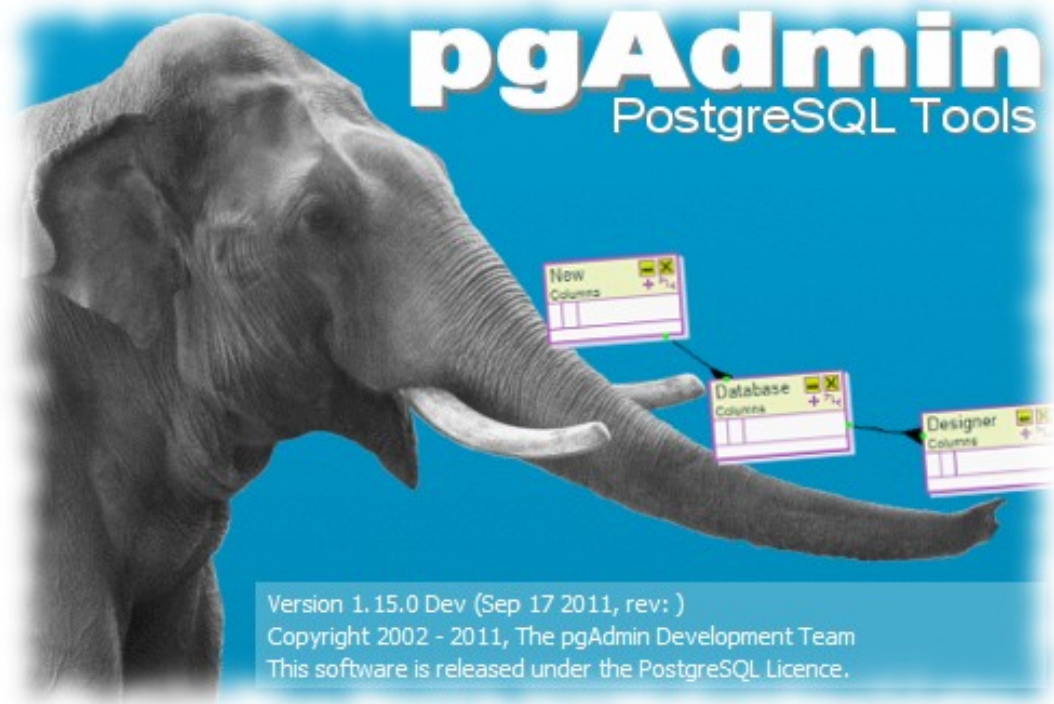


Relational Database Designer for pgAdmin



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?

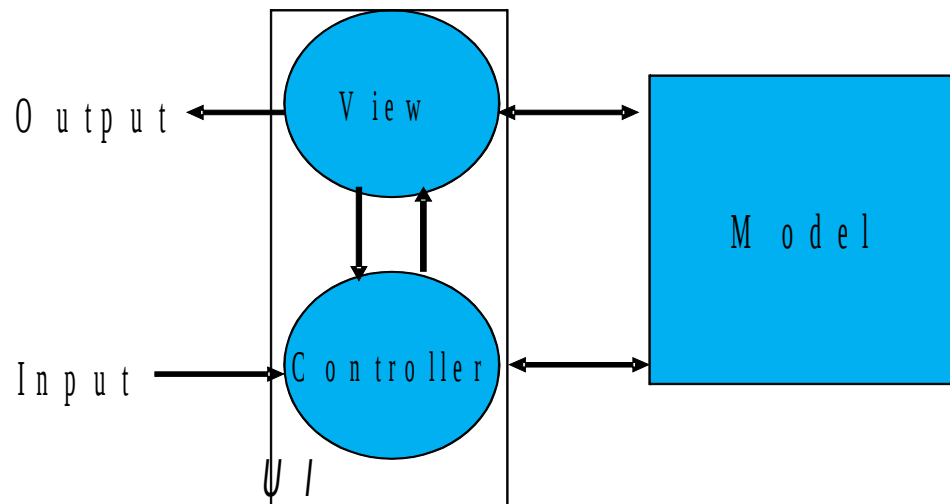
- Facilitates discussions
- Helps to avoid misunderstanding
- Allows to have a big picture of a database very quickly
- Faster to create a database than hand made DDL scripts

Why modeling a database?

- Facilitates 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.

