

Notas del curso de Introducción a la Inteligencia de Negocios

Sara Zambrano Gaona

Contenido

Qué es la Inteligencia de Negocios	5
Definición	5
Antecedentes históricos.....	5
Objetivos	7
Ventajas de la implementación de un esquema de BI en una empresa	7
Ciclo de los sistemas de Inteligencia de Negocios	8
Ciclo del desarrollo de un proyecto de Inteligencia de Negocios	9
El impacto del BI en las empresas	10
Data Warehouses y Datamarts	11
Qué es una ODS.....	11
Qué es un Almacén de Datos o Data Warehouse	11
Qué es un Datamart	12
Qué son los Metadatos	13
Diseño de un Data Warehouse	14
Mapeo lógico.....	15
Modelo dimensional	18
Diferencias entre el modelo dimensional y el modelo entidad-relación de bases de datos	21
Diseño de cubos	24
PostgreSQL para la construcción de Data Warehouses y Datamarts	26
Antecedentes históricos.....	26
Características de PostgreSQL.....	27
Instalación de PosgtreSQL.....	27
Instalación de PostgreSQL para distribuciones Linux desde archivos fuente (para compilar) .	28
Instalación de PostgreSQL para distribuciones Linux desde archivos compilados para distribuciones Linux de la familia Red Hat	29
Instalación de PostgreSQL desde archivos compilados para distribuciones Linux de la familia Ubuntu	29
Instalación de PostgreSQL utilizando el instalador	30
Interfaces de PostgreSQL	38
Psql	38
Comandos básicos de psql	38

pgAdmin	39
Sentencias de SQL Básicas.....	45
CREATE DATABASE	46
CREATE TABLE	46
CREATE INDEX	46
ALTER TABLE.....	47
DROP TABLE	47
DROP INDEX	48
INSERT INTO	48
UPDATE	48
DELETE.....	48
SELECT	49
Subconsultas	49
Tipos de join	50
Dentro de las sentencias SQL podemos incluir además:.....	50
¿PostgreSQL es adecuado para implementar en él un Datawarehouse?	53
Herramientas para proyectos de Inteligencia de Negocios	54
Open Source BI suites:	54
BI Suites de código cerrado:.....	54
Talend Open Studio.....	55
Eclipse BIRT	55
JasperSoft.....	55
Palo.....	55
SpagoBi.....	56
CloverETL.....	56
Pentaho	56
Prerrequisitos.....	58
Linux distribuciones Red Hat.....	58
Linux distribuciones Ubuntu	58
Windows.....	58
BI SERVER CE	61
PDI	63

PRD	65
WEKA.....	66
MONDRIAN.....	68
PENTAHO.....	69
Pentaho Data Integration.....	69
Pentaho Data Integration.....	69
Extracción, Transformación y Carga con el Pentaho Data Integration	70
Descripción del proceso ETL.....	70
Extract (Extracción)	70
Transform (Transformación)	71
Load (carga).....	74
Comenzando a trabajar con el Pentaho Data Integration	75
Configuración del repositorio.....	75
Menús.....	79
Conexiones.....	83
Transformaciones.....	85
Hops.....	86
Ejercicios de transformaciones	87
Trabajos (Jobs).....	90
Ejercicios de Jobs.....	91
Mapeos.....	93
Ejemplos de mapeo	94
Pan: Ejecución externa de transformaciones.....	96
Kitchen: Ejecución externa de trabajos.....	96
Objetivos del Datamart a diseñar:	98
Introducción al Pentaho Report Designer.....	104
Creando un reporte con el apoyo del asistente.....	109
Introducción a Mondrian	128
Pentaho Schema Workbench.....	128
Descripción del ambiente de trabajo del Schema Workbench.....	128
Cómo trabajar con el Schema Workbench.....	130
Creación de dimensiones compartidas.	132

Creación de Cubos.....	133
Publicación del Esquema y utilización desde el portal BI	135
Introducción al Dahsboard de Pentaho	137
Cuadro de Mando Integral	137
CDE de Pentaho.....	138
Diseño de un dashboard	138
Ejemplo.....	142
Bibliografía sugerida.....	146

Qué es la Inteligencia de Negocios

Definición

Llamamos **Business Intelligence o Inteligencia de Negocios (BI)** al conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa que sirva de respaldo para la toma de decisiones.



Antecedentes históricos

La raíz conceptual de la Inteligencia de Negocios está resumida en esta cita del Arte de la Guerra del filósofo y militar chino Sun Tzu (400-320 A.C.):



“... para poder ser exitoso en la guerra se deben conocer completamente las fortalezas y debilidades de uno mismo, así como las del enemigo. El no saber alguna de ellas podría significar la derrota.”

Sun Tzu aconseja: “Si quieres estar seguro de ganar, ataca un lugar que tu enemigo no defienda.”

El “terreno” hay que entenderlo como el “mercado”, que además de los competidores está integrado por clientes, sistemas comerciales, etc. El “clima” se identifica como el “entorno” donde están presentes las tendencias tecnológicas, económicas, políticas, sociales, medioambientales, regulaciones, etc.

Asimilando esto al campo de los negocios, los especialistas del “management” reiteran que, para formular sus estrategias, las empresas deben conocer con profundidad, la situación de los mercados, las fortalezas y debilidades de sus competidores, así como tener un diagnóstico preciso sobre sus fuerzas y debilidades internas. Es en este marco que las herramientas informáticas adquieren un gran valor en el análisis y generación del “autoconocimiento” de la empresa dentro de la disciplina de la *Inteligencia de negocios*.

La primera ocasión en que alguien usó el concepto de Inteligencia de negocios fue en 1865 cuando Richard Millard Devens en su “Encyclopedia of Anecdotes Comerciales y de Negocios” lo utilizó para referirse a los beneficios obtenidos por el banquero Sir Henry Furnese al actuar utilizando información de su entorno para anticiparse a sus competidores.

Sin embargo, antes de la era informática no había manera de automatizar la obtención y el manejo de la información, por lo que la toma de decisiones se basaba sólo en la intuición y experiencia del empresario.

Al inicio de la automatización los datos se hicieron más asequibles, pero obtenerlos era complicado debido a que no se poseía la tecnología para el intercambio de información dada la incompatibilidad de los sistemas. Los reportes tardaban meses en generarse.

En 1958, en un artículo, el Gerente de recuperación de información e investigador de IBM Hans Peter Luhn usó el término BI (Business Intelligence) para referirse a: “La capacidad de comprender las interrelaciones de los hechos presentados en tal forma como para orientar la acción hacia una meta deseada”.

Durante las décadas de 1960 hasta 1980 se desarrollaron los primeros Sistemas de Soporte a las Decisiones (DSS).

En 1989, Howard Dresner, investigador del Gartner Group popularizó el acrónimo BI (Business Intelligence) para englobar el conjunto de conceptos y métodos para mejorar la toma de decisiones en los negocios utilizando sistemas de apoyo basados en hechos.

El concepto ha evolucionado para incluir una gran cantidad de metodologías, aplicaciones y tecnologías para la compilación, acceso, transformación y análisis de datos; su explotación directa (reportes y consultas) o su conversión en conocimiento (minería).

Objetivos

La Inteligencia de Negocios facilita la toma de decisiones a cualquier nivel: Estratégico, Táctico u Operativo; gracias a que permite la extracción de datos, su análisis, así como la ejecución de búsquedas de manera eficiente y rápida para identificar tendencias y patrones.

En los negocios actualmente los retos que se presentan están relacionados con obtener los datos que sean necesarios, el identificar patrones y darles significado a los datos (generación de información), así como planear la estrategia a seguir con la información obtenida.



Pero BI también involucra poseer un conocimiento profundo de todos los factores que afectan a una organización tanto interna como externamente. Estos factores incluyen a clientes, competidores, socios de negocio, ambiente económico y operaciones internas.

Las metodologías y el uso de tecnologías junto con el conocimiento del negocio en conjunto sirven de guía para la toma de decisiones acertadas y efectivas, y al mismo tiempo son el origen de las ventajas competitivas.

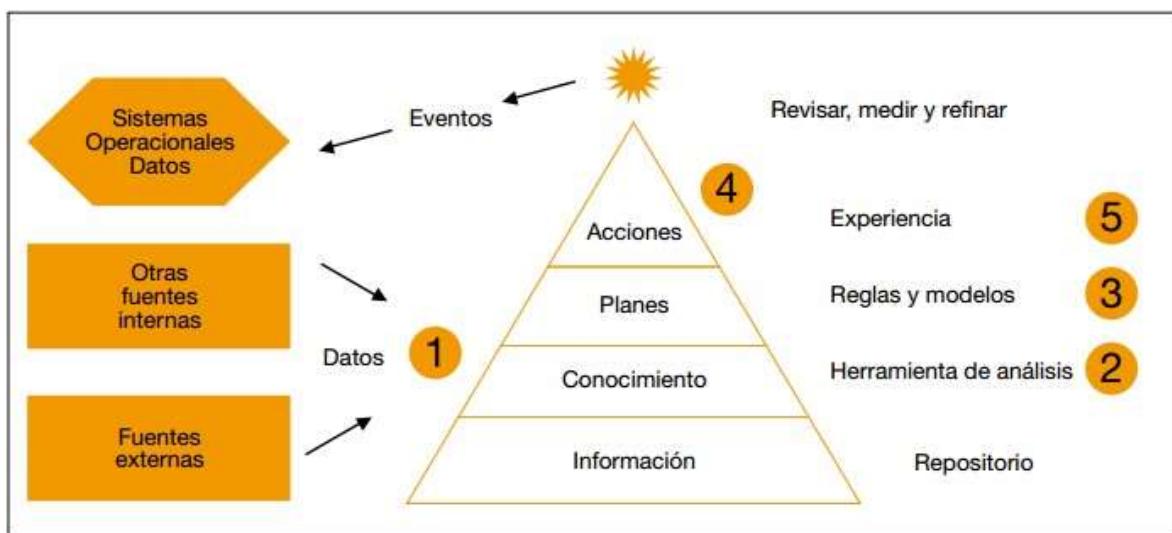
Ventajas de la implementación de un esquema de BI en una empresa

- Permite disponer de la información correcta y oportuna para la toma de decisiones sin tener que acudir a diferentes áreas porque concentra y almacena todos los datos en un único sitio para su extracción sencilla y eficiente.

- Permite evaluar distintos escenarios a la vez, con lo que es posible analizar posibles afectaciones en el negocio y revertir tendencias negativas.
- Se pueden definir indicadores para medir el desempeño del negocio.
- Permite agrupar información de diferentes procedencias en un solo reporte, lo que facilita calcular el impacto de un cambio de políticas o la reorientación de estrategias.
- Agiliza la capacidad de reacción a situaciones imprevistas con un nivel de riesgo menor. Con un análisis previo de los escenarios posibles se pueden planear las acciones a seguir con un conocimiento del riesgo que involucra aplicar dichas medidas.
- Se puede dar seguimiento al impacto de las decisiones tomadas lo cual retroalimenta nuestro conocimiento del negocio, y permite adquirir experiencia para futuros escenarios parecidos.

Ciclo de los sistemas de Inteligencia de Negocios

El ciclo de los sistemas de BI asemejan a una “refinería de datos”, es decir, están diseñados para admitir una materia prima (datos) y procesar una multiplicidad de productos de información:



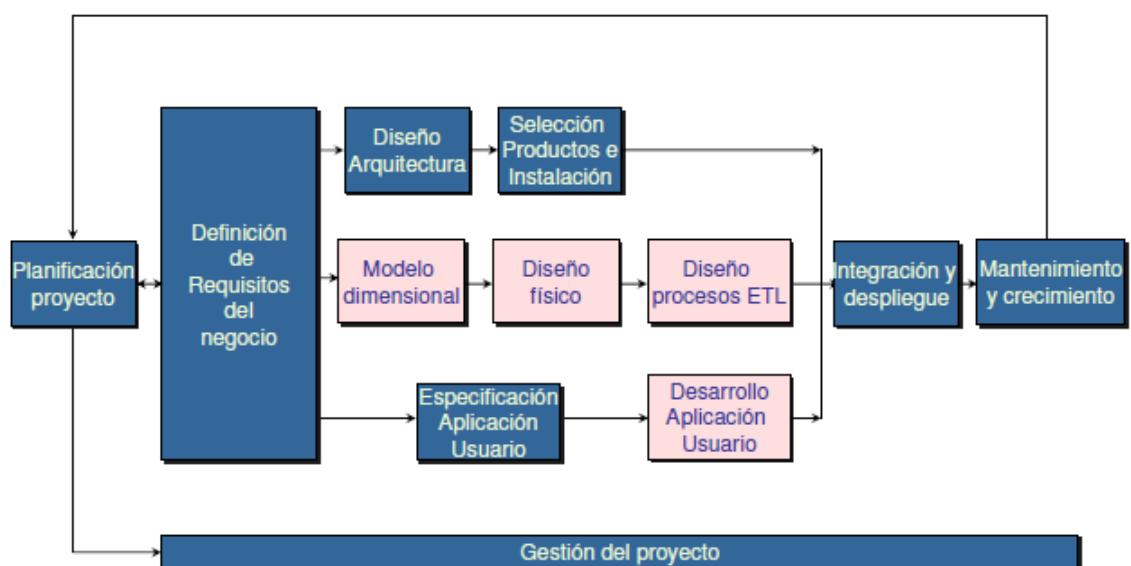
- Una herramienta **ETL (Extract, Transform & Load)** extrae los datos de múltiples fuentes o sistemas operacionales, los limpia, les da formato, los integra y los almacena en un **Data Warehouse** (Almacén de datos). La información debe ser transformada para su posterior utilización.
- Se selecciona la información útil y se definen los **indicadores** que nos sirven para el propósito de la empresa. Un indicador debe reflejar qué se quiere medir y tener una definición clara y concisa de cómo será calculado.
- Las herramientas de análisis (herramientas de consulta, OLAP y de minería de datos) acceden a la información y la procesan para identificar **patrones y excepciones**.

- Una vez descubiertas las reglas sobre las tendencias y patrones detectados se presentan en **informes** claros y concisos para su fácil entendimiento por parte de quienes tienen el poder de tomar las decisiones.
- Se pasa de las reglas a los planes de acción implementando las respuestas a los patrones encontrados, como por ejemplo: crear algoritmos que cambien dinámicamente los precios de los productos en respuesta a las condiciones del mercado u optimizar las rutas del transporte de mercancías, así como monitorear el movimiento periódico y no sólo las existencias de un producto e indicar cuando adquirir más.
- Se analizan los indicadores para medir los resultados de las acciones implementadas y retroalimentar el sistema.

Ciclo del desarrollo de un proyecto de Inteligencia de Negocios

Para recoger los datos de la organización y transformarlos en información útil es necesario hacer un plan de implementación de un proyecto de BI, lo cual implica:

- Comprender la naturaleza del negocio, establecer cuáles son los indicadores que requerimos y cómo obtenerlos.
- Diseñar la estructura de la base de datos de la cual se generarán los reportes: el almacén de datos.
- Recuperar los datos de los sistemas operacionales, depurarlos, transformarlos y cargarlos en la estructura de datos diseñada: procesos ETL (Extract, Transform & Load).
- Aplicar las herramientas para el análisis de los datos para obtener los indicadores y generar los reportes correspondientes.
- Es importante considerar también los tiempos de respuesta, la integración, la seguridad, la navegación, el entorno gráfico y la elección de las herramientas.



Este es el modelo de Kimbal para el ciclo de vida (KLC) de un proyecto de BI.

Las aplicaciones de BI son la cara visible de la inteligencia de negocios: los informes y aplicaciones de análisis proporcionan información útil a los usuarios. Las aplicaciones de BI incluyen un amplio espectro de tipos de informes y herramientas de análisis, que van desde informes simples de formato fijo a sofisticadas aplicaciones analíticas que usan complejos algoritmos e información del dominio.

Kimball divide a estas aplicaciones en dos categorías basadas en el nivel de sofisticación, y les llama informes estándar y aplicaciones analíticas.

El impacto del BI en las empresas

En marzo del 2007, la empresa SAS realizó un estudio para evaluar las tendencias actuales en el uso de información de negocios en particular sobre la inteligencia de negocios y el efecto que se ejerce en el desempeño de las compañías. Las conclusiones indican que la implementación de prácticas para aprovechar al máximo la información empresarial está íntimamente relacionada con un mejor desempeño de la compañía, pero también está claro que muchas organizaciones todavía tienen problemas con el manejo de su información.

A pesar de la frecuencia con la que se emplea la tecnología para administrar información, principalmente software de inteligencia de negocios, menos del 25 por ciento de los encuestados indicaron que su área de administración se basa en la inteligencia de negocios para tomar decisiones. Pero incluso más importante, casi el 80 por ciento de las organizaciones no han puesto en marcha de manera integral prácticas para garantizar la calidad de los datos.

El estudio también muestra que las organizaciones que son más eficaces para usar su información obtienen mejores resultados. Aunque las diferencias no son muy grandes, sí son constantes.

El crecimiento de la inteligencia de negocios en una organización ha dado como resultado un mercado de software de 5 mil quinientos millones de dólares.

Data Warehouses y Datamarts

Qué es una ODS

Una ODS (Operational Data Base) es una base de datos temporal que sirve para efectuar las operaciones de transformación previas a pasar los datos a un Data Warehouse, contiene operaciones atómicas de un periodo relativamente corto de tiempo, en ella se hacen todas las operaciones y validaciones necesarias, lo que permite mantener una actualización constante del DWH sin afectar su operación.

Qué es un Almacén de Datos o Data Warehouse

Un almacén de datos (data warehouse, DW) según Inmon, es una colección de datos orientada a undeterminado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata, sobre todo, de un historial completo de la organización.

Por otra parte, Kimball la define como “una copia de los datos transaccionales estructurados específicamente para consultas y análisis”.

Desde el punto de vista arquitectónico, la mayor diferencia entre los dos autores es el sentido de la construcción del DW, esto es comenzando por los Data marts o ascendente (Bottom-up, Kimball) o comenzando con todo el DW desde el principio, o descendente (Top-Down, Inmon).

Además, la metodología de Inmon se basa en conceptos bien conocidos del diseño de bases de datos relacionales, al contrario de la de Kimball que se basa en un modelado dimensional.

Un Data Warehouse (DWH almacén de datos) es un sistema que recibe y consolida datos de manera periódica desde sistemas fuente hacia un almacén de datos dimensional. Un DWH está diseñado para expresamente para optimizar la generación de reportes y el análisis (Data mining).

La información objetivo del análisis usualmente se encuentra dispersa en diferentes formatos y sistemas, y al concentrar la información en un almacén de datos se obtiene una visión integral de las operaciones de una organización.

Propiedades de un DWH:

- Conserva los **datos históricos completos**. Con lo que pueden reconstruirse las variaciones de los datos a lo largo del tiempo.

- Es un expediente completo de la organización más allá de la información transaccional y operacional. Esto incluye a los “**metadatos**”.
- Se **actualiza por lotes** o descargas periódicas de acuerdo con las necesidades de información y la carga de volumen de los datos de la empresa.
- Los datos deben estar **integrados**, es decir, deben ser consistentes, homogéneos y fiables.
- La **información es no volátil**, es decir, nunca se modifica o elimina ningún dato almacenado, la información es de sólo lectura.
- Está conformado por unidades independientes llamadas **Datamarts** con información especializada por temas.
- Está **diseñado para su consulta** con fines de inteligencia de negocios y otras actividades de análisis, por lo que la información se estructura en diferentes niveles de detalle (OLAP).
- El diseño del DW debe ser **adaptable y resistente al cambio**.
- El acceso al DW debe ser **seguro** para resguardar la información confidencial de la organización.
- Sólo existe un **resultado** verdadero en el DW, sólo siendo **único** es posible que sirva como soporte para la toma de decisiones.
- El DW debe ser **aceptado** por quienes toman las decisiones dentro de la Institución para ser considerado exitoso.

Qué es un Datamart

Un Datamart es una base de datos departamental, es decir, especializada en el almacenamiento de los datos de un área de negocio específica. Se caracteriza por disponer la estructura óptima de datos para analizar la información al detalle desde todas las perspectivas que afecten a los procesos de dicho departamento. Un Datamart puede ser alimentado desde los datos de un Data Warehouse (ser un subconjunto del DWH), o integrar por sí mismo un compendio de distintas fuentes de información.

Un Datamart es un subconjunto de un DWH con información de un área específica del negocio de una organización. Puede ser dependiente del DWH o independiente.

Datamart OLAP

Se basa en cubos OLAP y se construye agregando las dimensiones e indicadores necesarios de cada cubo relacional.

Datamart OLTP

Son un extracto del DW adicionado con algunas mejoras en su rendimiento (índices, filtros, totales, etc.) se forman con tablas *report* o *fact-tables* reducidas.

Ambos modelos se caracterizan por manejar poco volumen de datos con rapidez, mediante consultas sencillas, con validación directa de la información y la fácil reconstrucción histórica de los datos.

Qué son los Metadatos

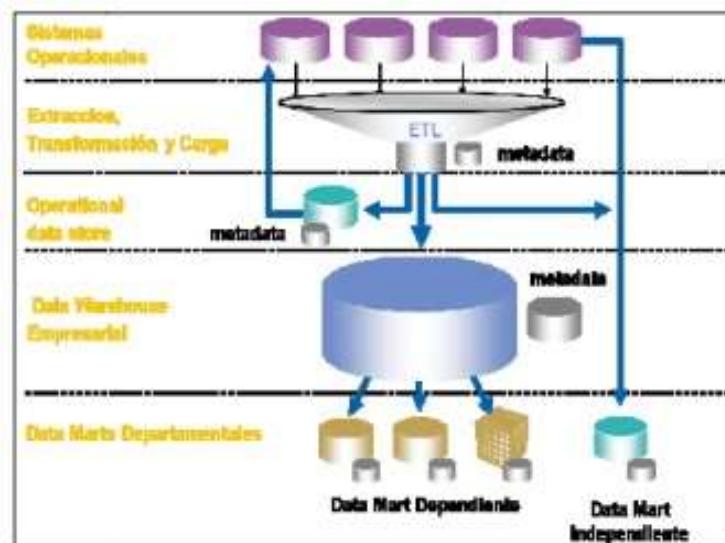
Literalmente son “datos sobre los datos”, es decir, se trata de datos que describen cuál es la estructura de los datos que se van a almacenar y cómo se relacionan.

El metadato documenta, entre otras cosas, qué tablas existen en una base de datos, qué columnas posee cada una de las tablas y qué tipo de datos se pueden almacenar. Los datos son de interés para el usuario final, el metadato es de interés para los programas que tienen que manejar estos datos.

La función de los metadatos es ayudar a ubicar datos.

En un Data Warehouse, los metadatos recogen todas las definiciones de la organización referente a:

- Tablas
- Columnas de tablas
- Relaciones entre tablas
- Jerarquías y dimensiones de los datos
- Entidades y relaciones



Diseño de un Data Warehouse

Data Profiling

El Data Profiling es el proceso de recopilación de estadísticas y otra información sobre los datos existentes en nuestros orígenes de información. Esta información va a ser de gran utilidad para el diseño de los procesos ETL. El Data Profiling también puede ser parte importante de cualquier iniciativa de calidad de datos, ya que antes de que la calidad de estos se pueda mejorar, habrá que establecer el estado actual de los datos.

Herramientas de Data Profiling:

Suelen incluir características de metadatos para recuperar la información del diccionario de datos de una BD, y previsualizar el contenido de las tablas de una forma rápida. Analizan los datos y las dependencias entre columnas y tablas.

- DataCleaner: Esta herramienta nos ayuda a ver si tenemos entradas duplicadas, si los datos de dirección, nombre, apellidos, correo electrónico, etc., tienen la estructura correcta, es decir, nos ayuda a encontrar discrepancias entre lo que deberíamos tener y lo que tenemos. Esta aplicación funciona con distintos tipos de plataformas y archivos de datos como son CSV, MS Excel (XLS y XLSX), MS Access, SAS, DBase, XML, Data, MonoDB, MySQL, Oracle y MS SQL Server.
- Talend Open Profiler: Se integra de tres módulos que permiten hacer el análisis de los datos, estadísticas sobre los valores de los campos, así como análisis de patrones predefinidos
 - Overview Analysis: analiza los esquemas de base de datos y proporciona información sobre tablas, filas, número de registros, índices, etc.
 - Table Analysis: analiza la definición de las tablas y verifica sus dependencias.
 - Column Analysis: realiza un análisis específico sobre el contenido de un campo. Para cada campo, se pueden seleccionar los indicadores de análisis (tales como valores estadísticos, número de registros, valores nulos, longitud mínima, longitud máxima, valores duplicados, etc.). Además, permite indicar patrones de validación sobre los campos (tanto con expresiones regulares como con sentencias deSQL).

Modelado de la Base de datos del Data Warehouse

Los Data Warehouses son bases de datos cuyo propósito es facilitar las consultas y el análisis para dar soporte a la toma de decisiones, lo cual implica que son diferentes a las bases de datos para

sistemas operacionales (que siguen el modelo Entidad-Relación) y por lo tanto su diseño debe seguir reglas distintas.

La información contenida en un DWH es resultado de transformaciones provenientes de las bases operacionales, sin embargo, las consultas que debe soportar un DWH son, en general, mucho más complejas. En los sistemas operacionales, por ejemplo, muchos valores pueden ser calculados a partir de otros datos para evitar redundancias, pero en un DWH es conveniente mantener los datos pre-calculados por cuestiones de performance. Otra diferencia es que los sistemas operacionales requieren estar actualizados constantemente, mientras que el DWH necesita incluir un elemento temporal que le permita mantener la información histórica completa.

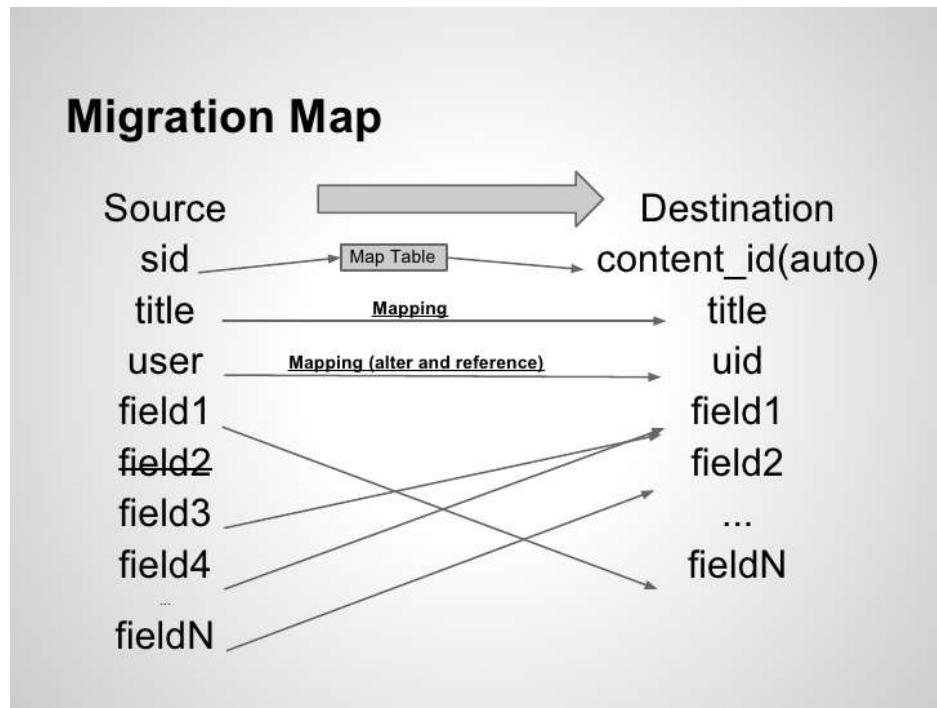
Existen dos tipos principales de arquitecturas que puede adoptar un DWH, la metodología de Kimball que consiste en construir el DWH de manera ascendente comenzando por la definición de los Datamarts, y la metodología de Inmon que plantea construir el DWH de manera descendente iniciando por todo el DWH partiendo del modelo relacional de bases de datos.

En particular, si se desea trabajar con cubos, el diseño del DWH requiere seguir los lineamientos definidos en el **modelo dimensional**. Con lo que el DWH puede adoptar un **diseño en estrella**, o bien, si se normaliza, adoptar un diseño **de copo de nieve o snowflakes**, o algún punto intermedio. El modelo de Kimball se basa en el modelo dimensional no normalizado.

La elección de la metodología a seguir dependerá de la definición del alcance de nuestro proyecto de BI, en una primera aproximación es conveniente comenzar con una extensión del modelo de Entidad-Relación e ir retroalimentándonos de la experiencia y luego hacer el diseño dimensional con la información que realmente nos va a ser de utilidad.

Mapeo lógico

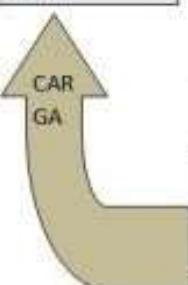
Antes de comenzar la construcción de un DW es necesario identificar correctamente el origen de cada dato, para ello vamos a elaborar un Mapa de Datos Lógico, en el cual estableceremos los criterios y las premisas que se deberán cumplir en todos los procesos de carga de datos al almacén.



Los elementos básicos del mapa lógico son:

- **Origen de datos** (tabla, columna, tipo de dato, tipo de objeto, criterios de datos a validar antes de considerarlos definitivos [calidad])
- **Destino** (se detallan las tablas y campos de las tablas de hechos y dimensiones, así como el tipo de dato)
- **Transformación a realizar** (toda operación sobre los datos: codificación/decodificación, sumas, tratamiento de cadenas, mapeo contra otras tablas, corrección de valores incorrectos, transformación de valores)

En la siguiente imagen se muestra una parte del mapeo lógico donde se explica la manera como se va a proceder una de las tablas del DWH.



Dim Producto	Atributo	Origen Datos	Tipo	Transformación	Observac.
Material (PK) Desc Material Familia Desc Familia Denominacion Origen Varietal Formato Venta Unidad Medida Litros Línea de Producto Desc Línea Producto Target	Material	MARA-MATNR	Char(18)	Conversion a Integer [7].	solo se utilizan 7 dígitos. Siempre numéricos. Estado en campo MARA-LVORM (una X es borrado). Fecha última modificación en MARA-LAEDA.
	Descripción Material	MAKT-MAKTX	Char(40)		Descripción dependiente del idioma (campo MAK-TSPRAS)
	Familia	MARA-MATKL	Char(9)	Conversion a Integer [6].	
	Descripción Familia	T023T-WGBEZ	Char(20)		Descripción dependiente del idioma (campo T023T-SPRAS)
	Denominac. Origen	T024X- LBTXT	Char(30)		A partir del campo MARA-LABOR se obtiene la denom. Origen en T024X. Depende del Idioma T024X-SPRAS.
	Varietal	MARA-FERTH	Char(18)		
	Formato Venta	MARA-NORMT	Char(18)		
	Unidad Medida	MARA-MEINS	Char(3)	Conversion a UM en Español, según tabla TD06A.	
	Litros	MARM-UMREN	Integer	Pasar a Litros, pues la conversión esta en ML.	Buscar en la tabla MARM la conversión de la unidad de medida base a la alternativa Millilitros (ML) y dividir por 1000.
	Línea de Producto	MARA-EXTWG	Char(18)	Conversion a Integer [6].	
	Descripción Línea Prod.	TWEWT- EWBEZ	Char(20)		
	Target	MARA-WRKST	Char(48)		

Fuera de las columnas básicas descritas, la cantidad de columnas y la información que se debe incluir en el mapeo es a criterio del diseñador, sin embargo, entre más completo y más detallado sea el mapeo se podrá hacer una migración más fiable de los datos al DWH, y detectar, en su caso, la procedencia de información corrupta o inconsistente.

Modelo dimensional

Una **medida** es un atributo (campo) de una tabla que se desea analizar, adicionando o agrupando sus datos, usando los criterios de corte conocidos como dimensiones. Las medidas habitualmente se vinculan con el nivel de granularidad y se encuentran las **tablas de hechos** (facts).

Las **tablas de dimensiones** tienen un conjunto de atributos (generalmente textuales) que brindan una perspectiva o forma de análisis sobre una medida en una tabla hechos.

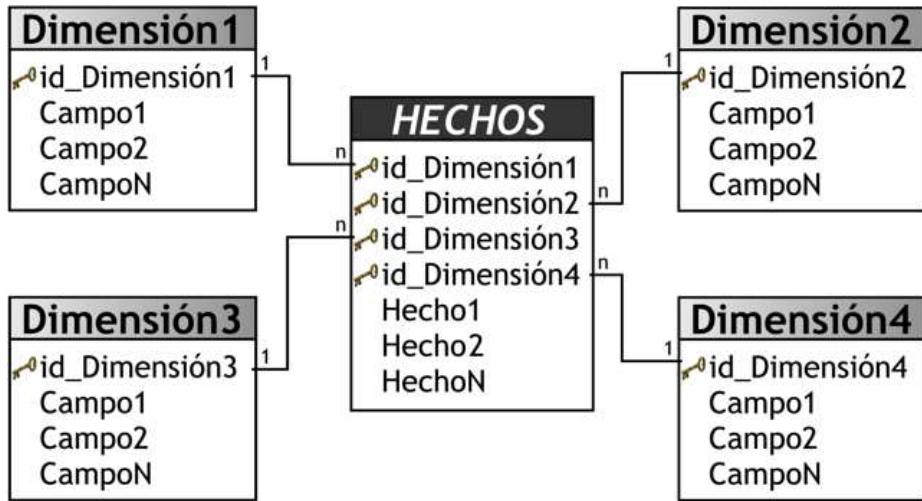
Del análisis de los requerimientos para construir el DWH se puede construir una matriz de procesos/dimensiones (Bus Matrix), por medio de la cual se pueden los datos numéricos a analizar, a estos datos les vamos a llamar medidas (measures). En esta matriz las filas representan los procesos del negocio y las columnas las dimensiones identificadas.

	Dimensiones					
Proceso de Negocio	Tiempo	Producto	Empleados	Ciudades (Revendedores)	Geografía de ventas	Importes
Proyección de ventas	X	X	X	X	X	X
Compras	X	X	X	X	X	X
Control de llamadas	X	X	X	X	X	
...						

El diseñador debe decidir qué nivel de fragmentación (**granularidad**) le va a corresponder a cada dimensión, tomando en consideración que los niveles de cada fragmento deben estar relacionados jerárquicamente. Por ejemplo, en la dimensión tiempo tenemos años, semestres, trimestres, meses, semanas, etc. De entrada se sugiere comenzar con el máximo nivel de granularidad para poder generar información con el mayor detalle posible.

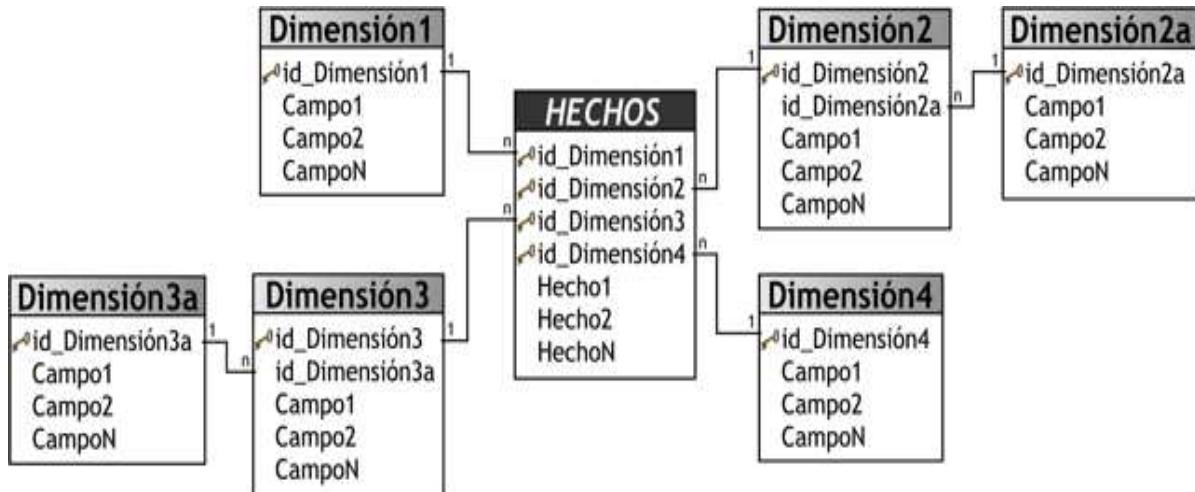
El diseñador debe decidir cuándo duplicar información de acuerdo a su estrategia de diseño.

La relación entre la tabla de hechos y sus respectivas tablas de dimensiones se puede visualizar con una estructura en forma de estrella: una tabla central (hechos) y un conjunto de tablas que la atienden radialmente (dimensiones).



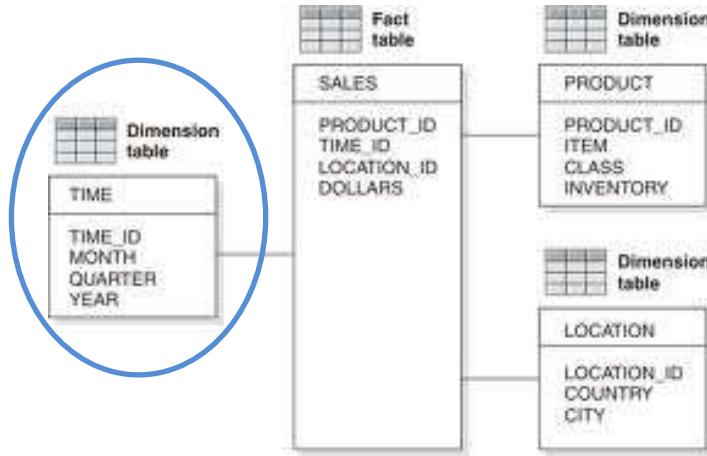
1 Esquema de estrella

Si es que nuestra base de datos está desnormalizada completamente; o bien como una estructura de copo de nieve (snowflake) si las dimensiones están normalizadas.



2 Esquema de copo de nieve

Salvo para casos particulares de dimensiones invariantes en el tiempo, debe considerarse siempre la existencia de la **dimensión tiempo**, su importancia radica en que nos va a permitir la reconstrucción histórica de los eventos en nuestro DWH.



Los campos llave en el modelo dimensional no pueden ser compuestos, sino que son campos tipo numérico (entero autoincremental o serial), y se les denomina **llaves subrogadas**.

Se definen nuevos campos llave en el DWH para evitar dependencias de los campos llave de los sistemas origen ya que en principio el almacén de datos se va a alimentar de diferentes fuentes (cada una con sus propias llaves definidas), y si además en éstos eventualmente se cambia la lógica de sus campos llave, se impactaría directamente al DWH. El utilizar campos llave seriales además agiliza las búsquedas y ahorra memoria con respecto de las llaves semánticas, sin mencionar que las bases de datos los generan automáticamente por defecto.

Cuando diseñamos un DWH completo, es posible que las tablas de hechos compartan tablas de dimensiones, es decir, tenemos **dimensiones conformadas**, pero para ello es preciso que toda dimensión signifique lo mismo para cada tabla de hechos con la que se relaciona.

Con respecto a los nombres de los campos hay que procurar que sean lo más descriptivos posible sin usar abreviaturas ni claves.

Diferencias entre el modelo dimensional y el modelo entidad-relación de bases de datos

Una vez que hemos visto las propiedades del modelo dimensional, podemos compararlo con las bases de datos relacionales.

Definición de base de datos

Conjunto de datos pertenecientes a un mismo contexto almacenados sistemáticamente.

Ejemplo: una biblioteca

Se pueden clasificar por sus cambios o movimientos

- Estáticas (Sólo lectura. Uso: proyección y análisis)
- Dinámicas (Cambian constantemente, se actualizan, se ingresan datos, se borran. Uso: manipulación y consulta)

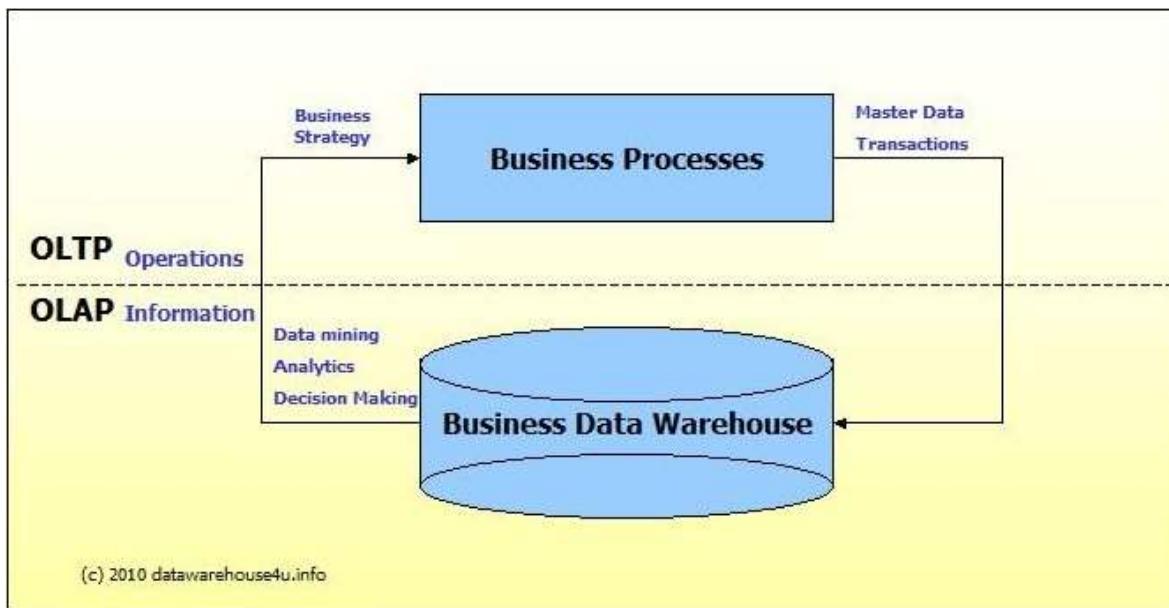
Se pueden clasificar por su arquitectura

- Jerárquicas y de red (obsoletas)
- Transaccionales (optimizan operaciones)
- Relacionales (normalización)
- Multidimensionales (hechos y dimensiones)

Características del modelo relacional	Características del modelo dimensional
<ul style="list-style-type: none">• Catálogos• Llaves primarias• Llaves foráneas• No admite redundancia• Sistemas operacionales	<ul style="list-style-type: none">• Tablas de dimensiones• Tablas de hechos• Llaves subrogadas• Admite redundancia• Data Warehouses y Data Marts

OLTP (On-Line Transactional Processing)

Los sistemas OLTP son bases de datos orientadas al procesamiento de transacciones. Una transacción genera un proceso atómico (que debe ser validado con un commit, o invalidado con un rollback), y que puede involucrar operaciones de inserción, modificación y borrado de datos. El proceso transaccional es típico de las bases de datos operacionales.



