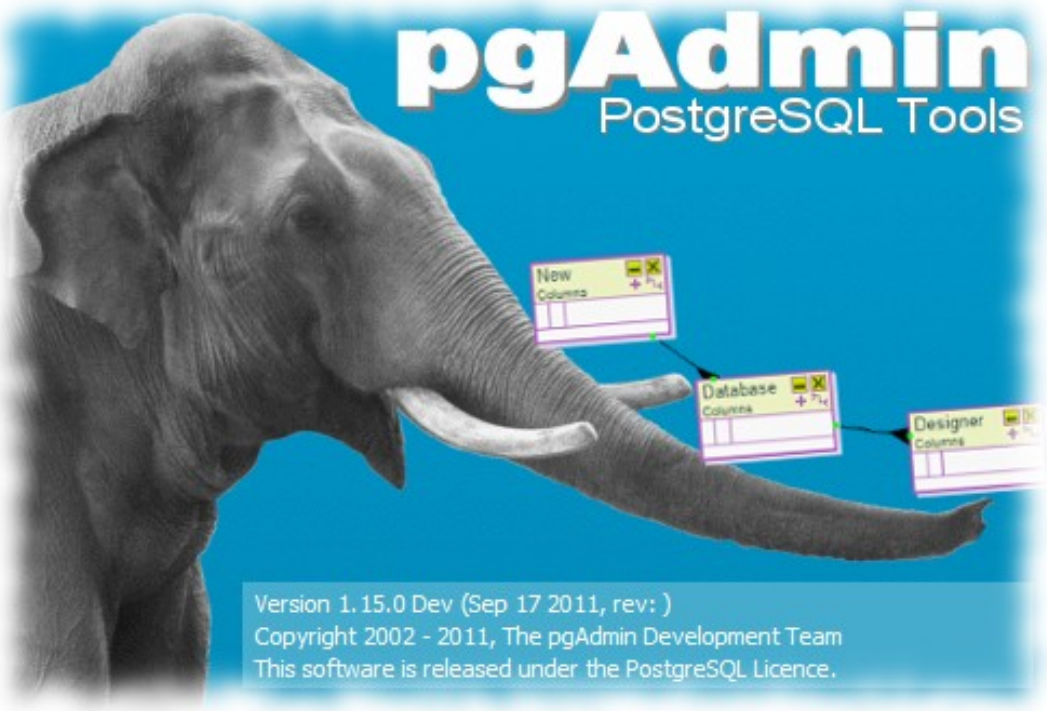
Relational Database Designer for pgAdmin



Guillaume Lelarge
Luis Ochoa

PGConf.EU 2011

GSoC

- Program designed to encourage students to participate in open source development

	2005	2006	2007	2008	2009	2010
Students	400	630	900	1125	1000	1026
Organizations	40	100	130	175	150	150
Countries	49	90	90	98	70	70
Success Rate	82%	82%	80%	83%	85%	90%

How does GSoC works?

- Students submit project proposals to the organizations, organizations rank the submissions (students paired with mentor from open source community)
- Execute to milestones laid out in accepted program application
- Google allocates a given number of slots to each organization, the students work all summer on their project in close mentored collaboration with that organization
- Program stipend allows students to concentrate on OSS development full-time

From student point of view GSoC Is about money?

- Working at GSoC is not (only) about money
- Stipend helps
- You still need to love what you're doing
  - big sacrifices in your summer time.
  - isn't a hourly paid job
- But the final result will help you a lot
  - learning a lot
  - quite helpful on the job market

pgAdmin

- Free and open source graphical front-end administration tool for PostgreSQL
- Supported on many OS
- Helps to administer a PostgreSQL database
- No tool for modeling designs of databases
  - Until now...

Why modeling a database?

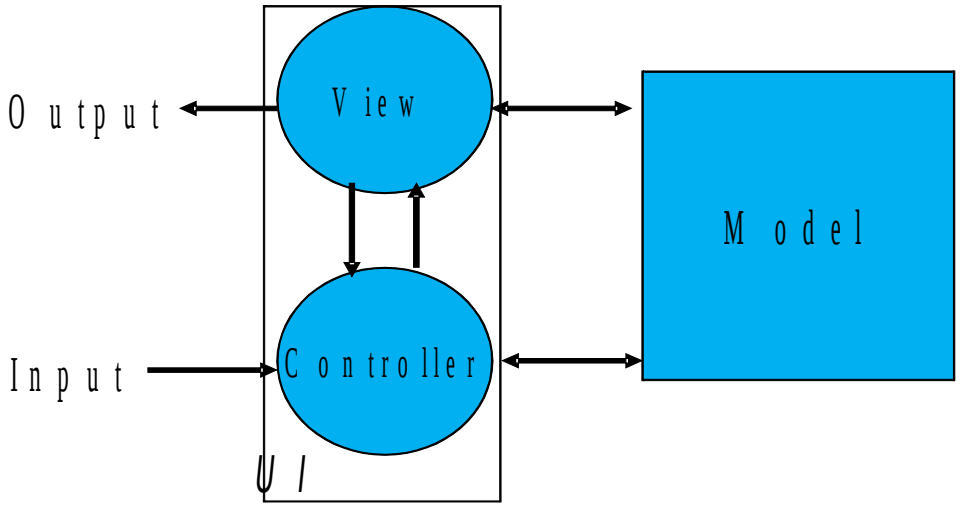
- Facilites discussions
- Helps to avoid misunderstanding
- Allows to have a big picture of a database very quickly
- Faster to create a database than hand made DDL scripts

Why modeling a database?

- Facilites discussions
- Helps to avoid misunderstanding
- Allows to have a big picture of a database very quickly
- Faster to create a database than hand made DDL scripts

Initial approach creating a modeler

- Initial approach... design pattern: Model View Controller (MVC)
- Used before for graphical query builder (gqb)



But :
It can be implemented in several ways, and sometimes this lead to problems as I learned with gqb, creating some inflexible source code difficult to extend.

What should I do?

- Graphical modeling is not new to pgAdmin
  - Graphical explain, Query Builder
- GQB is pure MVC patterns design
  - But not a very efficient and flexible implementation
- Graphical explain uses a library called WxOGL
  - Very good, very flexible, but isn't enough