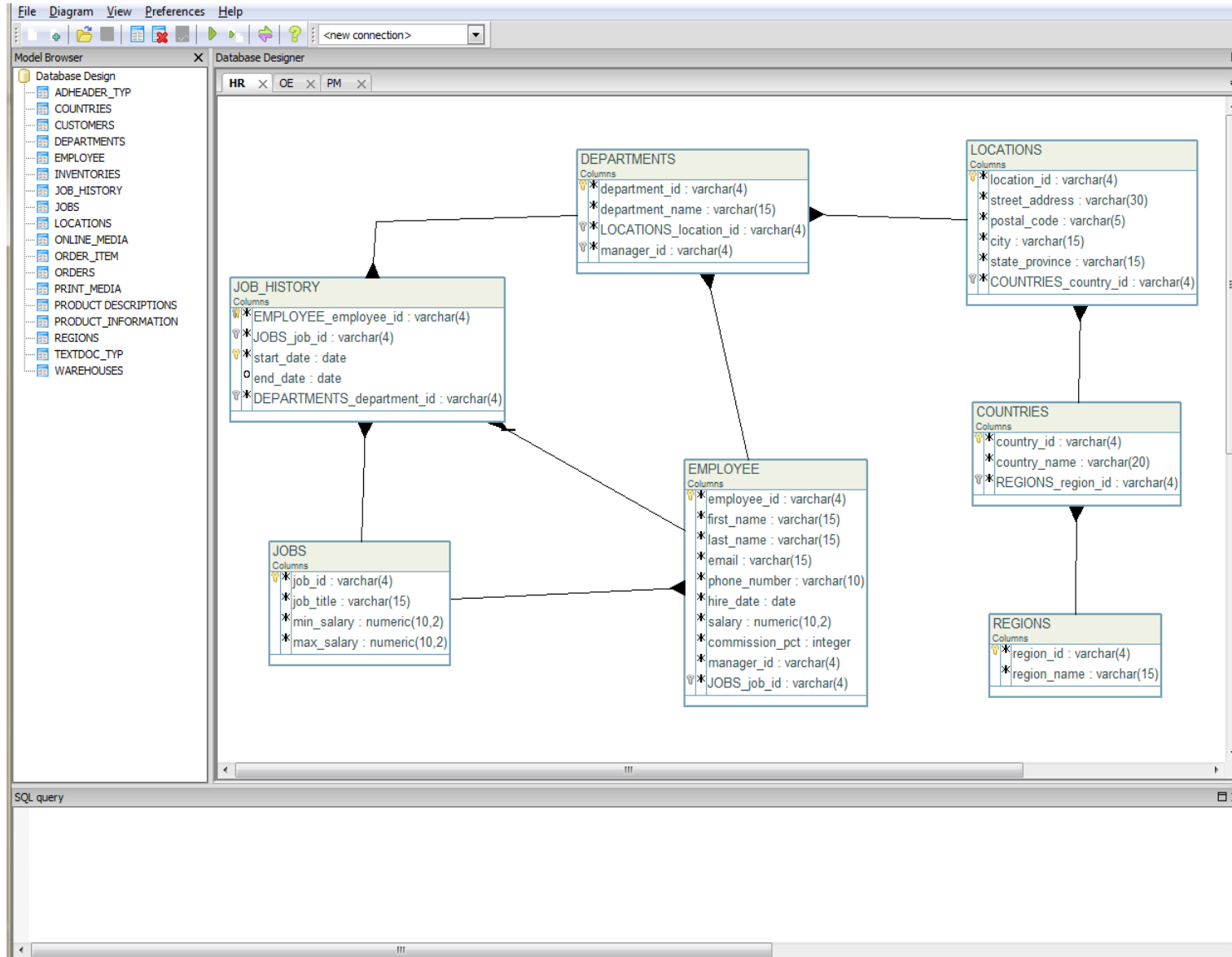
Interesting features

- 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

Menu

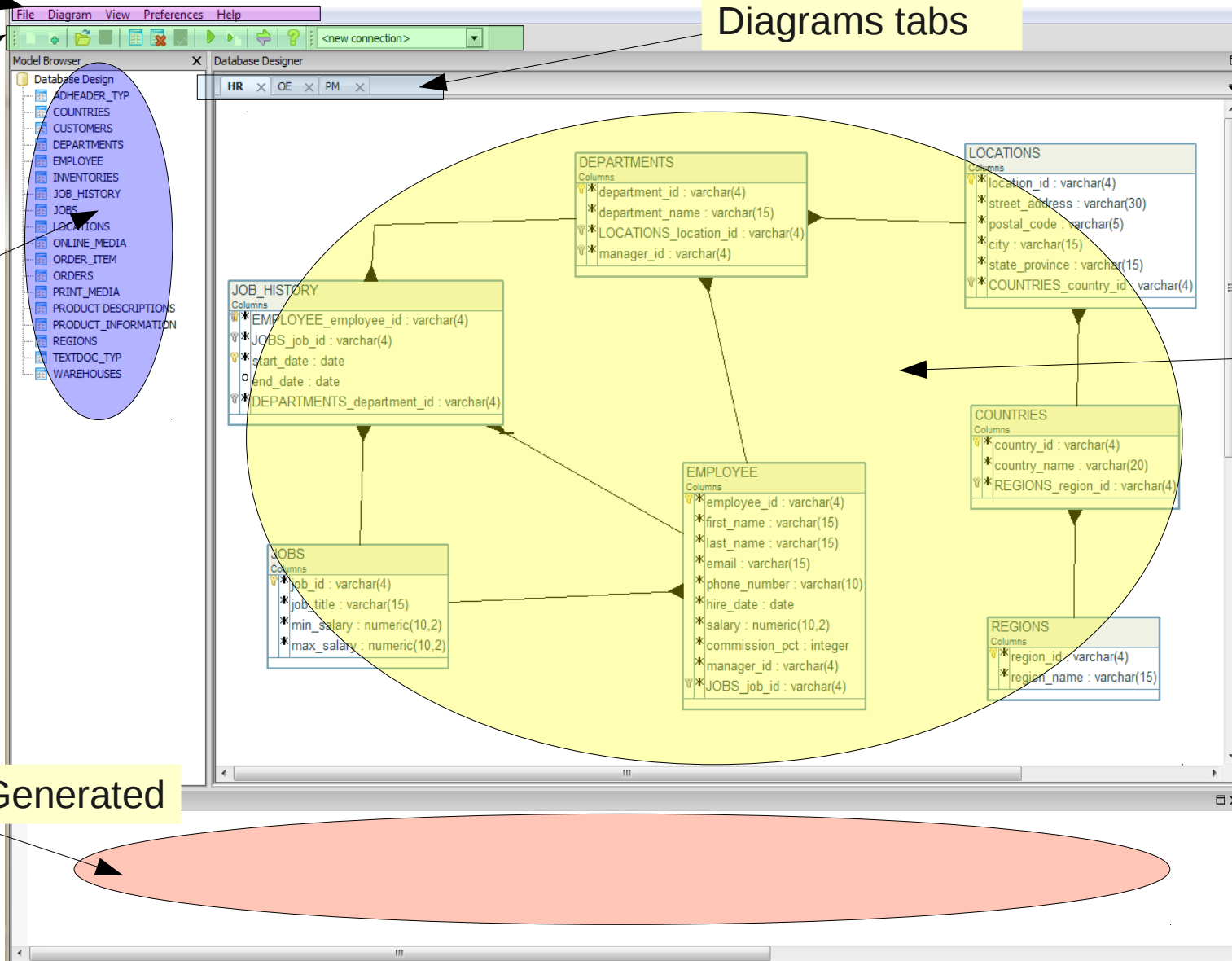
Toolbar

Model Tables

Diagrams tabs

Modeling Canvas

DDL Generated



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 important column metadatas:
 - Column name, datatype, optionality (Null or Not Null), constraints (Primary Key, Unique Key, Foreign Key,...)
- Have two possible states: Unselected or Selected.

EMPLOYEE (EMP)	
Columns	
* employee_id	: varchar(4)
* first_name	: varchar(15)
* last_name	: varchar(15)
* email	: varchar(15)
* phone_number	: varchar(10)
* hire_date	: date
* salary	: numeric(10,2)
* commission_pct	: integer
* manager_id	: varchar(4)
* JOBS_job_id	: varchar(4)

EMPLOYEE (EMP)	
Columns	
* employee_id	: varchar(4)
* first_name	: varchar(15)
* last_name	: varchar(15)
* email	: varchar(15)
* phone_number	: varchar(10)
* hire_date	: date
* salary	: numeric(10,2)
* commission_pct	: integer
* manager_id	: varchar(4)
* JOBS_job_id	: varchar(4)

Unselected Table State

Unselected State

Table Name





EMPLOYEE (EMP)	
Columns	
	* employee_id : varchar(4)
	* first_name : varchar(15)
	* last_name : varchar(15)
	* email : varchar(15)
	* phone_number : varchar(10)
	* hire_date : date
	* salary : numeric(10,2)
	* commission_pct : integer
	* manager_id : varchar(4)
	* JOBS_job_id : varchar(4)