OLAP (On Line Analytical Processing)

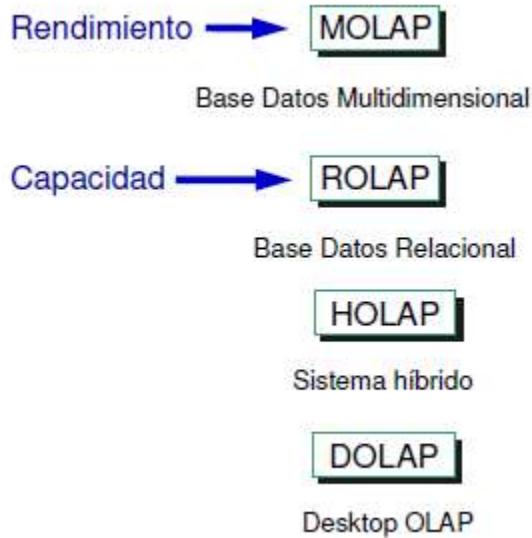
Los sistemas OLAP son bases de datos orientadas al procesamiento analítico. Este análisis suele implicar, generalmente, la lectura de grandes cantidades de datos para llegar a extraer algún tipo de información útil: tendencias de ventas, patrones de comportamiento de los consumidores, elaboración de informes complejos, entre otros. Se asocia principalmente su uso a Data Warehouses y Datamarts.

MOLAP

La arquitectura MOLAP usa bases de datos multidimensionales para proporcionar el análisis, su principal premisa es que el OLAP está mejor implantado almacenando los datos multidimensionalmente. Un sistema MOLAP usa una base de datos propietaria multidimensional,

en la que la información se almacena multidimensionalmente, para ser visualizada en varias dimensiones de análisis.

La arquitectura MOLAP requiere unos cálculos intensivos de compilación. Lee de datos precompilados, y tiene capacidades limitadas de crear agregaciones dinámicamente o de hallar ratios que no se hayan precalculados y almacenados previamente.



ROLAP (Relacional)

La arquitectura ROLAP, accede a los datos almacenados en un datawarehouse para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales.

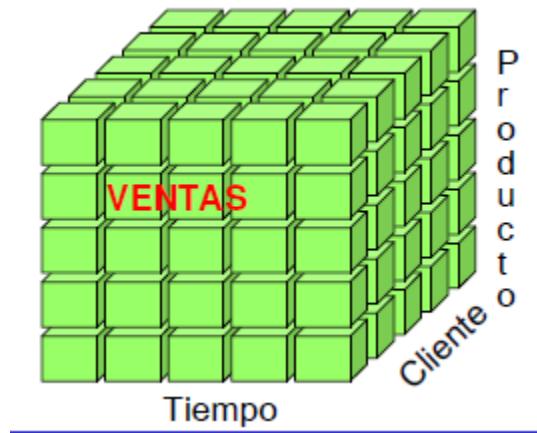
El sistema ROLAP utiliza una arquitectura de tres niveles. La base de datos relacional maneja los requerimientos de almacenamiento de datos, y el motor ROLAP proporciona la funcionalidad analítica. El nivel de base de datos usa bases de datos relacionales para el manejo, acceso y obtención del dato. El nivel de aplicación es el motor que ejecuta las consultas multidimensionales de los usuarios.

HOLAP (Híbrido)

Un desarrollo un poco más reciente ha sido la solución OLAP híbrida (HOLAP), la cual combina las arquitecturas ROLAP y MOLAP para brindar una solución con las mejores características de ambas: desempeño superior y gran escalabilidad. Un tipo de HOLAP mantiene los registros de detalle (los volúmenes más grandes) en la base de datos relacional, mientras que mantiene las agregaciones en un almacén MOLAP separado.

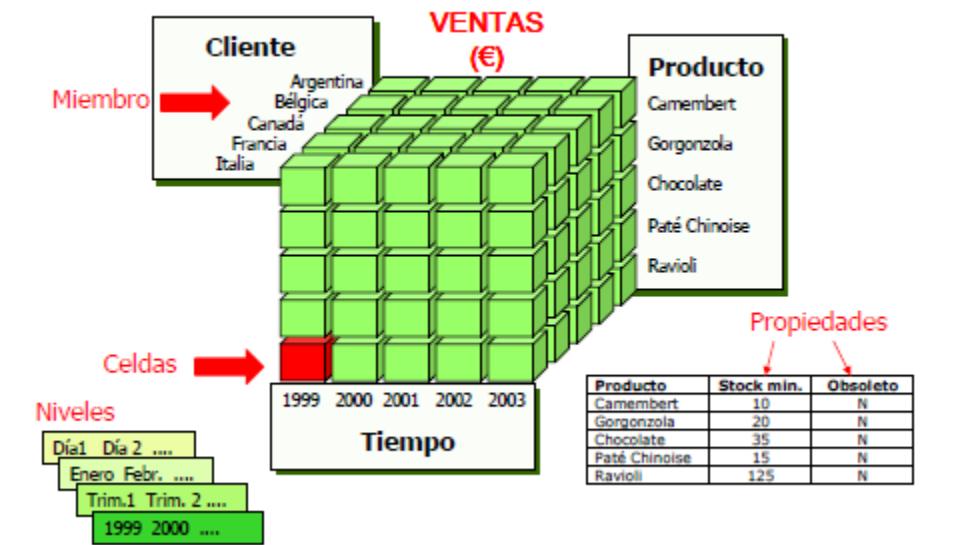
Diseño de cubos

Un cubo es un subconjunto de datos de un Data Warehouse que es almacenado en una estructura multidimensional. El cubo contiene los valores agregados de todos los niveles de todas las dimensiones.



Cada elemento del cubo (cubo elemental) representa una ocurrencia (fila) en la tabla de hechos.

Además de almacenar los valores de las medidas, también se guardan en ellos los acumulados según las dimensiones (por ejemplo de la dimensión tiempo se tienen acumulados anuales, trimestrales, diarios, etc.) como “medidas derivadas”.



Para hacer el diseño de un cubo para un datamart se requiere:

- Identificar las columnas de las tablas que participan en la dimensión.
- Definir las propiedades.
- Definir los niveles de detalle (granularidad).

El diseño de cubos ofrece:

- Una visión multidimensional de los datos (matricial).
- No imponer restricciones sobre el número de dimensiones.
- Simetría para las dimensiones.
- Permitir definir de forma flexible (sin limitaciones) sobre las dimensiones: restricciones, agregaciones y jerarquías entre ellas.
- Las consultas son muy rápidas.

Desventajas del uso de cubos:

- Su diseño requiere de más tiempo que el adaptar una base de datos relacional para reportes.
- Una vez poblada la base de datos no se puede modificar su estructura sin rediseñar el cubo.

PostgreSQL para la construcción de Data Warehouses y Datamarts



PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Antecedentes históricos

IBM había estado trabajando desde 1973 con los primeros conceptos, ideas y teorías sobre bases de datos relacionales.

Su proyecto "System R" fue entre otras cosas la primera implementación del lenguaje SQL (Structured Query Language).

Este proyecto, sus decisiones de diseño y muchos de los algoritmos usados, influenciaron muchos de los sistemas de bases de datos relacionales que aparecieron posteriormente.

El antecesor de PostgreSQL es el proyecto Ingres (INteractive Graphics REtrieval System) que estuvo activo de 1977 a 1985. Su creador, el profesor Michael Stonebraker de la Universidad de

Berkeley se basó inicialmente en el “System R”. Ingres compitió con Oracle por el mercado. En 1985 Stonebraker regresa a Berkeley e inicia el proyecto PostgreSQL.

En 1994 los estudiantes Andrew Yu y Jolly Chen depuran el código para que fuera 100% en ANSI C, implementan un intérprete de lenguaje SQL y desarrollan la interfaz pgsql para las consultas. Lo liberaron en 1995 como PostgreSQL95.

En 1996 se cambia a PostgreSQL y se crea el PostgreSQL Global Development Team.

Características de PostgreSQL

- Documentación completa.
- Licencia BSD (Admite el uso de su código fuente en software propietario).
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.
- Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de Oracle), PL/Perl, PL/Python y PL/Tcl.
- Numerosos tipos de datos y la posibilidad de definir nuevos tipos.
- Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.
- Soporta el almacenamiento de objetos binarios grandes.
- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.

Instalación de PosgtreSQL

La instalación de postgresSQL se puede hacer utilizando un instalador, ejecutando los archivos compilados o bien compilando los archivos fuente, dependiendo de nuestro propósito y del SO sobre el que vamos a trabajar.

Para descargar la versión que queremos instalar podemos ir a la página del proyecto <http://www.postgresql.org/download/>

Donate | Contact | Search



PostgreSQL

The world's most advanced open source database.

[Home](#) | [About](#) | [Download](#) | [Documentation](#) | [Community](#) | [Developers](#) | [Support](#) | [Your account](#)

[» Downloads](#)
[Binary](#)
[Source](#)
[» Software Catalogue](#)
[» File Browser](#)

Downloads

PostgreSQL Core Distribution

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

Binary packages

Pre-built binary packages are available for a number of different operating systems:

- BSD
 - [FreeBSD](#)
 - [OpenBSD](#)
- Linux
 - [Red Hat family Linux](#) (including CentOS/Fedora/Scientific/Oracle variants)
 - [Debian](#) GNU/Linux and derivatives
 - [Ubuntu](#) Linux and derivatives
 - [SuSE](#) and OpenSuSE
 - [Other](#) Linux
- [Mac OS X](#)
- [Solaris](#)
- [Windows](#)

Source code

The source code can be found in the main [file browser](#) or you can access the source control repository directly at [git.postgresql.org](#). Instructions for building from source can be found in the [documentation](#).

Beta/RC Releases and development snapshots (unstable)

There are source code and binary [packages](#) of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

3rd party distributions

Instalación de PostgreSQL para distribuciones Linux desde archivos fuente (para compilar)

Conectarse como root. Descargar y descomprimir los archivos fuente:

```
SU -
cd /usr/local
tar -xvzf postgresql-X.X.X.tar.gz
```

Seguir las instrucciones del archivo INSTALL

```
./configure
gmake
su
gmake install
```

```
adduser postgres --crea el usuario postgres  
mkdir /var/lib/pgsql/data --crea la carpeta de los datos  
chown postgres /var/lib/pgsql/data --otorga privilegios al usuario postgres sobre los datos  
su - postgres  
/usr/local/pgsql/bin/initdb -D /var/lib/pgsql/data --inicializa el cluster  
/usr/local/pgsql/bin/postgres -D /var/lib/pgsql/data >logfile 2>&1 &  
/usr/local/pgsql/bin/createdb test -- crea base de datos de prueba test  
/usr/local/pgsql/bin/psql test --abre la base de datos de prueba test
```

Instalación de PostgreSQL para distribuciones Linux desde archivos compilados para distribuciones Linux de la familia Red Hat

Elegir el repositorio, de preferencia el oficial de PostgreSQL y descargar los archivos de la versión de PostgreSQL

Abrir y editar el archivo gedit para que instale PostgreSQL desde el repositorio oficial y no del repositorio de la distribución (para tener la versión más reciente o alguna versión en particular):

```
sudo gedit /etc/yum.repos.d/fedora.repo ( en línea 10 insertar: exclude=postgresql*)
```

```
sudo gedit /etc/yum.repos.d/fedora-updates.repo ( en línea 9 insertar: exclude=postgresql*)
```

Descargar e instalar del repositorio el archivo compilado PGDG con extensión RPM (verificar versión en la página oficial: <http://yum.postgresql.org/repopackages.php>)

```
curl -O http://yum.postgresql.org/9.2/fedora/fedora-16-i386/pgdg-fedora92-9.2-5.noarch.rpm
```

```
sudo rpm -ivh pgdg-fedora92-9.2-5.noarch.rpm
```

[el comando curl automatiza la transferencia del archivo, el comando rpm ejecuta el administrador de instalación/actualización de paquetes]

Instalación de PostgreSQL desde archivos compilados para distribuciones Linux de la familia Ubuntu

Agregar el repositorio oficial de PostgreSQL

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ precise-pgdg main" >> /etc/apt/sources.list'  
wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo apt-key add -  
sudo apt-get update
```

Actualizar el repositorio: `sudo apt_get update`

Ejecutar la instrucción para instalar PostgreSQL y pgAdmin:

```
sudo apt-get install postgresql postgresql-contrib  
sudo apt-get install pgadmin3
```

Instalación de PostgreSQL utilizando el instalador

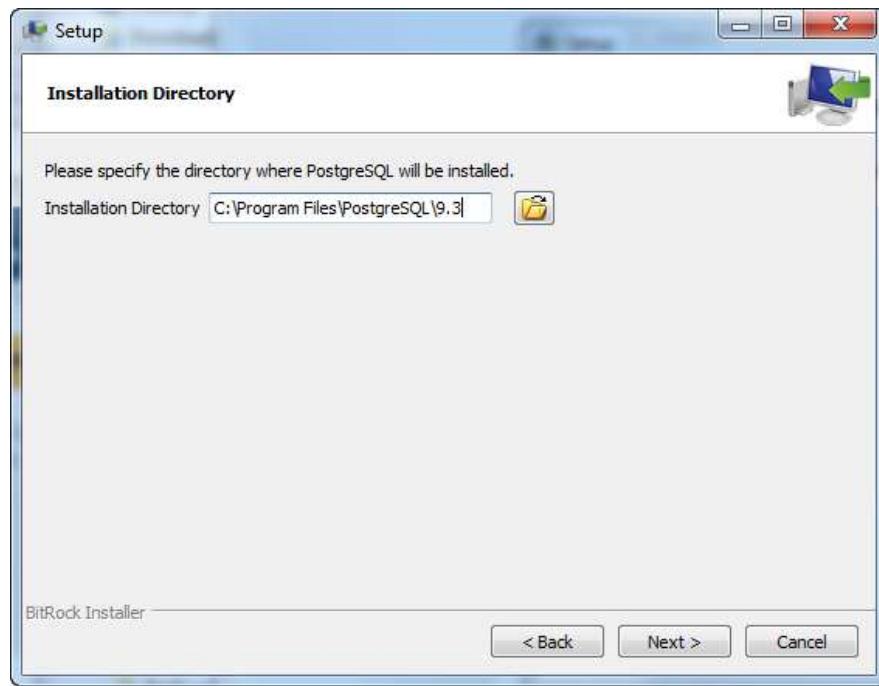
Se descarga el instalador de la página oficial de PostgreSQL y se ejecuta siguiendo las instrucciones como un usuario con permisos de instalación.



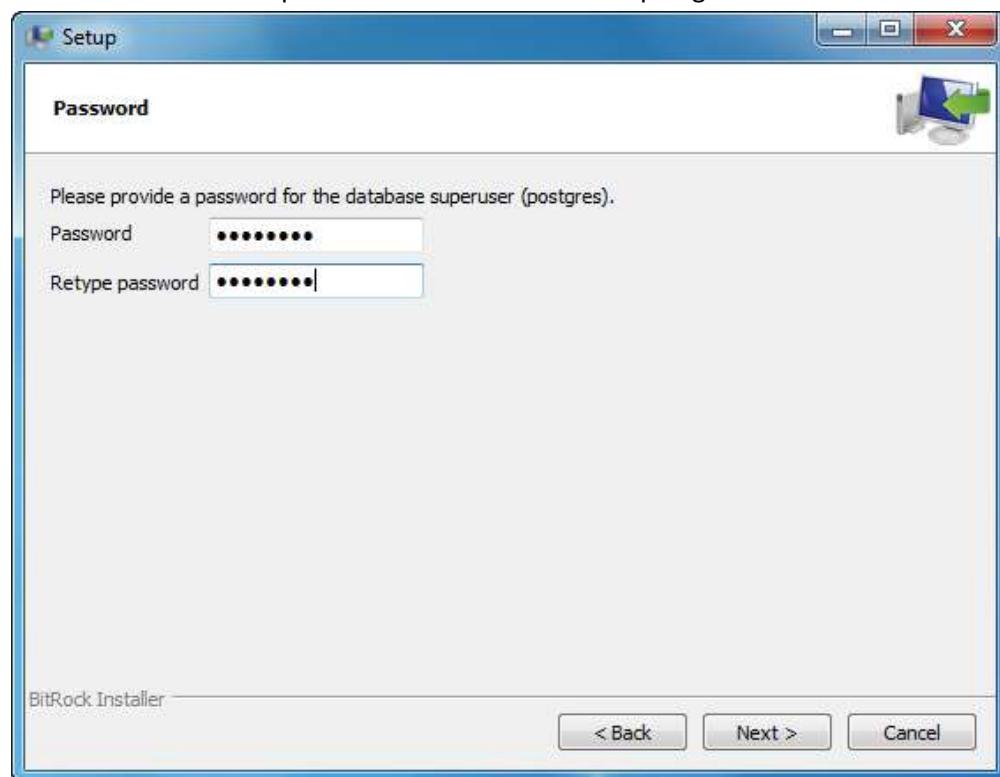
Se siguen las instrucciones del instalador:



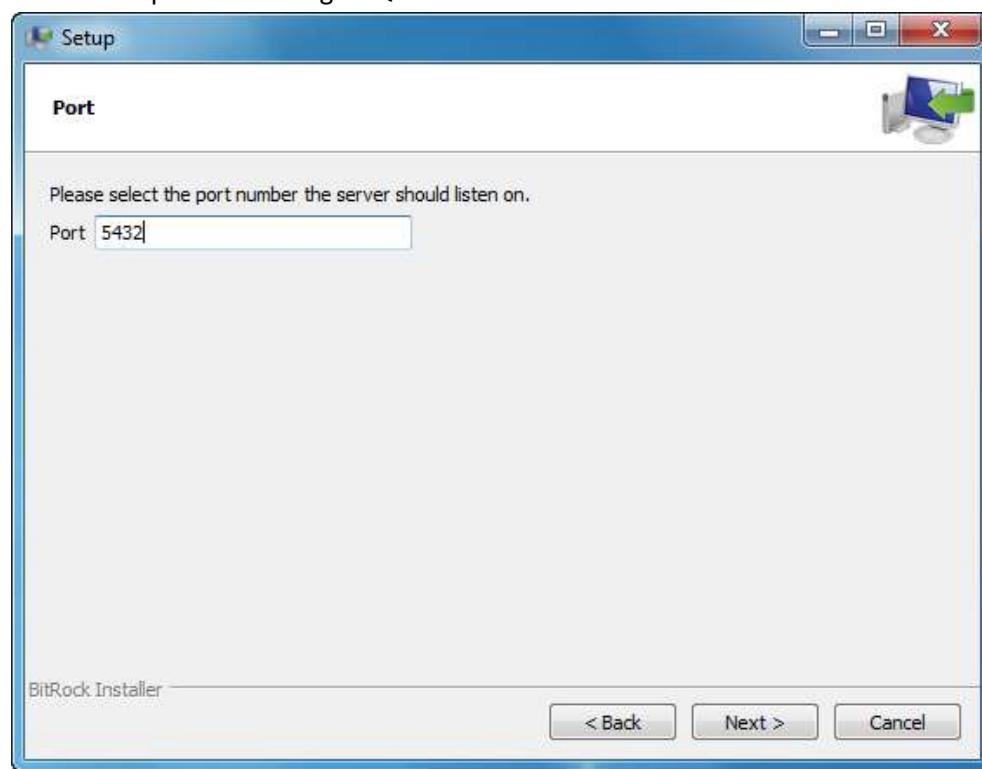
A menos que se tenga otra instalación previa de PostgreSQL, se recomienda dejar las rutas originales para la instalación:



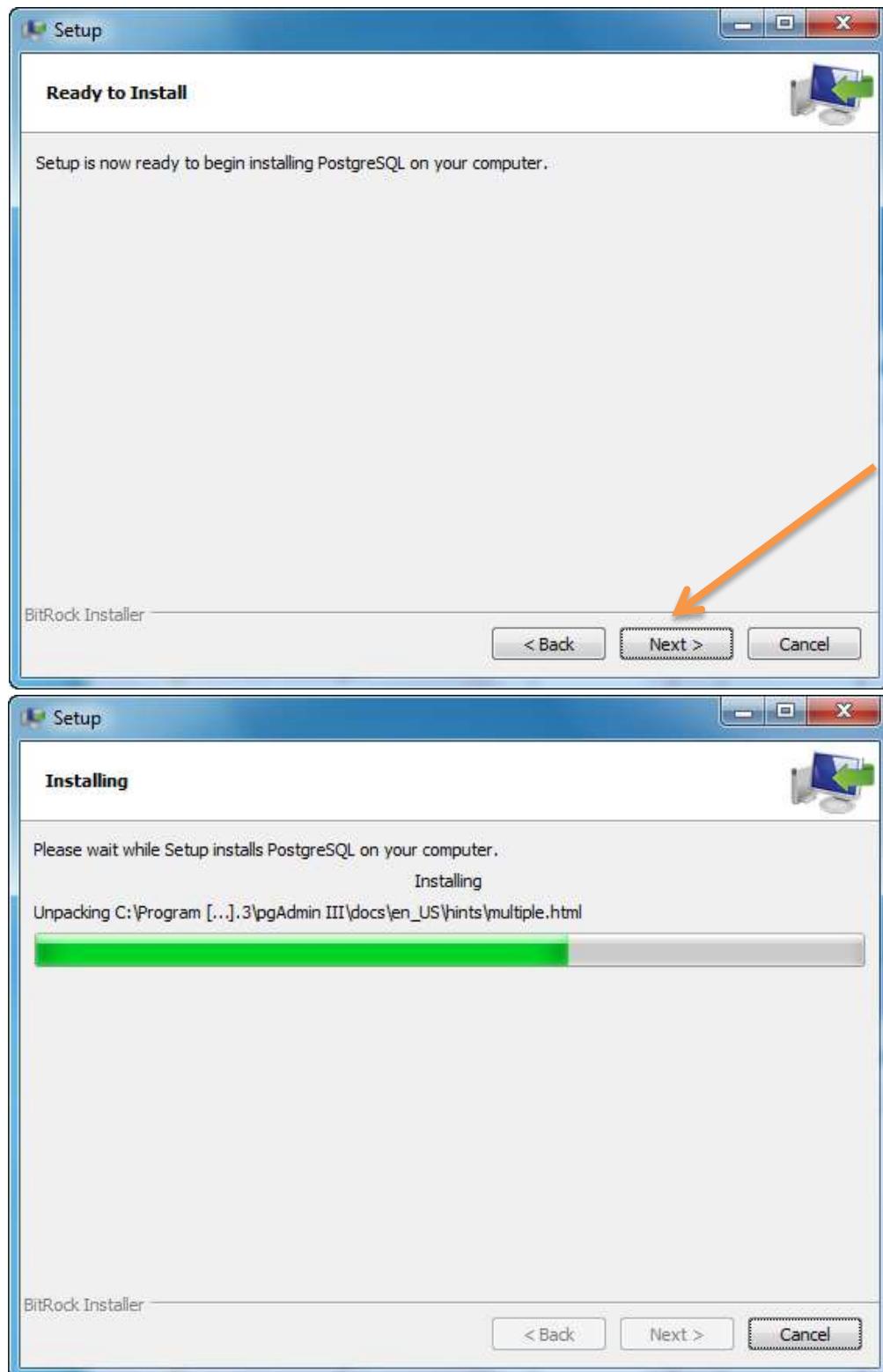
Se pide crear una contraseña para el usuario administrador postgres.



El puerto de conexión por default es el 5432, se recomienda conservarlo a menos que se cuente con una instalación previa de PostgreSQL.



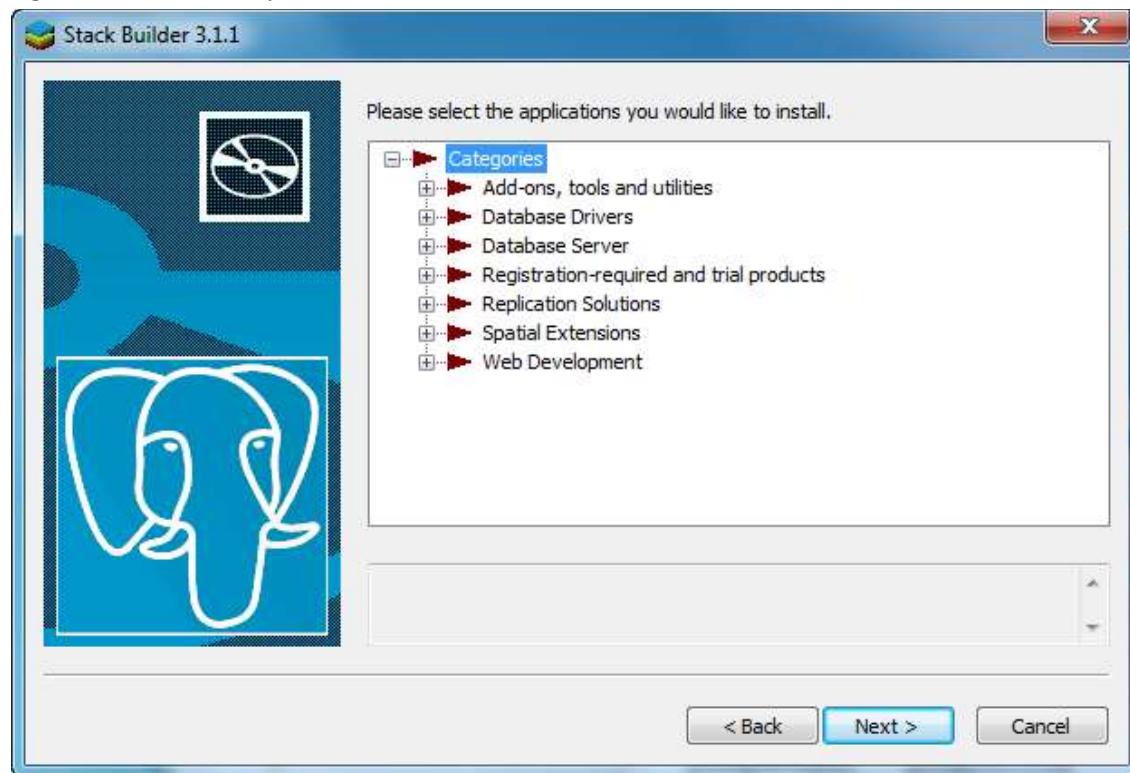
Cuando se han definido las rutas donde quedarán los archivos de instalación y los datos se instala el programa.



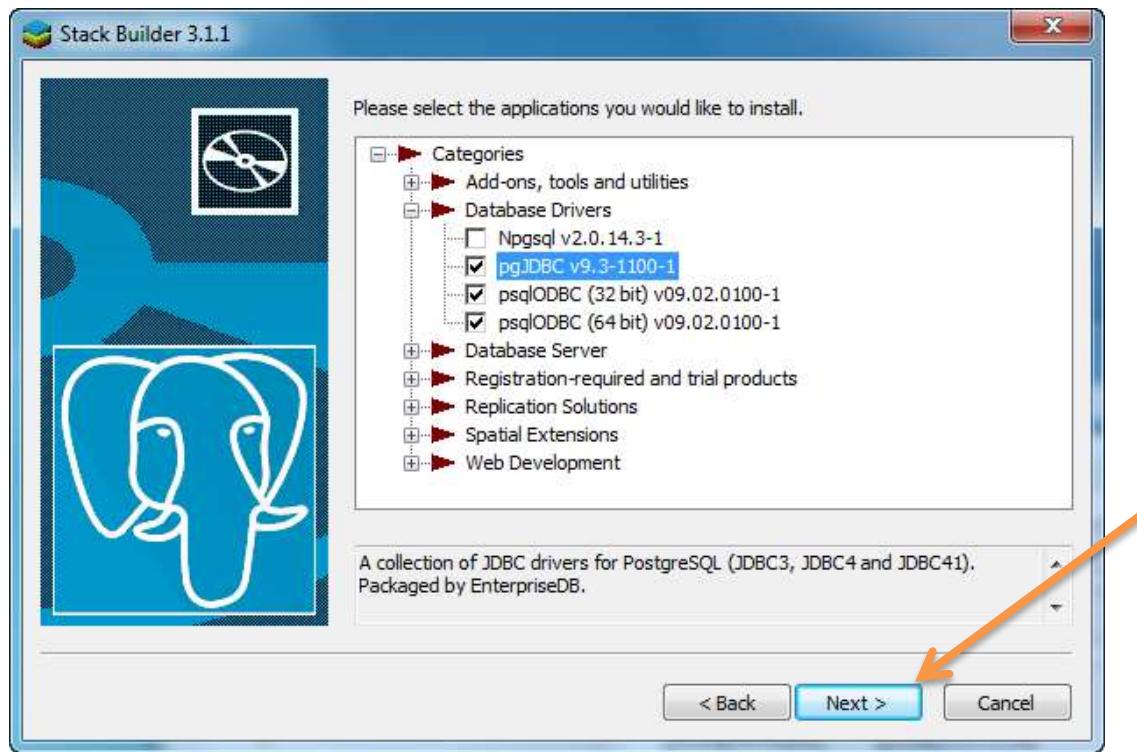
Al terminar la instalación de los archivos básicos, se activa el Stack Builder, que es el módulo para administrar actualizaciones y elementos adicionales de la configuración de PostgreSQL:



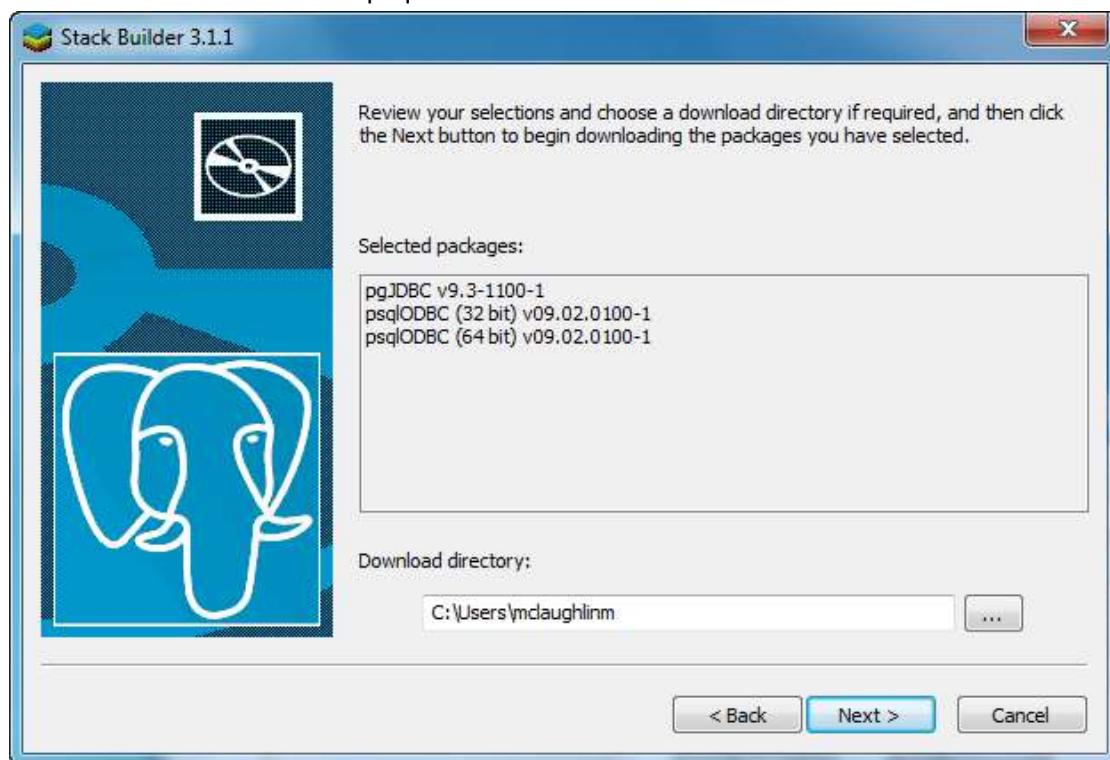
Entre los elementos a seleccionar se encuentran los drivers para la base de datos (psqlODBC, pgJDBC), la extensión para datos espaciales PostGIS, la compatibilidad con lenguajes de programación como Phyton, etc.



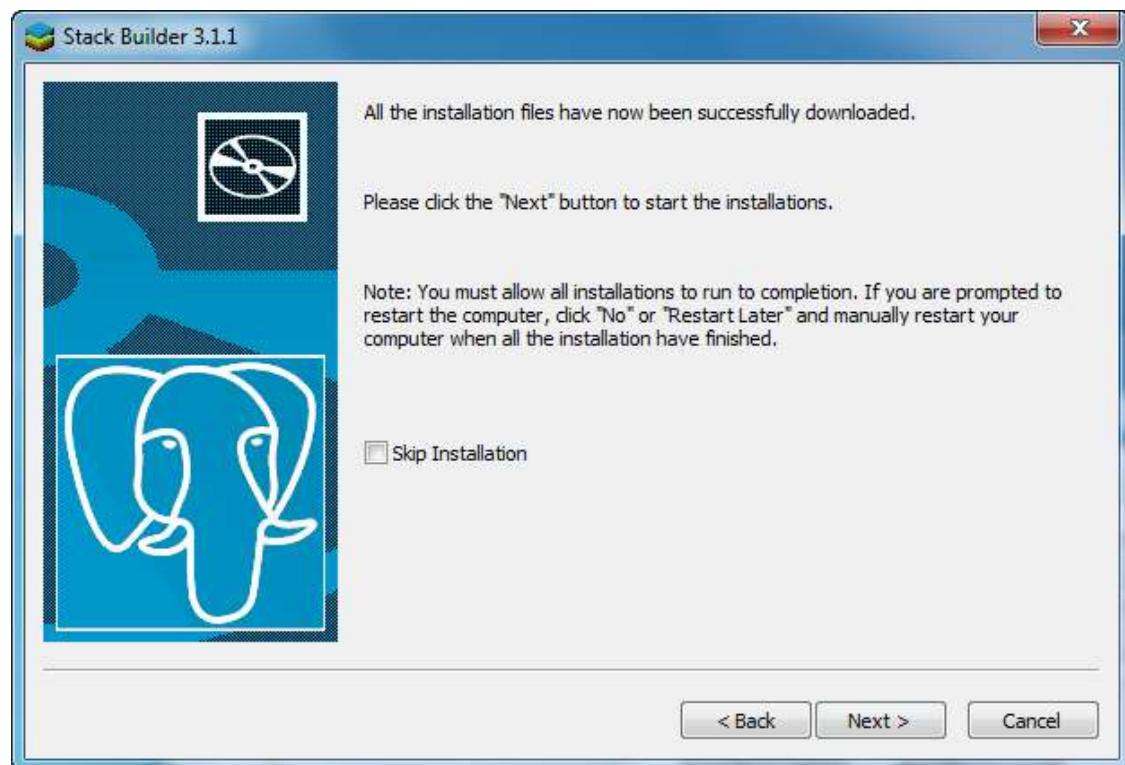
Una vez seleccionados los complementos a instalar se continúa con la instalación de los paquetes adicionales:



El sistema confirma el listado de paquetes adicionales antes de instalar:



Los archivos se descargan en la ruta indicada anteriormente.



Se instalan los paquetes adicionales:





Una vez instalados todos los componentes de PostgreSQL se cierra el instalador.



Interfaces de PostgreSQL

Para hacer solicitudes al motor de búsqueda de PostgreSQL existen dos interfaces: la interfaz de terminal **psql** y la interfaz gráfica **pgAdmin**.

Para acceder a la terminal psql desde Windows se utiliza el pgAdmin.

Psql



A screenshot of a terminal window titled "sara@Naomi:~". The user runs the command "sudo su - postgres" to switch to the postgres user. After entering the password, they run "psql" to enter the PostgreSQL command-line interface. The terminal is dark-themed with white text.

Comandos básicos de psql

Entrar desde una terminal cambiando al usuario postgres con: `sudo su - postgres`

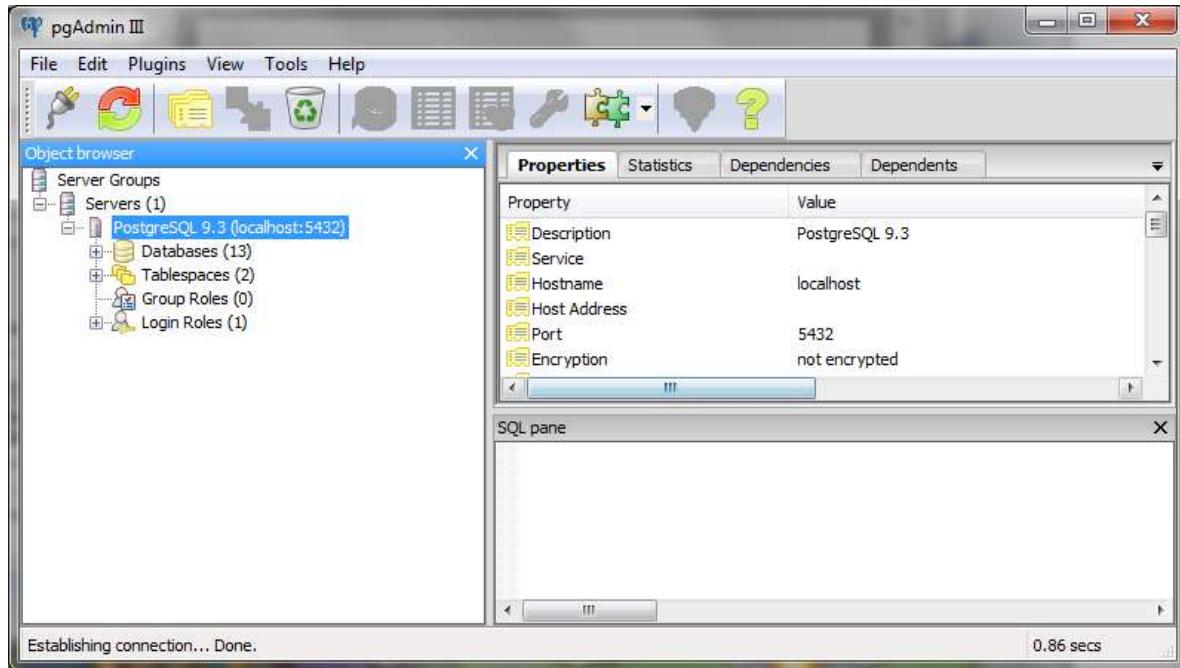
Como usuario postgres ejecutar la orden: `psql [nombredbd]` para ingresar a una base de datos en particular directamente.

El siguiente listado muestra las instrucciones más comunes de la interfaz psql:

\?	muestra el listado de los comandos de psql (salir con q)
\h	muestra la lista de órdenes para generar sentencias
\h [orden]	muestra la sintaxis de las sentencias en PostgreSQL
\q	salir de psql
\l	enlistar bases de datos en el servidor
\c [nombredbase]	conectarse a la base de datos con ese nombre
\c [nombredbase] - usuario	
\p	mostrar el contenido del búfer de consulta
\r	reiniciar (limpiar) el búfer de consulta

pgAdmin

En tanto que el pgAdmin es un entorno gráfico muy amigable y totalmente intuitivo. Está compuesto de varios páneles y menús.



En la parte superior tenemos los menús básicos para la conectarse al servidor, refrescar la conexión, abrir una ventana para ejecutar sentencias SQL entre otros, representados por iconos.



Y un grupo de opciones adicionales en las opciones: File, Edit, Plugins, View, Tools, Help.

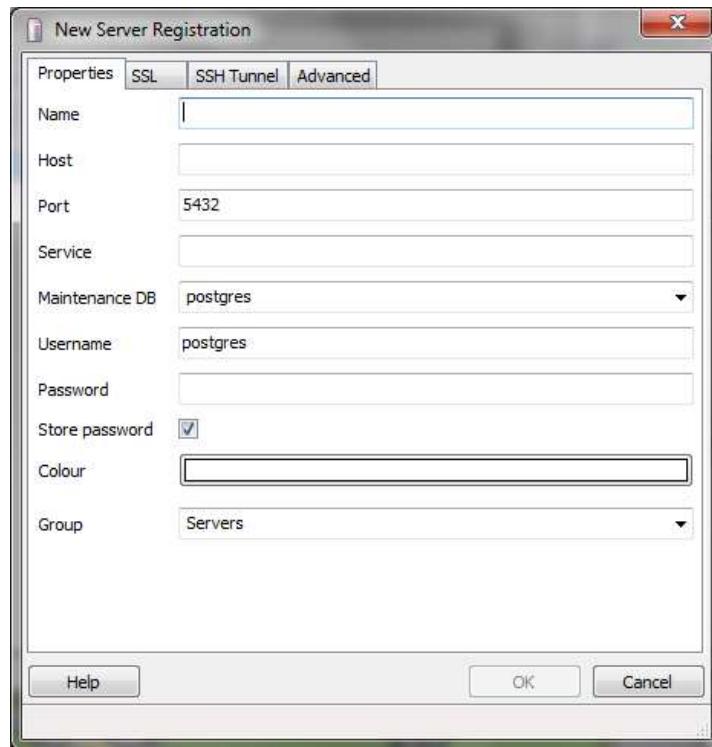


En particular, la opción Plugins tiene la única opción **PSQL CONSOLE** que nos permite desplegar una ventana con la interfaz psql.

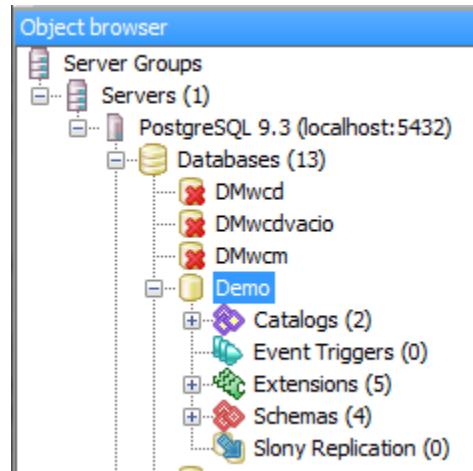
```
psql <9.3.4>
ADVERTENCIA: El código de página de la consola <850> difiere del código
de página de Windows <1252>.
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql <Notes for Windows users>
para obtener más detalles.
Digite <help> para obtener ayuda.