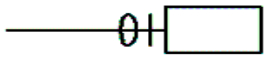
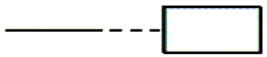
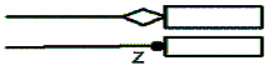
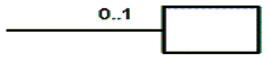
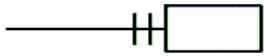
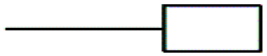
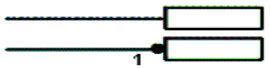
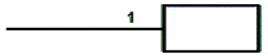
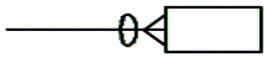
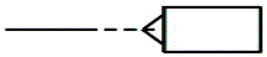
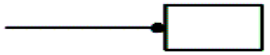
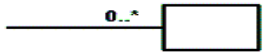
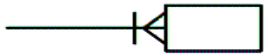
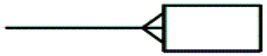
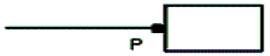
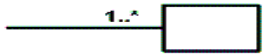
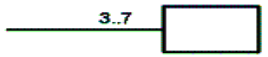
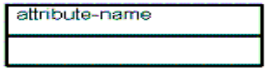
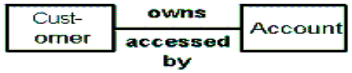
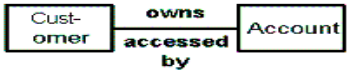
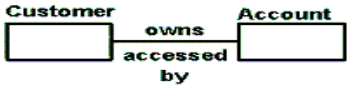
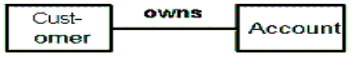
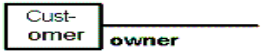
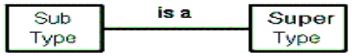
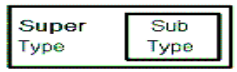
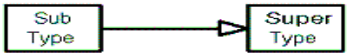
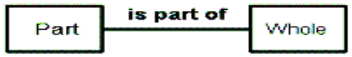
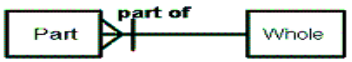
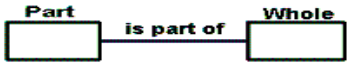
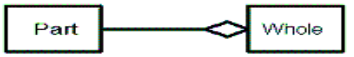
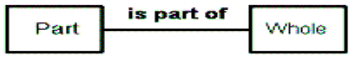
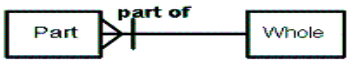
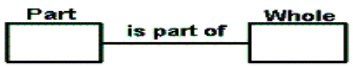
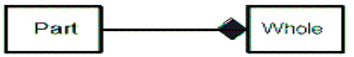
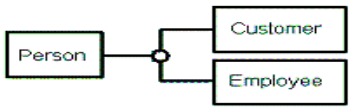
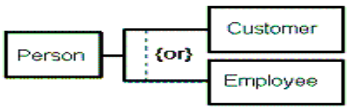
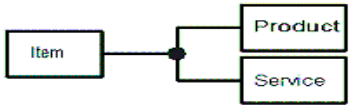
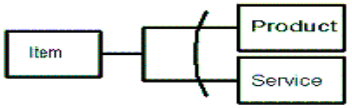
Hotdraw

- Two-dimensional graphics framework for structured drawing editors
- Several implementations of this framework:
  - monohotdraw, jhotdraw.
- No wxWidgets implementations
  - wxOGL is similar but lacks some features
- Need to implement some kind of hotdraw-like “framework”

Notations

- First decisions: which notation to use for modeling
- Several possible notations
- Would be best to have it all
- But first need to support at least one
- So, we need to choose one between...

Notations

Notation	Information Engineering	Barker Notation	IDEF1X	UML
Multiplicities:				
- Zero or one				
- One only				
- Zero or more				
- One or more				
- Specific range	N/A	N/A	N/A	
Attributes:				
Names	N/A	Attribute Name: Type	attribute-name: Type	attributeName: Type
Primary key/unique identifier	N/A	# Attribute Name		attributeName <<PK>> {order=#}
Foreign key	N/A	N/A	attribute-name (FK)	attributeName <<FK>> {to=tablename}
Associations:				
Labels				
Entity roles	N/A	N/A	N/A	
Subtyping				
Aggregation				
Composition				
Or Constraint		N/A	N/A	
Exclusive Or (XOR) Constraint			N/A	

Barker Notation

- Barker's notation: developed by Richard Barker, Ian Palmer, Harry Ellis et al. around 1981.
- Adopted by Barker when he joined Oracle
- Defined in his book Entity Relationship Modeling
- Variation of the "crows foot" style of data modelling
- Favoured by many because of its
  - Readability
  - and efficient use of drawing space

What to do next?

- We have a modeling framework (hotdraw)
- A notation for physical models (Barker)

Next step is to select the features of the database designer.

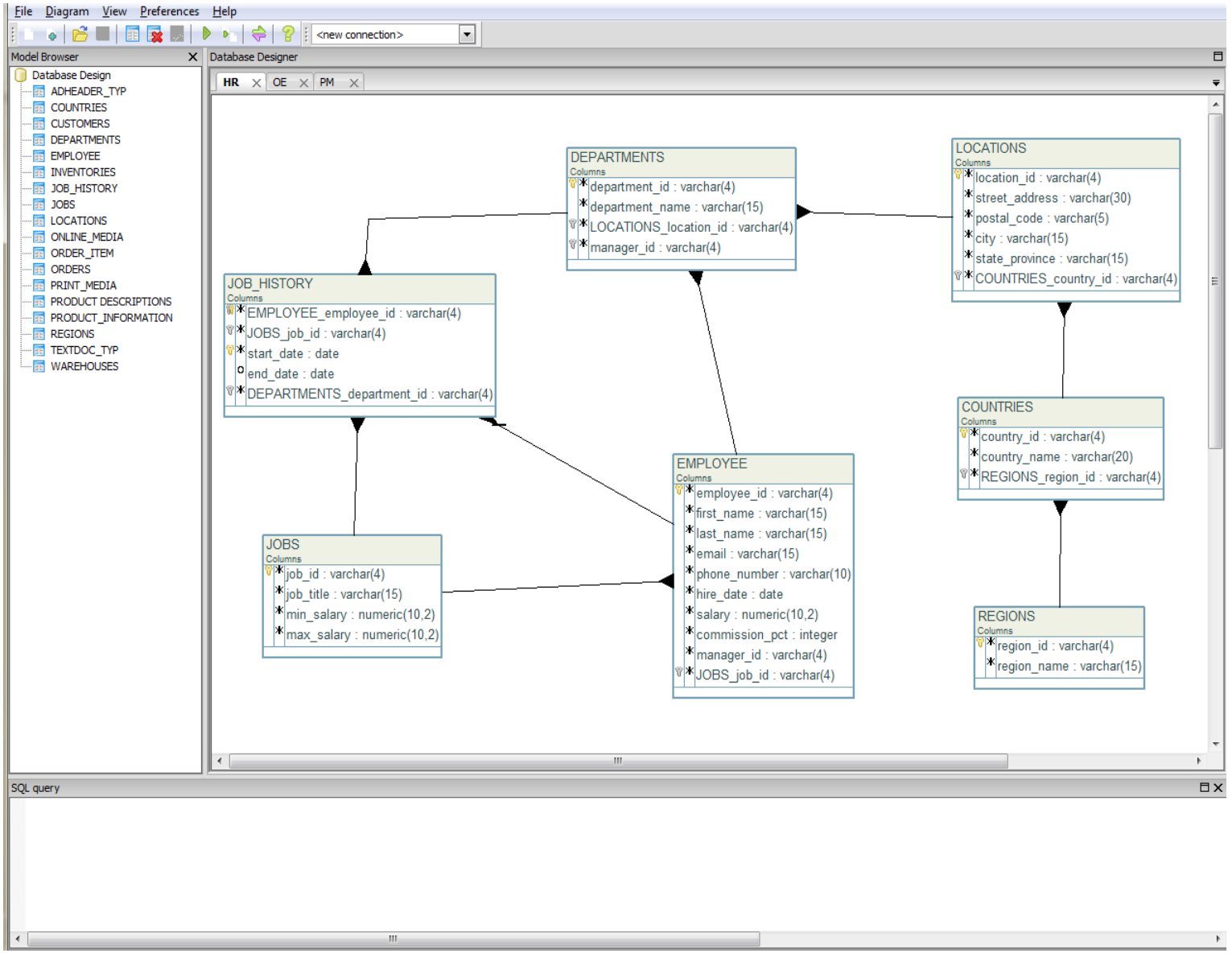
Interesting features

- Table and relationship modeling with a graphical UI
- Automatic foreign key handling or customized ones
- Model generation into DDL scripts
- Save / Load of diagrams
- Some way of handling big diagrams in an efficient way
- Import tables from database to diagram (reverse engineering)
- Updates tables in database with changes done at diagram level