Table optional short name (for auto fk naming)





Optionality:

-  Null
-  Not Null

Columns:
Name + Datatype
[+ length, precision]

Constraints:

-  Primary Key (pk)
-  Unique Key (uk)
(only when > 2)

-  Foreign Key from pk
-  Foreign Key from uk
-  Identifying Foreign Key from pk
-  Identifying Foreign Key from uk

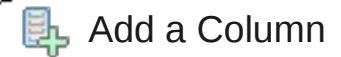
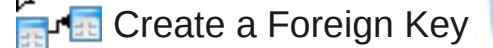
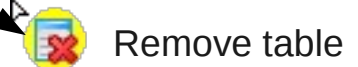
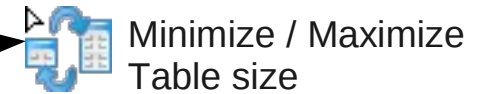
Selected Table State

- 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

Selected State

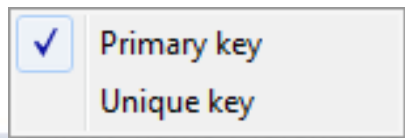
Double click sensitive area



Right click on option space

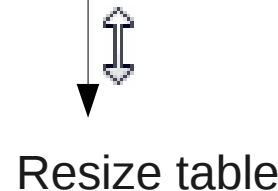
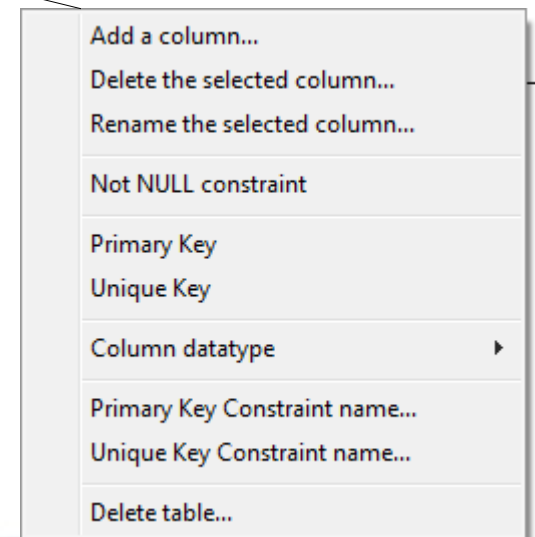


Right click on constraint space



EMPLOYEE (EMP)	
Columns	
<input checked="" type="checkbox"/>	employee_id : varchar(4)
<input checked="" type="checkbox"/>	first_name : varchar(15)
<input checked="" type="checkbox"/>	last_name : varchar(15)
<input checked="" type="checkbox"/>	email : varchar(15)
<input checked="" type="checkbox"/>	phone_number : varchar(10)
<input checked="" type="checkbox"/>	hire_date : date
<input checked="" type="checkbox"/>	salary : numeric(10,2)
<input checked="" type="checkbox"/>	commission_pct : integer
<input checked="" type="checkbox"/>	manager_id : varchar(4)
<input checked="" type="checkbox"/>	JOBS_job_id : varchar(4)

Right click on a column



Modifying table size: Scrollbar

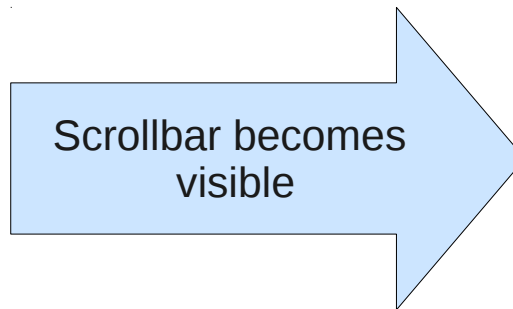
EMPLOYEE (EMP)

Columns

- * employee_id : varchar(4)
- * first_name : varchar(15)
- * last_name : varchar(15)
- * email : varchar(15)
- * phone_number : varchar(10)
- * hire_date : date
- * salary : numeric(10,2)
- * commission_pct : integer
- * manager_id : varchar(4)
- * JOBS_job_id : varchar(4)

A dashed yellow oval highlights the bottom edge of the table definition window, with a double-headed vertical arrow pointing downwards from it.