Demo=#
```

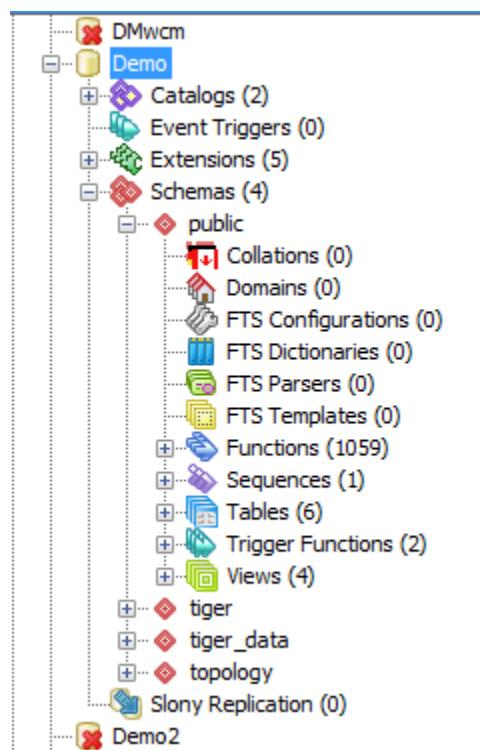
Para establecer la conexión al servidor se debe dar clic en el icono  para que se despliegue la ventana donde ingresar la ruta y contraseña del dueño de la base de datos.



En el panel lateral izquierdo se puede ver el árbol con la estructura de PostgreSQL y todos sus componentes: el servidor, las bases de datos que están en él, etc.

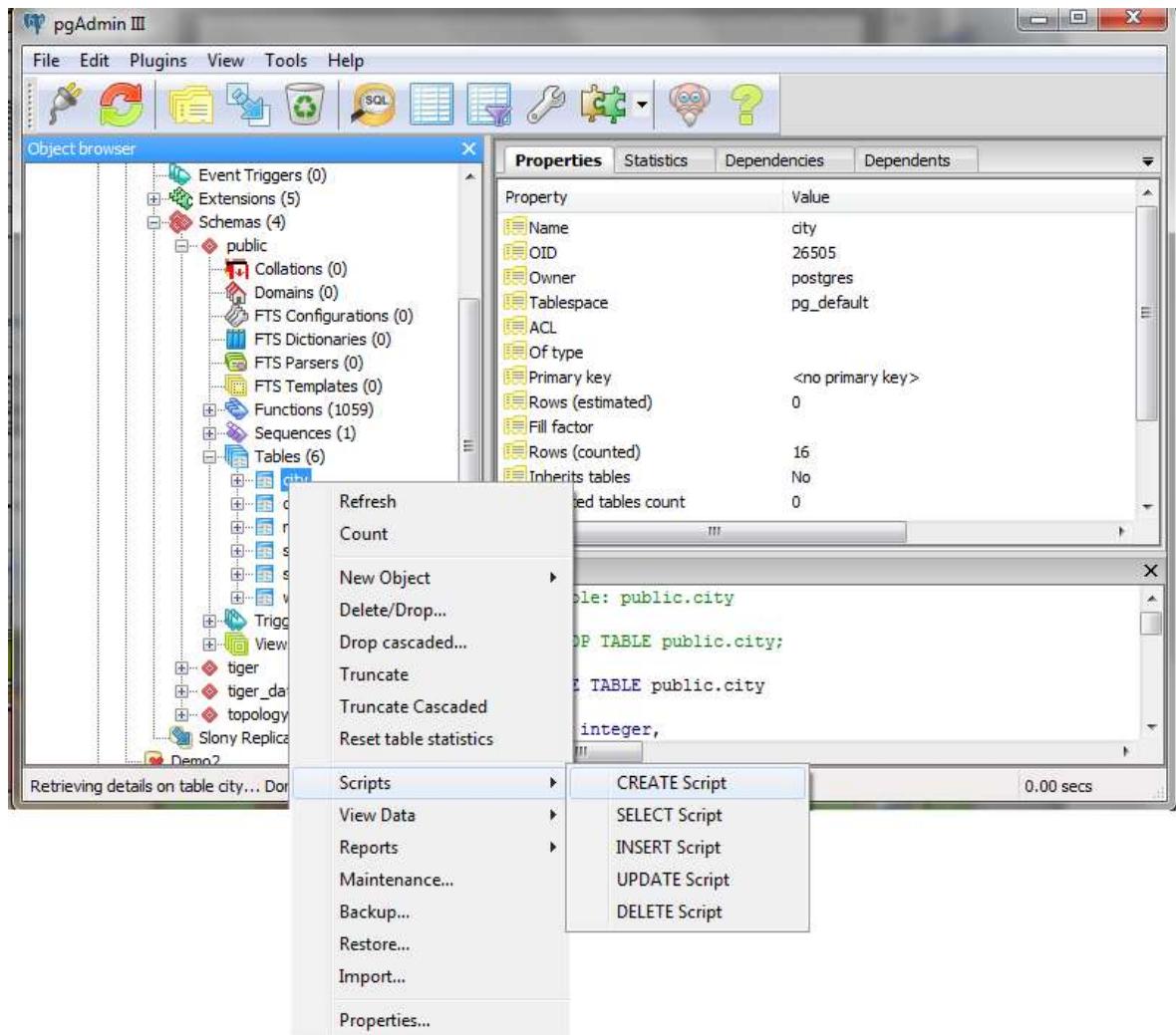


Las tablas que se encuentran dentro de los Schemas (por default se cargan en el schema public si no se indica uno distinto).

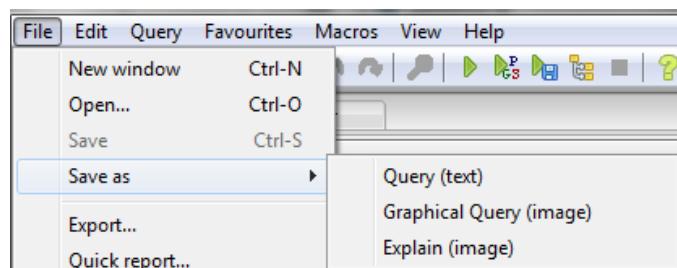


Al ir expandiendo el árbol, podemos ver que dentro del esquema se encuentran las tablas, templates, funciones, etc.

Al posicionarnos sobre una tabla y dar clic derecho se va a desplegar un menú que nos permite ejecutar diferentes sentencias SQL sobre dicha tabla:



Al seleccionar una de las sentencias SQL del submenú Scripts se va a desplegar una ventana con varios menús y páneles donde se muestra la sintaxis de la sentencia. Podemos editar la instrucción SQL para personalizar la consulta, debugearla para verificar su ejecución y ejecutarla con los botones del menú superior de la ventana. También podemos guardar las instrucciones que generemos en un archivo tipo SQL para su posterior ejecución o para crear tareas programadas.



Cuando ejecutamos la sentencia SQL se muestra el resultado en el panel inferior, junto con otras pestañas con los mensajes generados por el motor PostgreSQL y el historial de ejecuciones.

The screenshot shows the pgAdmin III application window. The top menu bar includes File, Edit, Query, Favourites, Macros, View, and Help. Below the menu is a toolbar with various icons. The main area has tabs for SQL Editor and Graphical Query Builder, with SQL Editor selected. A 'Previous queries' dropdown and 'Delete' buttons are visible. The SQL Editor pane contains the following query:

```
SELECT gid, area, perimeter, cities_, cities_id, city_name, gmi_admin,
       admin_name, fips_cntry, cntry_name, status, pop_rank, pop_class,
       port_id, the_geom
  FROM public.city;
```

The bottom half of the window is the 'Output pane', which is currently set to 'Data Output'. It displays the results of the query as a table:

gid	area	perimeter	cities_	cities_id	city_name	gmi_admin	admin_n
integer	double precision	double precision	numeric(38,9)	numeric(38,9)	character varying	character varying	character
1	0	0	0.20.000000000	53.000000000	Bismarck	USA-NDA	North D
2	1	0	0.24.000000000	54.000000000	Pierre	USA-SDA	South D
3	2	0	0.74.000000000	28.000000000	Winnipeg	CAN-MNT	Manitob
4	3	0	0.56.000000000	98.000000000	Saint Paul	USA-MNN	Minneso
5	4	0	0.61.000000000	38.000000000	Minneapolis	USA-MNN	Minneso
6	5	0	0.08.000000000	24.000000000	Milwaukee	USA-WIS	Wiscons
7	6	0	0.09.000000000	20.000000000	Madison	USA-WIS	Wiscons

Below the table, the status bar shows 'OK.', 'Unix', 'Ln 1, Col 1, Ch 1', '16 rows.', and '11 ms'.

Query - Demo on postgres@localhost:5432

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries

```
SELECT gid, area, perimeter, cities_, cities_id, city_name, gmi_admin,
       admin_name, fips_cntry, cntry_name, status, pop_rank, pop_class,
       port_id, the_geom
  FROM public.city;
```

Scratch pad

Output pane

Data Output Explain Messages History

Total query runtime: 11 ms.
16 rows retrieved.

OK. Unix Ln 5, Col 1, Ch 191 16 rows. 11 ms

Query - Demo on postgres@localhost:5432

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries

```
SELECT gid, area, perimeter, cities_, cities_id, city_name, gmi_admin,
       admin_name, fips_cntry, cntry_name, status, pop_rank, pop_class,
       port_id, the_geom
  FROM public.city;
```

Scratch pad

Output pane

Data Output Explain Messages History

-- Executing query:
SELECT gid, area, perimeter, cities_, cities_id, city_name, gmi_admin,
 admin_name, fips_cntry, cntry_name, status, pop_rank, pop_class,
 port_id, the_geom
 FROM public.city;

Total query runtime: 11 ms.
16 rows retrieved.

OK. Unix Ln 5, Col 1, Ch 191 16 rows. 11 ms

En caso de que no se pueda acceder a PostgreSQL se recomienda verificar que se ha levantado el servicio, de igual manera, para hacer actualizaciones se recomienda detener el servicio.

También se puede configurar el PostgreSQL para iniciar el servicio al arrancar el servidor o bien para que se tenga que hacer de manera manual.



```
GNU nano 2.2.6 Archivo: .../pgsql/9.3/main/start.conf

# Automatic startup configuration
# auto: automatically start/stop the cluster in the init script
# manual: do not start/stop in init scripts, but allow manual start/stop via pg_ctlcluster
# disabled: do not allow manual startup with pg_ctlcluster (this can be circumvented and is only meant to be a small protection against accidents).

auto
```

Se recomienda contar con un servidor dedicado para un proyecto de BI, pero eso dependerá del volumen de datos y de las posibilidades de la empresa.

Sentencias de SQL Básicas

- **CREATE DATABASE**
- **CREATE TABLE**
- **CREATE INDEX**
- **ALTER TABLE**
- **DROP TABLE**
- **DROP INDEX**
- **INSERT**
- **UPDATE**
- **DELETE**
- **SELECT**
 - Cláusulas SELECT, FROM, INTO, WHERE
 - Cláusulas GROUP BY, HAVING, ORDER BY
 - Tipos de JOIN
- **Subqueries**

Para conocer la sintaxis de alguna instrucción podemos consultar el manual en la interfaz psql utilizando la instrucción: \h [orden]

Y en la interfaz pgAdmin se va a mostrar la sintaxis en el panel de SQL de la ventana de edición para que la personalicemos.

CREATE DATABASE

Crea una nueva base de datos con el nombre que indiquemos, se puede dar la ubicación alternativa para almacenar la nueva base en el sistema de archivos. El creador de la nueva base es su propietario por default.

```
CREATE DATABASE name [ WITH LOCATION = 'dbpath' ]
```

Para eliminar una base de datos se usa: **DROP DATABASE** name. Sólo puede ser ejecutado por el propietario de la base. Una vez ejecutado es irreversible.

CREATE TABLE

El comando fundamental para definir datos es el que crea una nueva relación (una nueva tabla). La sintaxis del comando CREATE TABLE es:

```
CREATE TABLE table_name  
(name_of_attr_1 type_of_attr_1  
[, name_of_attr_2 type_of_attr_2  
[, ...]]);
```

CREATE INDEX

Se utilizan los índices para acelerar el acceso a una relación. Si una relación R tiene un índice en el atributo A podremos recuperar todas las tuplas t que tienen $t(A) = a$ en un tiempo aproximadamente proporcional al número de tales tuplas t más que en un tiempo proporcional al tamaño de R.

Para crear un índice en SQL se utiliza el comando CREATE INDEX. La sintaxis es:

```
CREATE INDEX index_name ON table_name ( name_of_attribute );
```

ALTER TABLE

Cambia la definición de una tabla:

```
ALTER TABLE [ ONLY ] nombre [ * ]  acción [, ... ]
ALTER TABLE [ ONLY ] nombre [ * ]
    RENAME [ COLUMN ] columna TO nueva_columna
ALTER TABLE nombre
    RENAME TO nuevo_nombre
```

donde acción es uno de:

```
ADD [ COLUMN ] columna tipo [ restricción_columna [ ... ] ]
DROP [ COLUMN ] columna [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] columna [ SET DATA ] TYPE tipo [ USING expresión ]
ALTER [ COLUMN ] columna SET DEFAULT expresión
ALTER [ COLUMN ] columna DROP DEFAULT
ALTER [ COLUMN ] columna { SET | DROP } NOT NULL
```

ADD COLUMN

Esta forma agrega una nueva columna a la tabla, usando la misma sintaxis que CREATE TABLE.

DROP COLUMN

Esta forma elimina una columna de una tabla. Los índices y restricciones involucrando la columna serán también automáticamente eliminados. Será necesario escribir CASCADE si algo fuera de la tabla depende de la columna, por ejemplo, una referencia de llave foránea o vista.

SET DATA TYPE

Esta forma cambia el tipo de una columna de una tabla.

Entre otras, para ver la definición completa se puede consultar el manual incluído en PostgreSQL, o en el manual del grupo de desarrollo oficial de PostgreSQL.

DROP TABLE

Se utiliza el comando `DROP TABLE` para eliminar una tabla (incluyendo todas las tuplas almacenadas en ella):

```
DROP TABLE table_name;
```

DROP INDEX

Se utiliza el comando DROP INDEX para eliminar un índice: `DROP INDEX index_name;`

INSERT INTO

Una vez que se crea una tabla (vea Create Table), puede ser llenada con tuplas mediante el comando INSERT INTO. La sintaxis es:

```
INSERT INTO table_name (name_of_attr_1  
[, name_of_attr_2 [...]])  
VALUES (val_attr_1  
[, val_attr_2 [, ...]]);
```

UPDATE

Para cambiar uno o más valores de atributos de tuplas en una relación, se utiliza el comando UPDATE. La sintaxis es:

```
UPDATE table_name  
SET name_of_attr_1 = value_1  
[, ... [, name_of_attr_k = value_k]]  
WHERE condition;
```

DELETE

Para borrar una tupla de una tabla particular, utilizamos el comando DELETE FROM.

La sintaxis es:

```
DELETE FROM table_name  
WHERE condition;
```

SELECT

El comando más usado en SQL es la instrucción SELECT, que se utiliza para recuperar datos. La sintaxis es:

```
SELECT [ALL|DISTINCT]
{ * | expr_1 [AS c_alias_1] [, ...
[, expr_k [AS c_alias_k]]]
FROM table_name_1 [t_alias_1]
[, ... [, table_name_n [t_alias_n]]]
[WHERE condition]
[GROUP BY name_of_attr_i
[,... [, name_of_attr_j]] [HAVING condition]]
[ {UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]
[ORDER BY name_of_attr_i [ASC|DESC]
[, ... [, name_of_attr_j [ASC|DESC]]]];
```

El particionamiento de las tuplas en grupos se hace utilizando las palabras clave **GROUP BY** seguidas de una lista de atributos que definen los grupos. Si tenemos GROUP BY A1, ..., Ak habremos particionado la relación en grupos, de tal modo que dos tuplas son del mismo grupo si y sólo si tienen el mismo valor en sus atributos A1,..., Ak.

La cláusula **HAVING** trabaja de forma muy parecida a la cláusula WHERE, y se utiliza para considerar sólo aquellos grupos que satisfagan la cualificación dada en la misma. Las expresiones permitidas en la cláusula HAVING deben involucrar funciones agregadas. Cada expresión que utilice sólo atributos planos deberá recogerse en la cláusula WHERE. Por otro lado, toda expresión que involucre funciones agregadas debe aparecer en la cláusula HAVING.

La cláusula **ORDER BY** permite al usuario especificar si quiere los registros ordenados de manera ascendente o descendente utilizando los operadores de modo ASC y DESC.

El alias es un nombre alternativo para la tabla precedente table. Se utiliza para abreviar o eliminar ambigüedades en uniones dentro de una misma tabla.

Subconsultas

En las cláusulas WHERE y HAVING se permite el uso de subconsultas (subselects) en cualquier lugar donde se espere un valor. En este caso, el valor debe derivar de la evaluación previa de la subconsulta. El uso de subconsultas amplía el poder expresivo de SQL.

Ejemplo:

```
SELECT *
FROM PART
WHERE PRICE > (SELECT PRICE FROM PART
WHERE PNAME='Tornillos');
```

Cuando revisamos la consulta anterior, podemos ver la palabra clave SELECT dos veces. La primera al principio de la consulta - a la que nos referiremos como la SELECT externa - y la segunda en la cláusula WHERE, donde empieza una consulta anidada - nos referiremos a ella como la SELECT interna. Para cada tupla de la SELECT externa, la SELECT interna deberá ser evaluada. Tras cada evaluación, conoceremos el precio de la tupla llamada 'Tornillos', y podremos ver si el precio de la tupla actual es mayor.

Tipos de join

Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

Su sintaxis es:

```
SELECT [campos de ambas tablas] FROM tabla1 JOIN tabla2 ON campo_tabla1=campo_tabla2
```

Hay tres tipos de combinaciones:

- JOIN ó INNER JOIN combina cada fila de la tabla 1 con cada fila de la tabla 2 seleccionando las filas que cumplen con la condición determinada.
- LEFT JOIN ó LEFT OUTER JOIN combina los valores de la primera tabla con los valores de la segunda que cumplan con la condición. Si no existe ninguna coincidencia el lado derecho contendrá un null.
- RIGHT JOIN ó RIGHT OUTER JOIN combina los valores de la primera tabla con los valores de la segunda. Siempre devolverá las filas de la segunda tabla, incluso aunque no cumplan la condición.
- FULL JOIN ó FULL OUTER JOIN combina todos los valores de la primera tabla con todos los valores de la segunda, devolviendo las filas de las dos tablas cumplan o no con la condición.
- CROSS JOIN retorna el producto cartesiano de las dos tablas.

Dentro de las sentencias SQL podemos incluir además:

Funciones de agregación:

SUM([ALL DISTINCT] expresión)	Calcula el total de una expresión numérica para todas las filas o sólo las distintas.
AVG([ALL DISTINCT] expresión)	Calcula el promedio de una expresión numérica para las filas involucradas.
MIN([ALL DISTINCT] expresión)	Calcula el mínimo valor de una expresión numérica para las filas involucradas
MAX([ALL DISTINCT] expresión)	Calcula el máximo valor de una expresión numérica para las filas involucradas.
COUNT([ALL DISTINCT] expresión)	Cuenta el número de veces que se repite el valor de la expresión.
COUNT(*)	Cuenta las filas seleccionadas

Operadores lógicos: AND, OR, NOT

Operadores matemáticos:

Operator	Description	Example	Result
+	addition	2 + 3	5
-	subtraction	2 - 3	-1
*	multiplication	2 * 3	6
/	division (integer division truncates the result)	4 / 2	2
%	modulo (remainder)	5 % 4	1
^	exponentiation	2.0 ^ 3.0	8
/	square root	/ 25.0	5
/	cube root	/ 27.0	3
!	factorial	5 !	120
!!	factorial (prefix operator)	!! 5	120
@	absolute value	@ -5.0	5
&	bitwise AND	91 & 15	11
	bitwise OR	32 3	35
#	bitwise XOR	17 # 5	20
~	bitwise NOT	~1	-2
<<	bitwise shift left	1 << 4	16
>>	bitwise shift right	8 >> 2	2

Operadores de comparación:

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
=	equal
<> or !=	not equal

Funciones matemáticas:

Function	Function	Description
<code>abs(x)</code>	<code>acos(x)</code>	inverse cosine
<code>cbrt(dp)</code>	<code>asin(x)</code>	inverse sine
<code>ceil(dp or numeric)</code>	<code>atan(x)</code>	inverse tangent
<code>ceiling(dp or numeric)</code>	<code>atan2(y, x)</code>	inverse tangent of y/x
<code>degrees(dp)</code>	<code>cos(x)</code>	cosine
<code>div(y numeric, x numeric)</code>	<code>cot(x)</code>	cotangent
<code>exp(dp or numeric)</code>	<code>sin(x)</code>	sine
<code>floor(dp or numeric)</code>	<code>tan(x)</code>	tangent
<code>ln(dp or numeric)</code>		
<code>log(dp or numeric)</code>		
<code>log(b numeric, x numeric)</code>		
<code>mod(y, x)</code>		
<code>pi()</code>		
<code>power(a dp, b dp)</code>		
<code>power(a numeric, b numeric)</code>		
<code>radians(dp)</code>		
<code>random()</code>		

Funciones para operar con cadenas de caracteres:

Function	Return Type	Description
<code>string string</code>	<code>text</code>	String concatenation
<code>string non-string or non-string string</code>	<code>text</code>	String concatenation with one non-string input
<code>bit_length(string)</code>	<code>int</code>	Number of bits in string
<code>char_length(string) or character_length(string)</code>	<code>int</code>	Number of characters in string
<code>lower(string)</code>	<code>text</code>	Convert string to lower case
<code>octet_length(string)</code>	<code>int</code>	Number of bytes in string
<code>overlay(string placing string from int [for int])</code>	<code>text</code>	Replace substring
<code>position(substring in string)</code>	<code>int</code>	Location of specified substring
<code>substring(string [from int] [for int])</code>	<code>text</code>	Extract substring
<code>substring(string from pattern)</code>	<code>text</code>	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.
<code>substring(string from pattern for escape)</code>	<code>text</code>	Extract substring matchingSQLregular expression. See Section 9.7 for more information on pattern matching.
<code>trim([leading trailing both] [characters] from string)</code>	<code>text</code>	Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string
<code>upper(string)</code>	<code>text</code>	Convert string to uppercase

Funciones para operar con fechas:

Function	Return Type		
age(timestamp, timestamp)	interval		
age(timestamp)	interval		
clock_timestamp()	timestamp with time zone		
current_date	date		
current_time	time with time zone		
current_timestamp	timestamp with time zone		
		justify_days(interval)	interval
		justify_hours(interval)	interval
		justify_interval(interval)	interval
		localtime	time
		localtimestamp	timestamp
		now()	timestamp with time zone
		statement_timestamp()	timestamp with time zone
		timeofday()	text
		transaction_timestamp()	timestamp with time zone

¿PostgreSQL es adecuado para implementar en él un Datawarehouse?

Sí, debido a que PostgreSQL está apegado a las normas ACID y SQL92, además es totalmente integrable a herramientas externas para la carga y el análisis de los datos.

Herramientas para proyectos de Inteligencia de Negocios

Actualmente existen en el mercado varias opciones tanto de software abierto como de software propietario. A continuación describiremos algunas de las suites más importantes.



Open Source BI suites:

- Talend Open Studio
- Eclipse BIRT
- JasperSoft
- Palo
- SpagoBI*
- Pentaho Data Integration*
- CloverETL Community*

BI Suites de código cerrado:

- Oracle Warehouse Builder (OWB)
- SAP Data Integrator
- Microsoft DTS
- Data Studio

*También existen versiones comerciales

Talend Open Studio

Talend fue fundada en 2005 por Bertrand Diard y Fabrice Bonan. Fue el primer vendedor de software de Integración de Datos open source.

El Talend Open Studio for Data Integration se liberó en octubre de 2006, para enero de 2012 se habían hecho 20 millones de descargas y la compañía tiene más de 3500 clientes en todo el mundo.

Talend es patrocinador de la Apache Software Foundation y otros proyectos de open source.

Talend publica el código de sus core modules bajo la licencia pública GNU, y vende funcionalidades adicionales (monitoreo, balanceo de cargas) y servicios bajo una licencia de suscripción, a esto se le denomina “modelo de negocios open core”.

Eclipse BIRT

Corre en la plataforma Eclipse IDE, es muy flexible y ampliamente utilizado en el ambiente de reportes, pero se debe usar la versión de pago para obtener más. El motor de BIRT es una colección de clases y APIs de Java que permiten hacer reportes con cálculos, agrupamientos, tablas cruzadas y gráficas. La versión de paga incluye BIRT Design, BIRT iHub y BIRT User Experience, que brindan diseño de reportes, soporte móvil y visualizadores y analizadores de datos. Los diseños de informes se hacen en XML. El proyecto es soportado por los desarrolladores de Eclipse.org

JasperSoft

Existen varias versiones de JasperSoft BI Suite, la nueva versión se basa en la plataforma Oracle NetBeans y el Jaspersoft Studio es un plugin para el IDE (Integrated development environment) de Eclipse. La edición Community provee un ambiente de reportes y gráficas, y la versión comercial se paga anualmente, aunque también se puede utilizar por horas a través de Amazon.

Palo

Palo suite (OLAP Server, Palo Web, Palo ETL Server y Palo Excel) de la firma alemana Jedox, es un motor multidimensional diseñado para trabajar sobre grandes hojas de cálculo en Excel u Open Office. Es totalmente gratuito pero es una solución limitada.

SpagoBi

Es una plataforma de integración con estructura modular, integra motores de análisis específicos para cada área: informes y reportes, minería de datos, tableros o dashboards, herramientas ETL, etc. y se adapta para su integración con soluciones propietarias. No existe una versión propietaria de Spago Bi, es un Open Source empresarial.

CloverETL

Desarrollado en Java, con una plataforma independiente, está diseñado para la transformación, limpieza y la distribución de datos para aplicaciones, bases de datos y data warehouses. Es una herramienta ETL con una versión Open Source y otra versión comercial. Las gráficas, los metadatos y las conexiones se almacenan en el formato XML.

Pentaho

Entre las herramientas ETL de software libre disponibles en el mercado, Pentaho ha incrementado muy rápidamente su funcionalidad en los últimos dos años, convirtiéndose así en el mayor competidor de los softwares propietarios (ref. <http://www.etltool.com/open-source-etl-tools/>)

Pentaho es un conjunto programas libres que abordan las distintas fases de la BI (Inteligencia de Negocios). La principal diferencia entre la versión comercial y la libre, es que la primera tiene integradas todas sus funciones, en tanto que la segunda consta de componentes dedicados que deben instalarse y configurarse por separado.

Los componentes más comunes de Pentaho son:

- Pentaho BI Server (Pentaho Administration Console)
- Pentaho Data Integration (Kettle)
- Pentaho Report Designer
- Pentaho Design Studio
- Mondrian Schema Workbench
- Weka Data Mining
- Pentaho Dashboard Editor

Existen otros componentes en la versión comunitaria (libre) que se han desarrollado por equipos independientes de Pentaho y que hacen las mismas funciones o parecidas que los componentes de la versión “oficial”.

La versión community de Pentaho y toda la documentación disponible asociada se puede descargar de la página oficial del proyecto:

<http://wiki.pentaho.com/display/COM/Community+Wiki+Home>

The screenshot shows the homepage of the Pentaho Community Wiki. At the top, there's a navigation bar with links for 'View', 'Edit', and 'Log In'. Below the header, there's a section titled 'Community Wiki Home' featuring the Pentaho logo. A large text block welcomes visitors to the wiki, mentioning the Pentaho Open Source BI Suite Community Edition (CE) and its various components like Kettle, BI Platform, Mondrian, Reporting, Weka, CDE, CBF, Saiku Analytics, Flash Charts, Commons, Data Access, Agile BI, Big Data, and Product Management. Each component has a link to its documentation, how-to guides, forums, and FAQ/JIRA pages. To the right of this main content area is a sidebar titled 'Resources' containing a list of links related to development, documentation, forums, and community engagement.

Welcome to the Pentaho Community wiki. This wiki contains documentation and information for the Pentaho Open Source BI Suite Community Edition (CE). The suite includes ETL, OLAP analysis, metadata, data mining, reporting, dashboards and a platform that allows you to create complex solutions to business problems. The Pentaho community is an extraordinary group of people with many different talents who are dedicated to delivering a complete, well integrated, and high quality suite of business intelligence software. By using Pentaho CE software, you are also participating in the Pentaho community and contributing to an open source project.

Expectations - If you are unfamiliar with open source, this article is a good place to start. The open source community thrives on participation and cooperation. There are several communication channels available where people can help you, but they are not obligated to do so. You are responsible for your own success which will require time, effort and a small amount technical ability. If you prefer to have a relationship with a known vendor who will answer questions over the phone, help you during your evaluation and support you in production, please visit www.pentaho.com.

Project Launch Pad

Project Home	Documentation	How To	Forum	FAQ	JIRA
Kettle	Home - es ko ru	Tech Tips	User - Dev	FAQ	JIRA
BI Platform	Home	Tech Tips	User - Dev	FAQ	JIRA
Mondrian	Home - Wiki	Tech Tips	User - Dev	FAQ	JIRA
Reporting	Home	Tech Tips	User - Dev	FAQ	JIRA
Weka	Home	Tech Tips	Forum	FAQ	JIRA
CDE	Home	Gallery	Forum	FAQ	JIRA
CBF	Home			FAQ	
Saiku Analytics	Home	Live Demo	Forum	REDMINE	
Flash Charts	Home		Dev	Google	
Commons	Wiki			JIRA	
Data Access	Wiki				
Agile BI	Wiki		Dev		
Big Data	Wiki				
Product Management	Wiki				

Resources

- [In Development](#) - What's next
- [Downloads](#) - Get the code
- [CI Builds](#) - Last Dev Build (unstable)
- [FAQ](#) - List of FAQs
- [Community Forums](#) - Ask questions
- [IRC irc freenode.net #pentaho](#) - Live Chat
- [IRC Office Hours](#) - Chat with Pentaho Devs
- [Pentaho Books](#) - Read all about it
- [Community Events](#) - Collaborate
- [Demos and more](#) - Magic from pre-sales
- [JIRA](#) - Report bugs, track changes
- [Case Tracking Process](#) - More detail
- [Wiki Doc Guide](#) - Writing great pages
- [Java Doc](#) - API documentation
- [Ways to Contribute](#) - Get involved
- [Contribution Process](#) - How to contribute
- [Active Contributors](#) - Be a rock star
- [Open Projects](#) - Fun sized projects
- [Community Logo](#) - Show your support
- [Engineering](#) - Development resources
- [UI Design](#) - Pixel Management
- [Bee Keeper](#) - Pentaho open source model
- [Open Scrum](#) - Pentaho Dev model

Prerrequisitos

Linux distribuciones Red Hat

- Conectarse como usuario root
- Instalar el Java Runtime Environment JRE: `rpm -ivh jre-7u7-linux-x64.rpm`
- Configurar la variable de entorno JAVA_HOME:
`sudo su root -c "echo 'export JAVA_HOME=/usr/lib/jvm/jre-7u7-linux-x64'" >> /etc/environment"`
- Verificar la instalación del JRE: `java -version`
- Editar el archivo environment: `gedit /etc/environment`
`export PENTAHO_JAVA_HOME=/usr/lib/jvm/java-7-sun`
- Verificar la instalación del JRE: `env | grep PENTAHO_JAVA_HOME`
- Crear una base de datos en blanco en cualquier motor de base de datos

Linux distribuciones Ubuntu

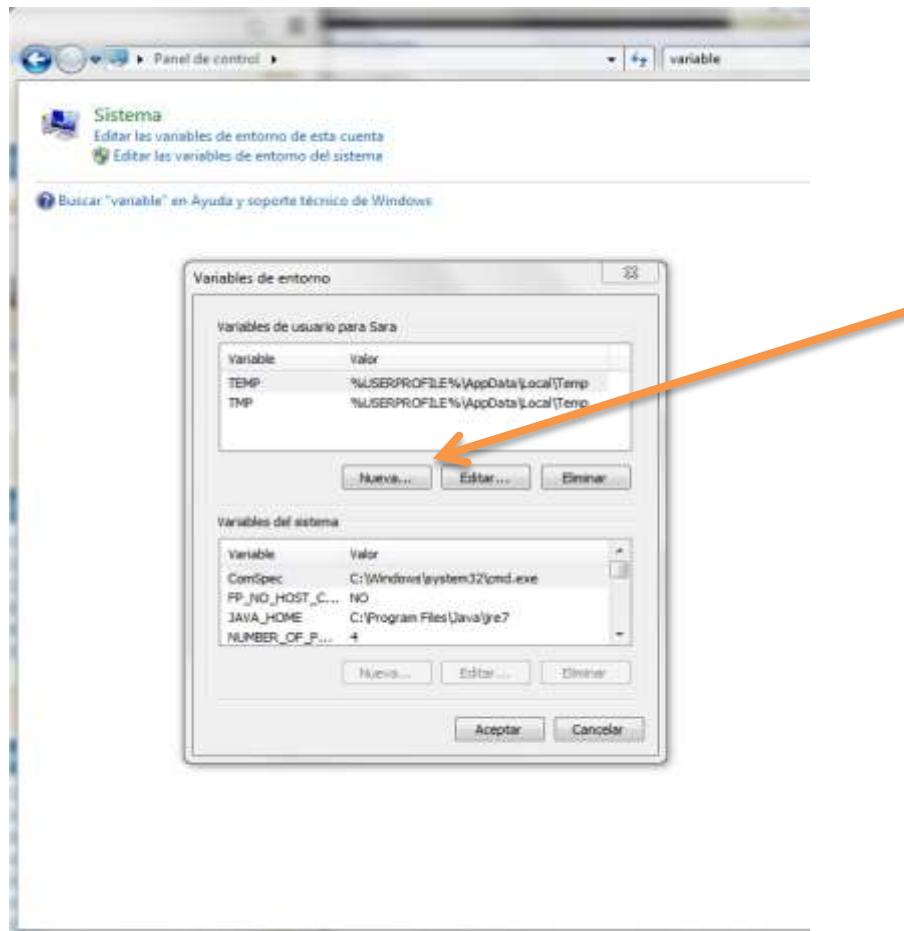
- Instalar el Java Runtime Environment JRE: `sudo apt-get install openjdk-6-jdk`
- Configurar la variable de entorno JAVA_HOME:
`sudo su root -c "echo 'export JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-amd64'" >> /etc/environment"`
- Verificar la instalación del JRE: `java -version`
- Editar el archivo environment: `gedit /etc/environment`
`export PENTAHO_JAVA_HOME=/usr/lib/jvm/java-7-sun`
- Verificar la instalación del JRE: `env | grep PENTAHO_JAVA_HOME`
- Crear una base de datos en blanco en cualquier motor de base de datos

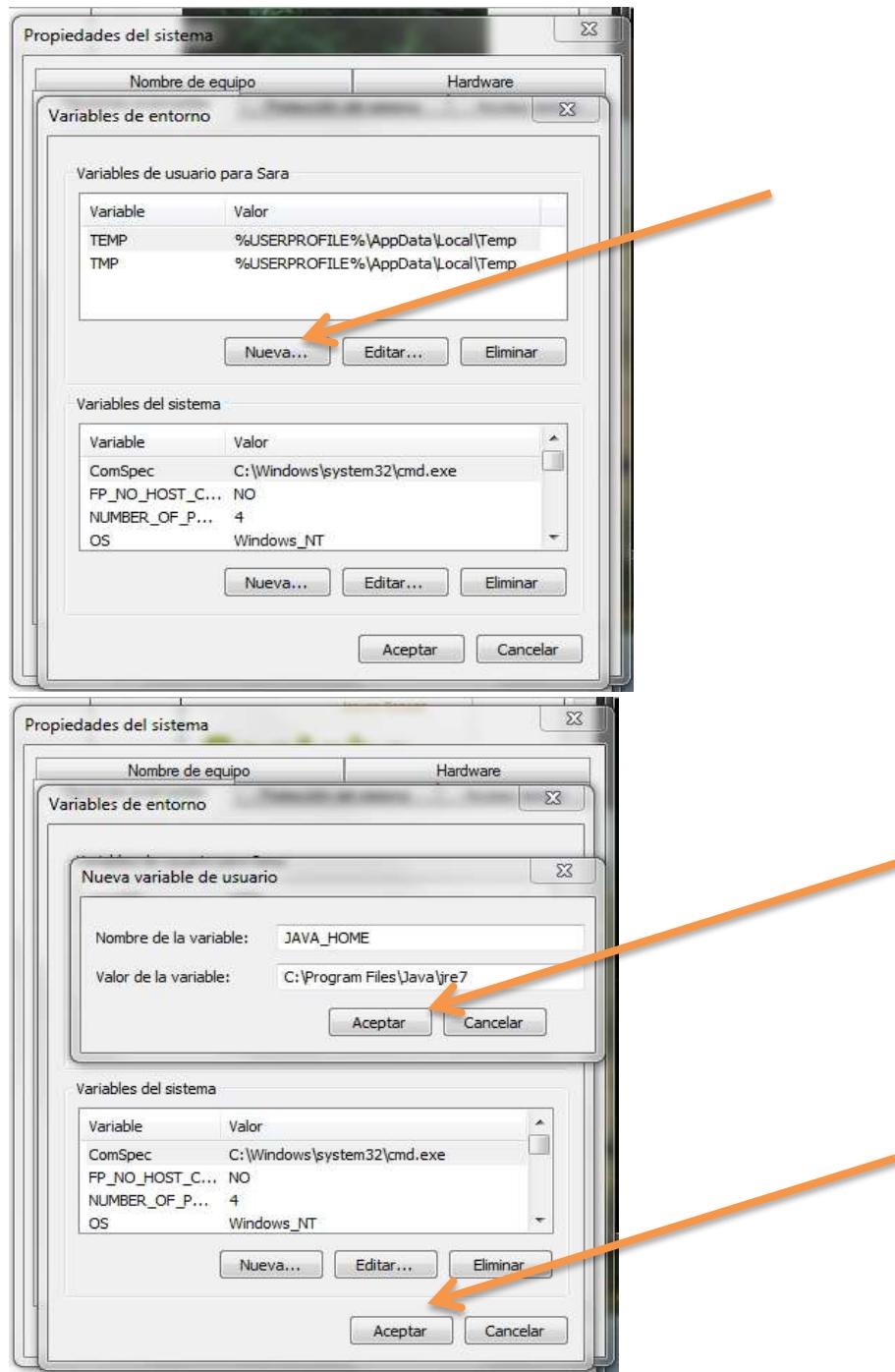
Windows

- Instalar el Java Runtime Environment JRE



- Configurar la variable de entorno JAVA_HOME





- Crear una base de datos en blanco en cualquier motor de bases de datos

BI SERVER CE

Es el módulo de administración de Pentaho, incluye entre otros componentes:

- **Pentaho Administration Console (PAC)** Permite la administración de los módulos de Pentaho. El PAC simplifica las tareas administrativas como la creación de usuarios, la asignación de roles, la conexión de bases de datos y la gestión de servicios, permite automatizar tareas. Se conecta con el PRD y puede generar reportes de manera periódica.
 - **Pentaho User Console (PUC)** Tiene el **CDE (Community Dashboard Editor)** el cual facilita las vistas de reportes y análisis. Se conecta con el PAC, el PRD y el Editor de Metadatos directamente.
 - **Community Dashboard Editor (CDE)** El CDE es un editor de cuadros de mando a través de una interfaz gráfica web. Es un proyecto independiente que se desarrolla por una empresa portuguesa liderada por Pedro Alves. Sus antecedentes son el CDF (Community Dashboard Framework), CDA (Community Data Access) y CCC (Community Chart Component). Actualmente el CDE forma parte del paquete de instalación de la PUC. Para crear un cuadro de mandos necesitamos definir cuáles indicadores (PKIs) nos interesa mostrar y cómo se desea que se visualicen.
 - **Jpivot** Una vez definidos los modelos dimensionales, podemos ejecutar nuestros análisis utilizando Jpivot a nivel de interfaz de usuario y Mondrian a nivel del servidor (es la parte que recibe las solicitudes de información de Jpivot, realiza las consultas contra la base de datos y devuelve la información en formato multidimensional).

Actualmente se han integrado muchas funcionalidades a este módulo, entre ellas la posibilidad de actualizar y agregar otros módulos al proyecto ya sean los oficiales o no a través de su interface, también permite la programación de tareas y la administración de usuarios y permisos.



Promoting innovation, participation, and cooperation

This release includes the Pentaho Marketplace where community members can explore and install community-developed Pentaho plugins with just one click.



Instalación

Se descarga el archivo comprimido biserver-ce-X.X.X-XXX.zip con la versión del módulo de administración Community Edition.

Se descomprime el archivo .zip donde se quiere ubicar el biserver.

Para trabajar en la plataforma Linux se debe ejecutar el archivo: /biserver-ce/start-pentaho.sh

Para trabajar en la plataforma Windows se debe ejecutar el archivo:
\biserver-ce\start-pentaho.bat



Para ingresar al PAC (Pentaho Administration Console) se abre un navegador de internet y se ingresa la dirección:

<http://localhost:8080/pentaho/Login>

Usuario: admin Password: password

PDI

En el año 2001, el belga Matt Casters empezó el desarrollo de una herramienta para uso personal, dadas las dificultades que había tenido durante su experiencia laboral como constructor de Data Warehouses para la integración de sistemas.

Durante los siguientes años, fue desarrollando la herramienta utilizando Java y su librería gráfica AWT, para finalmente pasar a la SWT.

La herramienta fue creciendo en funcionalidades: acceso a bases de datos, tratamiento de ficheros y componentes.

En el año 2004 se libera la versión 1.2. El proyecto fue publicado en Javaforge para su descarga como una herramienta Open Source.

En la versión 2.0 se incluyó un sistema de plugins para permitir el desarrollo de conectores de Kettle con otros sistemas (como SAP).

En el año 2005 fue liberado el código y puesto a disposición de todos en Javaforge. A partir de entonces, el proyecto creció con rapidez con una comunidad muy activa.

Casters vendió Kettle a Pentaho en el año 2006, para ser incluido como su herramienta ETL en su suite de productos. Desde entonces, Casters ha continuado trabajando en Pentaho como parte del equipo de desarrollo de su arquitectura.

El nombre de Kettle viene de KDE Extraction, Transportation, Transformation and Loading Environment, pues originariamente la herramienta iba a ser escrita para KDE, el famoso escritorio de Linux.

Data Integration - Kettle

Data Integration (or Kettle) delivers powerful Extraction, Transformation, and Loading (ETL) capabilities, using a groundbreaking, metadata-driven approach.



El producto ha sido renombrado como Pentaho Data Integration y a partir de ahora nos referiremos a él como PDI.

El Pentaho Data Integración se encarga del procesamiento ETL y a su vez está integrado por:

- **Spoon** (antes Kettle) es la interfaz gráfica de ETL que permite diseñar transformaciones y trabajos que serán ejecutados por las herramientas del PDI.
- **Pan** es un motor de transformación de datos que ejecuta las funciones de lectura, manipulación y escritura, es decir las transformaciones de datos, desde varias fuentes.
- **Kitchen** es el programa encargado de ejecutar los trabajos (jobs) diseñados en spoon con extensión XML o desde un repositorio. Se utiliza para ejecutar los jobs programados por lotes para que sean ejecutados en intervalos regulares de tiempo. Se ejecuta desde el sistema operativo.
- **Carte** esta herramienta permite ejecutar transformaciones y jobs de manera remota.

Instalación

Se descarga el archivo comprimido pdi-ce-X.X.X-XXX.zip con la versión del módulo de administración Community Edition.

Se descomprime el archivo .zip donde se quiere ubicar el pdi.

Para trabajar en la plataforma Linux se debe ejecutar el archivo: /data-integration/spoon.sh

Para trabajar en la plataforma Windows se debe ejecutar el archivo: \data-integration\spoon.bat



Usuario: admin

Password: password

PRD

Es la herramienta para la elaboración de reportes y puede generarlos en los formatos PDF, Excel, HTML, Texto, XML y CSV. Este editor permite integrar gráficas y tablas de consultas SQL sobre una BD para la presentación de nuestros reportes, así como imágenes (logos) y texto para dar una presentación profesional a los reportes para su publicación.

Reporting

Suite of Open Source reporting tools that enables the creation of relational and analytical reports from a wide range of data-sources.



El diseñador de reportes ofrece un entorno gráfico con herramientas intuitivas y fáciles de utilizar, el reporte tiene una estructura bien definida pero flexible para dar libertad en el diseño y generación de reportes que se adapten a las necesidades del cliente.

El diseñador es una herramienta “drag & drop” que cuenta con un asistente y plantillas de reportes precargadas.

Instalación

Se descarga el archivo comprimido prd-ce-X.X.X-XXX.zip con la versión del módulo de administración Community Edition.

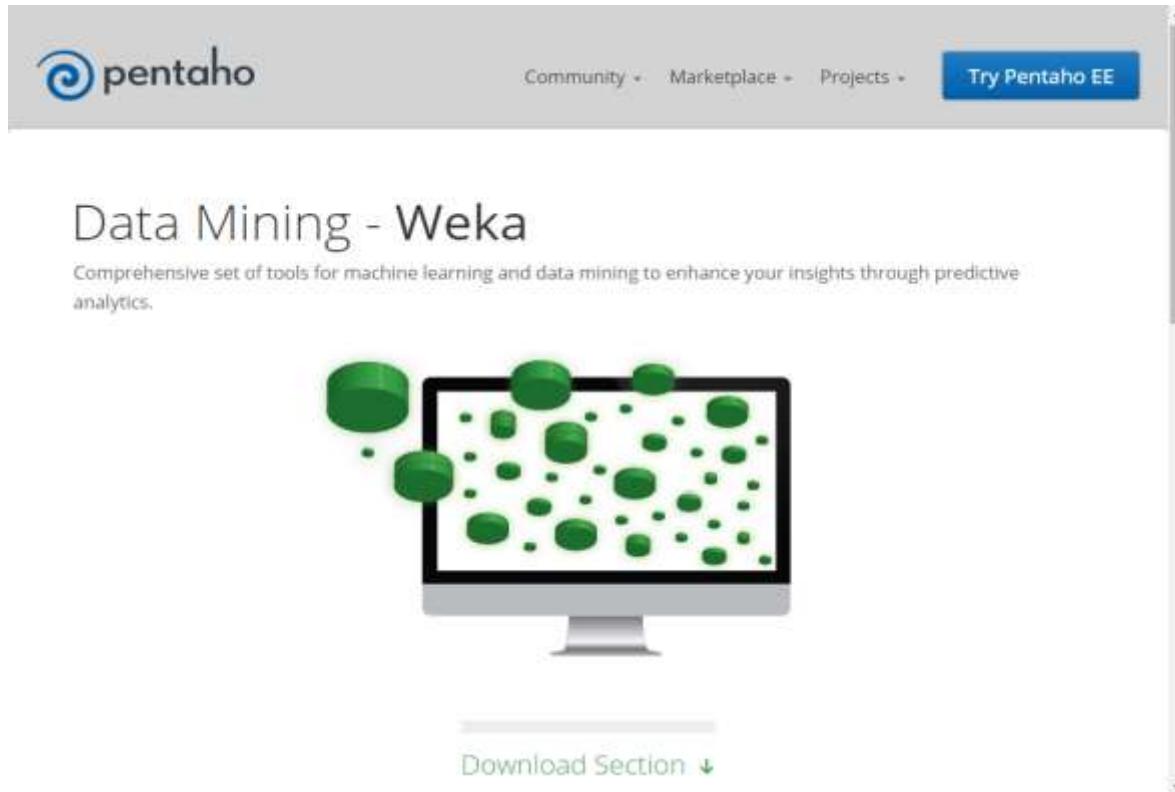
Se descomprime el archivo .zip donde se quiere ubicar el prd.

Para trabajar en la plataforma Linux se debe ejecutar el archivo: /report-designer/report-designer.sh

Para trabajar en la plataforma Windows se debe ejecutar el archivo: \report-designer\report-designer.bat

WEKA

Waikato Environment for Knowledge Analysis (Ambiente para Análisis del Conocimiento de la Universidad de Waikato).



Es un conjunto de herramientas de visualización y algoritmos que se enfoca a la minería de datos y el aprendizaje de máquina escrito en Java. Se puede seleccionar una de 4 interfaces:

- SimpleCli (Interfaz simple de línea de comandos)
- Explorer. Formado por varios páneles:
- Preprocess [importa datos de fuente y aplica algoritmos de filtrado y transformación de datos]
- Classify [Aplica algoritmos de clasificación estadística y análisis de regresión, estima la exactitud del modelo predictivo]