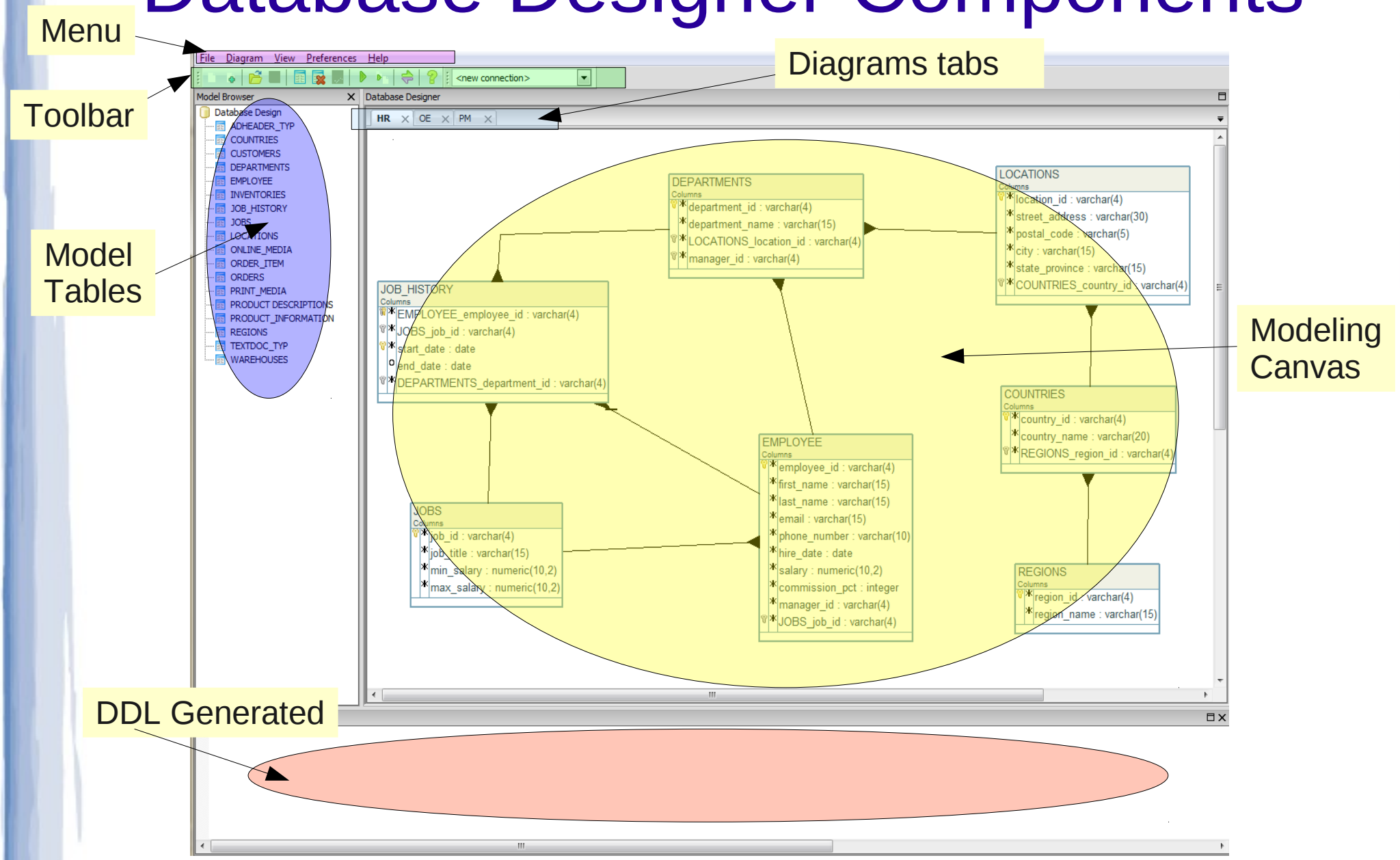
The Result

- A physical database designer, created by using the hotdraw like framework that creates models with diagrams inside it.
- This database designer have two important concepts:
  - **Model:** A collection of tables and relationships that eventually will become a Postgresql database.
  - **Diagram:** A view of some tables and relationships in a model, each diagram belongs to an unique model. A table can be used into several diagrams.
- Any changed made to a table / relationship inside a model will affect all diagrams.
- A table can be removed from a diagram or from a model, but if removed from a diagram, it still exists at model and can be added again.
- Relationships can be removed from model only.
- When a table is removed from a model, all related relationship (use as source this table) are removed too.