When a table size is changed



EMPLOYEE (EMP)

Columns

- * employee_id : varchar(4)
- * first_name : varchar(15)
- * last_name : varchar(15)
- * email : varchar(15)
- o phone_number : varchar(10)
- * hire_date : date
- 1 2 * salary : numeric(10,2)
- 1 * commission_pct : integer
- 1 * manager_id : varchar(4)

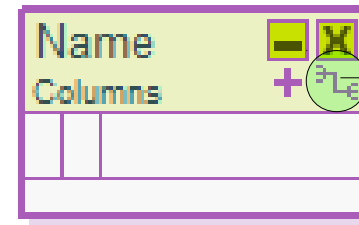
A vertical scrollbar is visible on the right side of the table definition window, with a black arrow pointing to it from the text 'Scrollbar' below.

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



Foreign Key Handle

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

Select Destination table of foreign key

EMPLOYEE	
Columns	
* employee_id	: varchar(4)
* first_name	: varchar(15)
* last_name	: varchar(15)
* email	: varchar(15)
* phone_number	: varchar(10)
* hire_date	: date
* salary	: numeric(10,2)
* commission_pct	: integer
* manager_id	: varchar(4)
* JOBS_job_id	: varchar(4)

Select source of fk

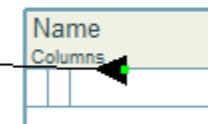
Select Foreign Key Mapping for each Column

From Primary

employee_id Automatically Generated

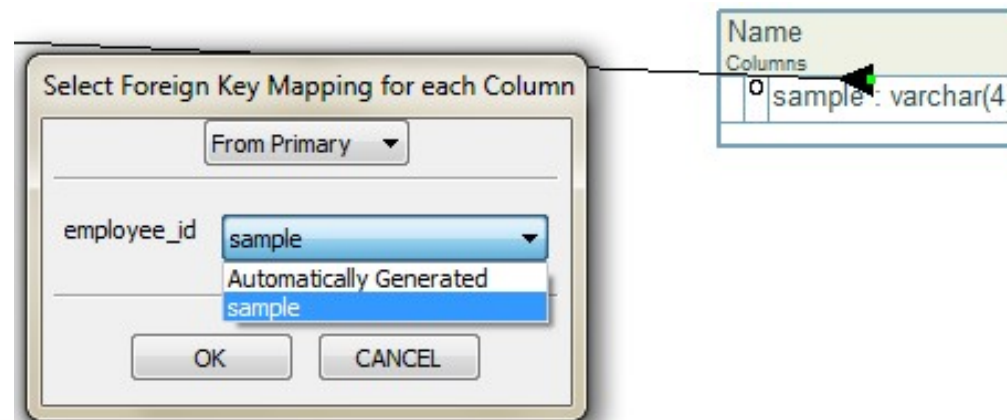
OK CANCEL

Select destination:
Automatically generated
Column or existing one at
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.



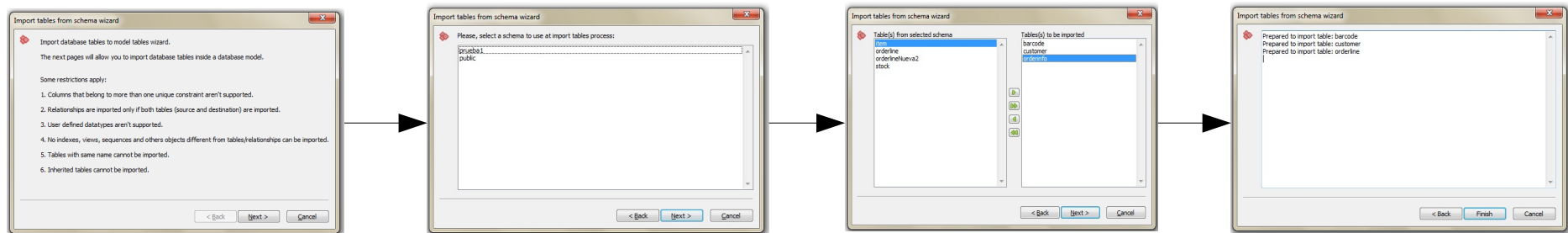
Database Designer Toolbar



- ↑ Create a new model (New file with a model)
- ↑ Create a new diagram inside a model.
- ↑ Open an existing model file
- ↑ Save a model to a file / Save changes inside a model
- ↑ Add an empty table to the current diagram into a model.
- ↑ Delete selected table(s)/relationship(s)
- ↑ Generate DDL for model (wizard)
- ↑ Generate DDL for current model (wizard)
- ↑ Import Wizard
- ↑ Select connection