- Associate [Da acceso a las reglas de asociación aprendidas e intenta identificar todas las interrelaciones importantes entre los atributos de los datos]
- Cluster [Aplica técnicas de agrupamiento o clustering como el K-means]
- Selected attributes [Aplica algoritmos para identificar los atributos más predictivos de un conjunto de datos]
- Visualize [muestra los datos con una matriz de puntos dispersos “scatterplot” y permite la selección de puntos para su análisis individualizado]

- Experimenter (Permite la comparación sistemática de una ejecución de algoritmos predictivos de Weka sobre una colección de datos)
- Knowledge Flow (Como el anterior pero permite el aprendizaje incremental)

Instalación

Descargar el instalador y ejecutarlo siguiendo las instrucciones.



MONDRIAN

Actualmente se le está renombrando como Pentaho Analysis Services (PAS).

Es la herramienta para el diseño de cubos, consiste en un JAR (un archivo que permite ejecutar instrucciones escritas en lenguaje Java) que actúa como JDBC (Java Database Connectivity, una API que permite ejecutar operaciones sobre una BD en Java independientemente del SO en SQL nativo) para OLAP (Procesamiento Analítico en Línea, solución de BI para agilizar la consulta de grandes cantidades de datos), proporciona conexiones y ejecuta consultas SQL en una BD relacional. Los cubos se componen de archivos XML y se ejecutan sobre un servidor Web para permitir la comunicación entre aplicaciones OLAP con BDs.

Mondrian

Online Analytical Processing server (OLAP). Allows business users to analyze large and complex amounts of data in real-time.



Instalación

Se descarga el archivo comprimido psw-ce-X.X.X-XXX.zip con la versión del módulo de administración Community Edition.

Se descomprime el archivo .zip donde se quiere ubicar el Mondrian Schema Workbench.

Para trabajar en la plataforma Linux se debe ejecutar el archivo: /schema-workbench/workbench.sh

Para trabajar en la plataforma Windows se debe ejecutar el archivo:

\schema-workbench\workbench.bat

PENTAHO

Pentaho Data Integration



Pentaho Data Integration
Previously Kettle

Pentaho Data Integration

Pentaho es un conjunto de programas libres que abordan las distintas fases de la BI (Inteligencia de Negocios). La principal diferencia entre la versión comercial y la de pago, es que la primera tiene integradas todas sus funciones, en tanto que la segunda consta de componentes dedicados que deben instalarse y configurarse por separado:

Pentaho Data Integración se encarga del procesamiento ETL y a su vez está integrado por:

Spoon (antes Kettle) es la interfaz gráfica de ETL que permite diseñar transformaciones y trabajos que serán ejecutados por las herramientas del PDI

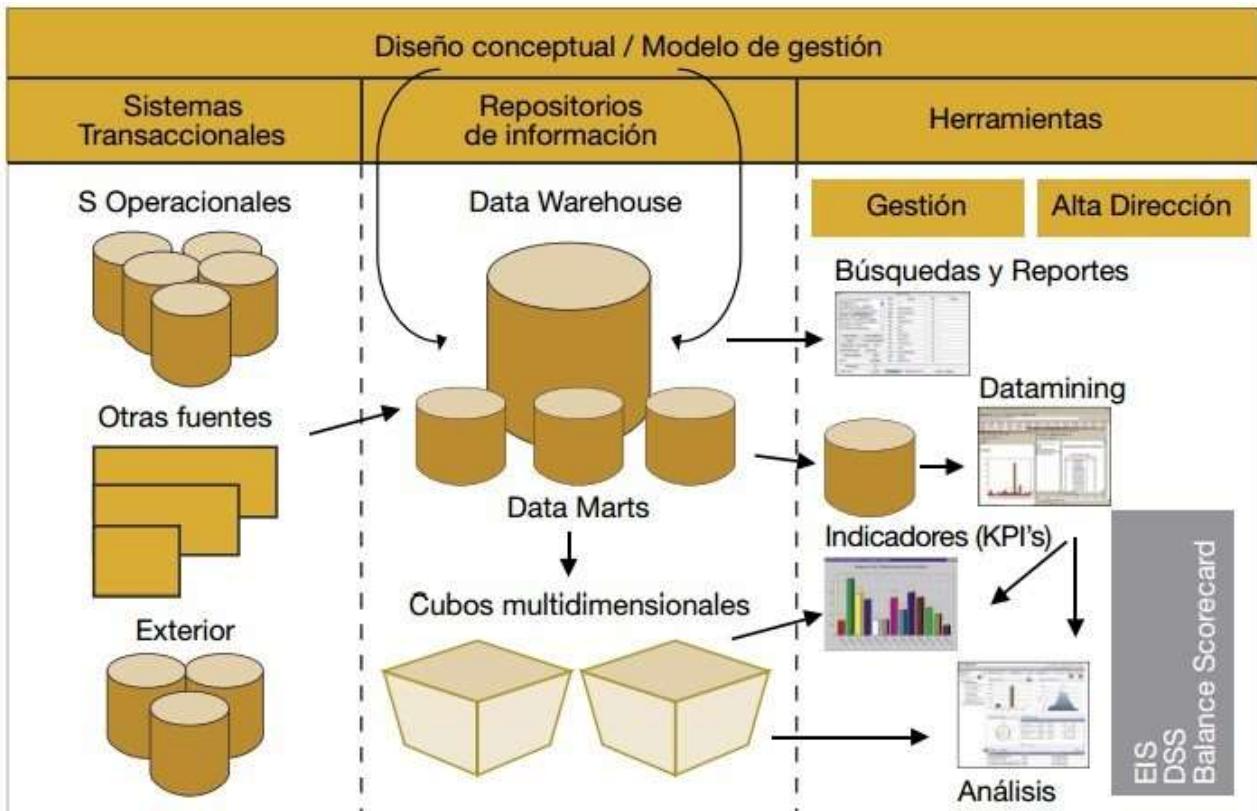
Pan es un motor de transformación de datos que ejecuta las funciones de lectura, manipulación y escritura, es decir las transformaciones de datos, desde varias fuentes.

Kitchen es el programa encargado de ejecutar los trabajos (jobs) diseñados en spoon con extensión XML o desde un repositorio. Se utiliza para ejecutar los jobs programados por lotes para que sean ejecutados en intervalos regulares de tiempo. Se ejecuta desde el sistema operativo.

Carte esta herramienta permite ejecutar transformaciones y jobs de manera remota.

Extracción, Transformación y Carga con el Pentaho Data Integration

Descripción del proceso ETL



Extract (Extracción)

Extracción de datos de los sistemas origen (BD Relacionales, BD no relacionales, archivos planos) sin importar el formato en que estén, se convierten a un formato preparado para hacer la transformación de los datos.

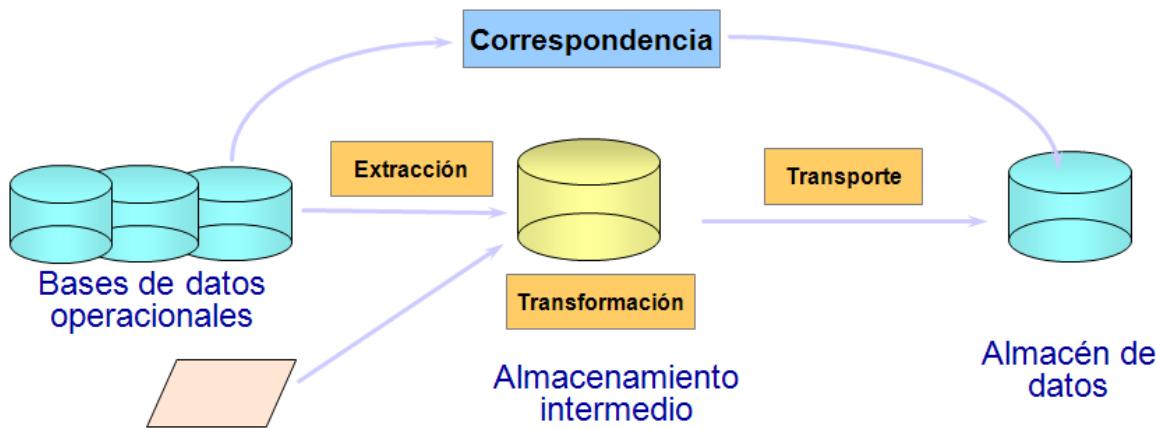
Consiste en la obtención de los datos de sus fuentes. Para facilitar su manipulación y no afectar el rendimiento de los sistemas de producción, los datos se cargan en una ODS provisionalmente y fuera del horario de la jornada laboral.

Una base de datos ODS (Operational Data Store) sirve para que en ella se efectúen las operaciones de transformación y validación previas a pasar los datos al Data Warehouse.

Los datos origen pueden estar en tablas de bases de datos o no, estos son los formatos soportados por Kettle:

- Bases de datos relacionales (SQL): PostgreSQL, MySQL, SQL Server, Oracle, etc.
- Sistemas ERP (Enterprise Resource Planning) como el SAP, Oracle financial, JD Edwards. A veces se requieren licencias para acceder a herramientas que nos comuniquen con ellos.
- Hojas de cálculo (Excel [.xls, .xlsx], Open Office [.csv])
- Bases de datos de escritorio (Access [.mdb, .accdb], Open Office Base, Fox Pro, DBase [.dbf], etc.)
- Datos estructurados externos
- Datos en formato XML
- Datos Online
- Archivos de texto separados por comas, pipes [|], espacios, etc.

Es importante que el impacto de la extracción sobre los sistemas origen sea nulo o mínimo (sobre todo en las BDs de producción para no interferir con la operación normal de la empresa). Por eso se eligen horarios fuera de la jornada laboral para hacer la extracción.



Transform (Transformación)

Son las operaciones necesarias para integrar los datos aplicando las reglas del negocio, es decir, son funciones que se aplican sobre los datos extraídos para:

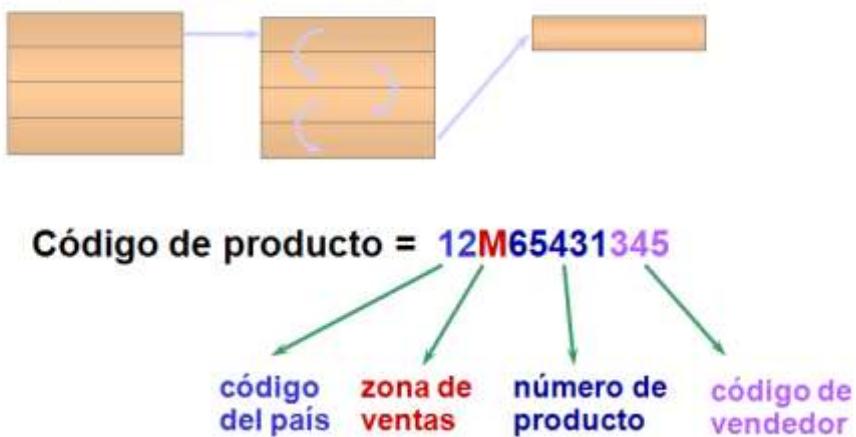
- selección de las columnas a cargar
- traducción de códigos (h, m → hombre, mujer; 1,2 → hombre, mujer)
- codificar valores (hombre → h)
- obtener nuevos valores calculados (consumo_gasolina = litros * precio)
- unir datos de múltiples fuentes

- calcular totales
- generar campos claves para la BD destino
- dividir columnas (nombre → nombres + apellido paterno + apellido materno)
- consolidar columnas (día + mes + año → fecha)
- crear índices
- trasponer o “pivotear” tablas (convertir columnas en renglones y viceversa)
- desagregación de columnas en diferentes tablas
- validación de datos: estructura del RFC, CURP
- manejo de datos erróneos: manejo de excepciones y datos nulos.

Ejemplos:

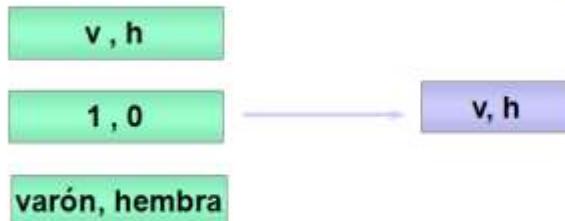
Transformación.

- Claves con estructura: descomponer en valores atómicos



Transformación.

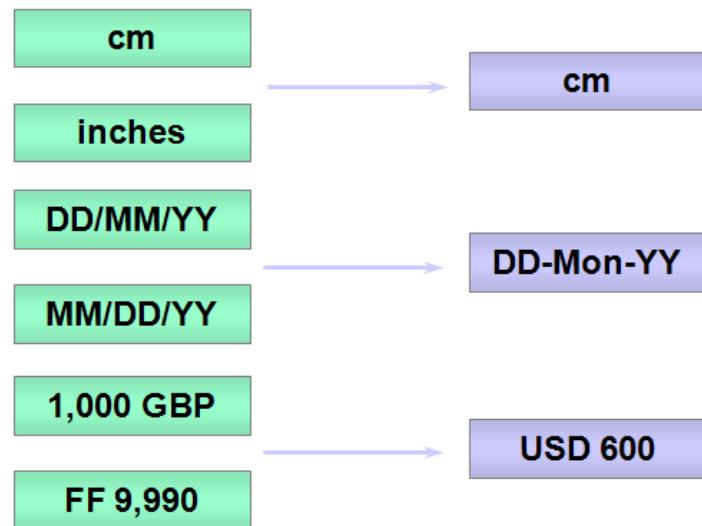
- Unificar codificaciones: existencia de codificaciones múltiples.



- Deben detectarse los valores erróneos.

Transformación.

- Unificar estándares: unidades de medida, unidades de tiempo, moneda,...



Transformación.

- Valores duplicados: deben ser eliminados.
 - SQL
 - restricciones en el SGBDR



Transformación.

- Integridad referencial: debe reconstruirse.

Departamento	Emp	Nombre	Departamento
10	1099	Smith	10
20	1289	Jones	20
30	1234	Doe	50
40	6786	Harris	60

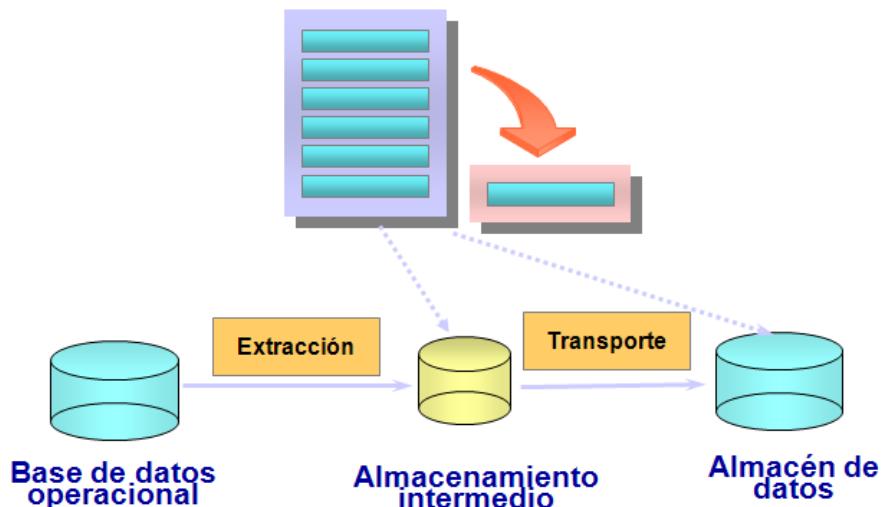
Load (carga)

En esta parte del proceso, es cuando se introducen los datos ya depurados en el sistema destino. Puede efectuarse de dos formas:

- **acumulación simple**: se acumulan las transacciones y se recalcularan los totales
- **rolling**: se usa cuando se quieren mantener “niveles de granularidad”, la información se almacena en niveles jerárquicos o varias dimensiones (totales diarios, semanales, mensuales, etc.)

Procesos posteriores a la carga: obtención de agregados.

- Durante la extracción.
- Despues de la carga (transporte).



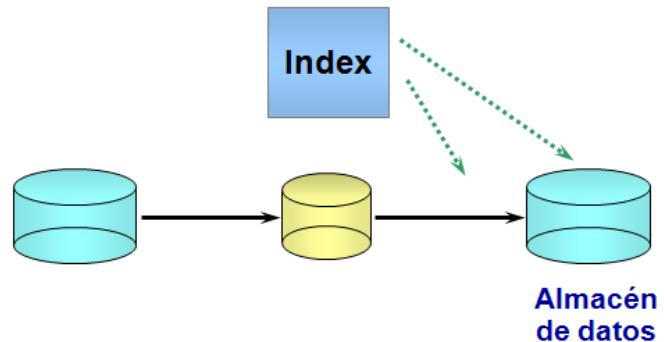
Al efectuarse la carga se activan todas las restricciones y triggers (disparadores) y se validan: valores únicos, integridad referencial (en su caso), campos obligatorios, rango de valores, etc.

Las herramientas ETL permiten el procesamiento en paralelo puntos de sincronización y puntos de actualización.

La carga de los datos, dependiendo de sus características y volumen puede darse en paralelo o secuencialmente.

Procesos posteriores a la carga: indexación.

- Durante la carga:
 - carga con el índice habilitado
 - proceso tupla a tupla. (lento)
- Despues de la carga:
 - carga con el índice deshabilitado
 - creación del índice (total o parcial). (rápido)



Comenzando a trabajar con el Pentaho Data Integration

Configuración del repositorio

La primera ocasión que se ingresa a Kettle (spoon.bat o spoon.sh) se ofrece la opción de configurar el repositorio. Se ofrecen dos opciones para el almacenamiento de nuestras transformaciones y trabajos.

Para configurar el repositorio, debe darse clic al botón "+" para agregar uno nuevo. Se muestra una ventana emergente donde se pregunta si se desea una base de datos o un sistema de archivos.

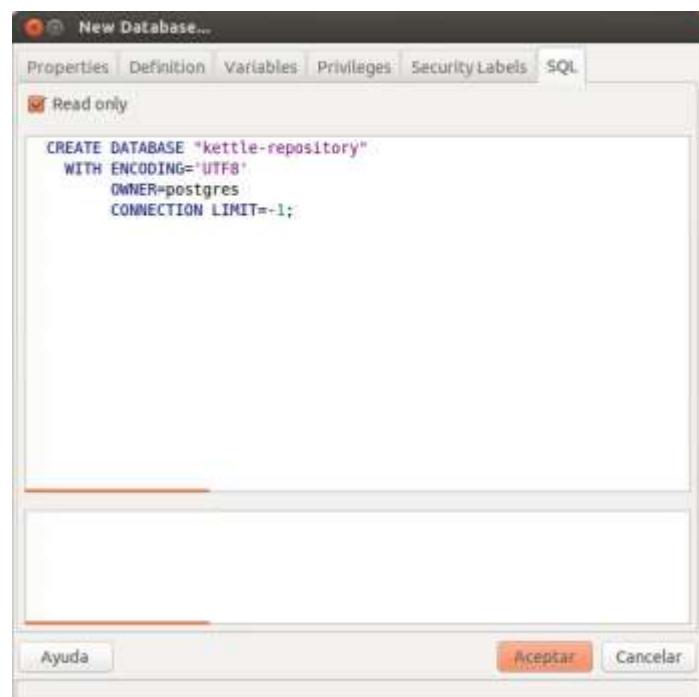
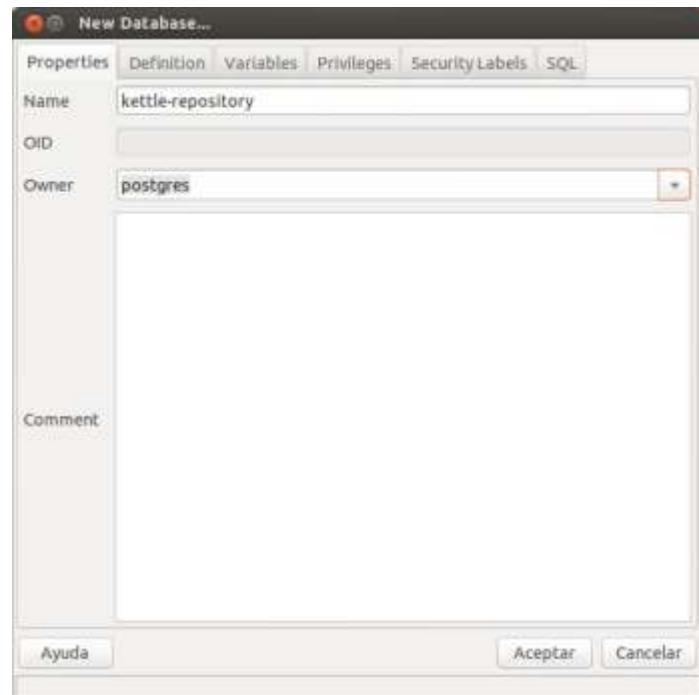


Una vez configurado el repositorio hay que desmarcar el cuadro que indica que queremos ver la configuración del repositorio cada vez que se inicie el PDI.

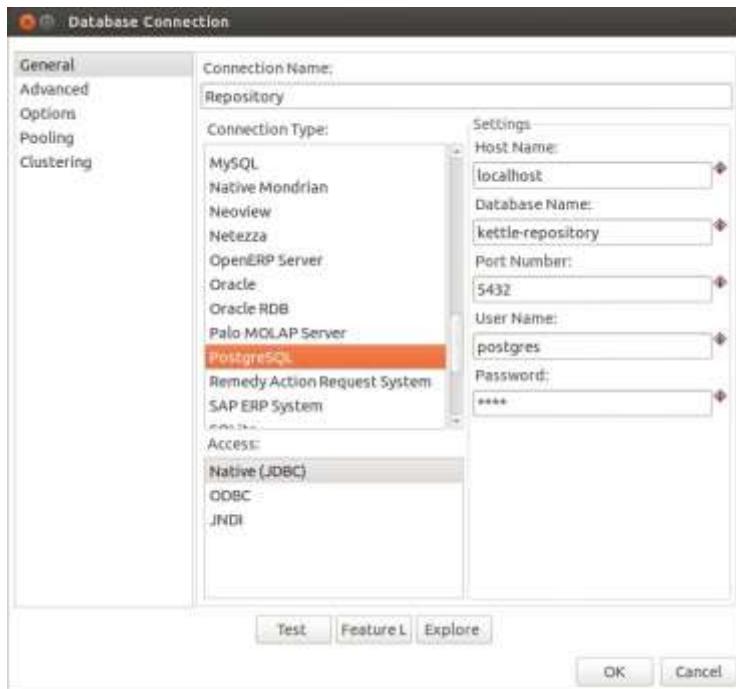


Repositorio en BD: Todo se almacena en una base de datos, con una estructura especial, donde son guardadas las transformaciones y trabajos construidos. Puede ser útil para el trabajo en equipo y para disponer de un lugar centralizado donde se va registrando todo lo realizado. Además se trata de una alternativa más segura (los archivos .ktr y .kjb no son compilados por lo que pueden ser editados manualmente).

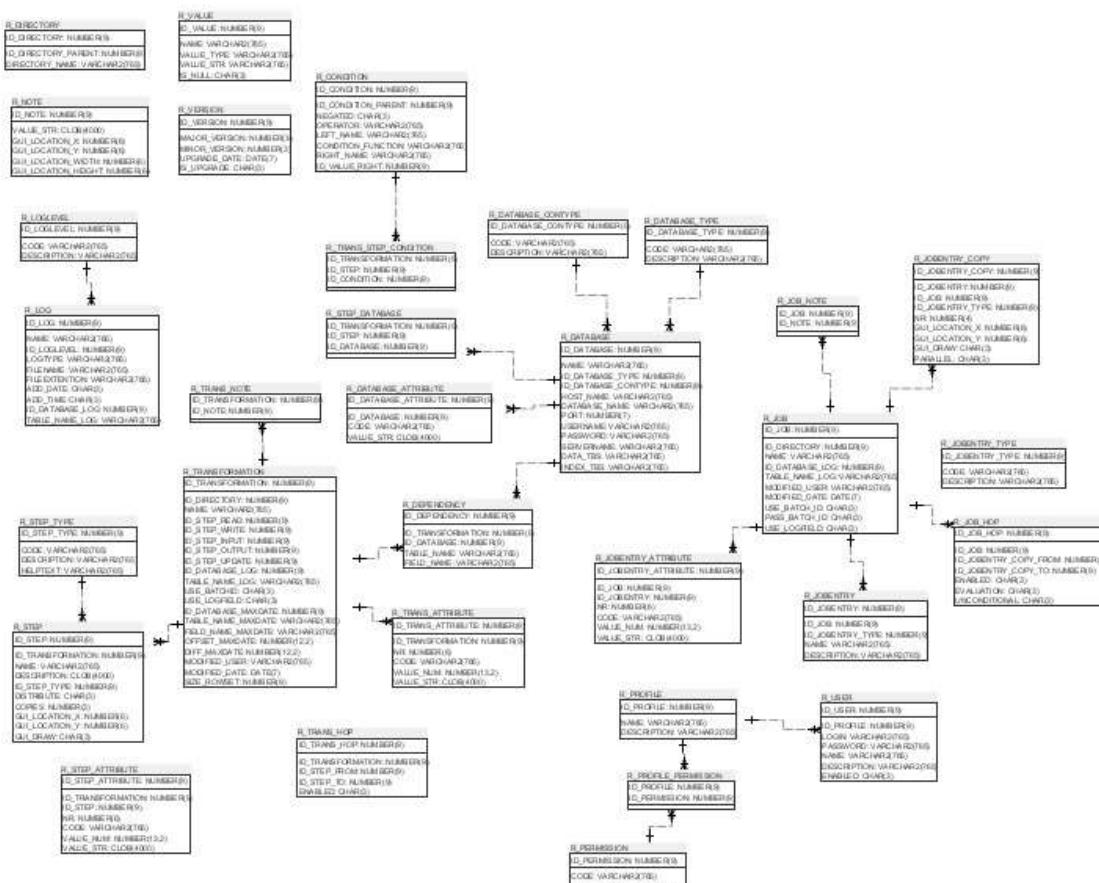
Para crear un repositorio en BD primero se debe crear una base de datos limpia en el pgAdmin.



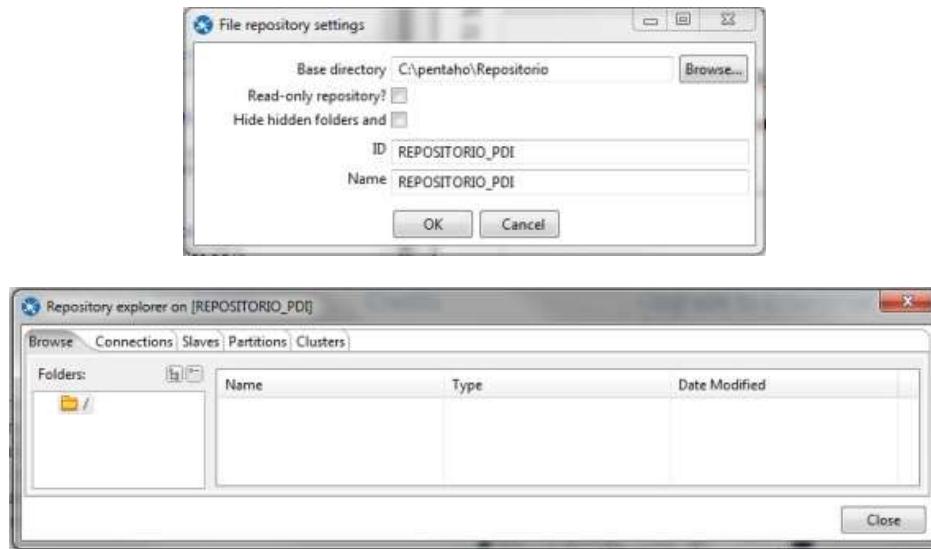
Luego se debe elegir la opción del repositorio de base de datos en el PDI y establecer la conexión con la base de datos que creamos para tal propósito.



Al terminar el proceso, el PDI habrá implementado la estructura de tablas a la base de datos que creamos.



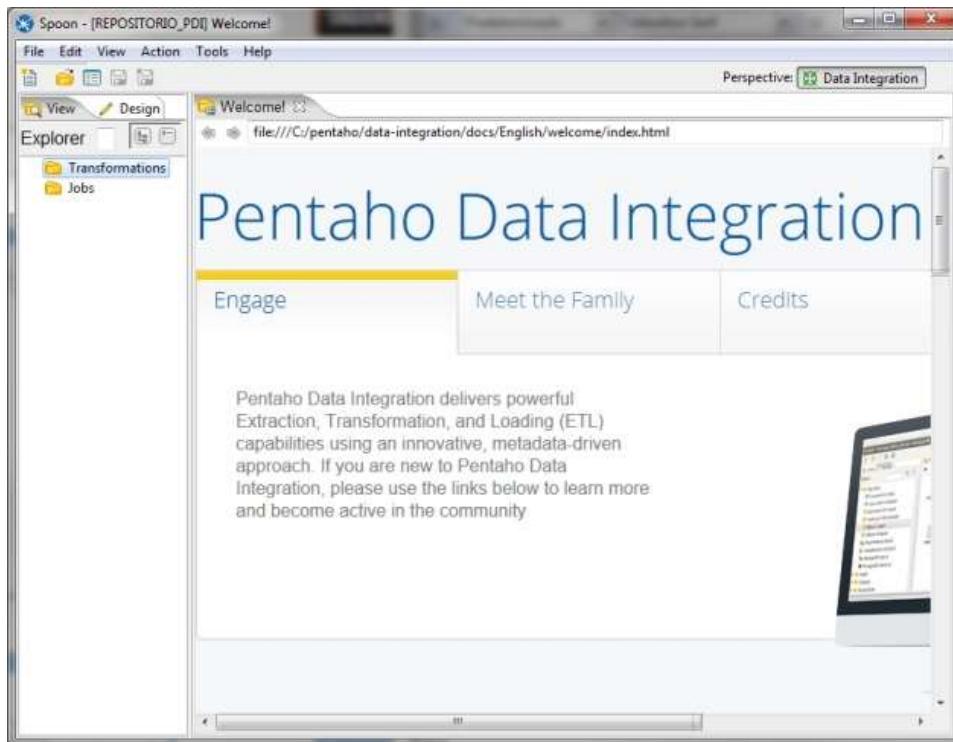
Repositorio en Archivos: las transformaciones y trabajos son guardados a nivel del sistema de ficheros, en archivos XML (con extensión .ktr para las transformaciones y .kjb para los jobs). Cada transformación y trabajo tiene un archivo asociado, que incluye en formato XML los metadatos (que definen su comportamiento).



Se debe crear una carpeta dedicada para almacenar los archivos de las transformaciones (.ktr) y los trabajos (.kjb).

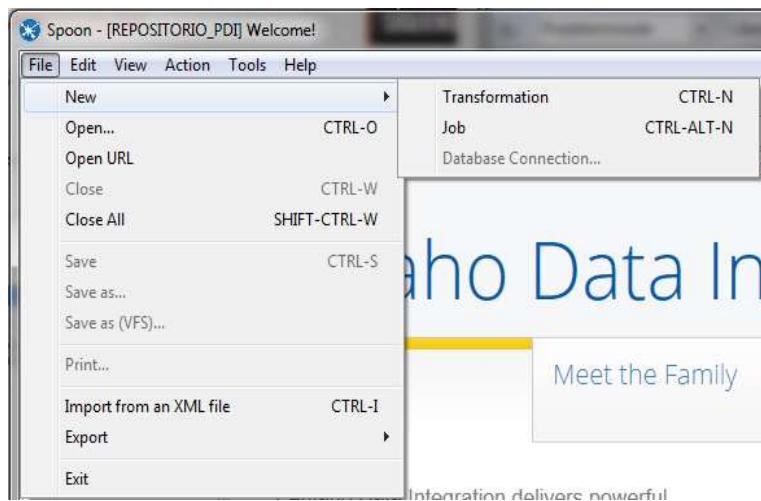
Menús

Una vez configurado el repositorio podemos comenzar a trabajar con el PDI.

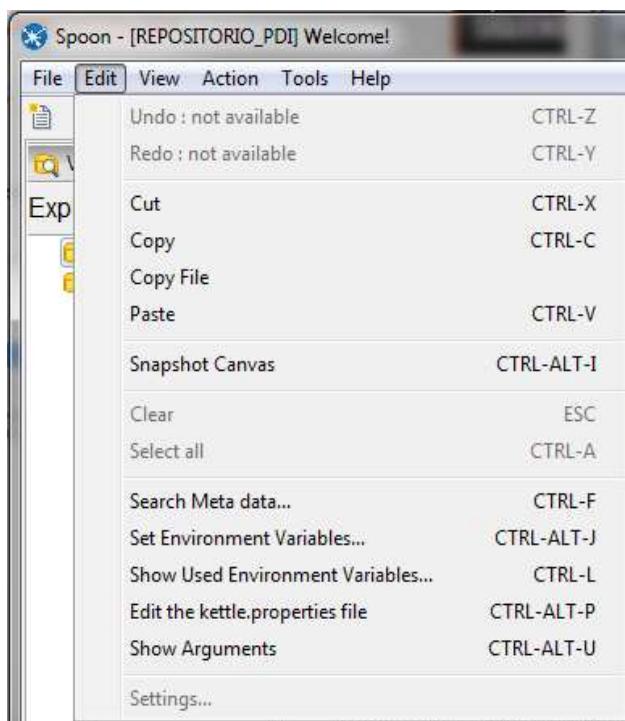


En la parte superior podemos ver una serie de menús: File, Edit, View, Action, Tools y Help.

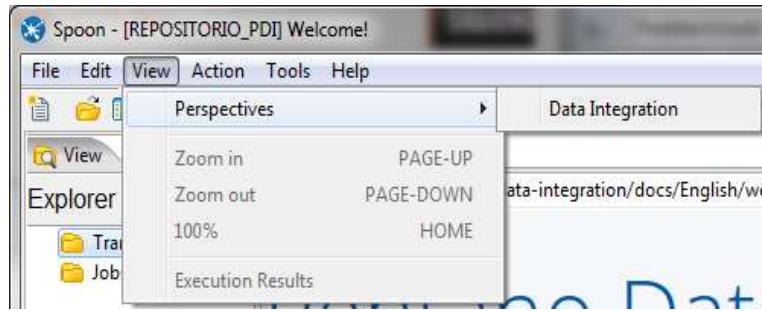
Menú File: Aquí se indica dónde se puede crear una nueva transformación o Job (trabajo).



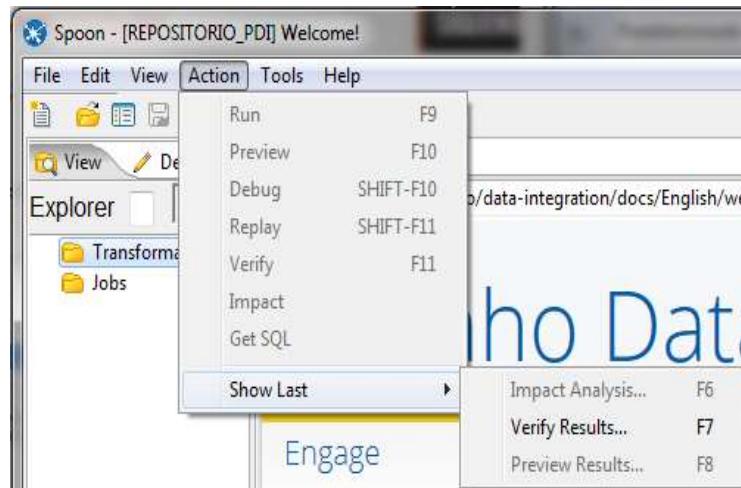
En el menú Edit se pueden copiar y pegar componentes, así como establecer las preferencias del entorno.



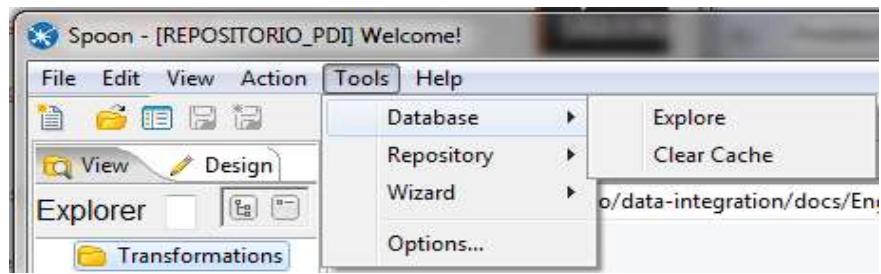
En el menú View se puede amplificar o disminuir la escala para visualizar una transformación o job.

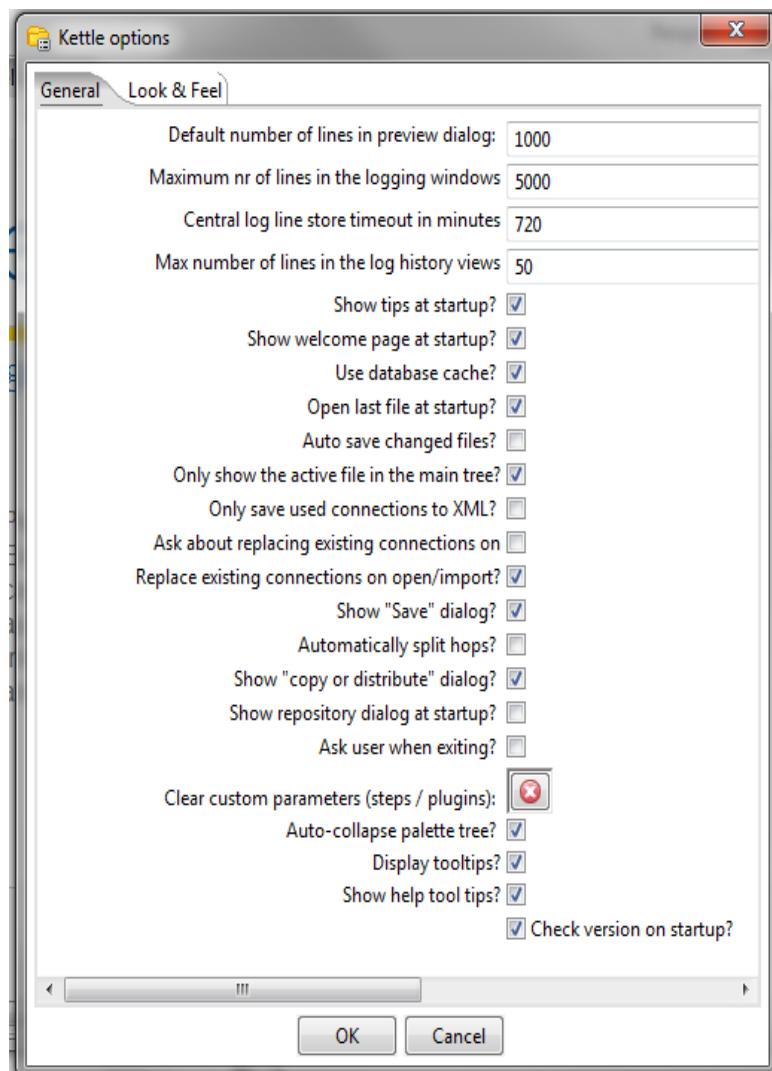
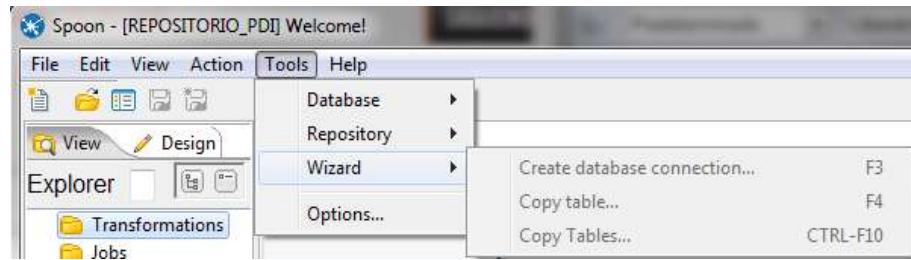


En el menú Action se puede ejecutar en modo normal o de depuración una transformación o job.



En el menú Tools se pueden establecer las conexiones a bases de datos para la carga/descarga de datos, reestablecer la configuración del repositorio y las opciones de personalización del PDI.



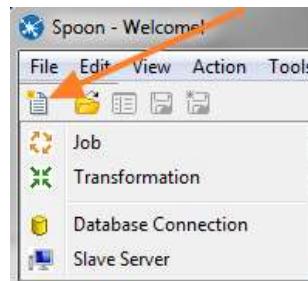


Icono	Descripción
	Crea un nuevo Trabajo o Transformación
	Abre una Transformación/Trabajo desde un archivo si no está conectado a un Catálogo o desde el Catálogo si se está conectado a uno.
	Guarda la Transformación/Trabajo en un archivo o en el Catálogo.
	Guarda la Transformación/Trabajo con un nombre diferente y/o en diferente lugar.
	Ejecuta la Transformación/Trabajo actual desde el archivo XML o Catálogo.
	Pone en pausa la ejecución de la actual Transformación.
	Detiene la ejecución de la actual Transformación/Trabajo.
	Vista previa de la Transformación: ejecuta la Transformación actual desde la memoria. Puede obtener una vista previa de las filas generadas por los pasos seleccionados.
	Ejecuta la Transformación en el modo de depuración, lo cual permite detectar problemas y/o errores en la ejecución.
	Repite el procesamiento de una Transformación para una determinada fecha y hora. Esto hará que algunos pasos (entrada archivo de texto y entrada Excel) sólo procesen las filas que no fueron interpretadas correctamente durante la ejecución en una fecha y hora en particular.
	Verifica la Transformación: Spoon ejecuta varias pruebas para cada paso para ver si todo va a funcionar como debería.
	Ejecuta un análisis de impacto: analiza qué impacto tendrá la Transformación sobre las bases de datos utilizadas.
	Genera el SQL necesario para ejecutar la actual Transformación/Trabajo.
	Abre el explorador de bases de datos y permite realizar una vista previa de los datos, ejecutar las consultas SQL, generar DDL (Lenguaje de Definición de Datos), etc.
	Muestra u oculta el panel de resultados de la ejecución de la Transformación/Trabajo.
	Representa el tamaño de visualización (en porcentaje) del área de trabajo.

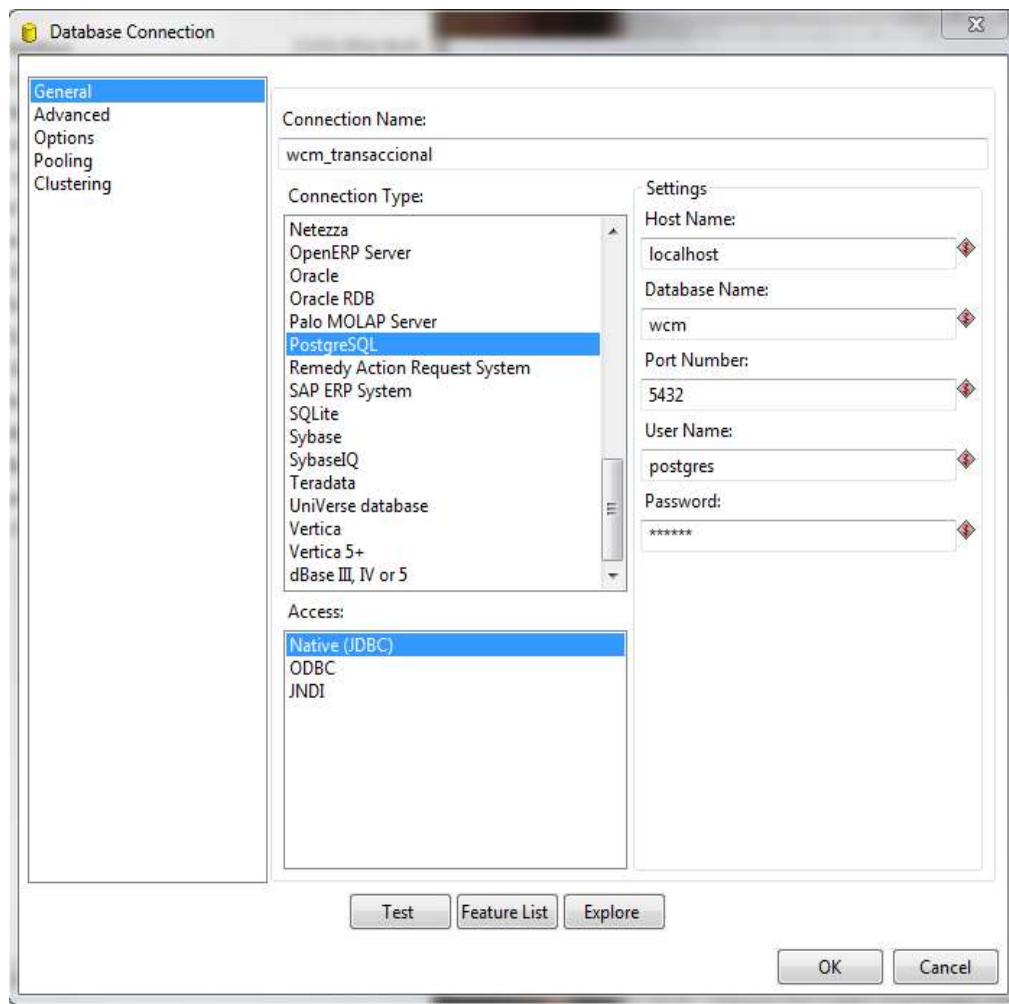
Tabla de iconos del PDI

Conexiones

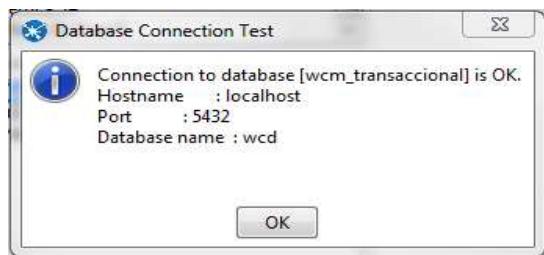
Para establecer una conexión a una base de datos, se puede dar clic en el ícono debajo de File y seleccionar la opción “Database Connection”.



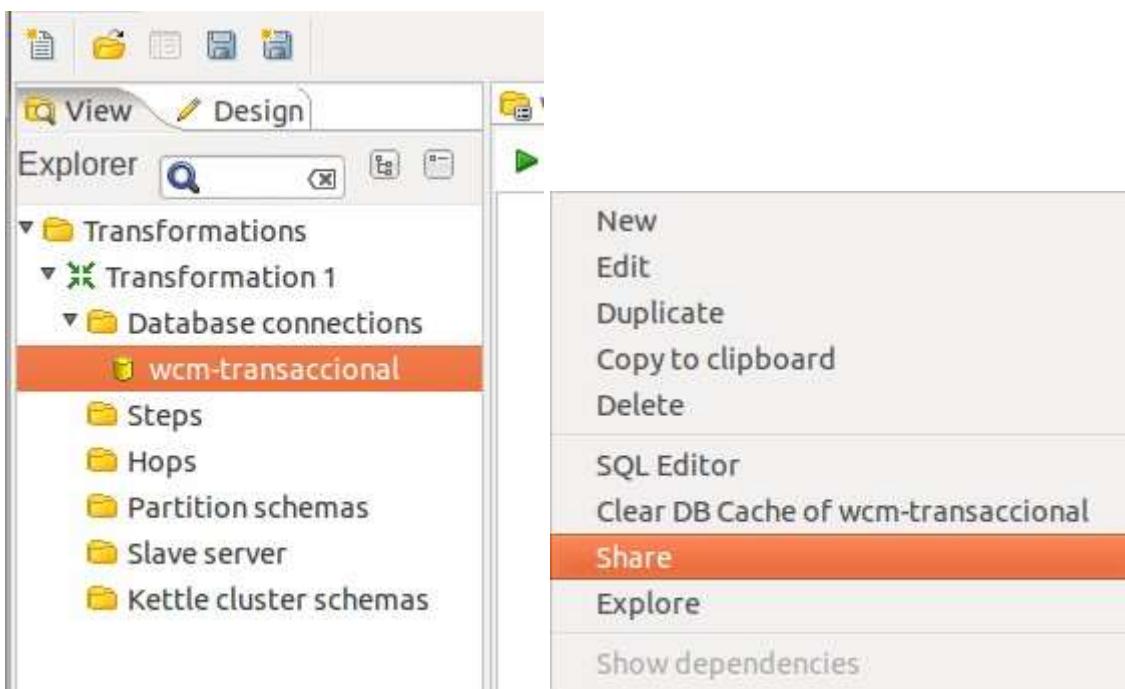
Con lo cual se desplegará una ventana para configurar la conexión a la BD, en este caso pondremos un nombre a la conexión, y nos conectaremos al motor PostgreSQL.



Al finalizar la captura debemos probar la conexión dando clic en el botón Test.



Una vez que se ha establecido la conexión, es importante compartirla (share) para que podamos utilizarla en todas las transformaciones y jobs que lo requieran.



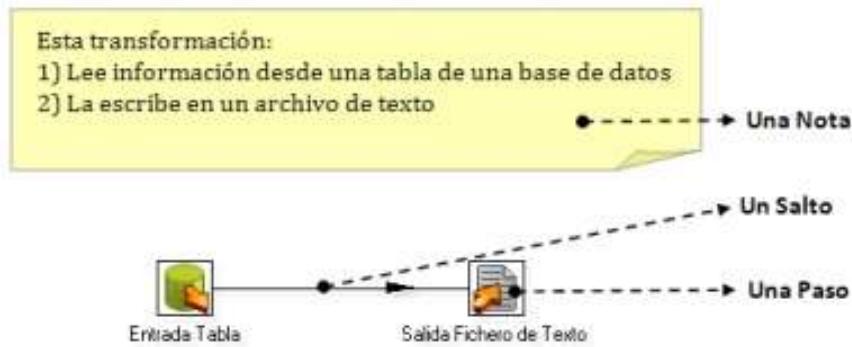
Transformaciones

La **transformación** es el elemento básico del diseño de un proceso ETL, está compuesta de **pasos** (steps) que están conectados entre sí a través de los **saltos** (hops) → . Una **transfomación no es un programa o un ejecutable, es un conjunto de metadatos en XML** que le indican al motor de PDI las acciones a realizar. Los steps están agrupados por categorías, y cada uno de ellos está diseñado para cumplir una función determinada. Cada paso tiene su ventana de configuración donde se indican los elementos a tratar y su comportamiento. Los steps de una transformación se procesan de manera simultanea y asíncrona.

Definiciones de Transformación

La siguiente tabla contiene una lista de definiciones de Transformación:

Transformación	Descripción
Valor	Valores que forman parte de una fila y que pueden contener cualquier tipo de datos: cadenas, números de punto flotante, números grandes de precisión ilimitada, enteros, fechas o valores booleanos.
Fila	Una fila consiste de 0 o más valores procesados mediante una sola entrada.
Flujo de Entrada	Conjunto de filas que ingresan a un paso.
Salto	Representación gráfica de uno o más flujos de datos entre 2 pasos; un salto siempre representa el flujo de salida de un paso y el flujo de entrada de otro (la cantidad de flujos es igual a las copias del paso destino, una o más).
Nota	Texto descriptivo que se puede agregar a la Transformación.



Hops

Los **hops** o saltos son el elemento a través del cual fluye la información entre los diferentes pasos. Hay un código de colores para identificar las propiedades de los hops:

Color de hop	Significado
Verde	Filas distribuidas: Si múltiples <u>hops</u> dejan un paso, las columnas de datos se distribuyen en todos los pasos objetivo.
Rojo	Copia filas: Si múltiples <u>hops</u> dejan un paso, todas las columnas de datos serán copiadas a todos los pasos objetivos.
Amarillo	Provee información para el paso, en columnas distribuidas.
Magenta	Provee información para el paso, en columnas copiadas.
Gris	El <u>hop</u> está deshabilitado.
Negro	El <u>hop</u> tiene un paso objetivo con nombre.
Azul	Candidato a <u>hop</u> dando clic al <u>mouse</u> y arrastrando.
Naranja (--)	El <u>hop</u> no se usa nunca porque no hay datos que lleguen ahí.
Rojo (--)	El <u>hop</u> se usa para llevar columnas que causaron errores en los pasos origen.

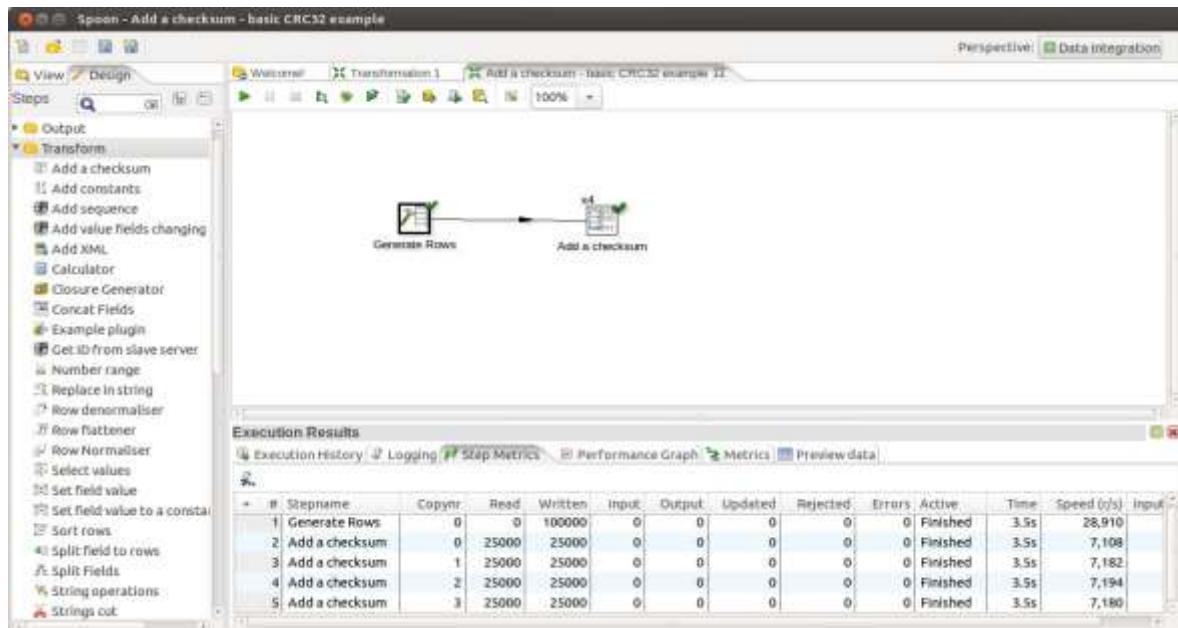
Ejercicios de transformaciones

Seleccionamos del menú la opción de crear una nueva transformación.



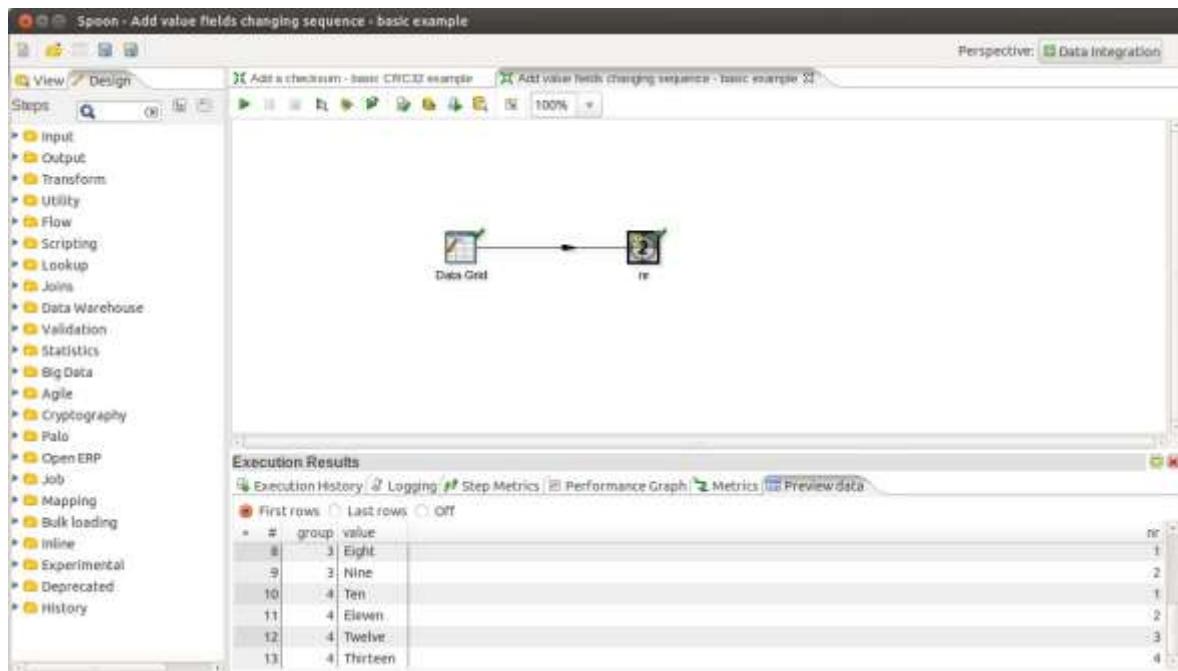
Analizaremos los ejemplos que se encuentran en la carpeta: file → open Url →
file:///C:/Users/Sara/Documents/Cursos/Curso BI/software curso BI/data-integration/samples/transformations/

Transformación: Add a checksum (Add a checksum - Basic CRC32 example.ktr)

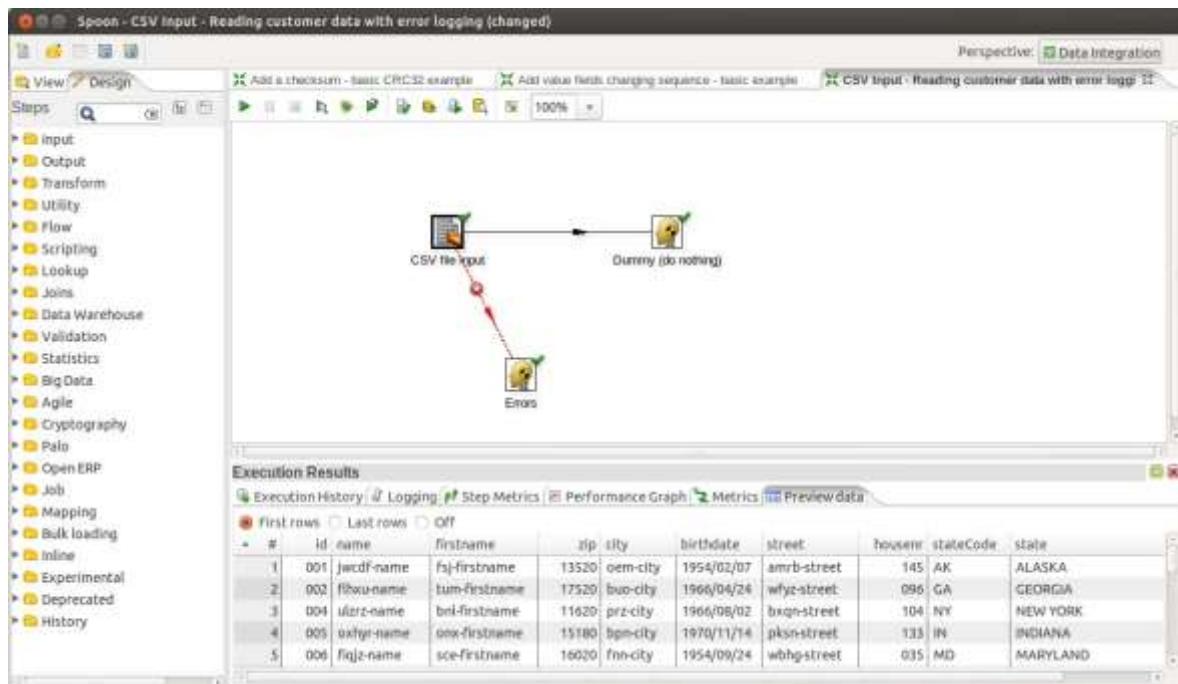


#	Stepname	Copysr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (ops)	Input
1	Generate Rows	0	0	1000000	0	0	0	0	0	Finished	3.5s	28,910	
2	Add a checksum	0	25000	25000	0	0	0	0	0	Finished	3.5s	7,108	
3	Add a checksum	1	25000	25000	0	0	0	0	0	Finished	3.5s	7,182	
4	Add a checksum	2	25000	25000	0	0	0	0	0	Finished	3.5s	7,194	
5	Add a checksum	3	25000	25000	0	0	0	0	0	Finished	3.5s	7,180	

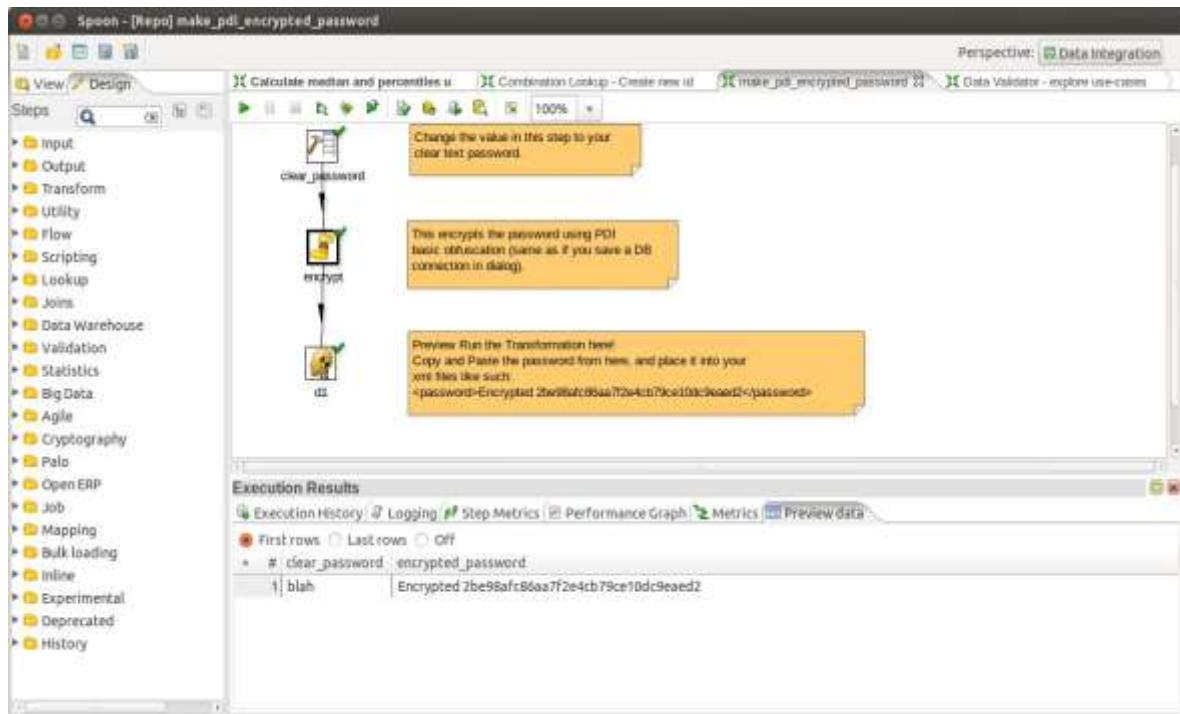
Transformación: Add value changing sequence (Add value fields changing sequence –basic example.ktr)



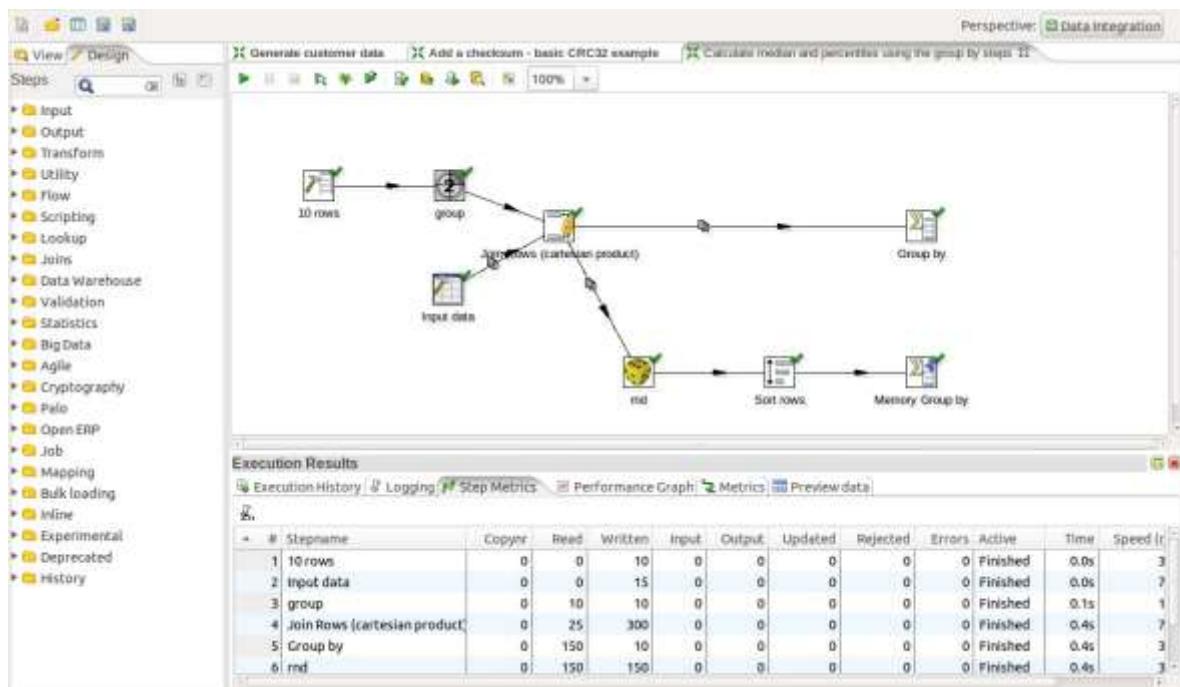
Transformación: Leer CSV con errores (CSV Input – Reading customer data with error logging.ktr)



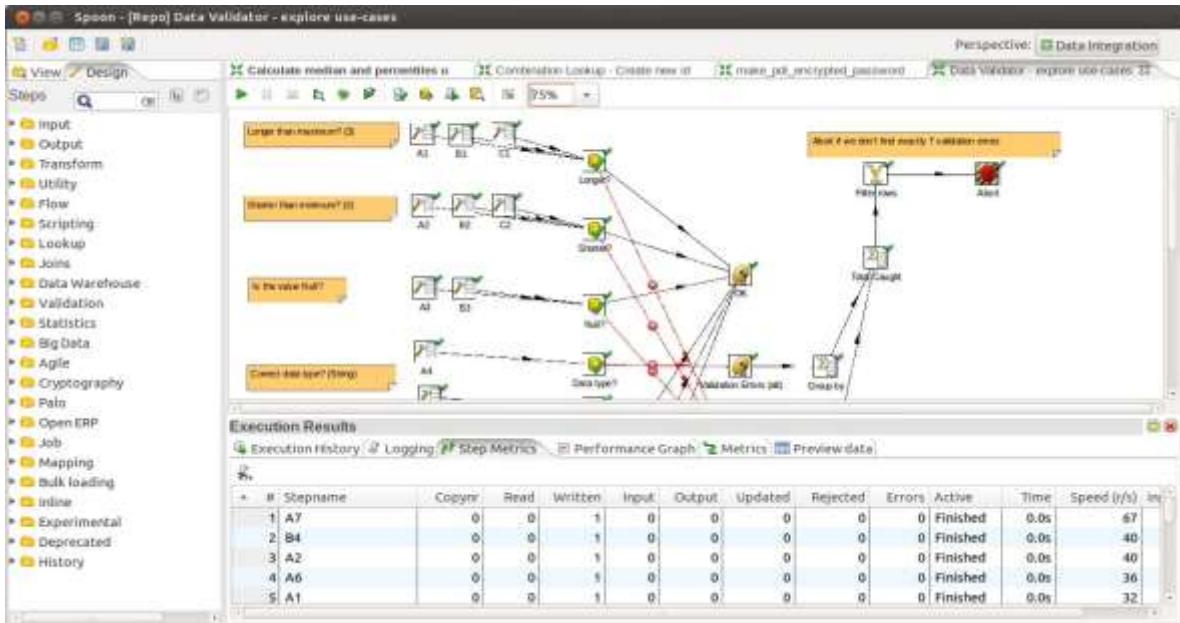
Transformación: Password encryptado (Encrypt Password.ktr)



Transformación: Calcular mediana y cuartiles (Calculate median and percentiles using the group by steps.ktr)



Transformación: Validación de datos (Data Validator – all usecases with error hanling.ktr)



Trabajos (Jobs)

Un trabajo (job) consiste en un conjunto sencillo o complejo de tareas con el objetivo de realizar una acción determinada. En los jobs se utilizan pasos específicos (distintos de los disponibles en las transformaciones), inclusive se pueden ejecutar una o varias transformaciones que hayamos diseñado. **Los jobs se ejecutan en la secuencia que le indiquemos** mediante hops; no se va a iniciar un paso hasta que no haya terminado el inmediato anterior, además es posible condicionar la ejecución de un paso al resultado de algún paso anterior debido a que la salida de cada job da un estado éxito/fracaso.

Los jobs están en un nivel superior que las transformaciones, pero al igual que éstas son metadatos en XML que le describen al motor del PDI la forma de realizar las acciones.

Entrada de job: es una parte del job que ejecuta una tarea específica. También se manejan los hops y las notas.

Para agregar un paso entre dos pasos unidos por un salto, se puede colocar el nuevo paso sobre el salto, y automáticamente la aplicación preguntará si se desea "**partir**" el salto.

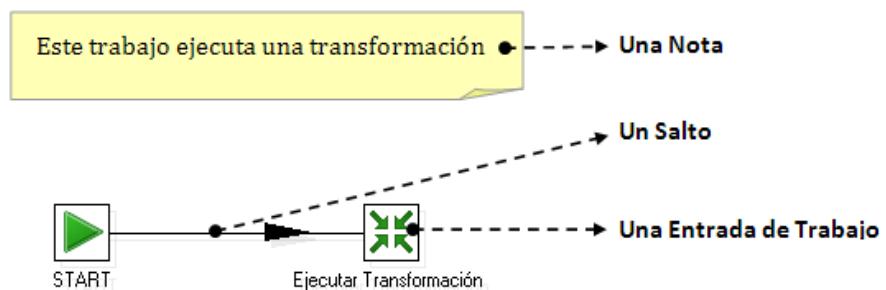
Una diferencia adicional entre las transformaciones y los trabajos es que en los primeros no se permiten bucles y en los trabajos sí.

También hay un trap detector de filas mezcladas, es decir, no se permite mezclar filas de orígenes distintos si éstas tienen estructuras diferentes.

Definiciones de Trabajo

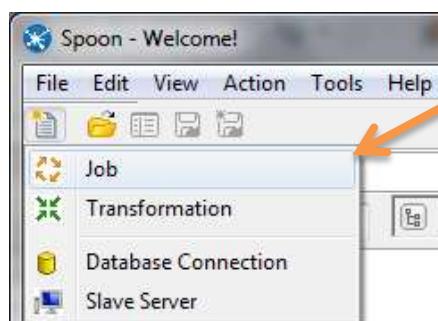
La siguiente tabla contiene una lista de definiciones de Trabajo:

Trabajo	Descripción
Entrada de Trabajo	Representa una parte de un trabajo que realiza una tarea específica.
Salto	Representación gráfica de uno o más flujos de datos entre 2 pasos; un salto siempre representa el flujo de salida de un paso y el flujo de entrada de otro (la cantidad de flujos es igual a las copias del paso destino, uno o más)
Nota	Texto descriptivo que se puede agregar a un Trabajo



Ejercicios de Jobs

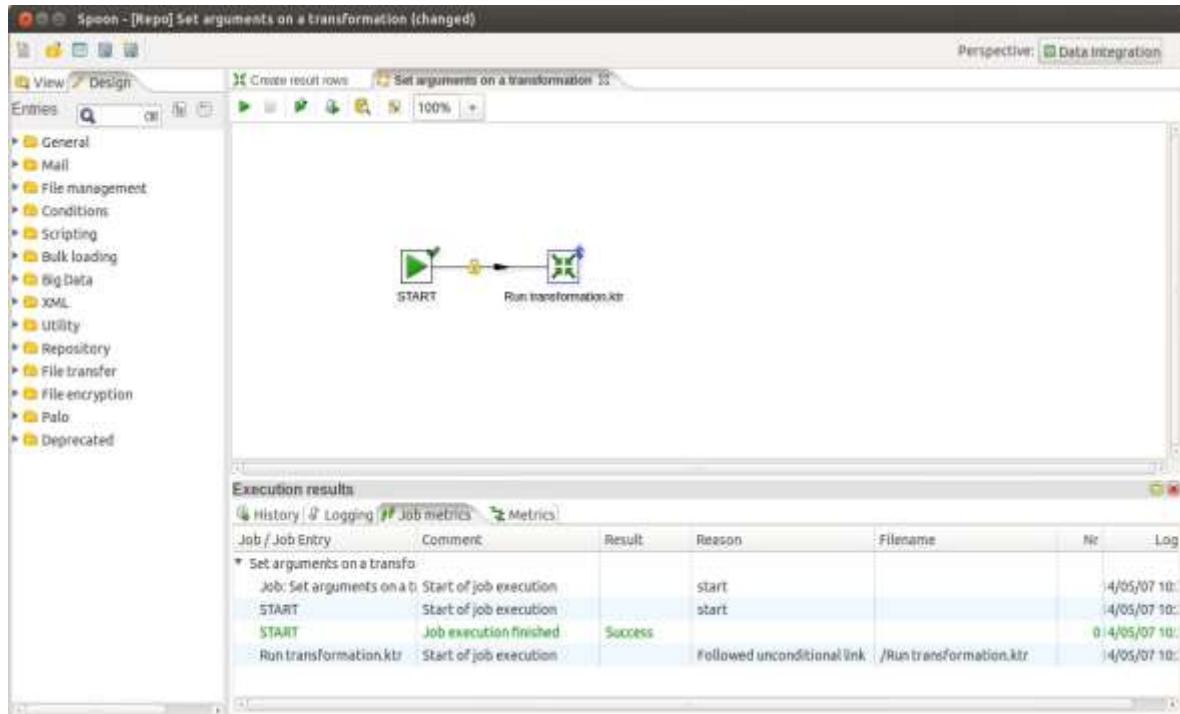
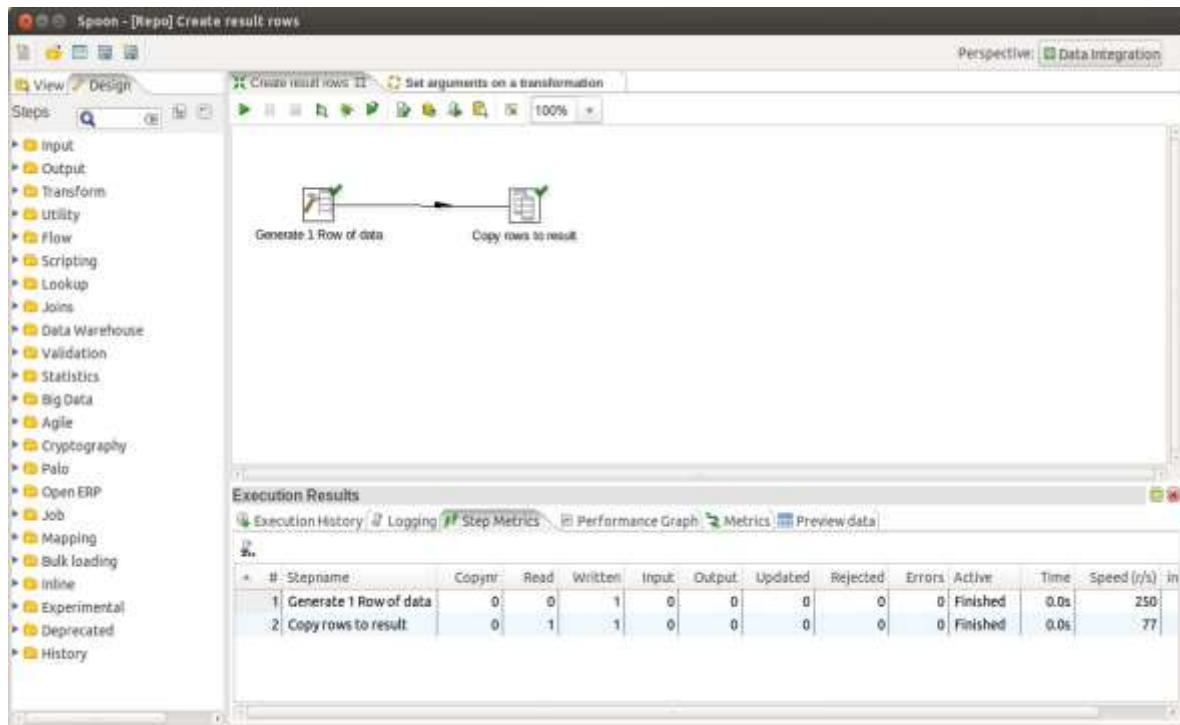
Elegir la opción de nuevo Job en el menú.



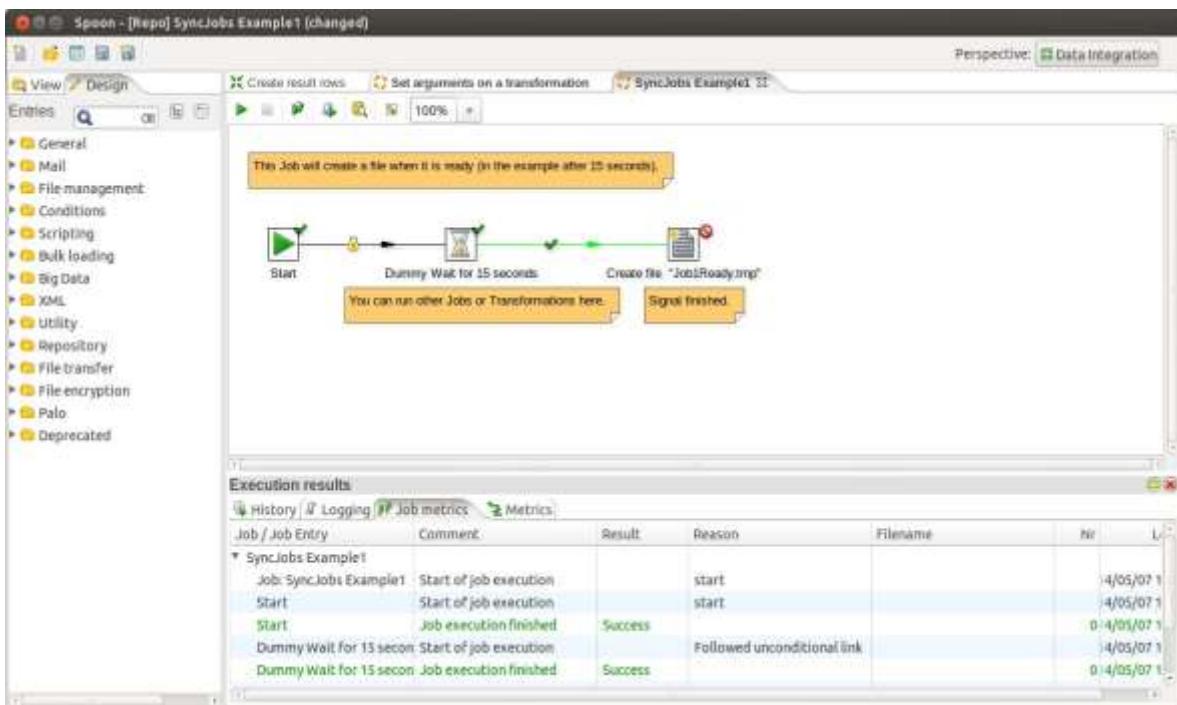
Analizaremos los ejemplos que se encuentran en la carpeta: file → open Url →

file:///C:/Users/Sara/Documents/Cursos/Curso BI/software curso BI/data-integration/samples/jobs/

Job: Ejecución de una transformación (Set arguments on a transformation en arguments/)



Job: Job con un retardo sincronizado (sync_jobs en la carpeta parallel synchronized Jobs/SyncJobs Example1.kjb)



Mapeos

Un mapeo funciona como una transformación regular, sólo que introducimos un paso previo para establecer un mapeo entre el paso de origen y el paso de destino de los datos, lo cual permite reutilizar esta misma transformación para llenar otros destinos o bien guardar cálculos complejos y asignarles un campo de destino.

El siguiente es un ejemplo de una Transformación simple en la que se busca generar el mapeo de los correspondientes flujos de datos a una tabla de salida:

*1) Hacer clic derecho sobre el paso Salida Tabla y seleccionar "Generar mapa contra este paso destino":



*2) Agregar todos los mapeos necesarios usando las herramientas de "Editar Mapeo" mostradas anteriormente y hacer clic en OK. Ahora verá que se agrega automáticamente al lienzo un nuevo paso "Selecciona/Renombra Valores" con el nombre "Mapeo Salida Tabla":

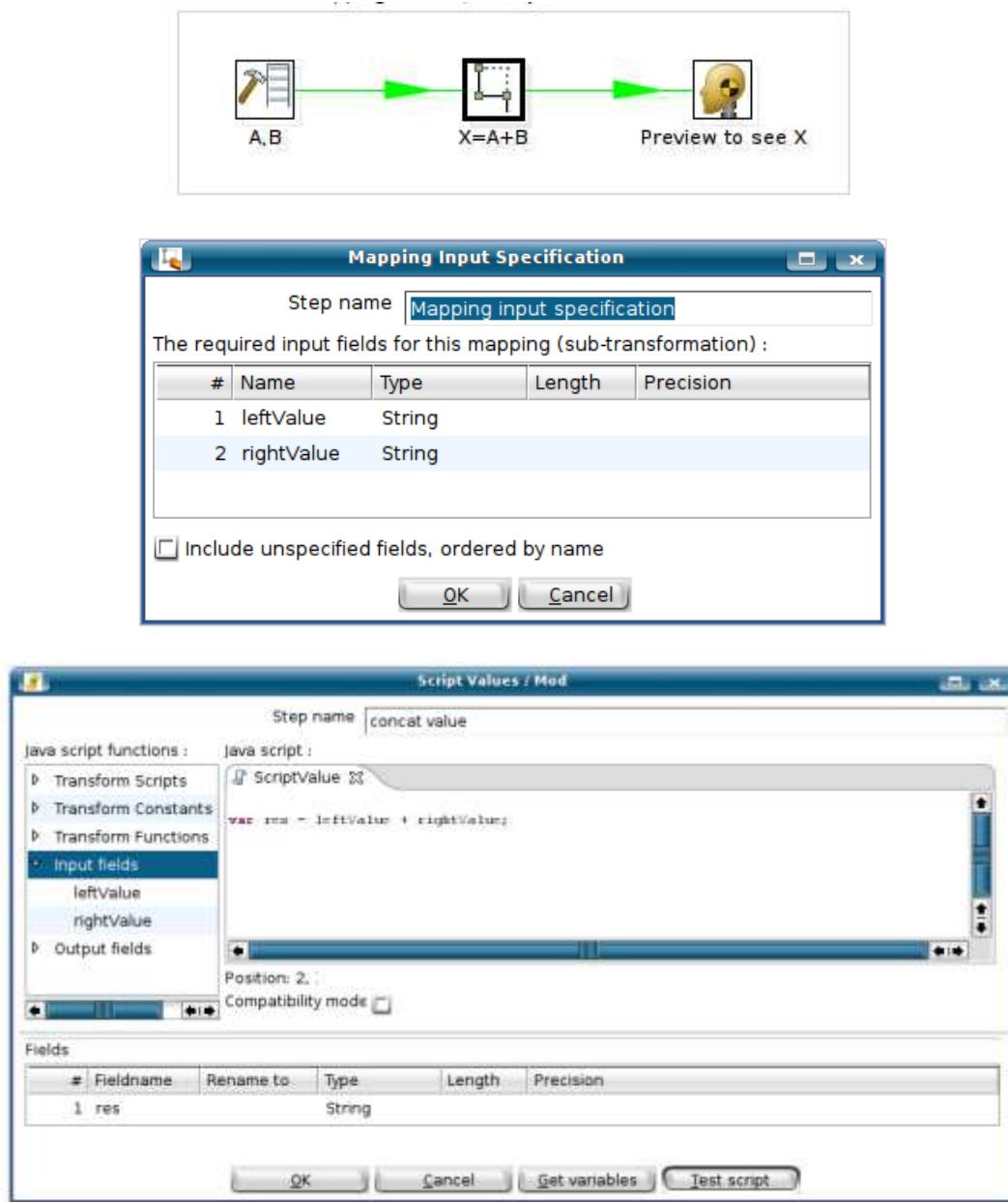


*3) Colocar el nuevo paso entre los dos pasos iniciales y configurar los saltos de la siguiente manera:



Como se puede observar, esta opción permite realizar de forma sencilla y gráficamente las correspondencias entre el flujo de datos de entrada y salida.

Entre el paso de entrada de datos y el de salida se ha introducido un paso intermedio de mapeo en el cual podemos definir la correspondencia entre los campos y cálculos.

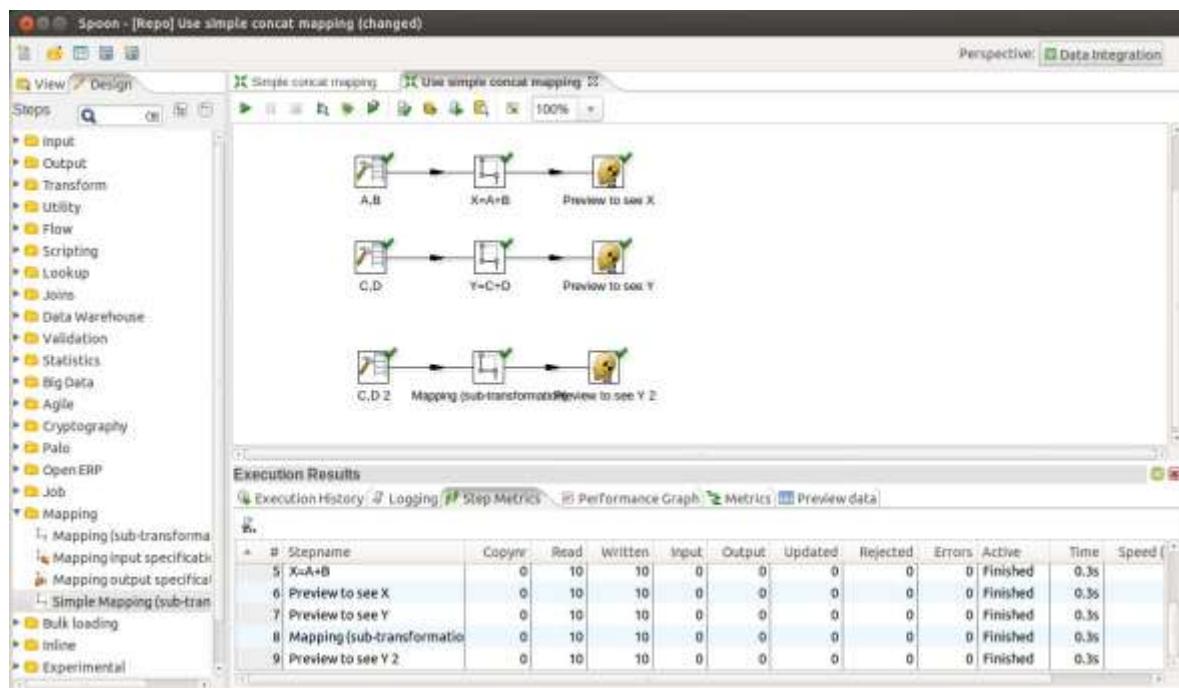
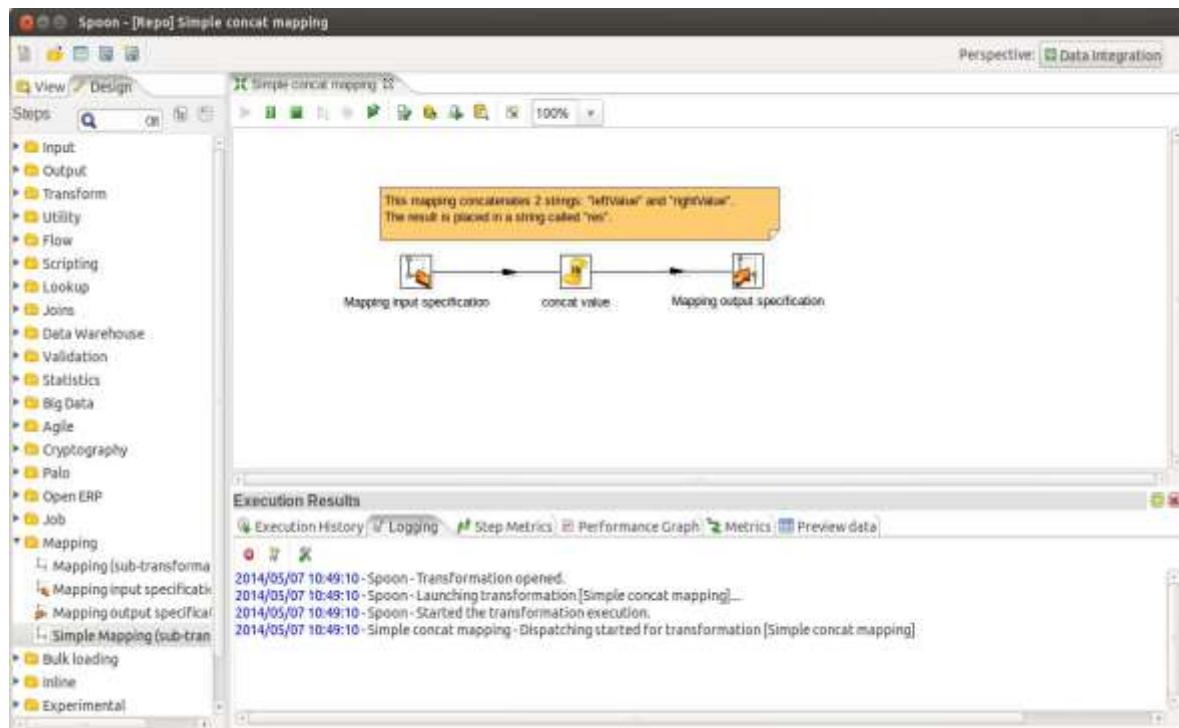


Ejemplos de mapeo

Recordemos que un mapeo es una transformación.

Analizaremos los ejemplos que se encuentran en la carpeta: file → open Url →

file:///C:/Users/Sara/Documents/Cursos/Curso BI/software curso BI/data-integration/sample/transformation/mapping/



Pan: Ejecución externa de transformaciones

Pan forma parte del PDI, es una herramienta que permite ejecutar los archivos .ktr (es decir, las transformaciones que creamos con Spoon) desde la ventana de comandos (cmd).

Desde una ventana de comandos se puede ejecutar una transformación ya sea desde un archivo XML (.ktr) o bien desde el repositorio del PDI, utilizando el programa pan.sh en linux o pan.bat en Windows.

Las instrucciones son:

Para ejecutar una transformación desde el archivo XML (ktr) hay que escribir la siguiente linea:

Windows