Database Designer User Interface

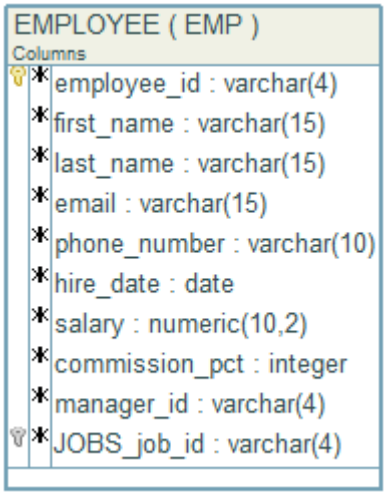


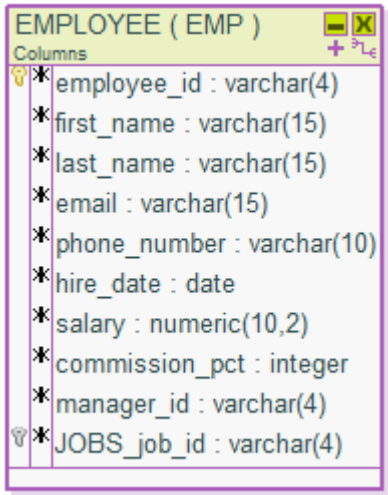
Database Designer Components



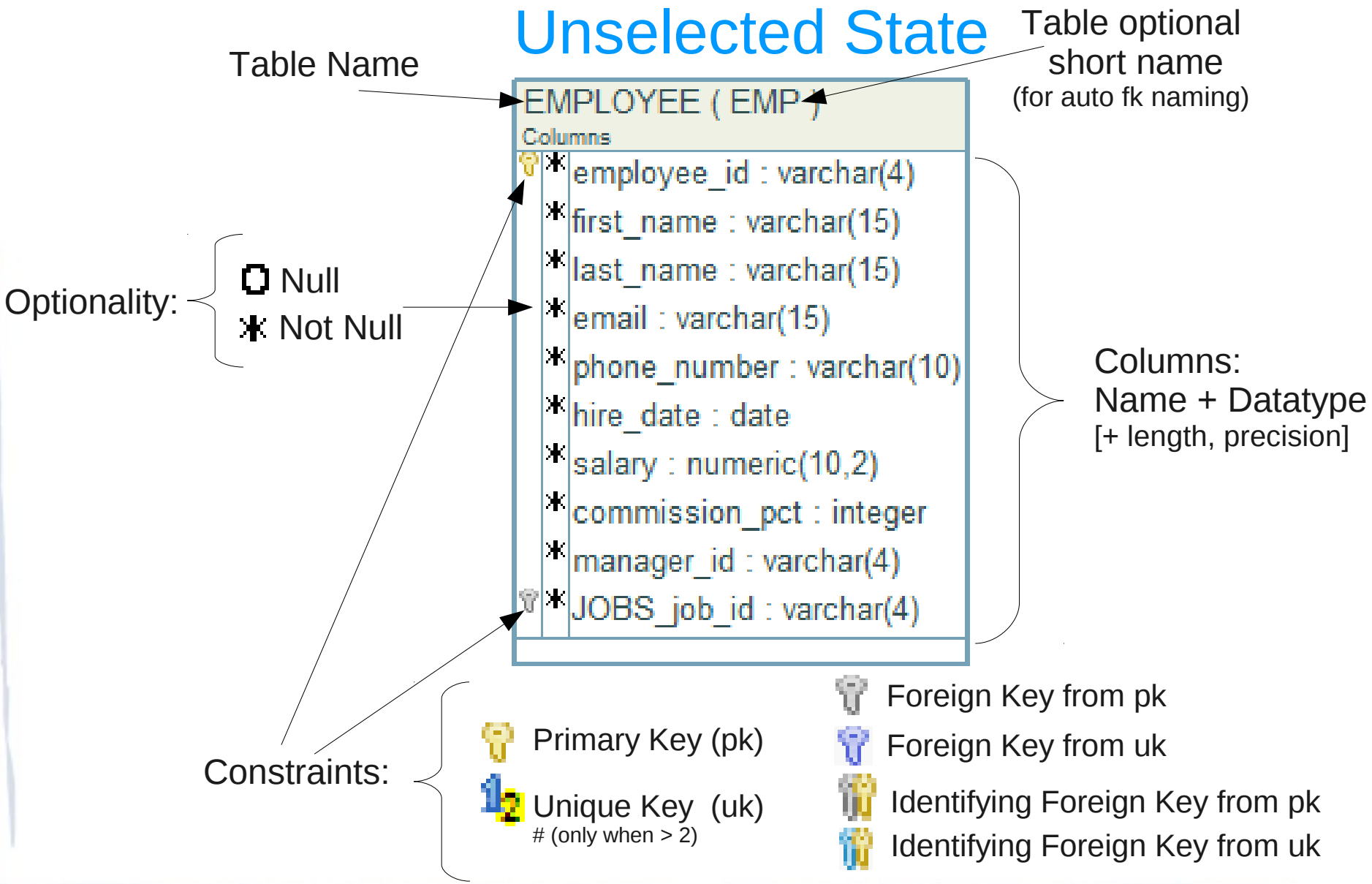
Tables

- Main graphical object of the modeler
  - can be added new into a diagram
  - or dragged and dropped from the browser.
- Allow to graphically show most importants column metadatas:
  - Column name, datatype, optionality (Null or Not Null), constraints (Primary Key, Unique Key, Foreign Key,...)
- Have two possible states: Unselected or Selected.