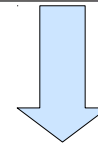
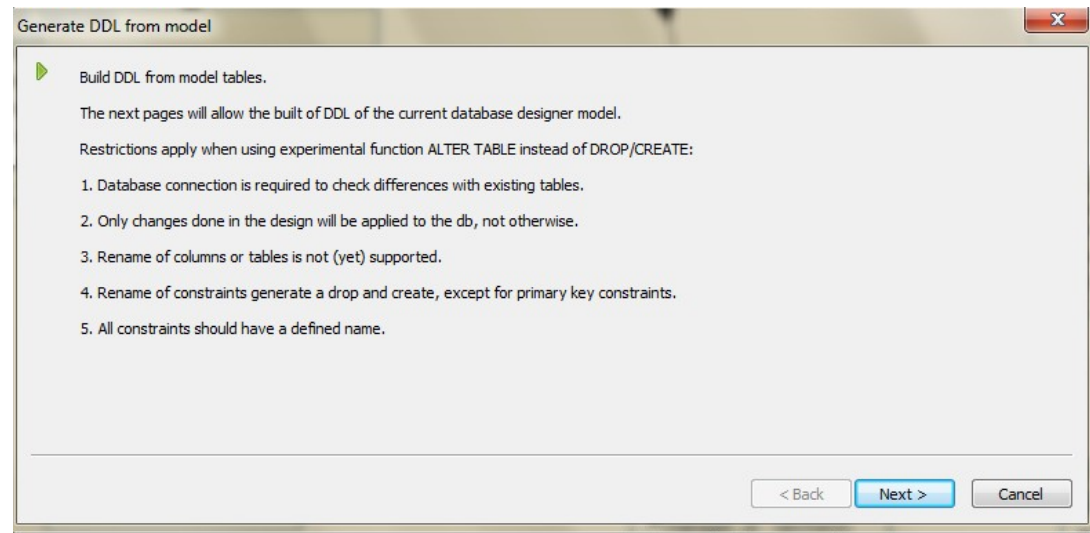
Import Tables

- 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)