```
pan.bat /file:"D:\Transformaciones\Dimension Clientes.ktr" /level:Basic > C:\LOG\trans.log
```

Linux

```
./pan.sh -file="/transformaciones/Dimension Clientes.ktr" -level=Minimal >> /LOG/trans.log
```

La salida se redirige a un archivo .log

Para ejecutar una transformación desde el repositorio hay que indicar la localización de un repositorio y aportar usuario con clave:

Windows:

```
pan.bat /rep:"DWDEV" /trans:"trsfcountry" /dir:/ /user:admin /pass:admin /level:Minimal  
(ejecutable "pan.bat"+ ruta)
```

Linux:

```
./pan.sh -rep:"test" -job:"SMTPtest" -dir:"/" -user:"admin" -pass:"admin" -level:"Minimal"
```

Kitchen: Ejecución externa de trabajos

Kitchen forma parte del PDI, es una herramienta que permite ejecutar los archivos .kjb (es decir, los jobs que creamos con Spoon) desde la ventana de comandos (cmd).

Desde una ventana de comandos se puede ejecutar una trabajo ya sea desde un archivo XML (.kjb) o bien desde el repositorio del PDI, utilizando el programa kitchen.sh en linux o kitchen.bat en Windows.

Las instrucciones son:

Para ejecutar un trabajo desde el archivo XML (ktr) hay que escribir la siguiente linea:

Windows

```
kitchen.bat /file:C:\trabajos\jpb.kjb /level:Basic > C:\LOG\trans.log
```

Linux

```
./kitchen.sh -file="/trabajos/job.kjb" -level=Minimal >> /LOG/trans.log
```

La salida se redirige a un archivo .log

Para ejecutar un trabajo desde el repositorio hay que indicar la localización de un repositorio y aportar usuario con clave:

Windows:

```
kitchen.bat /rep:"DWDEV" /job:"trsfcountry" /dir:/ /user:admin /pass:admin /level:Minimal  
(ejecutable "kitchen.bat"+ ruta)
```

Linux:

```
./kitchen.sh -rep:"test" -job:"SMTPtest" -dir:"/" -user:"admin" -pass:"admin" -level:"Rowlevel"
```

Objetivos del Datamart a diseñar:

Usaremos como ejemplo la base de datos World Class Movies (WCM), la cual usa dos bases de datos en su negocio, uno para la operación y otro para sus tiendas en la web. El manejo de productos está atado a los dos procesos.

WCM usa las tablas de las normas ISO 639 y 3166 para codificar y nombrar lenguajes, países y estados (regiones).

Características de la base de datos fuente:

- El núcleo son los clientes, productos y órdenes.
- Dos tipos de órdenes: de compra y de clientes.
- Los productos de compran a distribuidores y son recibidos en almacenes en donde son revisados por empleados.
- Cada orden de compra se hace por un empleado a cierto distribuidor, con varias partidas.
- Una partida de una orden de compra consiste de un ítem, una cantidad y el costo del producto.
- En la orden de compra se define el almacén destino.
- Los empleados cuentan con una descripción de su trabajo y trabajan en un almacén específico.
- Los almacenes, empleados, clientes y distribuidores tienen una dirección.
- Cada dirección está localizada en una cierta región, y cada región es parte de un país.
- Las órdenes de los clientes se hacen en un website y pueden contener una promoción.
- Las promociones son ciertos productos (versiones en DVD) y pueden tener un precio de venta rebajado, un precio de renta rebajado, un periodo extendido de renta o una combinación de ellos.
- Una orden de un cliente consiste de una o más partidas, con un producto diferente por partida.
- El negocio se basa en la renta y venta de películas
- El cliente las pide por medio de un catálogo en la web
- Se le cobra como venta
- Si después de verla decide sólo rentarla, la regresa y se abona a su cuenta la diferencia entre renta y venta.
- El almacén debe hacer un pedido nuevo por cada película vendida.

Debemos cargar la base de datos fuente en nuestro motor Postgresql.

1. Crear la base de datos de pruebas wcm con el psql como usuario postgres

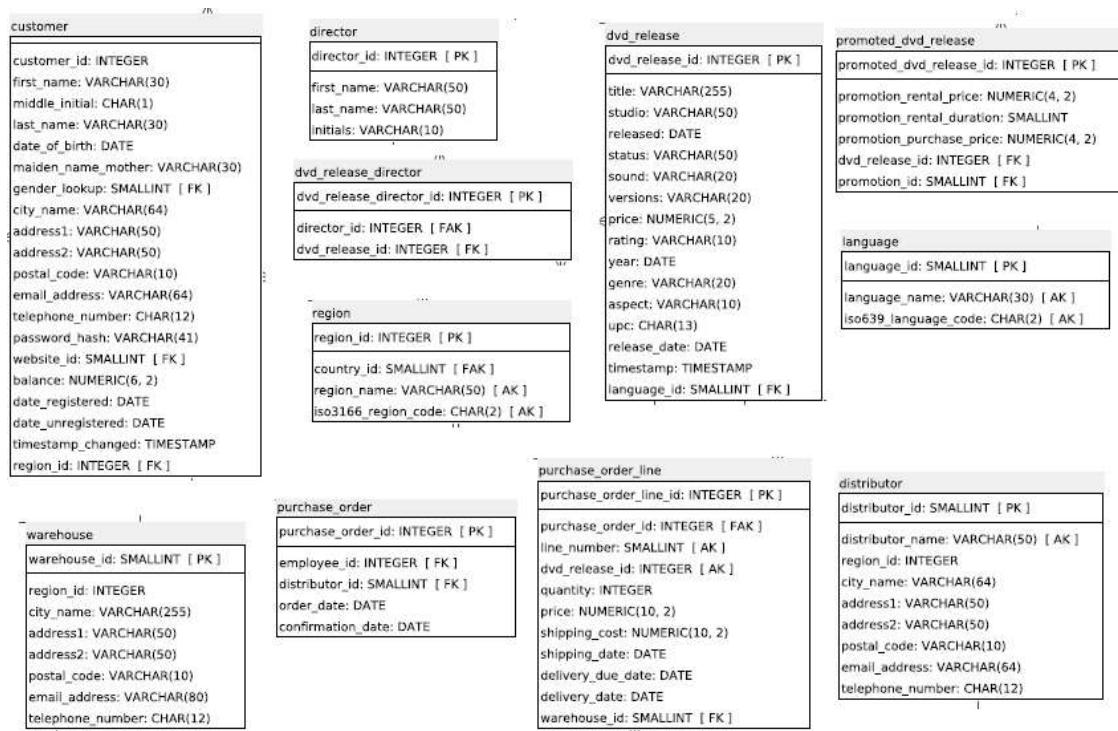
2. sudo -su postgres (Linux) ó abrir una sesión de pgAdmin
3. abrir la interfaz psql
4. CREATE DATABASE wcm WITH OWNER = postgres;
5. \c wcm
6. \i ../"ruta_archivo"/wcm_dump.sql

Ahora, nosotros trabajamos en el departamento de marketing, y nos interesa crear un datamart exclusivo sobre las ventas, clientes y distribuidores para encontrar nuestras “ventanas de oportunidad”.

Podemos plantearnos objetivos tales como los siguientes:

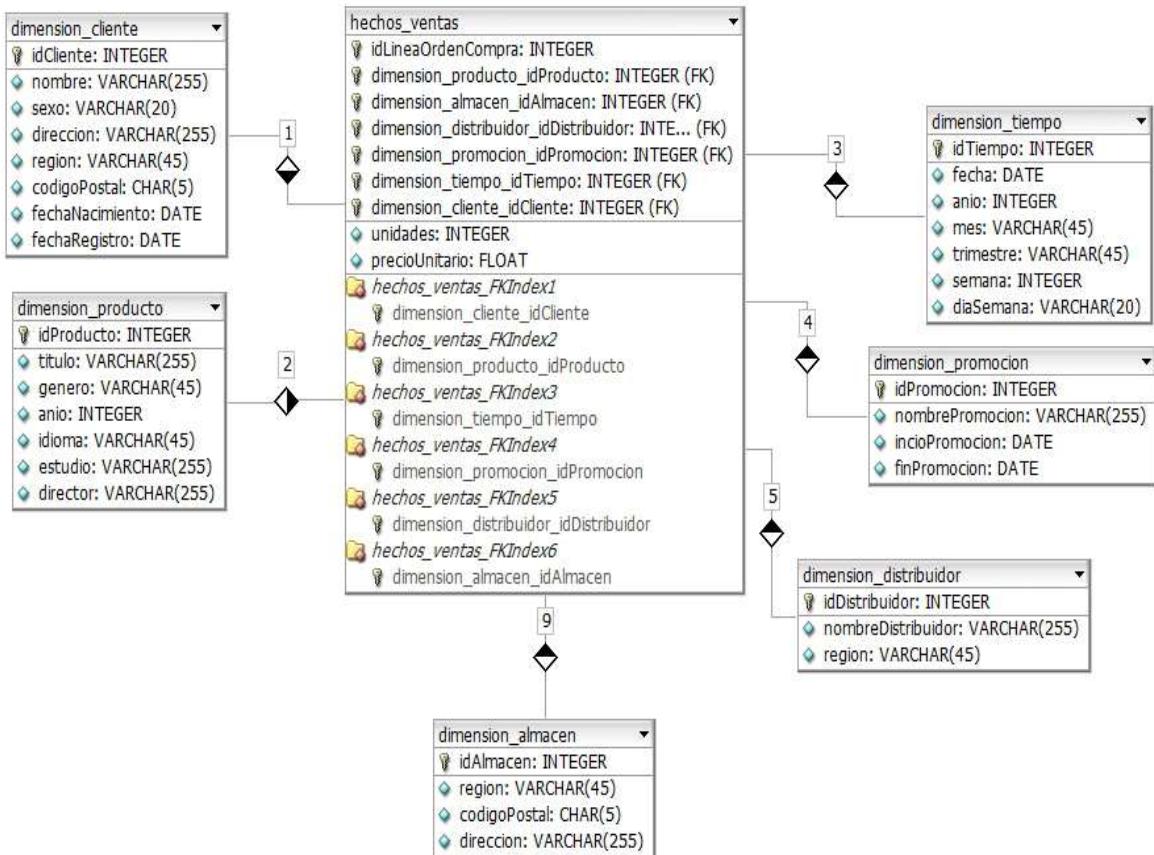
- Encontrar la relación entre las regiones donde se encuentran nuestros clientes y nuestros distribuidores.
- Encontrar la relación entre las regiones donde se encuentran nuestros distribuidores y nuestros almacenes.
- Conocer cuánto venden nuestros distribuidores por región por trimestre.
- Conocer cuál es la demanda de nuestros productos por género en cada región para saber cómo distribuir nuestra mercancía en los almacenes, en función del tiempo.
- Conocer el impacto de las promociones en la venta de nuestros productos por región.

Tablas de la base de producción WCM que contienen la información que nos interesa conocer para nuestros análisis:



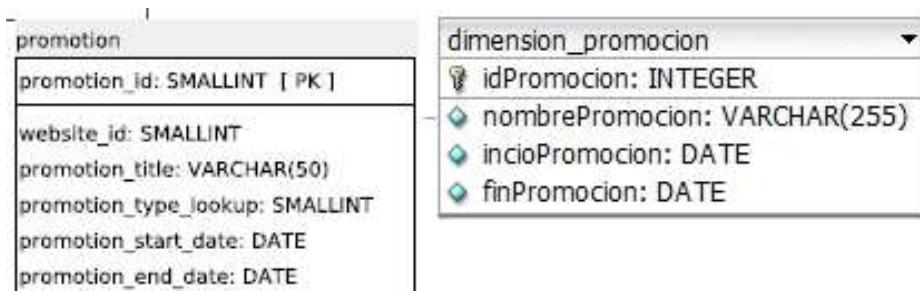
Utilizando el modelo dimensional hemos diseñado una base de datos en forma de estrella donde la tabla de hechos son las ventas y el resto de los atributos los incluimos en las tablas de dimensión.

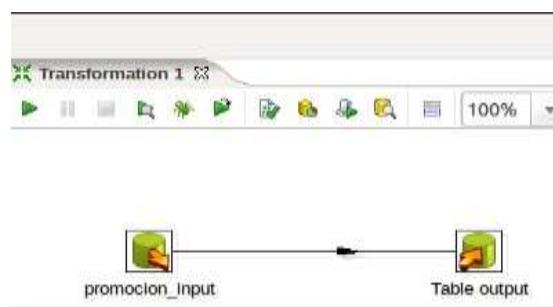
DataMart Ventas



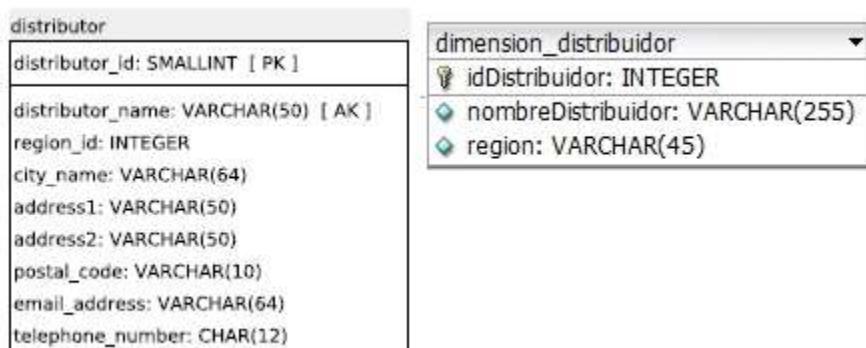
Para cargar cada una de las tablas deberemos crear las transformaciones correspondientes.

Transformación: dimension_promocion

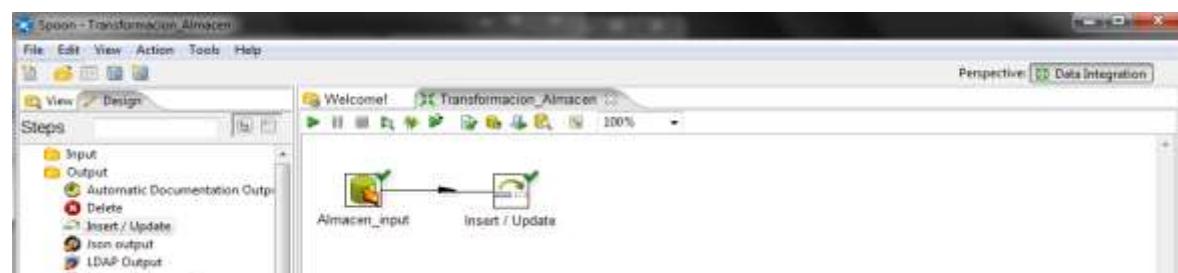
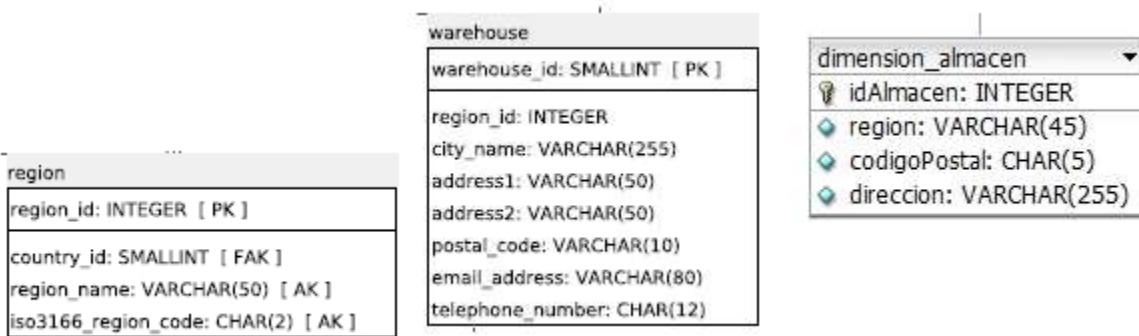




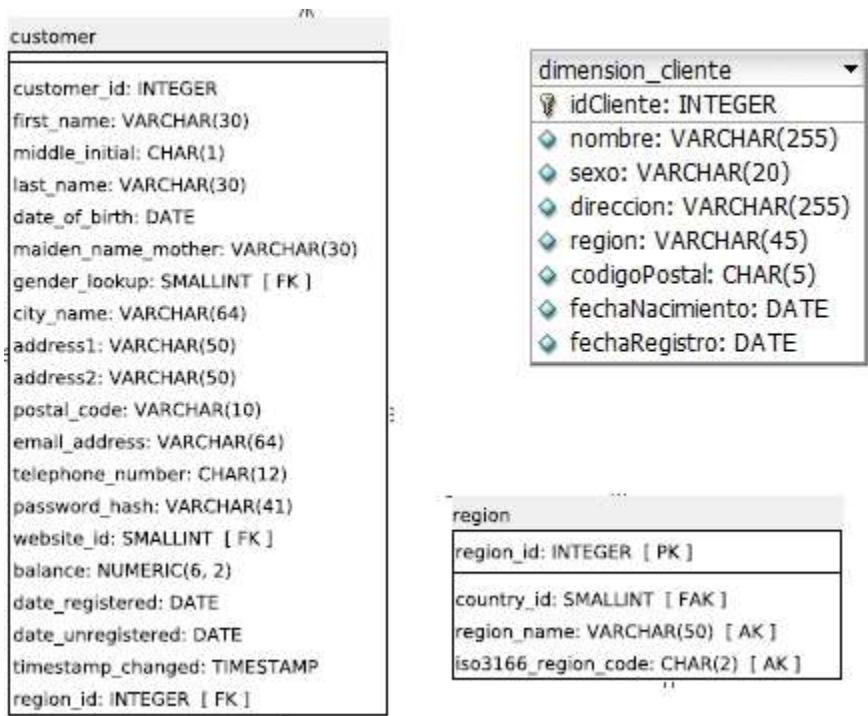
Transformación: dimension_distribuidor



Transformación: dimension_almacen



Transformación: dimension_cliente



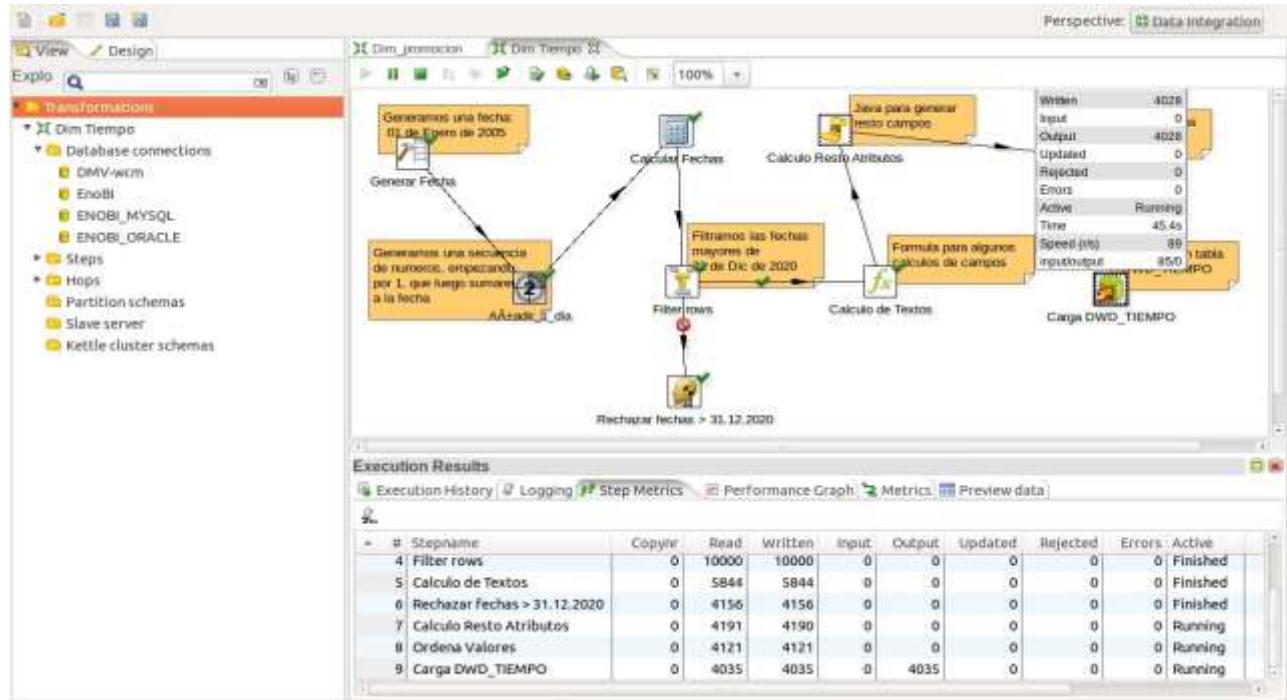
region
region_id: INTEGER [PK]
country_id: SMALLINT [FAK]
region_name: VARCHAR(50) [AK]
iso3166_region_code: CHAR(2) [AK]

Transformación: dimension_producto



language
language_id: SMALLINT [PK]
language_name: VARCHAR(30) [AK]
iso639_language_code: CHAR(2) [AK]

Transformación: dimension_tiempo

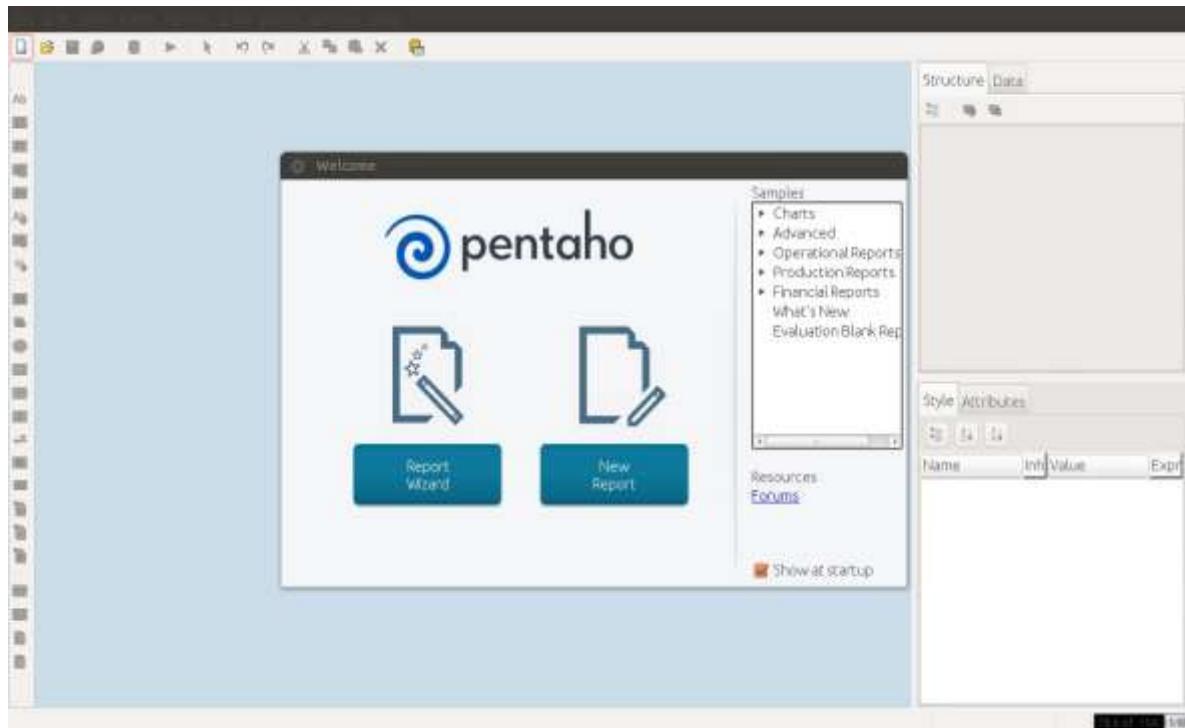


Transformación: hechos_ventas

Introducción al Pentaho Report Designer

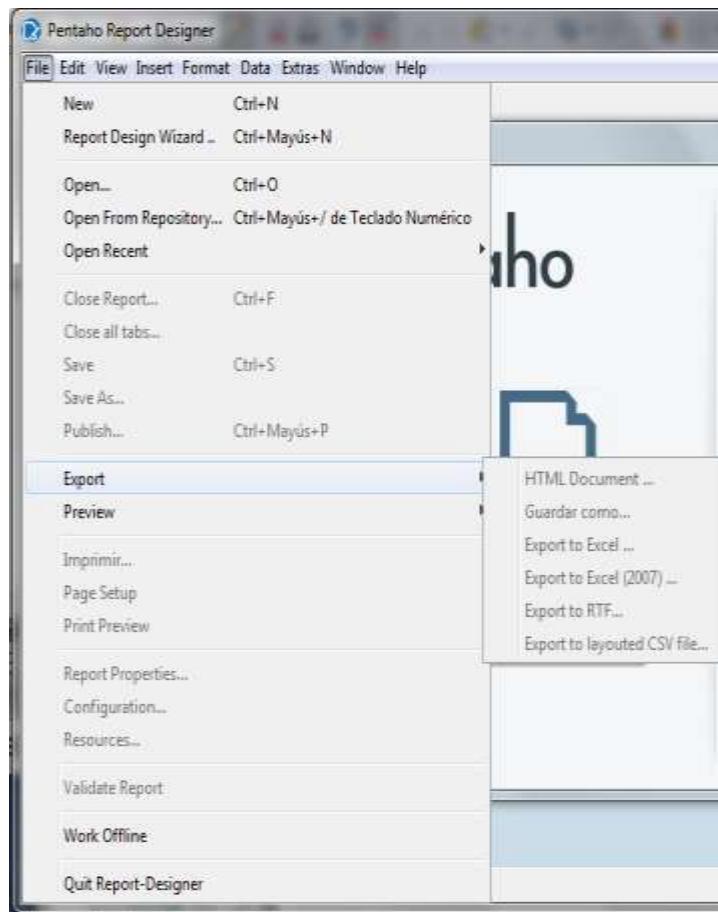
Pentaho Report Designer (PRD) es la herramienta para la elaboración de reportes y puede generarlos en los formatos PDF, Excel, HTML, Texto, XML y CSV. Este editor permite integrar gráficas y tablas de consultas SQL sobre una BD para la presentación de nuestros reportes, así como imágenes (logos) y texto para dar una presentación profesional a los reportes para su publicación.

El PRD brinda la opción de generar un reporte desde ceros, pero también ofrece un wizard que apoya al usuario en cada paso.

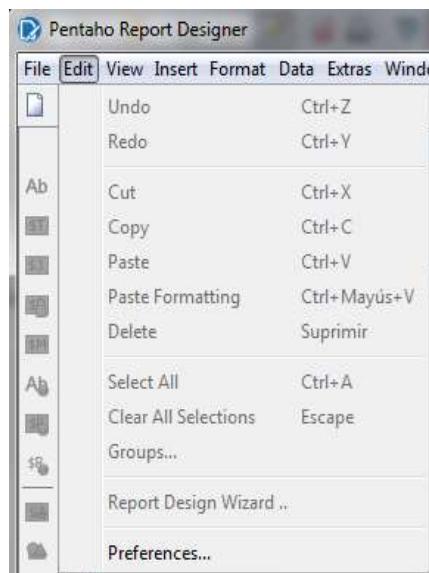


En el menú principal se ofrecen los menús: File, Edit, View, Insert, Format, Data, Extras, Window y Help.

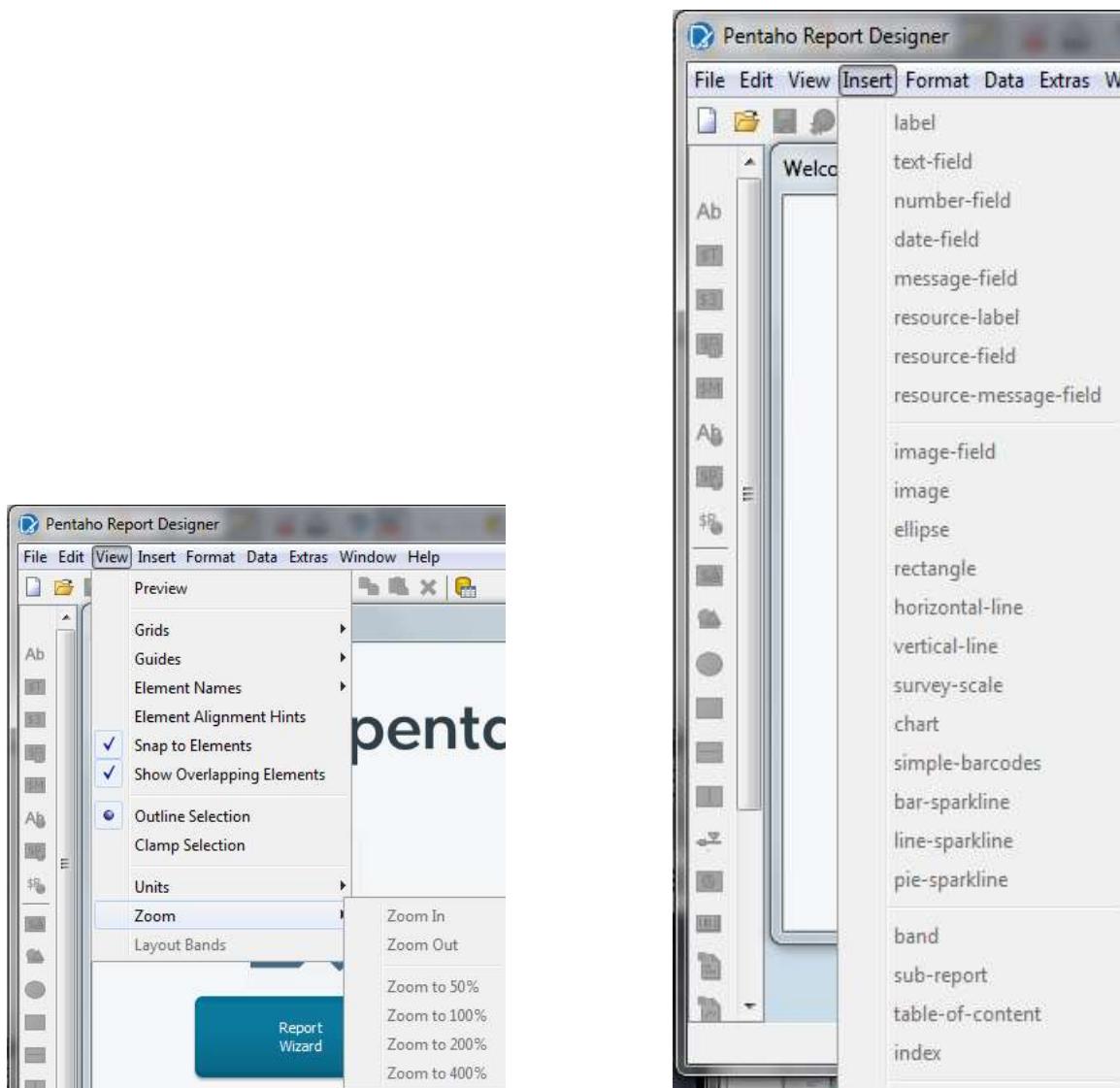
Menú File donde se puede crear un nuevo reporte o abrir uno ya existente, además permite guardar y exportar el reporte.



En el menú Edit se puede copiar, cortar y pegar objetos del reporte, además de invocar el wizard.

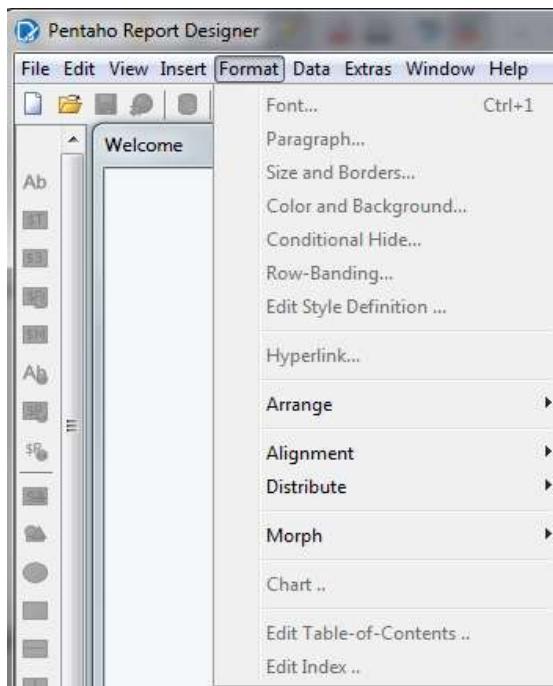


En el menú View se puede visualizar la cuadrícula y las guías para apoyar durante el diseño, acercar o alejar la imagen, alinear los objetos, etc.

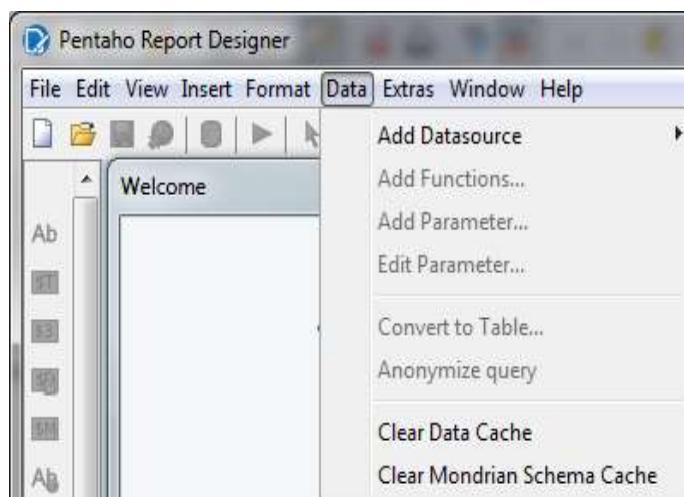


En el menú Insert se pueden agregar a la plantilla los diferentes objetos del diseño de reportes.

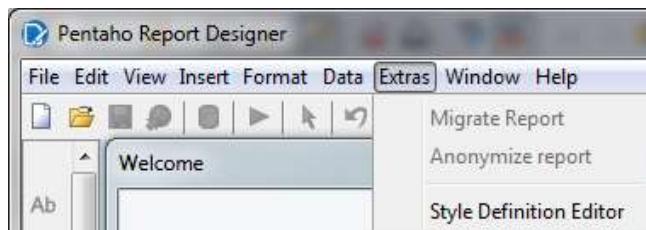
En el menú Format se puede definir el tamaño, tipo, color y estilo de letra, agregar características a las gráficas, insertar índices y tablas de contenido e imágenes.



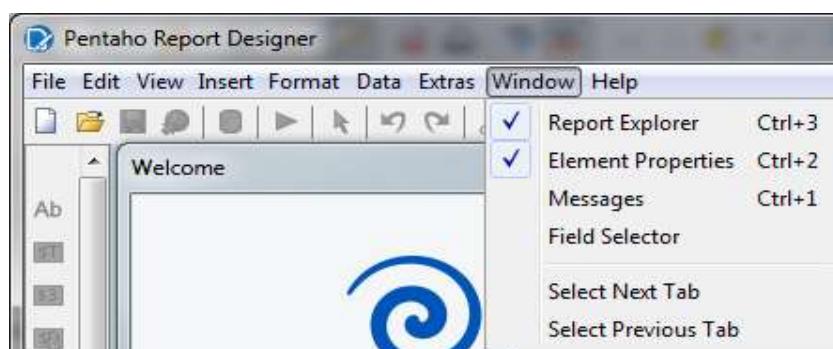
En el menú Data es donde se configuran los orígenes de datos para los reportes, ya sean de bases de datos, archivos planos, cubos, etc.



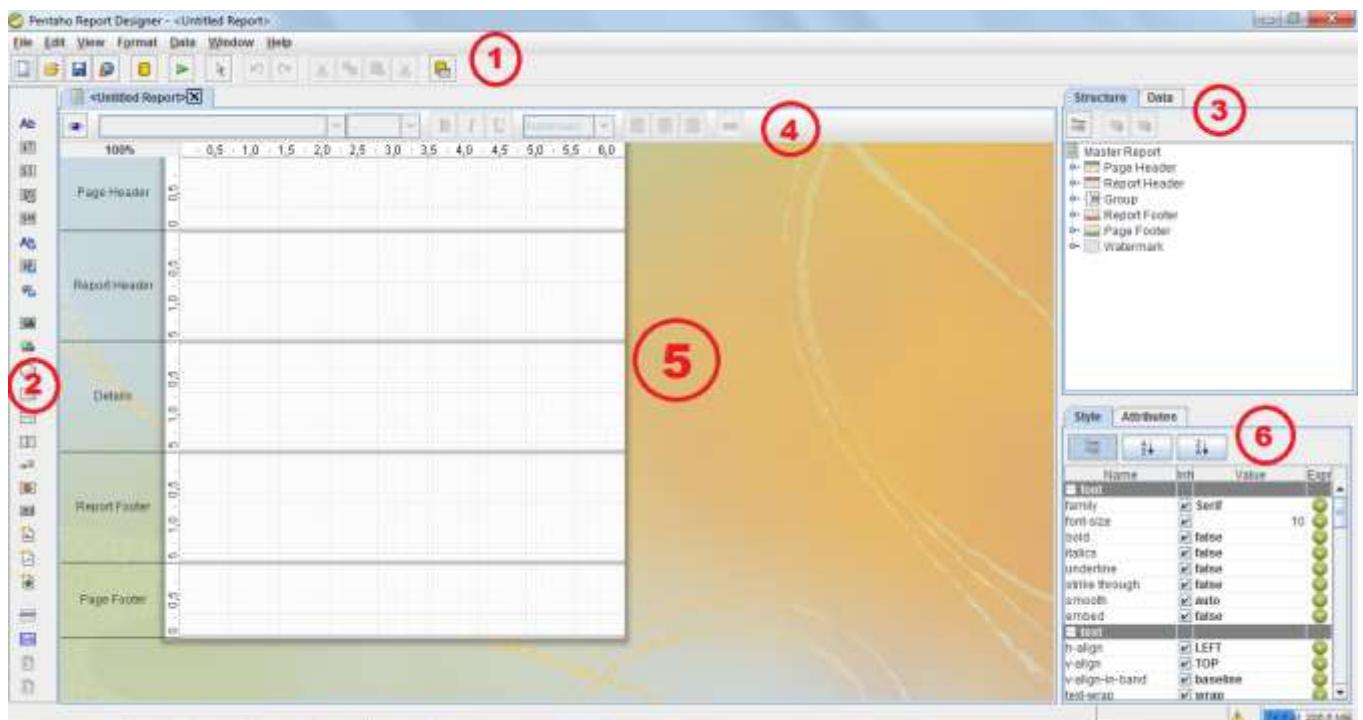
En el menú de Extras se puede definir un estilo definido para un reporte en el caso de que hayamos guardado alguna plantilla predeterminada que queramos usar.



En el menú Window podemos elegir las ventanas que deseamos ver en el editor PRD: el explorador de objetos, la ventana de propiedades de los objetos, etc.



El Pentaho Report Designer es un ambiente de trabajo muy fácil de comprender y utilizar para el diseñador, está formado por las siguientes secciones:

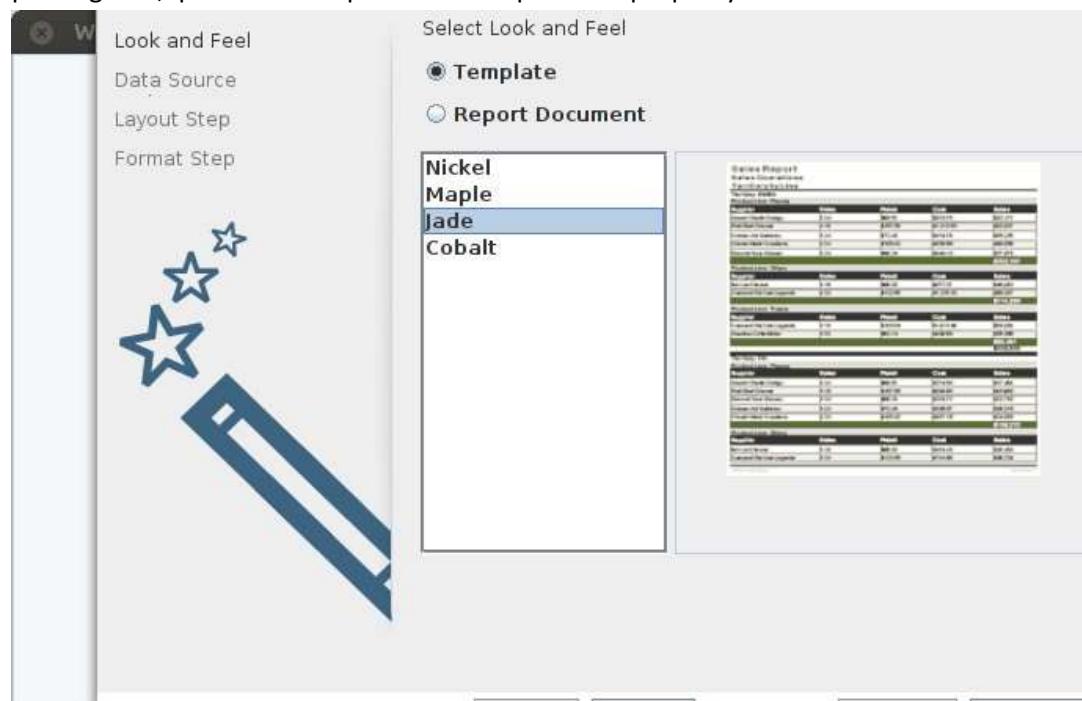


- 1) Barra de menús: Ofrece las opciones de abrir un nuevo reporte, cargar uno ya existente, guardar el actual. Además de la publicación de nuestro reporte en el BIServer o su exportación a formatos como el PDF y Excel, entre otros.
- 2) Barra de herramientas: En esta barra tenemos los elementos que podemos agregar a nuestro reporte según sea necesario: etiquetas de texto, tablas, imágenes, gráficas, entre otros.
- 3) Estructura y Datos: En esta ventana vamos a administrar las conexiones, variables, funciones y orígenes de datos que van a estar en nuestro reporte.
- 4) Menú de Formato: Es un atajo para darle formato (tipo de letra, tamaño, negritas, cursivas, alineaciones) a los elementos de nuestro reporte.
- 5) Área de trabajo: Es donde se edita propiamente el reporte, está dividido (al principio) en 5 secciones (Page Header [Encabezado de la página], Report Header [Encabezado del reporte], Details [Cuerpo del reporte], Report Footer [Pie del reporte], Page Footer [Pie de la página]).
- 6) Estilo y Atributos: Aquí se definen todos (y a detalle) los formatos (colores, letras, rellenos, bordes, formato de números y fecha), así como su comportamiento (eventos).

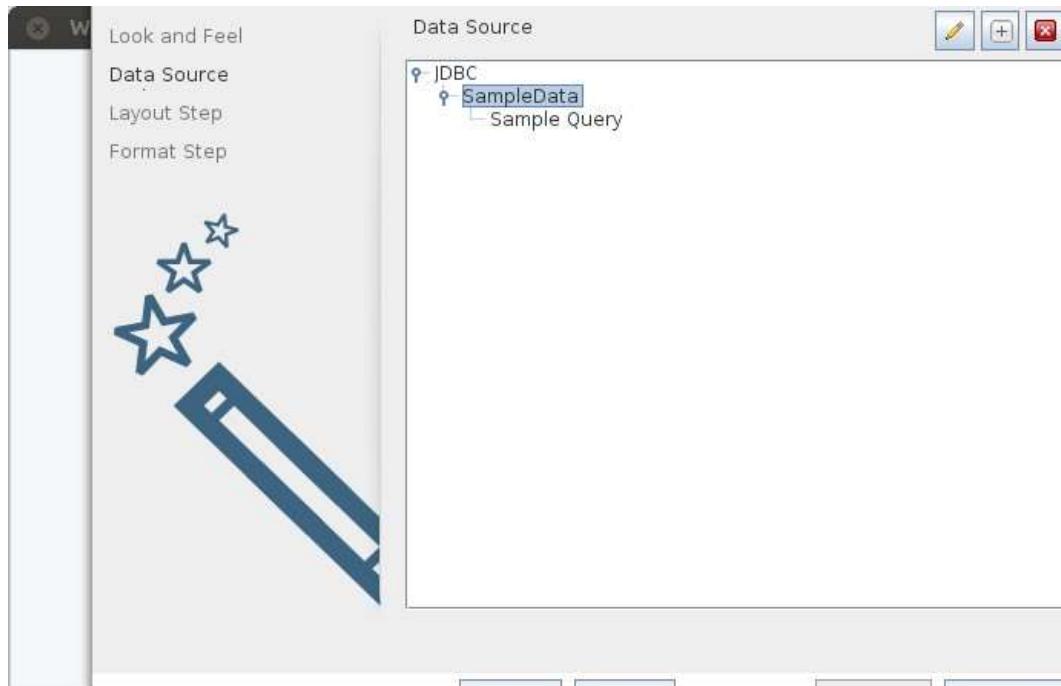
Creando un reporte con el apoyo del asistente.

Usando el Wizard para crear un nuevo reporte.

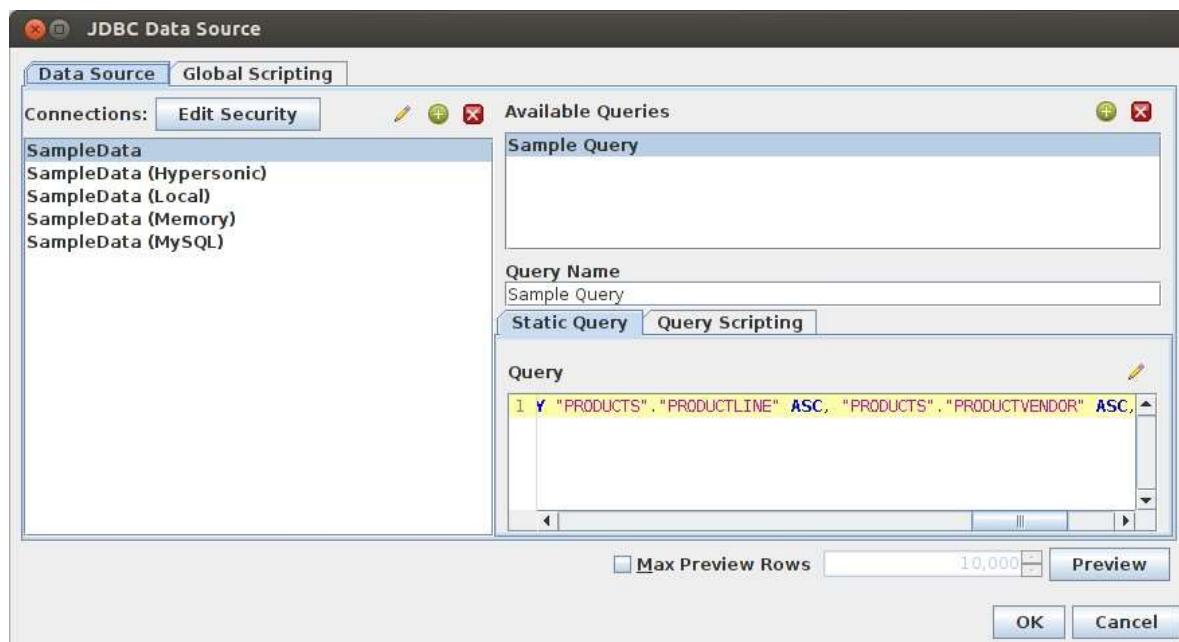
- Lo primero es elegir (si tenemos) una plantilla (template). El PRD tiene algunas precargadas, pero también permite crear plantillas propias y almacenarlas.



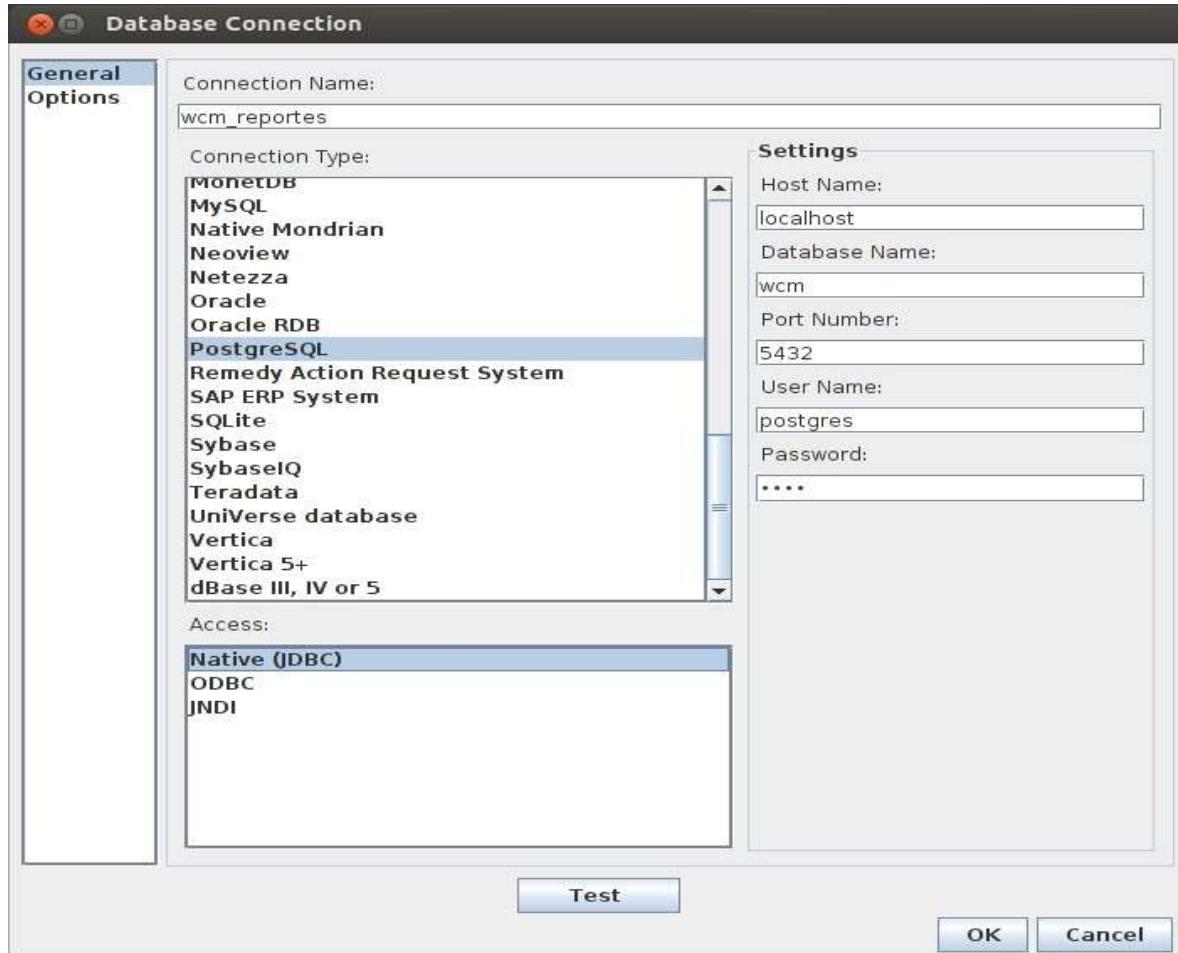
- Posteriormente el asistente solicita seleccionar los orígenes de datos (Bases de datos, archivos, etc.)



- Configuraremos el origen de los datos. Dando clic en el botón podemos configurar la conexión.



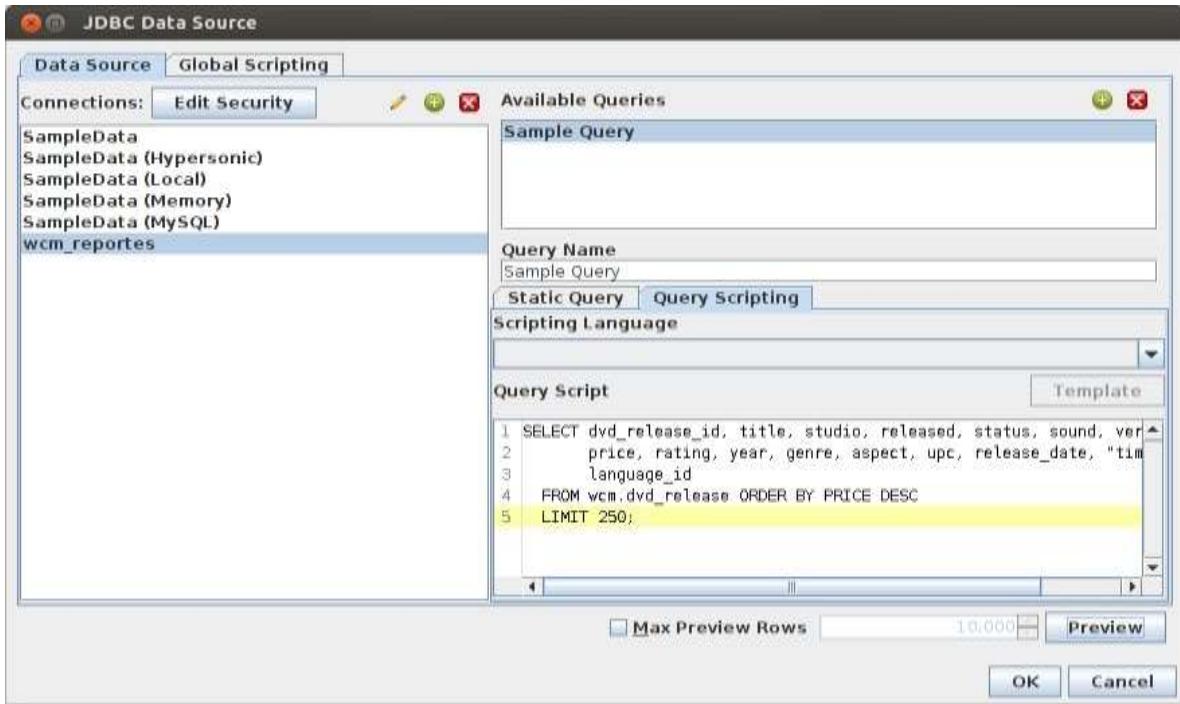
- Se debe configurar la conexión a la base de datos indicando el motor, el servidor, el usuario y el password.



- Al terminar debemos probar la conexión.



Una vez que se ha establecido la conexión se puede escribir una consulta SQL en la ventana Query Script, a la cual le podemos asignar un nombre para que el PRD la almacene y la ejecute para la generación del reporte.

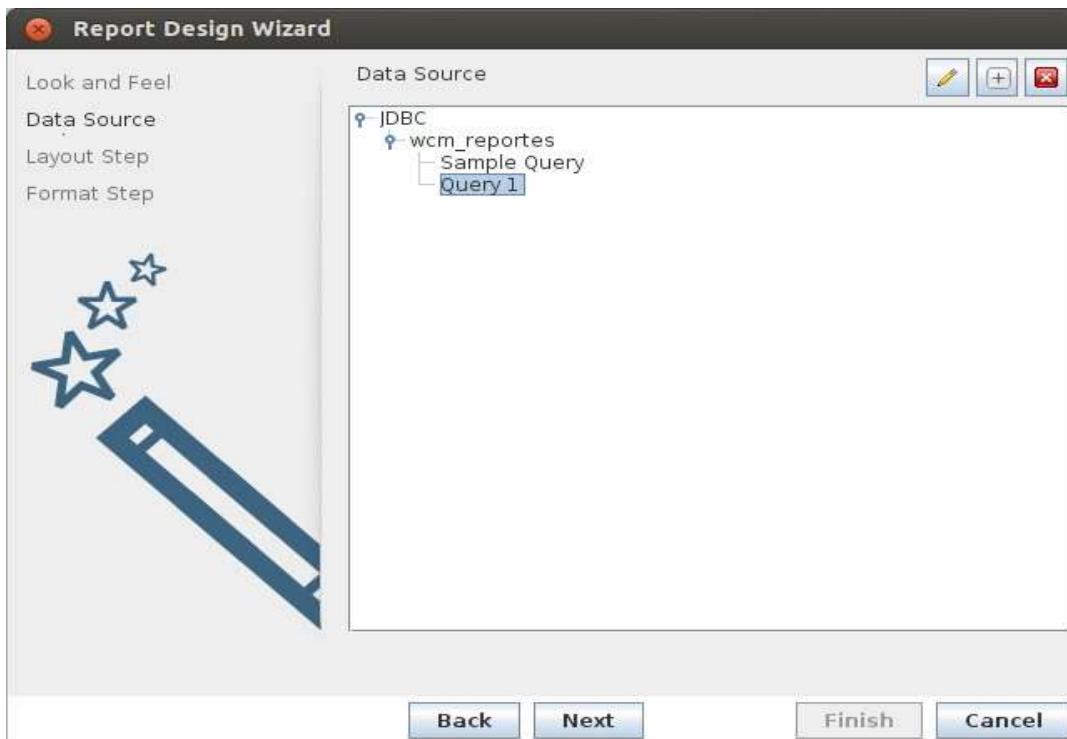


Probamos la consulta dando clic en el botón Preview para ver el contenido de nuestra búsqueda de manera parcial (podemos elegir el máximo número de columnas, no deben ser demasiadas pues sólo requerimos visualizar la información previamente para ver cómo se muestra).

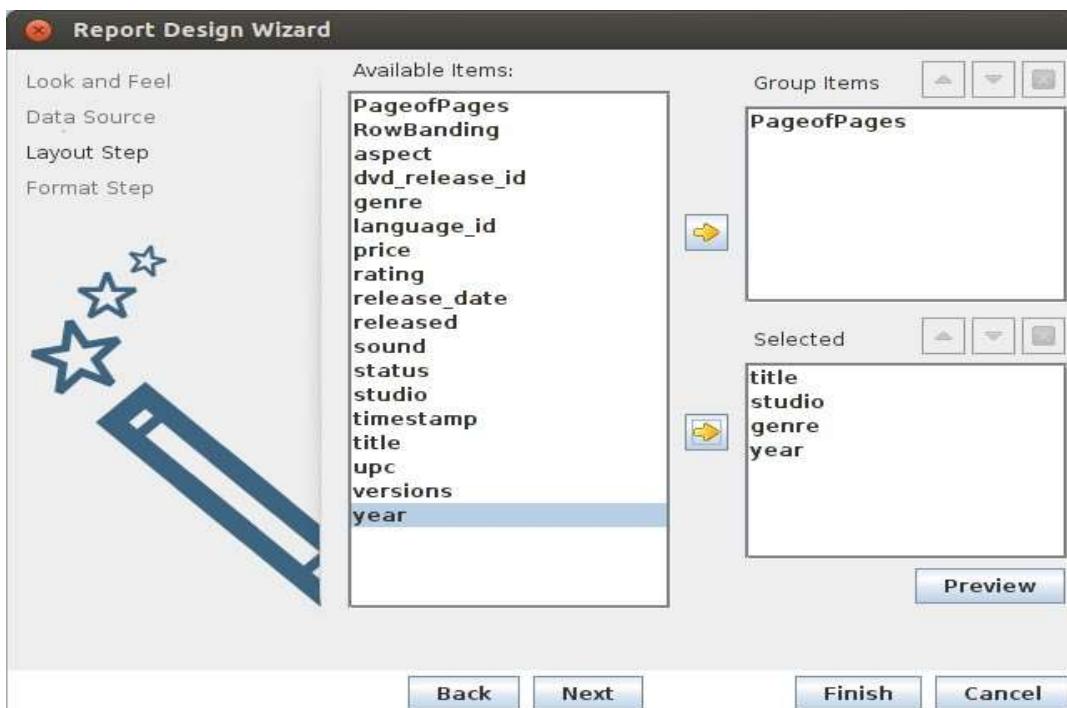
dvd_r...	title	studio	relea...	status	sound	versio...	price	rating	year	genre	aspect	upc	relea...	times...	langu...
7972 Ange...	Vang...	Out	2.0	4:3	19.95	NR	1/01/...	Drama	1.33:1	6587...	3/10/...	4/10/...	25	▲	
8163 Evolver	TriMark	Out	2.0	4:3	14.98	R	1/01/...	SciFi	1.33:1	0313...	1/10/...	9/12/...	36	=	
8566 Spac...	Image	Out	1.0	4:3	14.99	NR	1/01/...	SciFi	1.33:1	0143...	13/1...	11/0...	136	=	
9308 Neil Y...	WEA	Out	2.0	4:3	19.99	NR	1/01/...	Music	1.33:1	0759...	12/1...	10/0...	42	=	
9747 Kenn...	BMG ...	Out	2.0	4:3	14.98	NR	1/01/...	Music	1.33:1	0782...	20/0...	30/1...	16	=	
9837 Ride ...	Buen...	Out	2.0	LBX, ...	9.99	R	1/01/...	Com...	1.85:1	7869...	13/0...	18/0...	92	=	
10692 Misfits	MGM/...	Out	1.0	LBX	9.98	NR	1/01/...	Drama	1.66:1	0276...	19/0...	28/0...	15	=	
10841 Boes...	Kino	Out	SUR	4:3 P...	29.95	NR	1/01/...	Drama	2.35:1	7383...	8/05/...	7/05/...	85	=	
11036 Just ...	Wolfe	Out	2.0	4:3	24.95	UR	1/01/...	Late ...	1.33:1	7432...	1/05/...	11/0...	84	=	
11116 Bast...	Pion...	Disco...	DUB	4:3	29.98	MA15		Anime	1.33:1	0130...	5/06/...	6/04/...	19	=	
11567 Acid ...	Zeitg...	Out	2.0	LBX	29.99	NR	1/01/...	Drama	1.85:1	7959...	28/0...	14/1...	96	=	
11817 Dem...	Shrie...	Out	DUB	4:3	9.95	NR	1/01/...	Forel...	1.33:1	6315...	26/0...	13/0...	28	=	
12053 Sayo...	MGM/...	Out	1.0	LBX	14.98	NR	1/01/...	Drama	2.35:1	0276...	18/0...	28/0...	105	=	
12397 Euros	Wolfe	Out	2.0	4:3	24.95	UR		Late ...	1.33:1	7432...	4/09/...	11/0...	74	=	
12709 Amer...	TriMark	Out	2.0	4:3	14.98	PG-13	1/01/...	Drama	1.33:1	0313...	31/0...	10/0...	94	=	
14391 Conv...	TriMark	Out	2.0	4:3	14.98	R	1/01/...	Horror	1.33:1	0313...	11/1...	9/12/...	131	=	
14053 Tune...	Warn...	Out	2.0	4:3	9.95	NR		Music	1.33:1	7231...	23/1...	29/0...	69	=	
15444 Jame...	Plon...	Disco...	5.1	4:3	19.98	NR	1/01/...	Music	1.33:1	0130...	22/0...	10/0...	71	=	
15512 Made...	MGM/...	Out	1.0	LBX, ...	14.98	NR	1/01/...	Drama	2.35:1	0276...	5/03/...	17/1...	81	=	
15493 Bible ...	Kitty ...	Out	SUB	4:3	29.95	MA17	1/01/...	Anime	1.33:1	6315...	26/0...	8/04/...	139	=	
15852 Gothl...	Artisan	Out	SUR	4:3	14.98	R	1/01/...	Horror	1.33:1	0122...	26/0...	25/0...	102	=	
16290 Archi...	MPI	Out	2.0	4:3	29.98	NR		Docu...	1.33:1	0303...	26/0...	13/0...	30	=	
16512 Blo-D...	MGM/...	Out	2.0	LBX, ...	14.98	PG-13	1/01/...	Com...	1.85:1	0276...	16/0...	28/0...	39	=	
18112 Thre...	Facets	Out	SUB	LBX	29.95	PG	1/01/...	Drama	2.35:1	7368...	16/0...	22/0...	105	=	
18305 Good...	Kultur	Out	2.0	4:3	24.95	NR	1/01/...	Com...	1.33:1	0320...	28/0...	27/0...	28	=	
19385 Gabby ...	Good...	Out	2.0	4:3	5.98	NR		Anim...	1.33:1	0187...	27/0...	24/1...	62	=	
20341 Surge ...	First ...	Out	SUR	4:3	9.98	R	1/01/...	Horror	1.33:1	6877...	17/0...	12/0...	142	=	
20785 All Over	Kultur	Out	2.0	4:3	24.95	NR	1/01/...	Drama	1.33:1	0320...	3/09/...	2/09/...	51	=	
20731 Big K...	MGM/...	Out	1.0	4:3	19.98	NR	1/01/...	Drama	1.33:1	0276...	15/1...	14/1...	19	=	
20816 Typists	Kultur	Out	2.0	4:3	24.95	NR	1/01/...	Com...	1.33:1	0320...	3/09/...	2/09/...	78	=	
21396 Lost ...	A&E ...	Disco...	2.0	LBX	39.95	NR	1/01/...	SciFi	1.78:1	7339...	29/1...	28/1...	70	=	
22822 Peac...	Peac...	Out	2.0	4:3	14.99	UR		Late ...	1.33:1	8256...	19/1...	18/0...	39	=	
23419 O Yo...	Kultur	Out	2.0	4:3	24.95	NR	1/01/...	Drama	1.33:1	0320...	26/1...	18/1...	148	=	
23682 Carola	Kultur	Out	2.0	4:3	24.95	NR	1/01/...	Drama	1.33:1	0320...	25/0...	9/12/...	142	▼	

Close

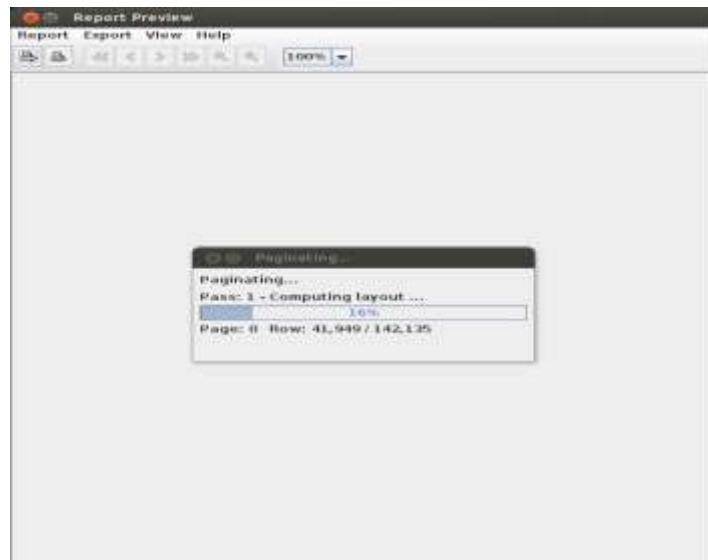
Posteriormente seleccionamos en el wizard las consultas que vamos a utilizar en nuestro reporte con el nombre que les hemos asignado.



Seleccionamos los campos a mostrar y cómo se van a mostrar agrupados los datos.



Finalmente, generamos el reporte.

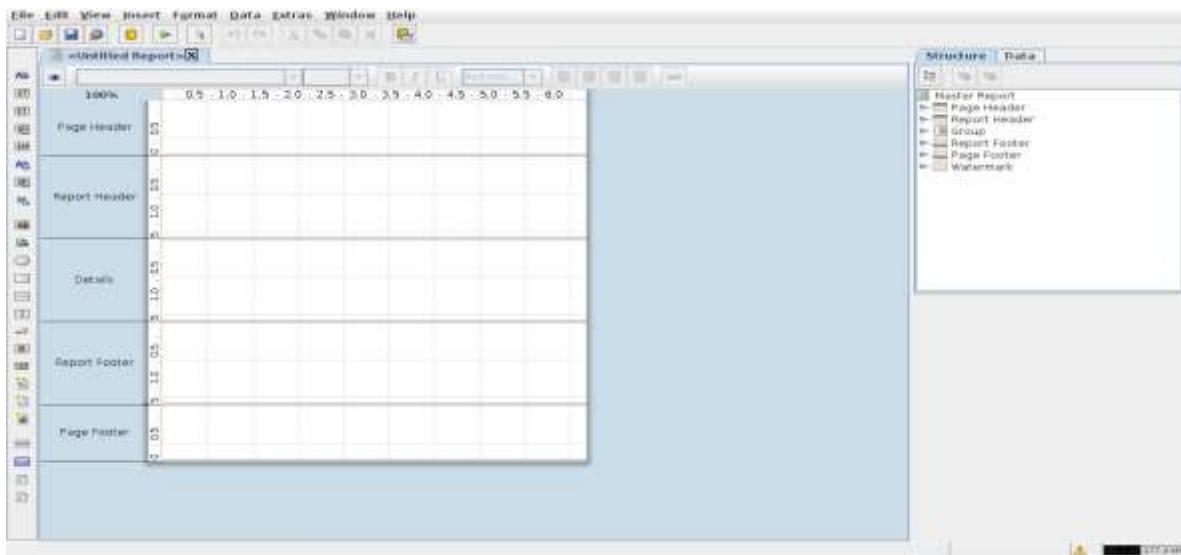


Así se ve el reporte si lo generamos como un archivo html.

A screenshot of a web browser window titled "prueba". The address bar shows "file:///C:/Users/Sara/Documents/Trab". The page content starts with the title "prueba" in a large blue font. Below it, there is generated information: "Generated: 21/02/2014 05:56:33 a.m." and "Database: wcd on postgres@localhost:5432". The main section is titled "Query results" with a table below it. The table has two columns: "count" and "employee_id".

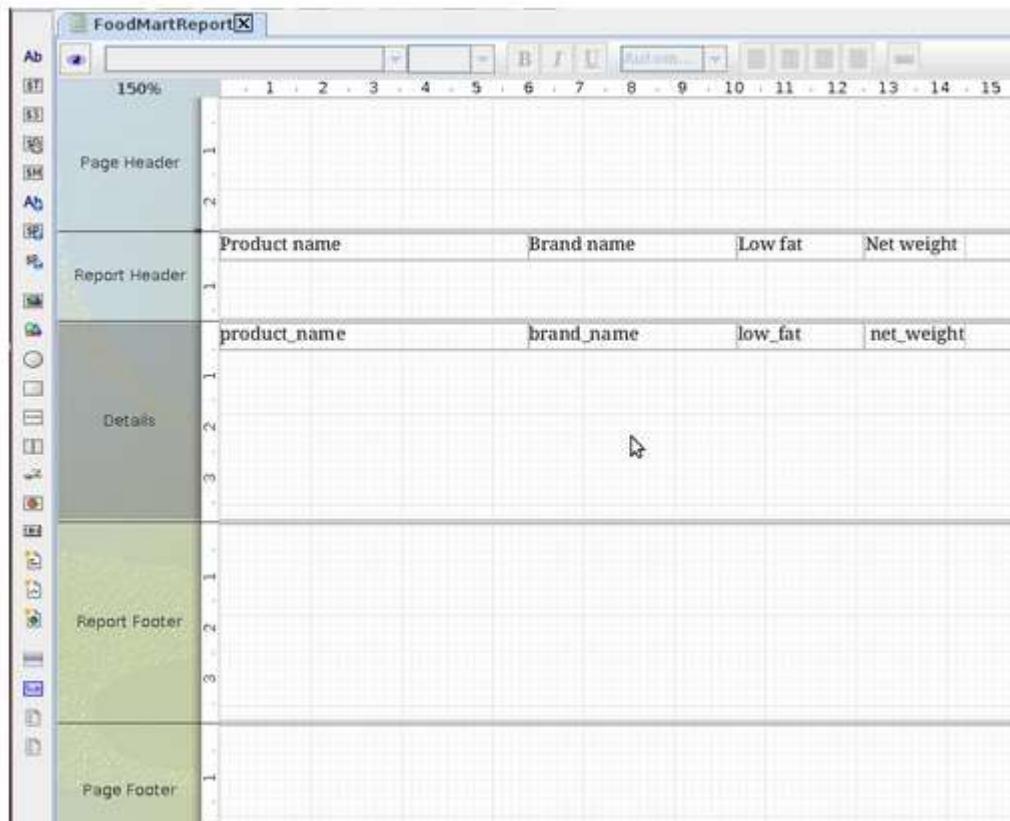
count	employee_id
10	681
12	744
14	809
7	985
5	866
11	887
5	707
13	601
15	802
7	664
10	647
13	971
9	881
3	551
4	930
5	683

Si queremos personalizar nuestro reporte y darle una apariencia más profesional podemos hacer nuestro propio diseño.



A continuación explicaremos cómo está conformada la ventana del PRD.