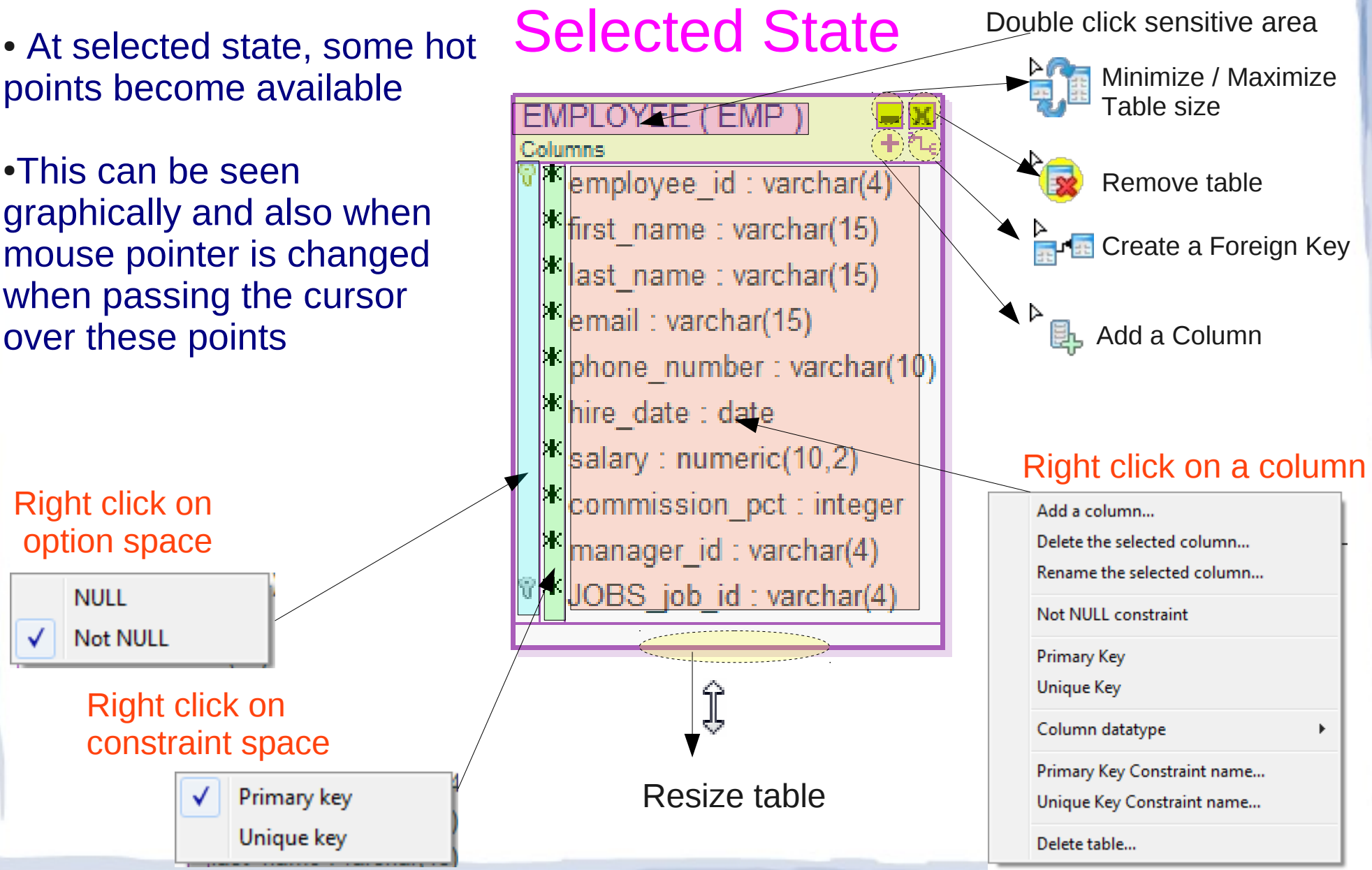
Unselected Table State



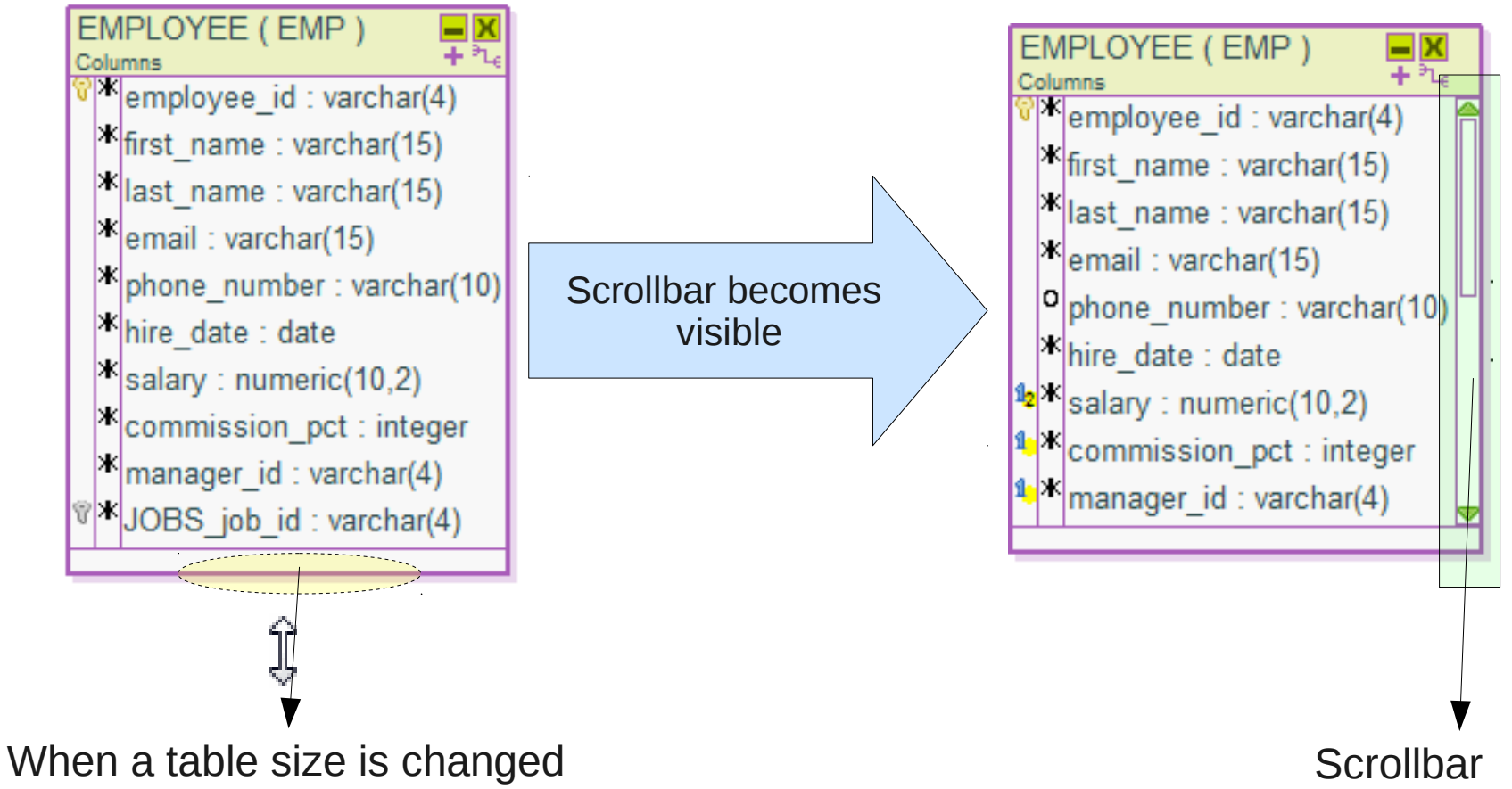
Selected Table State

- At selected state, some hot points become available

- This can be seen graphically and also when mouse pointer is changed when passing the cursor over these points



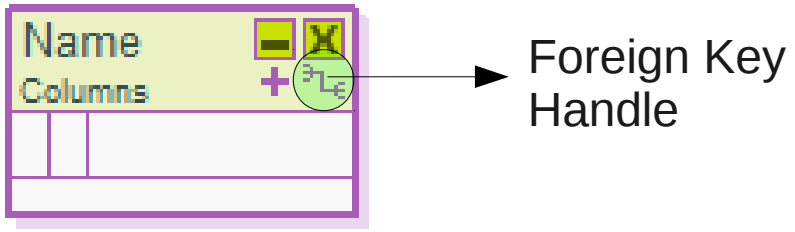
Modifying table size: Scrollbar



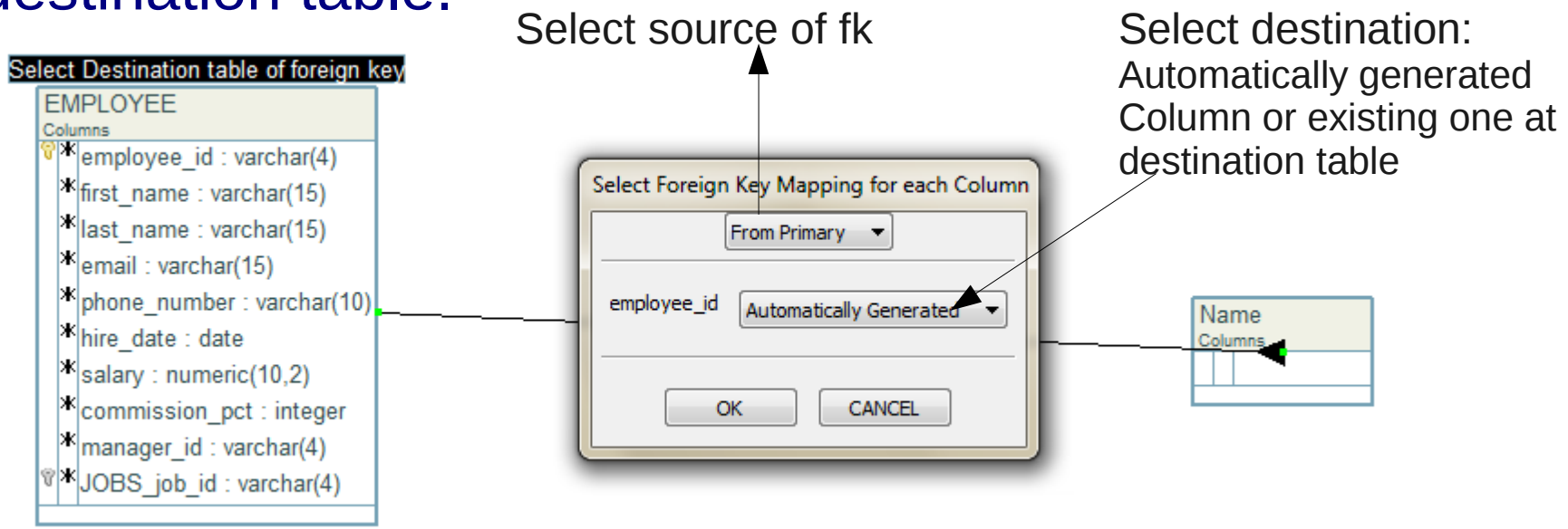
Creating a foreign Key

- In a relationship, you can set the following values:
  - Two kind of foreign keys:
    - Primary key as source.
    - Unique key as source.
  - Relationship kind:
    - Mandatory
    - Optional
  - Identifying relationship?
- You are able to create a fk that automatically creates a new column at destination table, or use a previously created column at destination table.
- There are two ways of creating a foreign key.