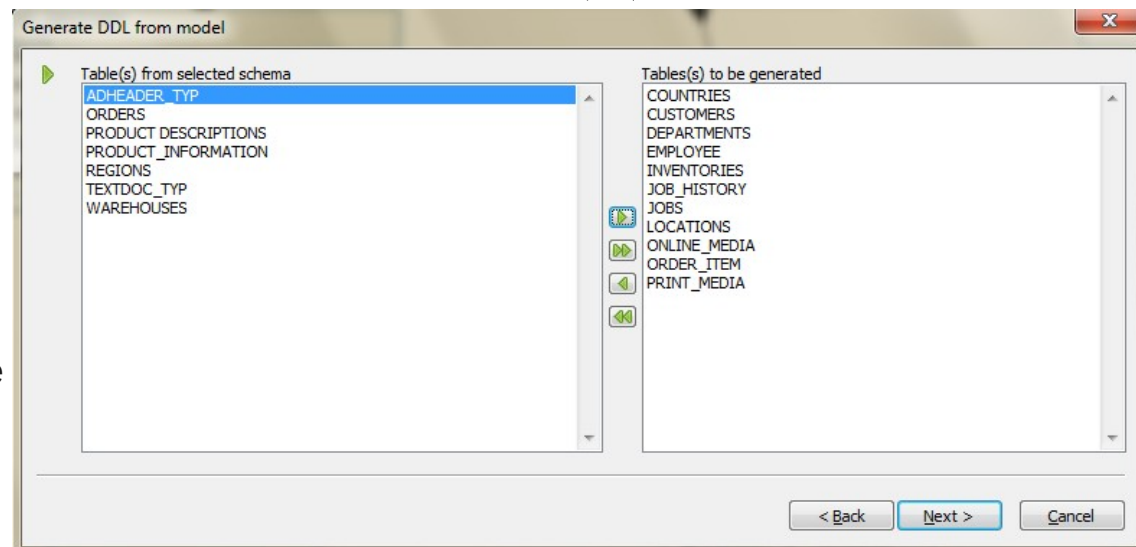


Generate DDL from model

Initial Screen of Wizard

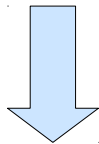
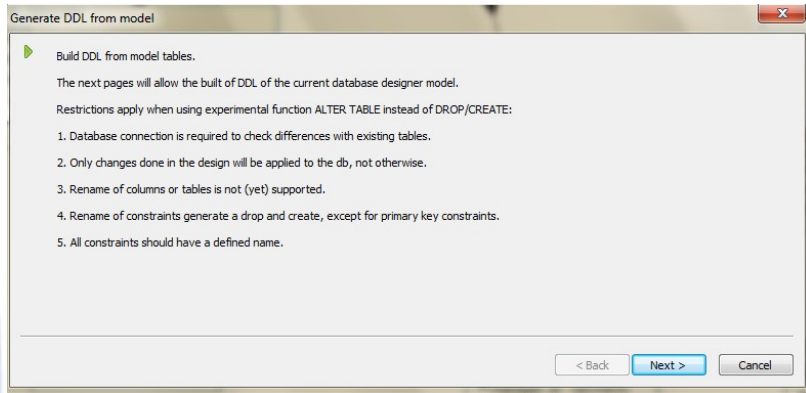


Select tables to be generated:
The user is responsible for any
inconsistency in the generated DDL code

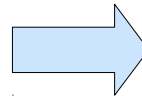
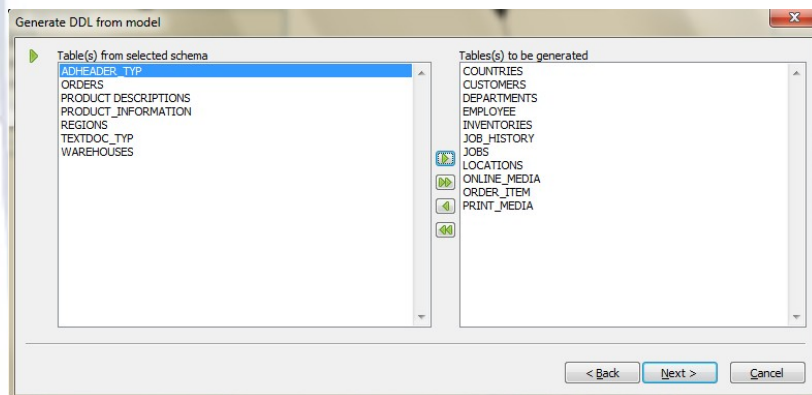


Generate DDL from model

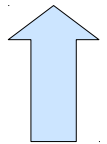
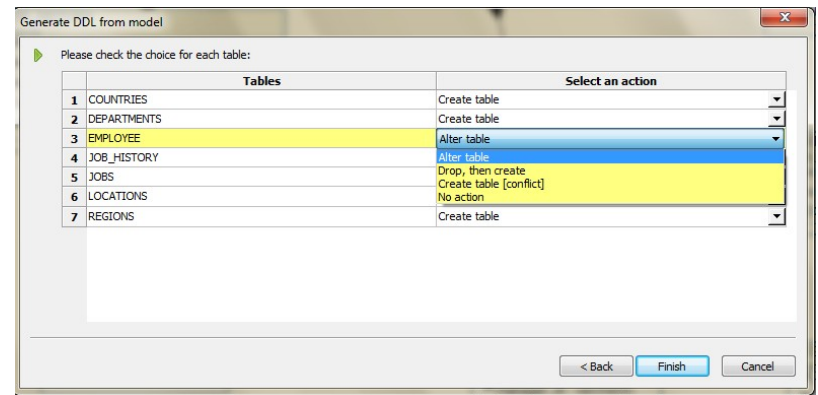
Initial Screen of Wizard



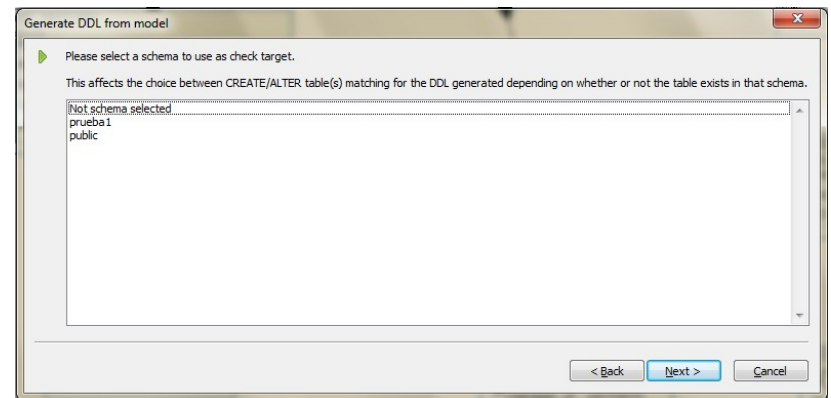
Select tables to be generated:
The user is responsible for any
inconsistency in the generated DDL code



