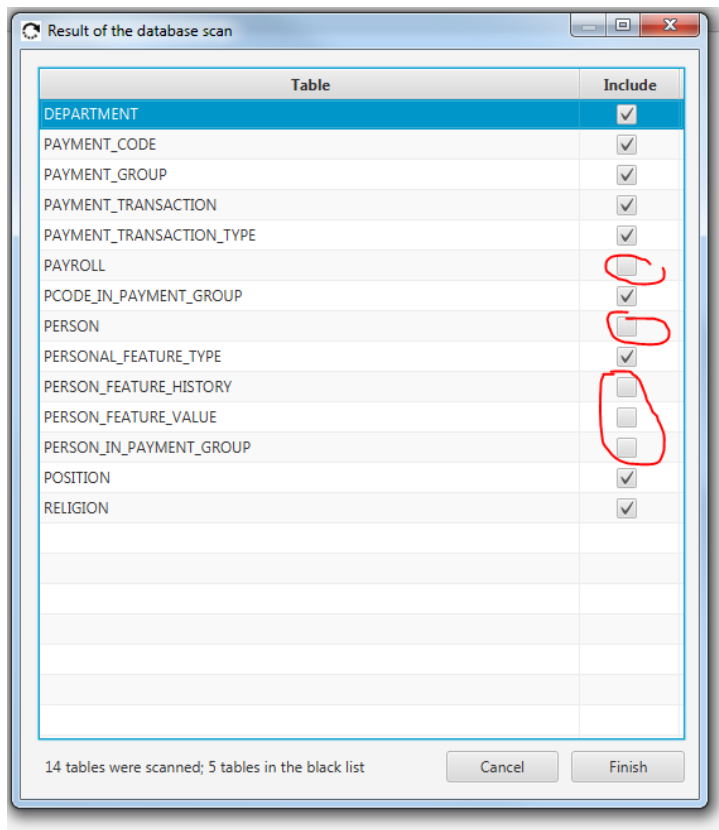
The Blacklist can be populated in two ways: explicitly via the form of editing of the Project (use the “Edit Black List” button). The entities, included into the Blacklist have to be “moved” to the right-side as shown on the screen below:



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:



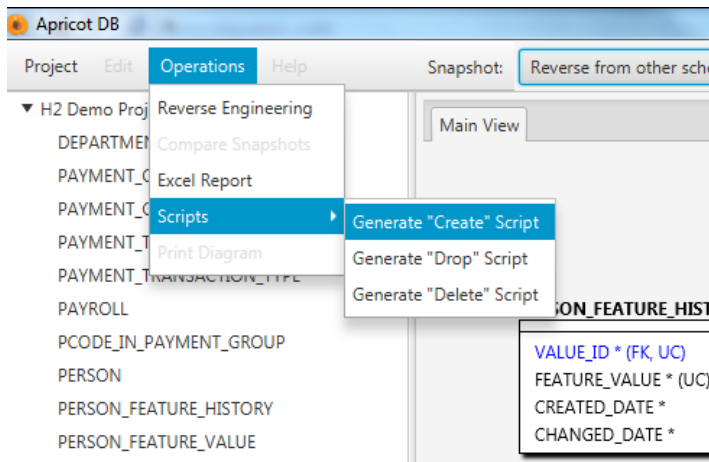
This means that “Apricot DB” scans the whole list of the tables in the source database, but uses the Blacklist and “suggests” excluding the previously excluded tables from the result set. User can check the unchecked entities in the form above, and they will be included into the Reverse Engineering result.

## Generation of the Database Scripts

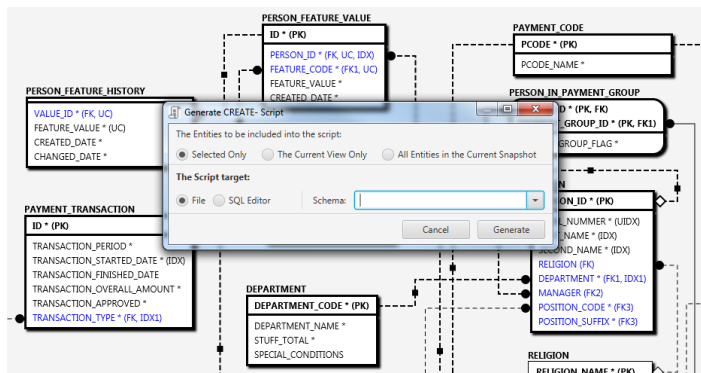
There are 3 types of database scripts which can be generated by “Apricot DB”:

- 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:



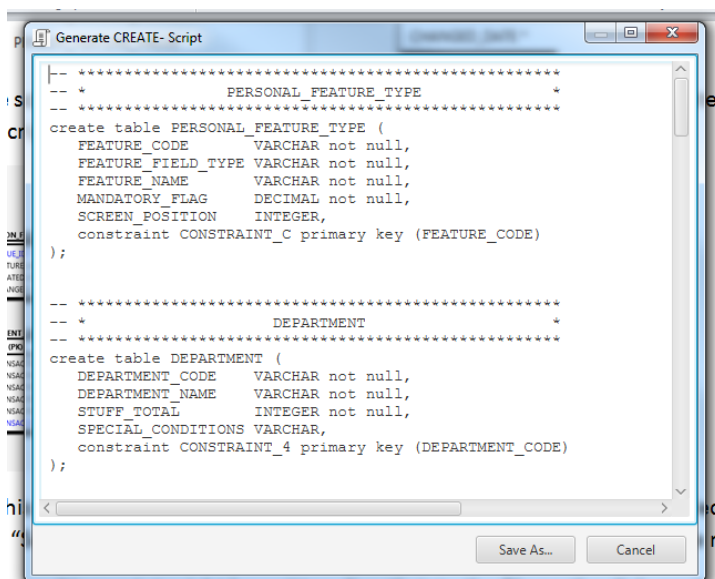
The script generation form is shown below (this form is common for Create, Drop and Delete types of scripts):



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.

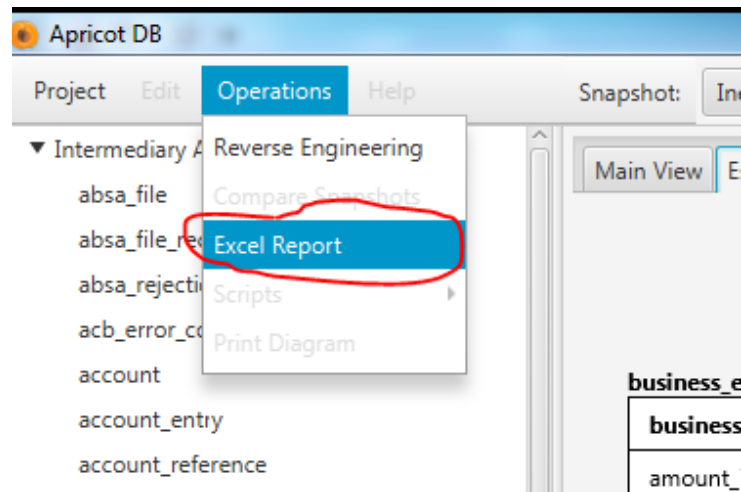
The resulting script might be written directly into the file on the disk, or shown in the simple SQL Editor:



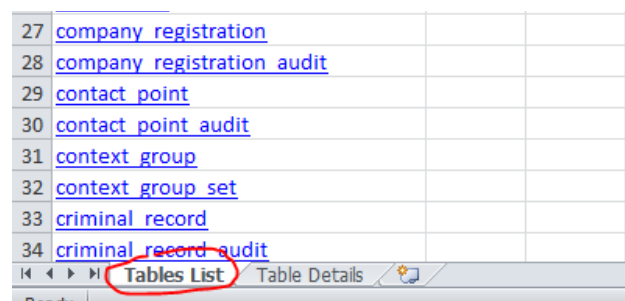
To run the resulting script on the target Database, use your favourite data access and administration tool (DB Visualizer?).

## 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	<a href="#">company_registration</a>		
28	<a href="#">company_registration_audit</a>		
29	<a href="#">contact_point</a>		
30	<a href="#">contact_point_audit</a>		
31	<a href="#">context_group</a>		
32	<a href="#">context_group_set</a>		
33	<a href="#">criminal_record</a>		
34	<a href="#">criminal_record_audit</a>		

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	<a href="#">batch_step_execution context</a>
215		2 version *	bigint		
216		3 step_name *	varchar (100)		
217	<a href="#">batch_job_execution</a>	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	<a href="#">batch_step_execution</a>	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.