How to create a Relationship

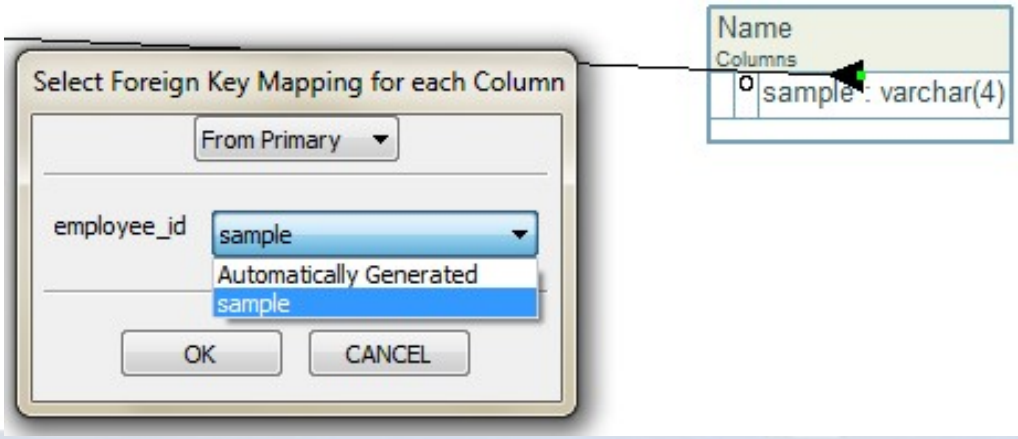


- Click source table fk handle [ ] and then click destination table.
- Drag and drop from source table fk handle [ ] to destination table.

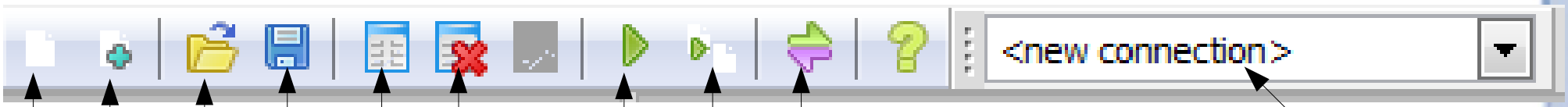


Foreign key automatically creation and synchronization

- When you select to create a FK, a dialog is opened to help you:
  - Add a new column to the target table
  - Use an existing column in the target table
- If you choose to automatically add a new column:
  - Column name can be generated automatically by using table short name
  - Changes at datatypes and columns propagate automatically.



Database Designer Toolbar



Select connection

Import Wizard

Generate DDL for current model (wizard)

Generate DDL for model (wizard)

Delete selected table(s)/relationship(s)

Add an empty table to the current diagram into a model.

Save a model to a file / Save changes inside a model

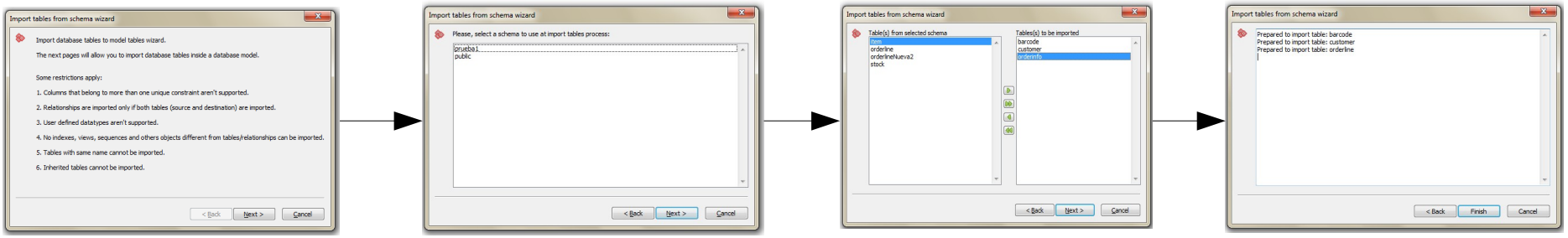
Open an existing model file

Create a new diagram inside a model.

Create a new model (New file with a model)

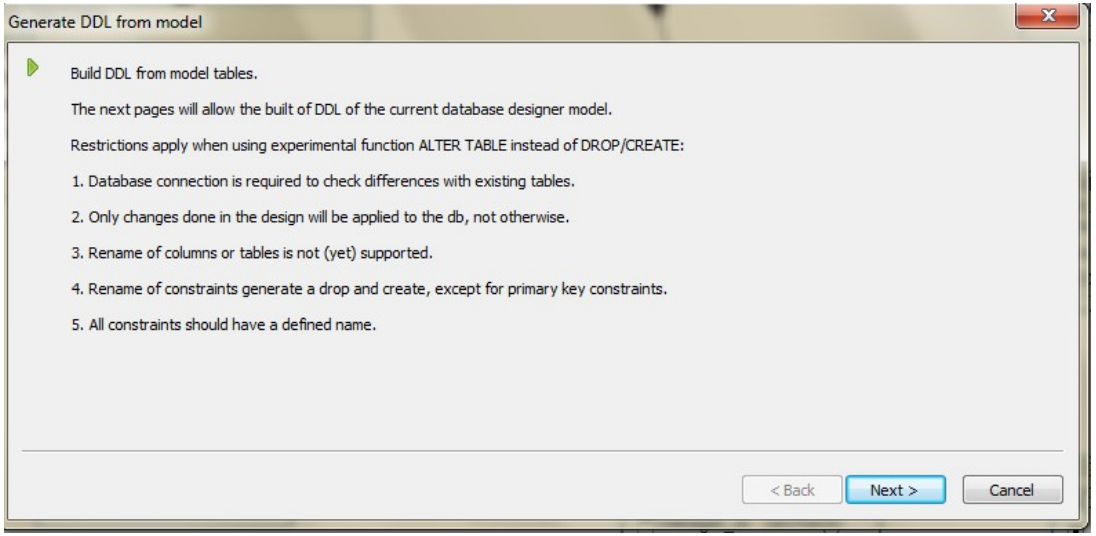
Import Tables

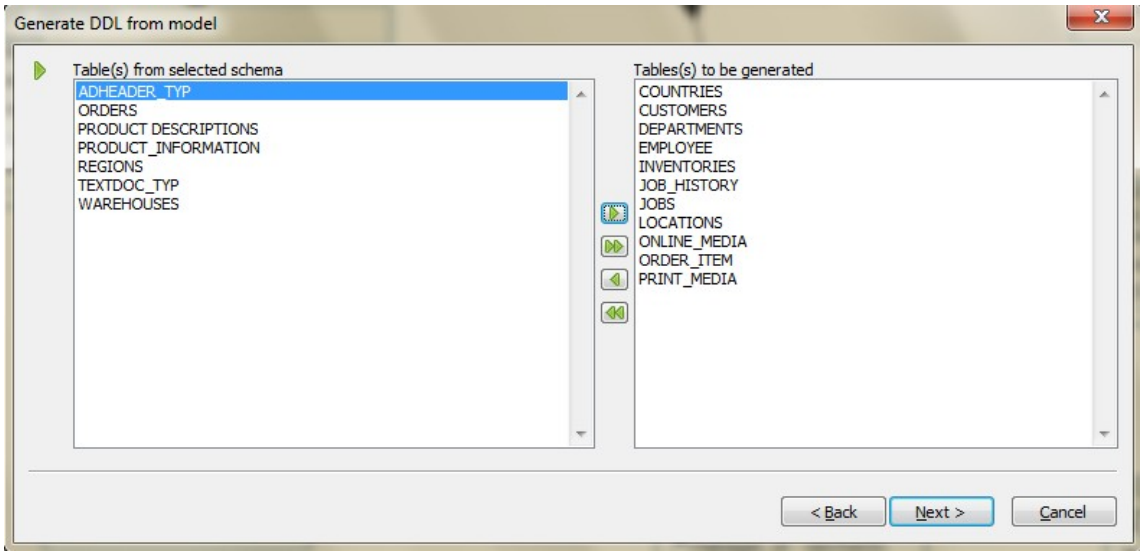
- A wizard like UI allow user to import tables from a schema to the model (some restrictions apply), steps are:
  - Select schema (database connection needed).
  - Select tables to be imported from schema.
  - Report results of import process.
  - Finish import (insert new tables into model)