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



Select schema to use as target



Generate DDL from model

The screenshot displays the Oracle Database Designer interface. On the left, the 'Model browser' shows a list of tables including ADHEADER_TYP, COUNTRIES, CUSTOMERS, DEPARTMENTS, EMPLOYEE, INVENTORIES, JOB_HISTORY, JOBS, LOCATIONS, ONLINE_MEDIA, ORDER_ITEM, ORDERS, PRINT_MEDIA, PRODUCT DESCRIPTIONS, PRODUCT_INFORMATION, REGIONS, TEXTDOC_TYP, and WAREHOUSES. The main workspace shows a database model with the following tables and their columns:

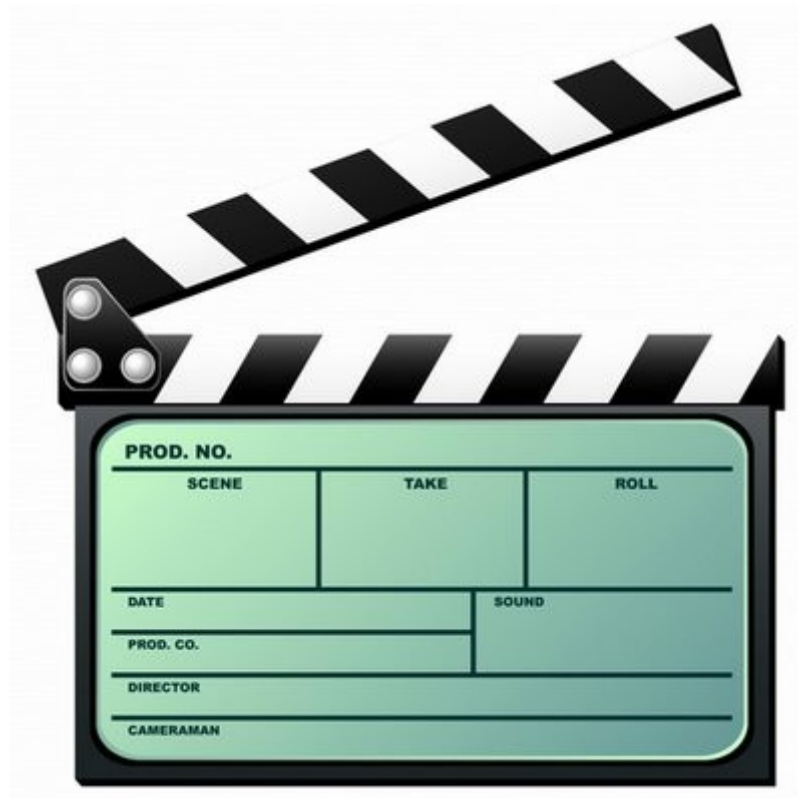
- DEPARTMENTS**: department_id (varchar(4)), department_name (varchar(15)), LOCATIONS_location_id (varchar(4)), manager_id (varchar(4)).
- LOCATIONS**: location_id (varchar(4)), street_address (varchar(30)), postal_code (varchar(5)), city (varchar(15)), state_province (varchar(15)), COUNTRIES_country_id (varchar(4)).
- COUNTRIES**: country_id (varchar(4)), country_name (varchar(20)), REGIONS_region_id (varchar(4)).
- EMPLOYEE**: employee_id (varchar(4)).
- JOB_HISTORY**: EMPLOYEE_employee_id (varchar(4)), JOBS_job_id (varchar(4)), start_date (date), end_date (date), DEPARTMENTS_department_id (varchar(4)).

The SQL query window at the bottom shows the following DDL code:

```
--  
-- Generating Create sentence(s) for table(s)  
--  
  
CREATE TABLE "prueba1"."REGIONS" (  
  "region_id" varchar(4) NOT NULL ,  
  "region_name" varchar(15) NOT NULL  
);  
  
CREATE TABLE "prueba1"."LOCATIONS" (  
  "location_id" varchar(4) NOT NULL ,  
  "street_address" varchar(30) NOT NULL ,  
  "postal_code" varchar(5) NOT NULL ,  
  "city" varchar(15) NOT NULL ,  
  "state_province" varchar(15) NOT NULL ,  
  "COUNTRIES_country_id" varchar(4) NOT NULL  
);  
  
CREATE TABLE "prueba1"."JOBS" (  
  "job_id" varchar(4) NOT NULL ,  
  "job_title" varchar(15) NOT NULL ,  
  "min_salary" numeric(10,2) NOT NULL ,  
  "max_salary" numeric(10,2) NOT NULL
```

DDL

Demo



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