Generate DDL from model

Initial Screen of Wizard



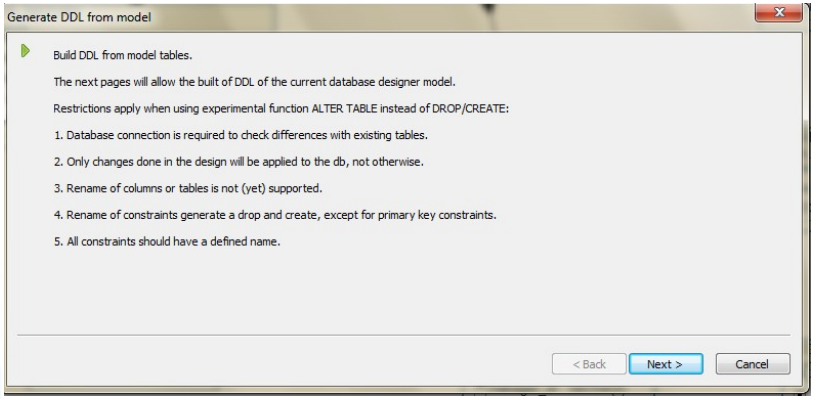


Select tables to be generated:

The user is responsible for any inconsistency in the generated DDL code

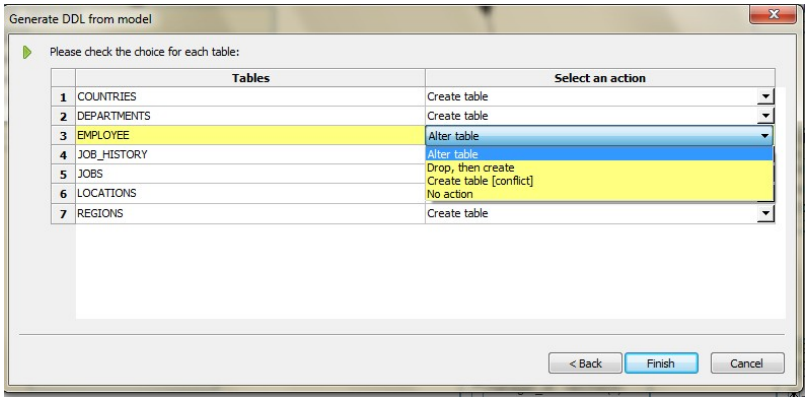
Generate DDL from model

Initial Screen of Wizard



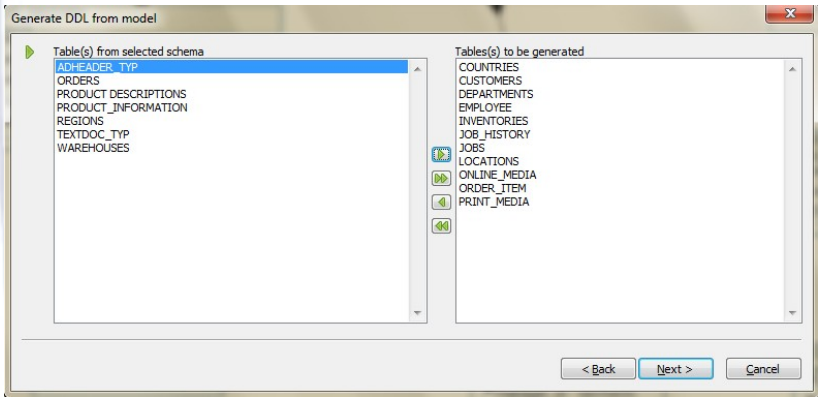
Select action:

- New tables only create.
- Existing Table:
  - Alter table
  - Drop, then create
  - Create
  - No action

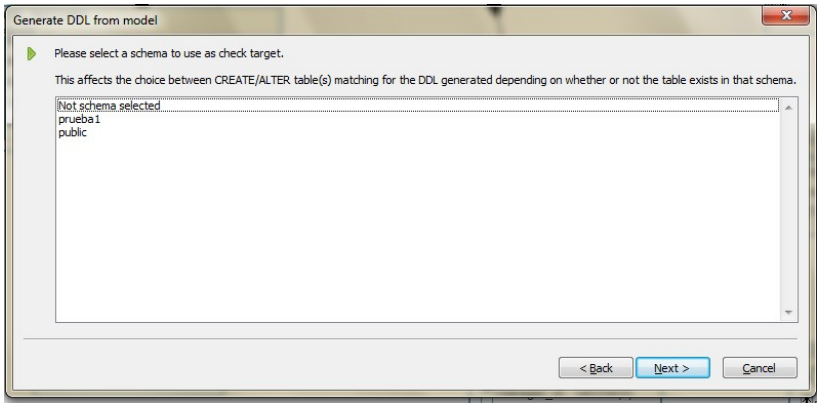


Select tables to be generated:

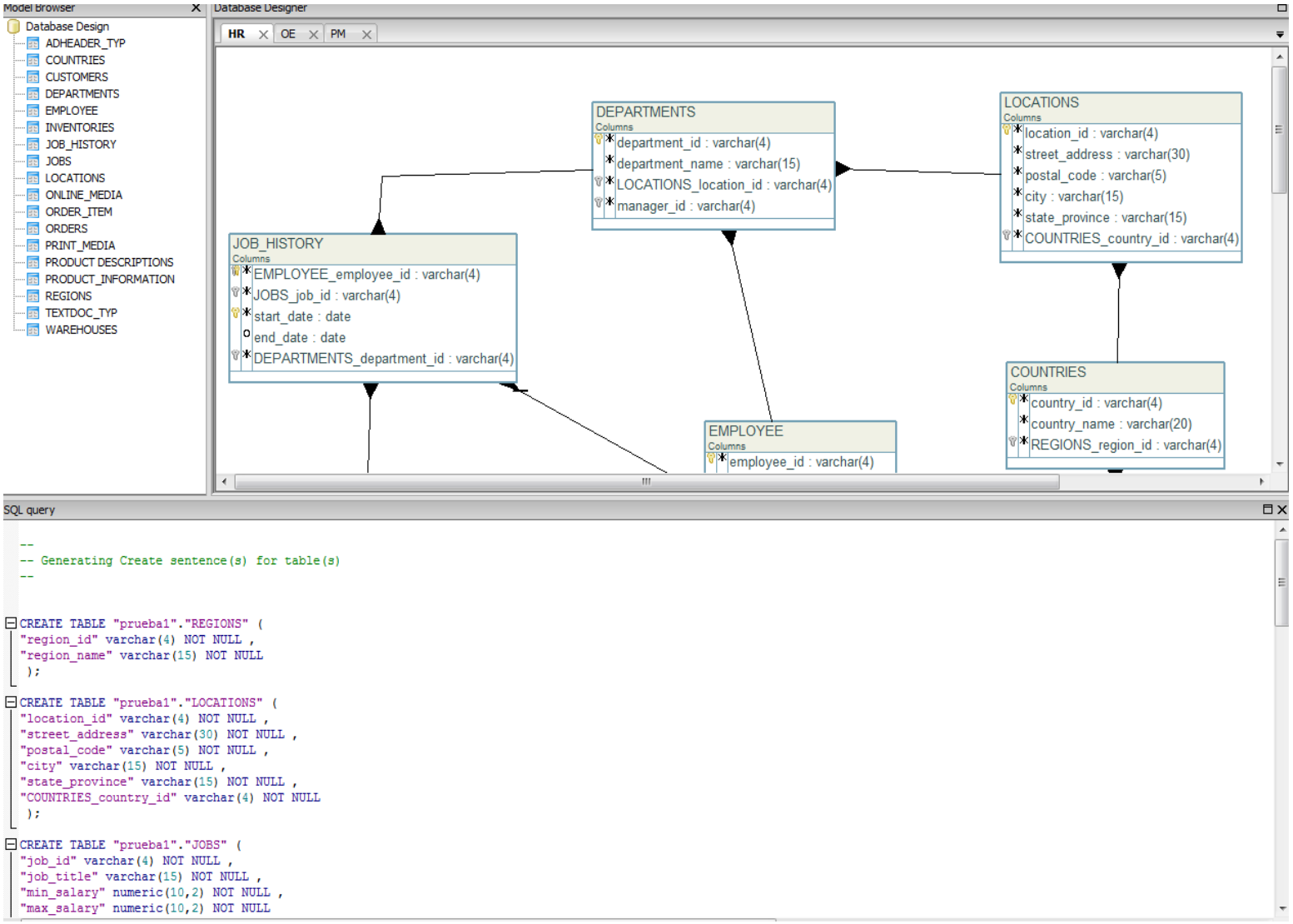
The user is responsible for any inconsistency in the generated DDL code



Select schema to use as target

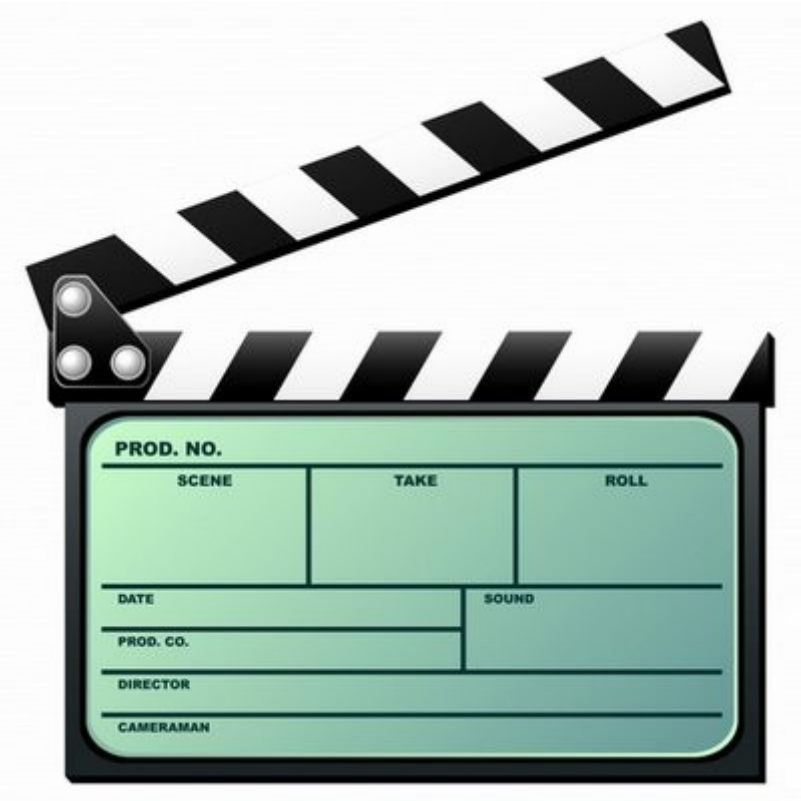


Generate DDL from model



DDL

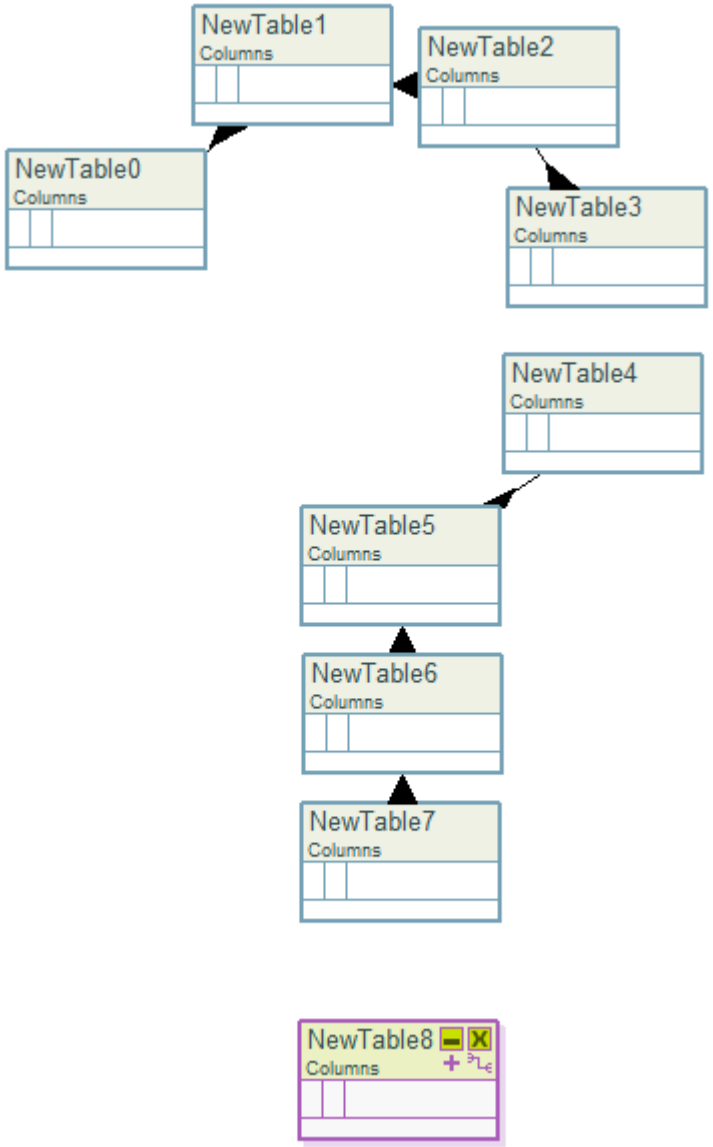
Demo


PROD. NO.
SCENE
TAKE
ROLL
DATE
SOUND
PROD. CO.
DIRECTOR
CAMERAMAN

Ideas about future of database designer

- Improve user interface.
- New kind of relationship lines (Orthogonal Lines).
- Allow self relationships.
- Improving of datatypes managing.
- Support of zoom feature.
- Support of exclusive Or (Xor) relationships.
- Undo / Rollback support.
- Logical Modeling.
- Automatic initial mapping from logical Model to physical Modeling.

Questions


NewTable1
Columns
NewTable2
Columns
NewTable0
Columns
NewTable3
Columns
NewTable4
Columns
NewTable5
Columns
NewTable6
Columns
NewTable7
Columns
NewTable8
Columns