Observamos la plantilla inicial dividida en secciones: Encabezado de página, Encabezado de reporte, Cuerpo del reporte (Details), Pie del reporte y pie de la página.

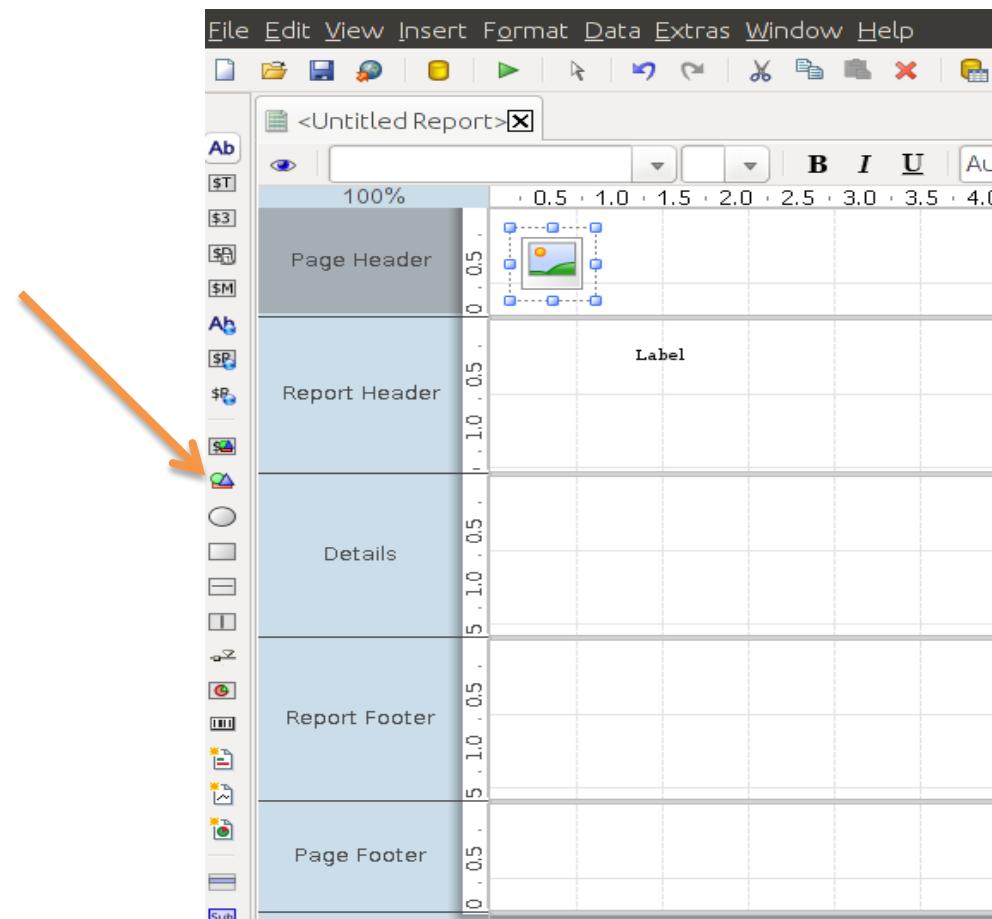


A esta plantilla podemos arrastrar los elementos que se encuentran en la barra lateral izquierda para agregar cuadros, gráficos, imágenes y texto, entre otros.

En cada sección se selecciona lo que queremos mostrar, agregar número de páginas, logos, títulos, subtítulos en los encabezados y los pies del documento.

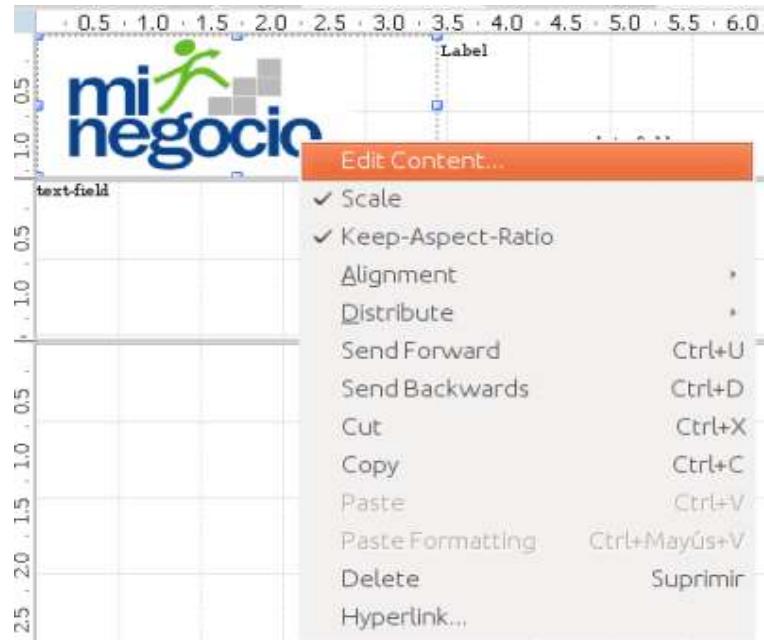
Gráficos, tablas y datos calculados en el cuerpo del reporte se colocan en la sección Details.

Agreguemos una imagen:

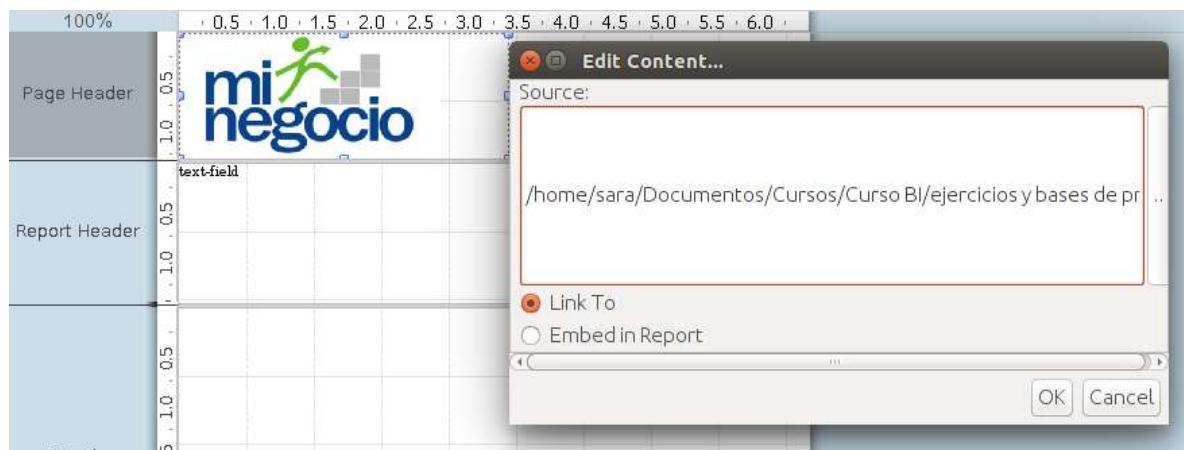


Seleccionamos el icono de imagen y lo arrastramos a la posición donde lo queremos colocar.

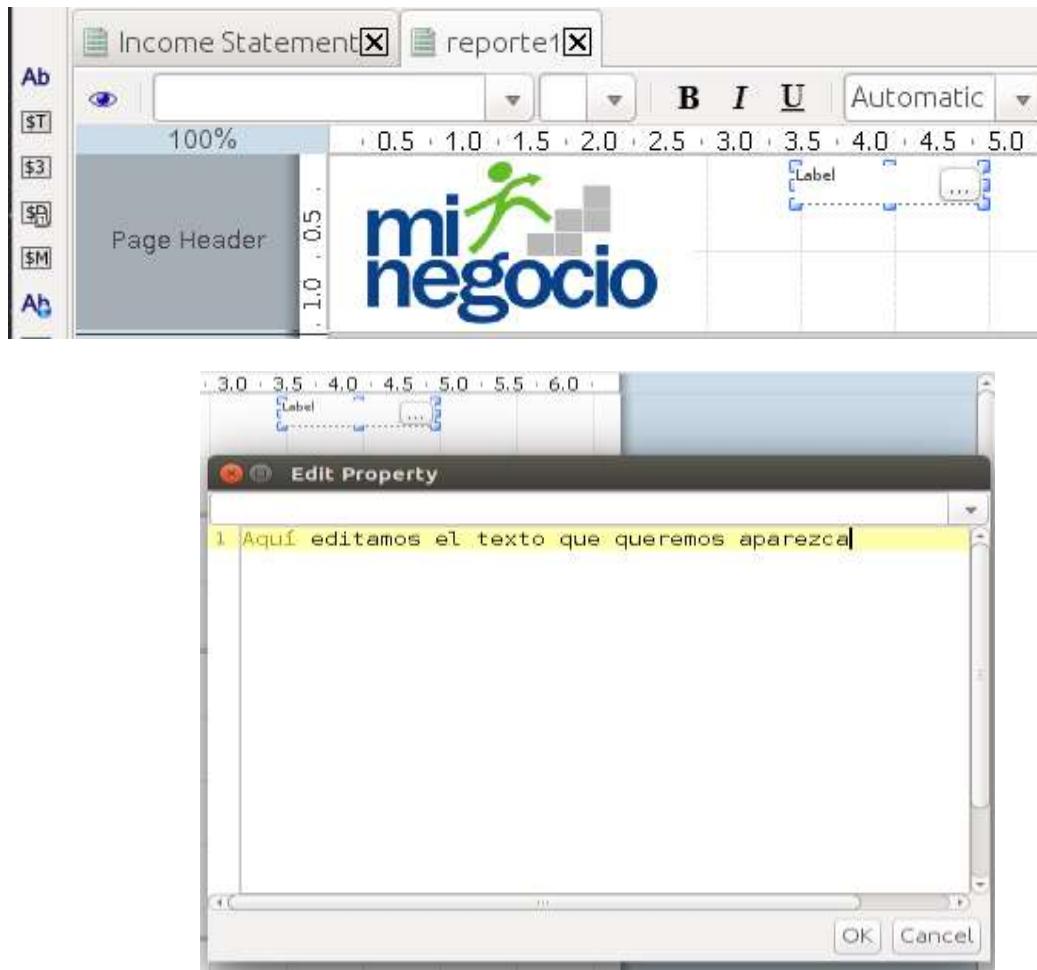
Dando clic derecho sobre el objeto recién creado se despliega un menú, y vamos a elegir "Edit Content" para seleccionar la imagen que queremos colocar ahí.



Debemos escribir en el recuadro la ruta física donde se encuentra el archivo de la imagen. Al terminar debe darse OK para que se agregue la imagen.



Para agregar títulos y texto se debe arrastrar un objeto tipo LABEL del mismo menú. Del mismo modo damos clic derecho sobre la etiqueta para que se despliegue un cuadro en el cual podemos editar el texto a agregar.



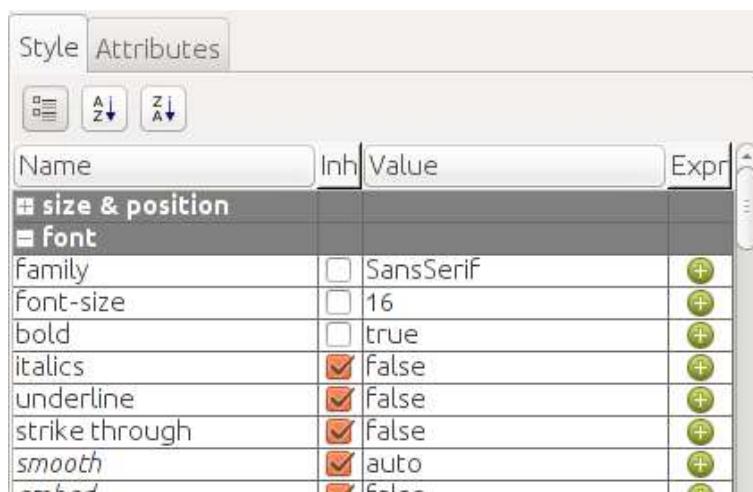
Para modificar el tamaño, tipo de letra, poner negritas, la alineación dentro de la etiqueta, etc. Debemos hacerlo en la sección de Style & Attributes.

Style Attributes			
<input type="button" value=""/>	<input type="button" value="A"/>	<input type="button" value="Z"/>	
Name	Inh	Value	Expr
+ size & position			
- font			
Family	<input checked="" type="checkbox"/>	Serif	<input type="button" value="+"/>
Font-size	<input checked="" type="checkbox"/>	10	<input type="button" value="+"/>
bold	<input checked="" type="checkbox"/>	False	<input type="button" value="+"/>
italics	<input checked="" type="checkbox"/>	False	<input type="button" value="+"/>
underline	<input checked="" type="checkbox"/>	False	<input type="button" value="+"/>
strike through	<input checked="" type="checkbox"/>	False	<input type="button" value="+"/>
smooth	<input checked="" type="checkbox"/>	auto	<input type="button" value="+"/>
embed	<input checked="" type="checkbox"/>	False	<input type="button" value="+"/>
- text			
h-align	<input checked="" type="checkbox"/>	LEFT	<input type="button" value="+"/>
v-align	<input checked="" type="checkbox"/>	TOP	<input type="button" value="+"/>

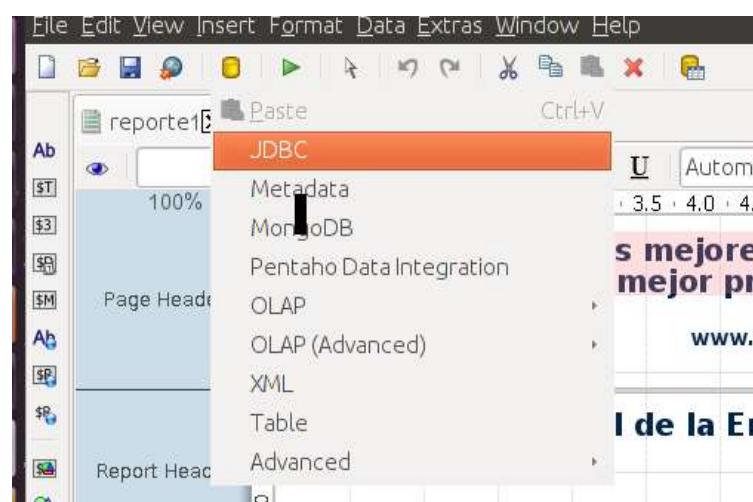
Al agregar un campo de tipo Date podemos agregar una fecha para nuestro reporte.



Para configurar el formato de la fecha demás propiedades del objeto debemos usar la ventana de atributos.

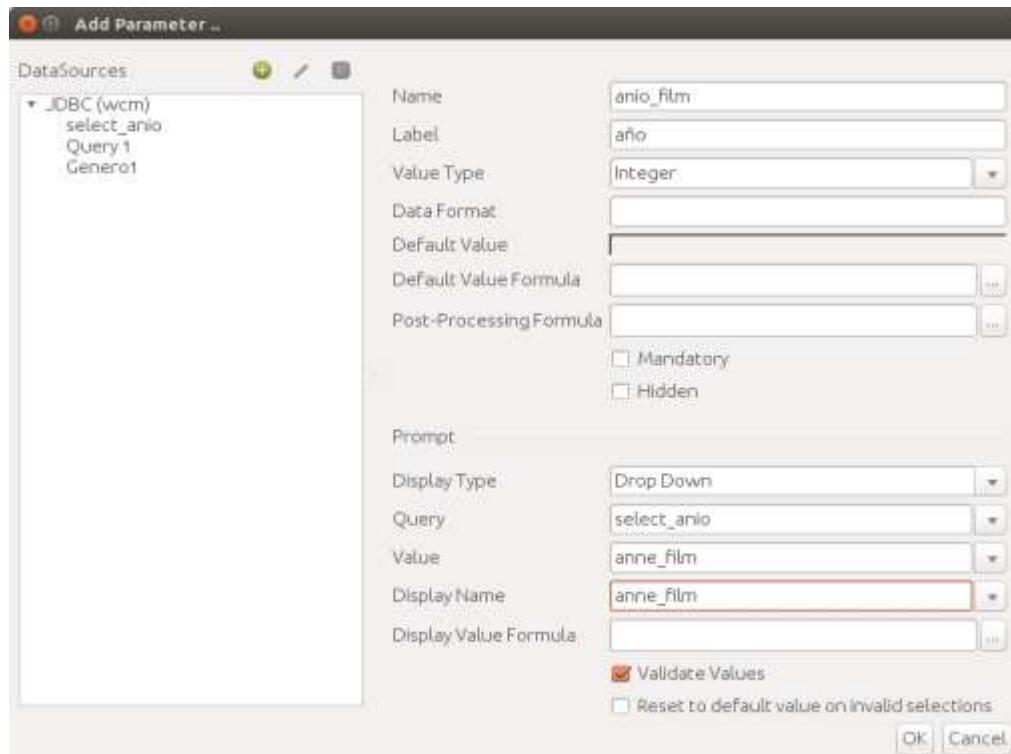
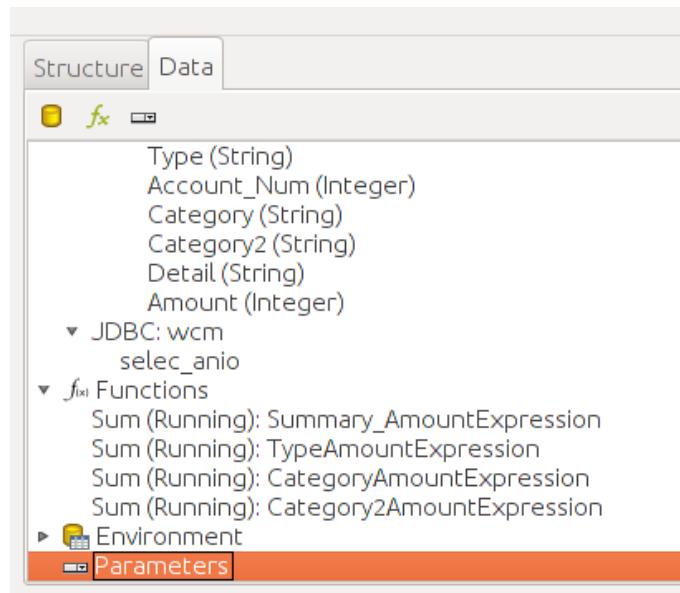


Para hacer la conexión a los datos, se requiere agregar un origen de datos, si se trata de una base de datos elegimos el icono .

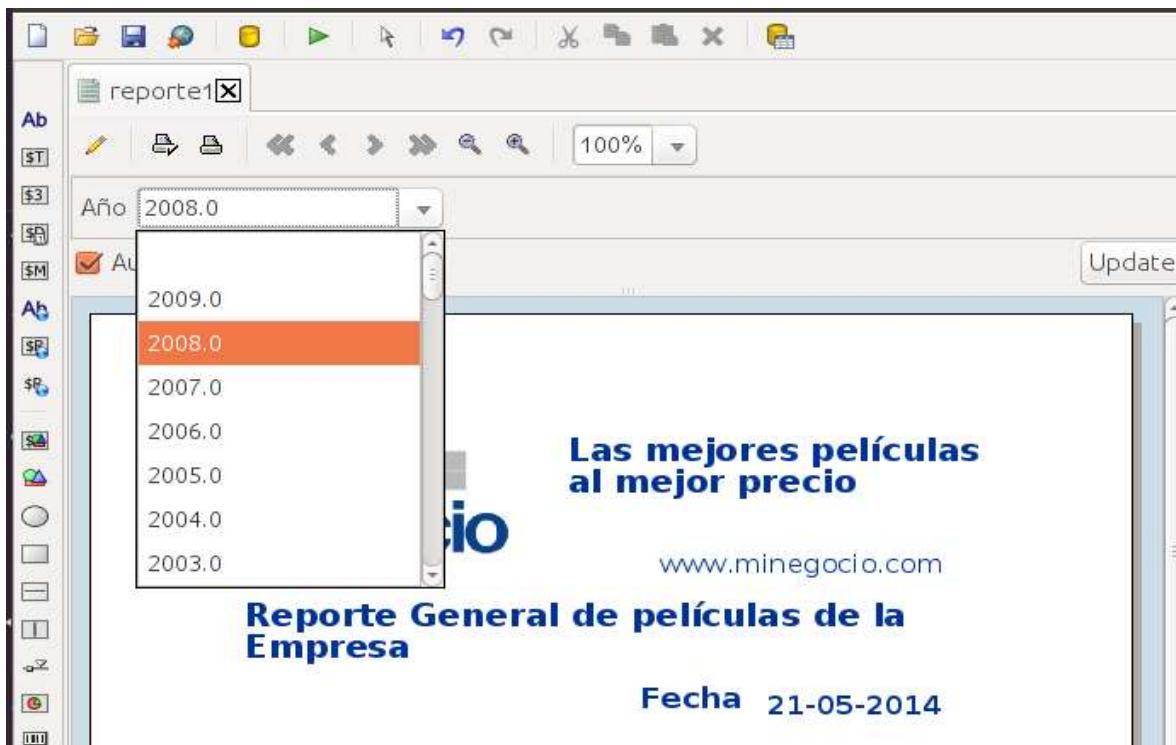


Vamos a hacer la conexión a la base de datos tal como vimos en la sección para crear un reporte con el wizard.

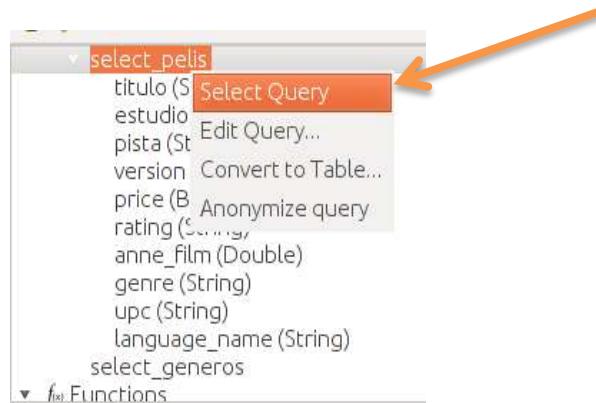
Ahora vamos a ver cómo agregar parámetros. Lo hacemos a través de la ventana de Estructura y Datos, creamos un parámetro dándole un nombre y asignándole la consulta a la cual se va a dirigir el valor para crear el reporte.



Cuando ejecutemos el reporte, podremos elegir el año en el ejemplo para que se haga la búsqueda y nos muestre los resultados correspondientes.



Ahora vamos a agregar el cuerpo del reporte para el parámetro año seleccionado: En data damos clic derecho y seleccionamos "Select Query".



The screenshot shows a report interface with a header for 'mi negocio' and a sub-header 'Las mejores películas al mejor precio'. The main title is 'Reporte General de películas de la Empresa'. Below it is a table with columns: Año, anne_film, Fecha, and report.date. An orange arrow points from the 'anne_film' column header to the 'anne_film' column in the table. To the right, a sidebar lists functions under 'select_pelis' and 'select_generos'.

Luego arrastramos los en el orden que queremos se muestren en el reporte sobre la zona "Details".

Para agregar el número de página en el reporte usaremos una función:

The screenshot shows the PDI interface with the 'Data' tab selected. On the left, there's a tree view with 'Structure' and 'Data' tabs, 'Data Sets' (including 'JDBC: wcm' and 'Genero1'), 'Functions' (which is highlighted), and 'Environment' (with various environment variables listed). On the right, a dialog box titled 'Add Function ..' is open, showing the 'Common' category with 'Page of Pages' selected. There are 'OK' and 'Cancel' buttons at the bottom of the dialog.

Arrastramos el objeto que se ha creado en el menú Data al sitio donde queremos aparezca la numeración:

Structure Data

The Structure tab displays the report's hierarchical structure:

- Ab label: Las mejores películas al mejor precio
- Ab label: www.mineocio.com
- image: /home/sara/Documentos/Cursos/Curso
- Report Header
 - Ab label: Reporte General de la Empresa
 - date-field: report.date
- { Group:
- Report Footer
 - band
 - table-of-content
- Page Footer
 - text-field: PageOfPagesFunction0
 - Ab label: Página
- Watermark

Style Attributes

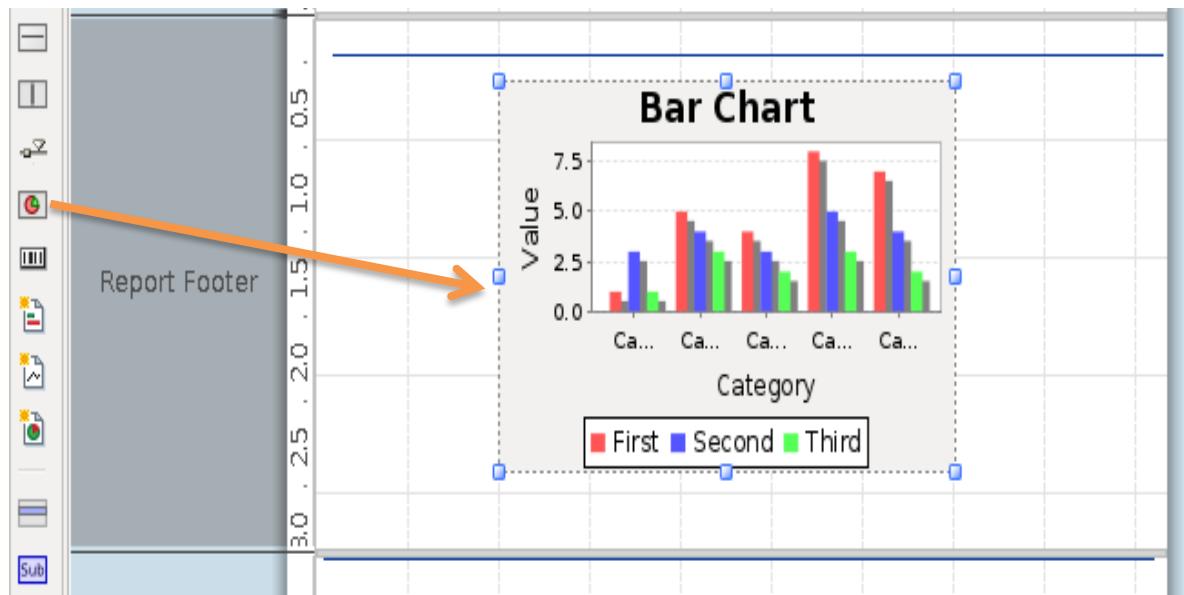
The Style tab shows the properties for the "Page Footer" component:

Name	Inh	Value	Expr
size & position			
layout	<input checked="" type="checkbox"/>		[+]
x	<input checked="" type="checkbox"/>		[+]
y	<input checked="" type="checkbox"/>		[+]
width	<input checked="" type="checkbox"/>	0.0	[+]
height	<input checked="" type="checkbox"/>	0.0	[+]
visible	<input checked="" type="checkbox"/>	true	[+]
font			
Family	<input checked="" type="checkbox"/>	Serif	[+]
Font-size	<input checked="" type="checkbox"/>	10	[+]
bold	<input checked="" type="checkbox"/>	false	[+]
italics	<input checked="" type="checkbox"/>	false	[+]

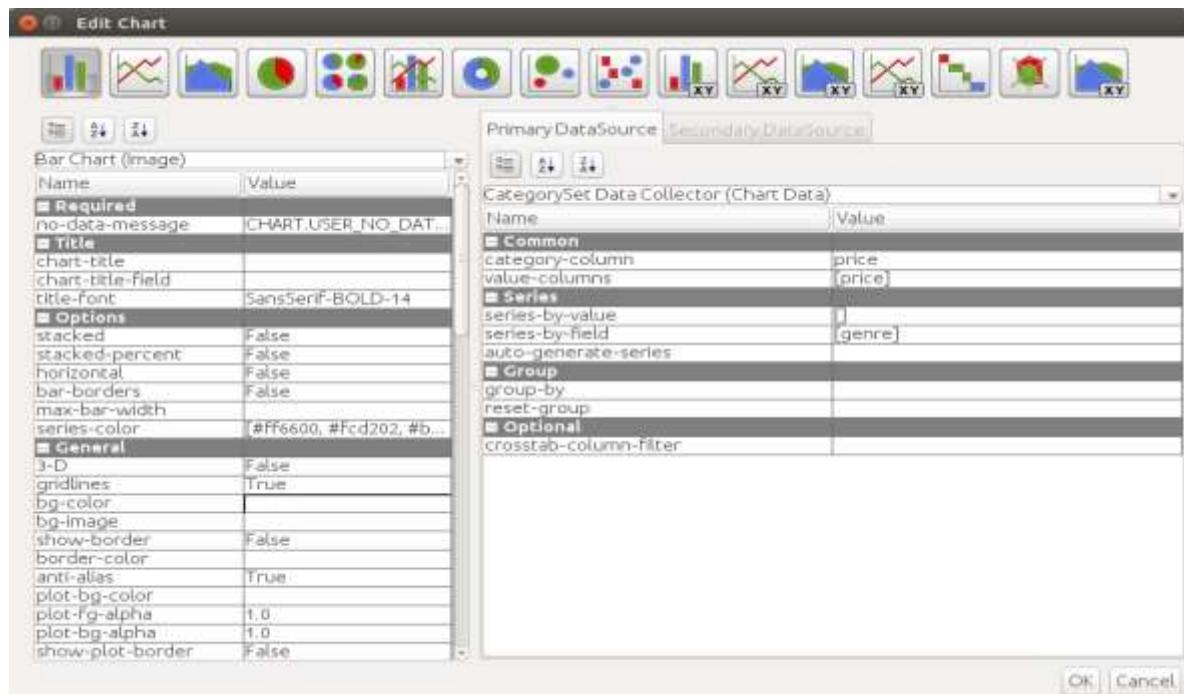


Los objetos tipo line nos ayudan a dar una mejor presentación insertando líneas para remarcar o separar las partes del reporte.

También podemos agregar gráficas para mejorar y aclarar nuestro reporte, arrastrando un objeto gráfica al cuerpo del reporte.



A través de la ventana de atributos podemos agregarle a la gráfica títulos, cambiar el tipo de gráfico, y configurar el contenido que debe mostrar de acuerdo a la consulta que previamente hayamos preparado con la información agrupada en las categorías que seleccionemos.



Al terminar de detallar nuestro reporte lo guardamos y vamos a la vista preliminar  :



The screenshot shows a report viewer window titled "reporte1". At the top, there are buttons for file operations (New, Open, Save, Print, etc.), navigation (Back, Forward, Home, etc.), and search. A zoom control shows "75%" with a dropdown arrow. Below the toolbar, a date selector shows "Año 1976.0" with a dropdown arrow. A checkbox labeled "Auto-Update on selection" is checked. The main content area displays the following information:

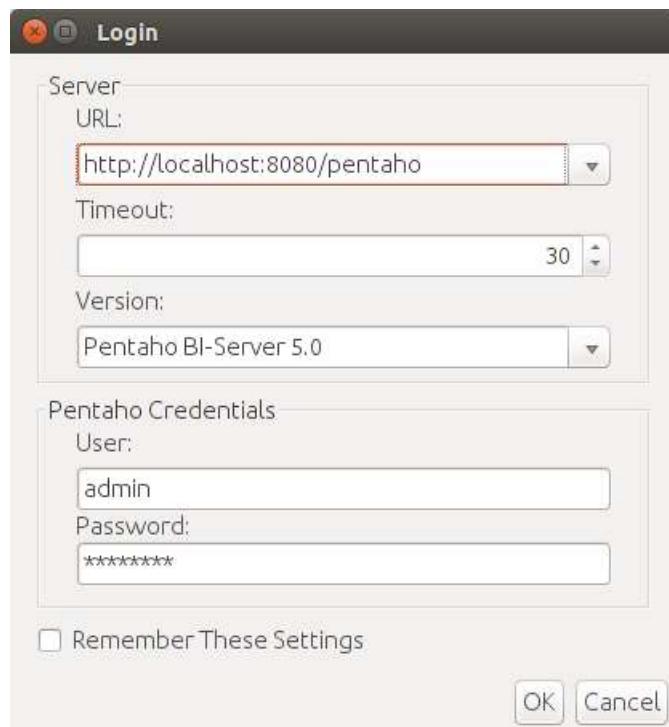
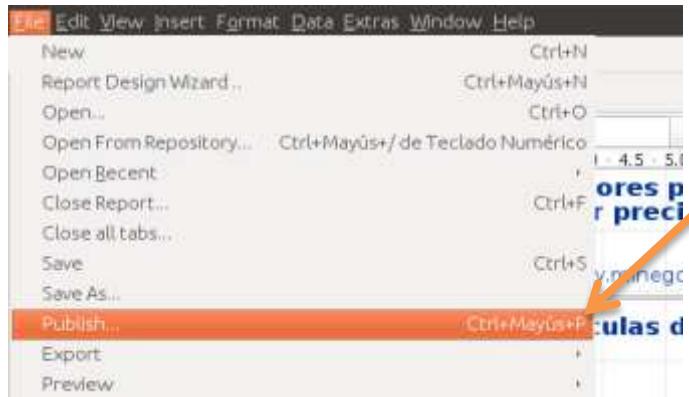
mi negocio *Las mejores películas al mejor precio*
www.mineocio.com

Reporte General de películas de la Empresa

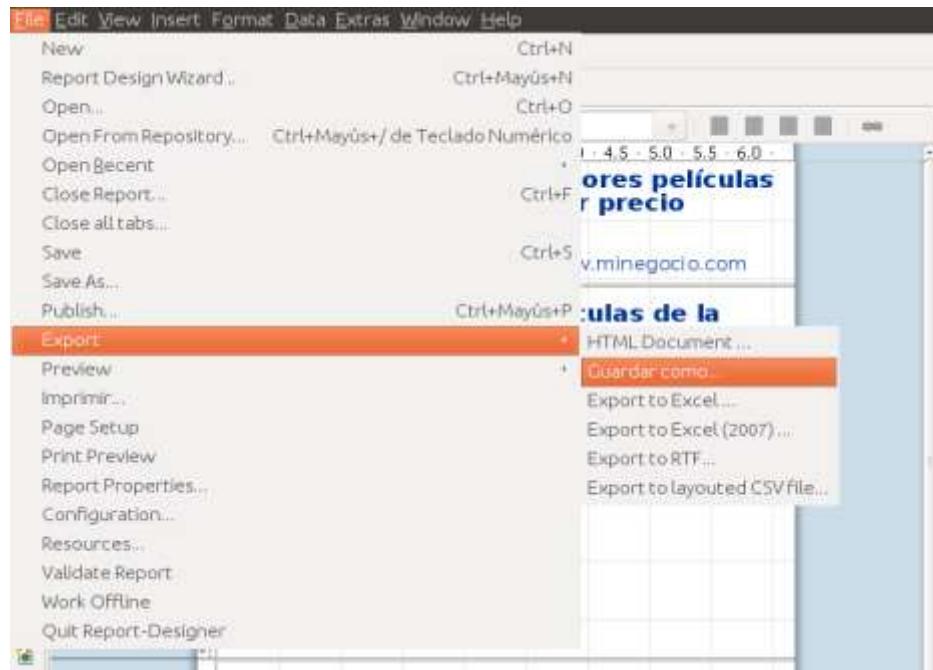
Año	1,976	Fecha	21-05-2014	
All Over	Kultur	Drama	Galician	24.95
Gumball Rally	Warner Brothers	Comedy	Sanskrit	14.97
All The President's Men	Warner Brothers	Drama	Malagasy	19.98
Annie Hall	MGM/UA	Action/Ad..	Tonga	14.98
Assault On Precinct 13 (1976/ Special Edition/ Old Version)	Image	Action/Ad..	Italian	29.99
Baker's Hawk (United American)	United American	Western	Tibetan	12.99
Bloodsucking Freaks (Special Edition)	Troma	Mystery/S..	Romanian	19.95

Página 1 / 61

Nuestro paso final es publicar el reporte a través del BI-Server:



O bien, guardar nuestro reporte en la gran variedad de formatos que nos ofrece el Pentaho Report Designer: html, .xls, .xlsx, .csv, .pdf, etc.



En la carpeta software curso BI/report-designer/samples/ se pueden ver varios ejemplos de reportes para comprender mejor el funcionamiento de la aplicación.

Introducción a Mondrian

Mondrian Schema Workbench es la herramienta para el diseño de cubos, consiste en un JAR (un archivo que permite ejecutar instrucciones escritas en lenguaje Java) que actúa como JDBC (Java Database Connectivity, una API* que permite ejecutar operaciones sobre una BD en Java independientemente del SO en SQL nativo) para OLAP (Procesamiento Analítico en Línea, solución de BI para agilizar la consulta de grandes cantidades de datos), proporciona conexiones y ejecuta consultas SQL en una BD relacional. Los cubos se componen de archivos XML y se ejecutan sobre un servidor Web para permitir la comunicación entre aplicaciones OLAP con BDs.

Pentaho Schema Workbench (PAS) proporciona en su plataforma BI una solución ROLAP a través de lo que llaman Pentaho Analysis Services. PAS está basado en Mondrian, que es el corazón de este, y en Jpivot, que es la herramienta de análisis de usuario, con el que realizamos la navegación dimensional sobre los cubos desde la plataforma BI y visualizamos los resultados de las consultas. Estas son ejecutadas por Mondrian, que traduce los resultados relacionales a **resultados dimensionales**, que a su vez **son mostrados** al usuario en formato **html por Jpivot**. El cual ya forma parte de la instalación del BI Server, así que basta con publicar el cubo desde Mondrian para que éste pueda ser visto en la consola del BI Server.

El elemento principal del sistema son los ficheros xml donde se representan los esquemas dimensionales. Para construir estos ficheros xml, se podría utilizar cualquier editor de texto o xml, o bien la herramienta que nos ofrece Pentaho, que se llama Schema Workbench.

Pentaho Schema Workbench.

Descripción del ambiente de trabajo del Schema Workbench

Al abrir la aplicación veremos una distribución en la cual hay un menú con iconos los cuales son:

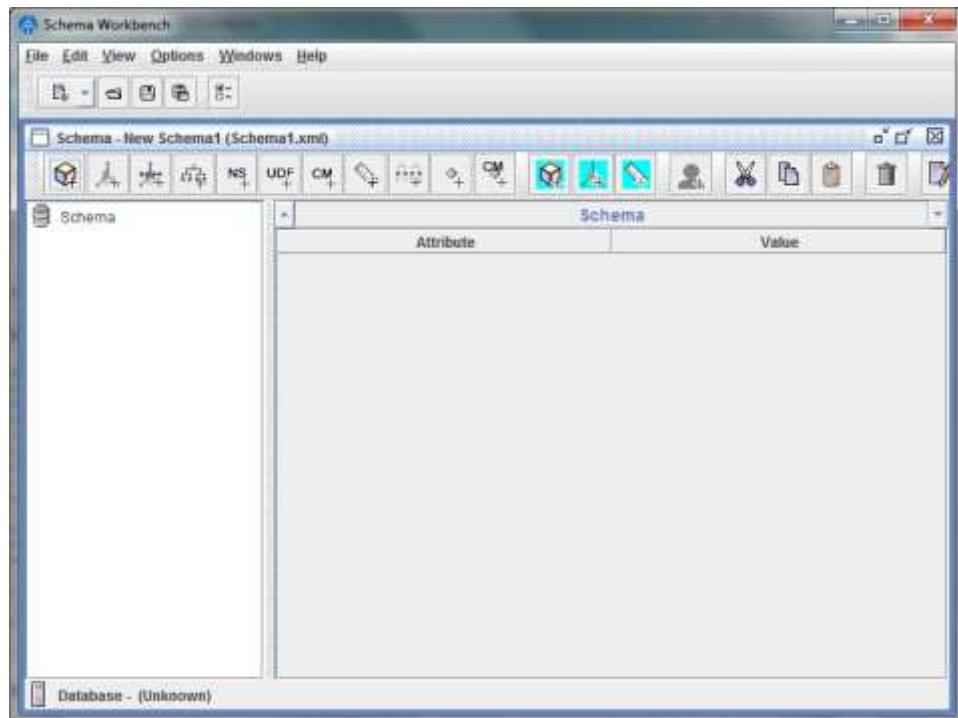


- Creación de un cubo nuevo
- Agregar una dimensión
- Agregar una dimensión compartida
- Creación de una nueva jerarquía

- Agregar una medida
- Agregar un nivel
- Agregar una propiedad
- Agregar una propiedad calculada
- Cortar, copiar, pegar, eliminar y editar

Del lado izquierdo podemos ver un árbol en el que iremos colocando una a una las dimensiones, jerarquías y medidas de nuestro cubo. De acuerdo en primer lugar al Schema, luego al cubo, y en él estarán la tabla de hechos y las dimensiones son sus correspondientes medidas y jerarquías.

Los nombres y propiedades de cada elemento se irán colocando en las tablas de atributos y valores correspondientes.

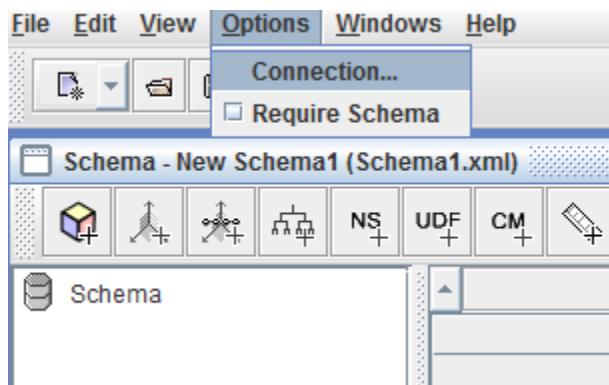


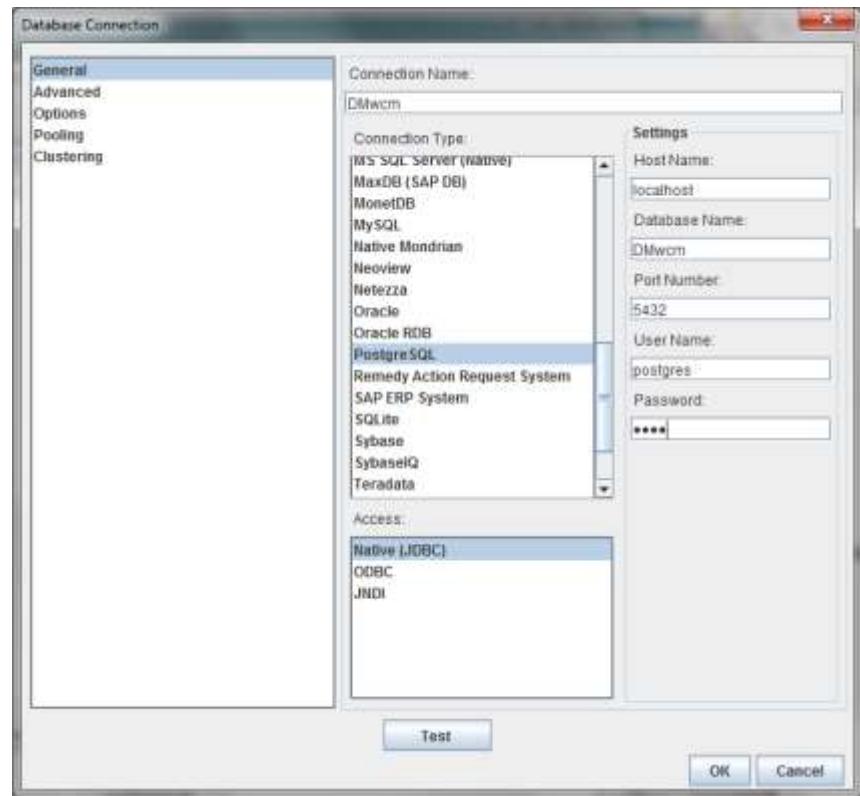
Cómo trabajar con el Schema Workbench

Como en todas las herramientas de Pentaho, en primer lugar hemos de definir las conexiones a base de datos como paso previo a la configuración de los esquemas.

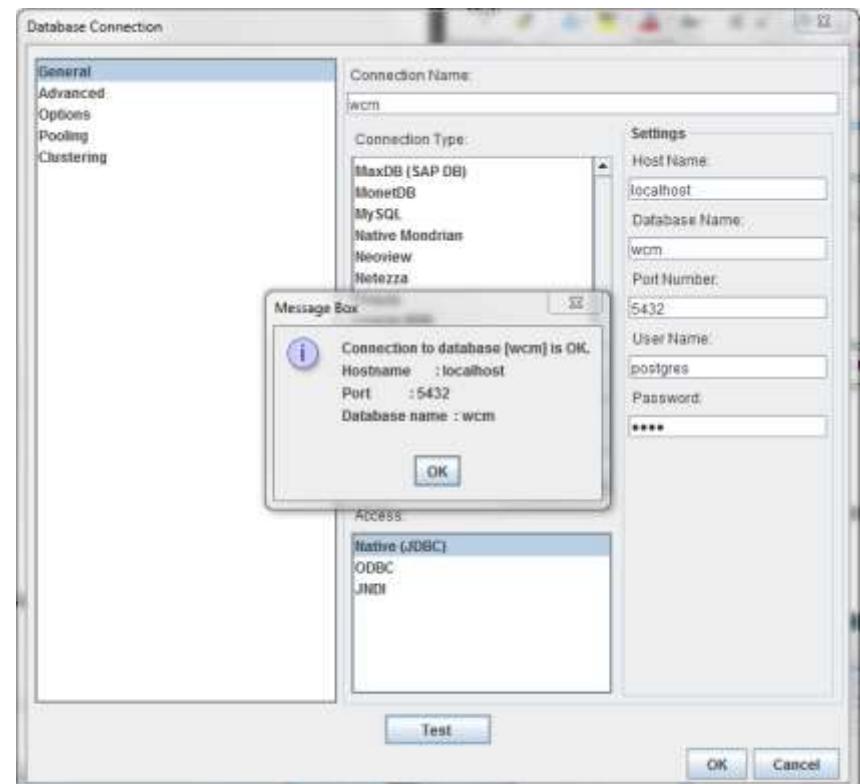
Debemos asegurarnos que los drivers están instalados (los JDBC no vienen por default en la distribución), de lo contrario debemos descargar la versión adecuada y pegarla en el directorio: //pentaho-installation/Mondrian/workbench/drivers/

En la opción del menú Options, Connection se despliega la ventana para configurar nuestro origen de datos.

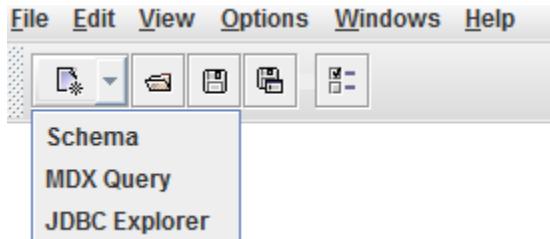




Y probamos la conexión para asegurarnos que podemos acceder a la base de datos.



Procederemos a crear el Esquema. Un esquema es un contenedor de cubos (tendrá un único archivo xml), donde podremos crear tantos cubos como deseemos. Las propiedades que se pueden indicar al crear un esquema son el nombre, la descripción, un nombre para la dimensión que agrupa las medidas y un rol por defecto para utilizar en las conexiones de base de datos.



Una vez creado el esquema, procederemos a la creación de los Cubos, aunque previamente hemos de hacer una consideración. En cada cubo, podemos definir la estructura de tabla de hechos, medidas, miembros calculados y dimensiones. La dimensiones y sus jerarquías podemos definirlas dentro de cada cubo, o crearlas de una forma general dentro del esquema, y luego utilizarlas en los cubos que nos interesen. Esto nos evita tener que definir varias veces lo mismo para cada uno de los cubos, así como reutilizar elementos ya definidos que se tratan en varios cubos. Por tanto, antes de crear los cubos hay que crear las dimensiones compartidas con sus correspondientes jerarquías.

Creación de dimensiones compartidas.

Para añadir las dimensiones, seleccionaremos el esquema y pulsaremos la opción Add Dimension. Le daremos un nombre significativo a la dimensión, y seleccionaremos su tipo y una descripción.

A continuación, iremos creando las diferentes jerarquías que tenga la dimensión. Por ejemplo, en nuestra dimensión tiempo tenemos la jerarquías: Año – Mes – Dia, Semana – Dia, Año – Trimestre – Mes – Dia, etc. Podemos definir tantas jerarquías como deseemos.



Las jerarquías son los niveles de análisis y detalle de la información de nuestro modelo dimensional que luego nos permitirán realizar el análisis y la navegación por los datos con el Mondrian.

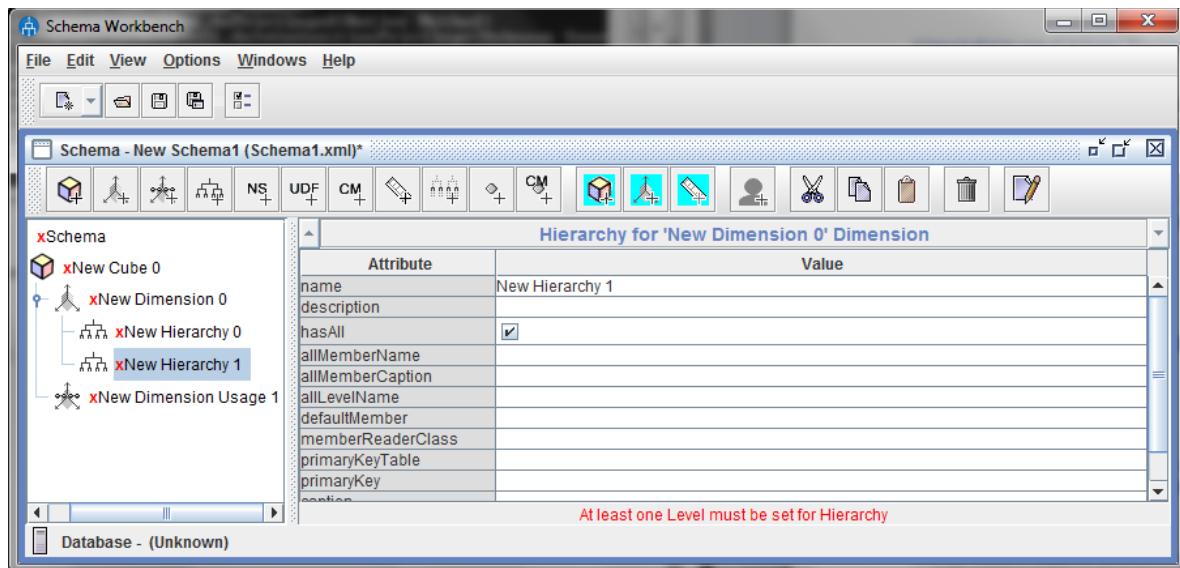
En cada jerarquía, indicaremos una serie de parámetros (los más importantes son el `hasAll`, si queremos que haya un agrupador de todos los valores de la jerarquía, y su descripción en el caso de que esté marcado (`allMemberName`)). Igualmente importante es la clave de la jerarquía y una descripción que luego nos aparecerá al configurar la ejecución del cubo.

Para cada jerarquía, indicaremos una tabla de la dimensión, y a continuación iremos creando los diferentes Niveles (levels) que componen la jerarquía. Para cada nivel, iremos indicando la columna de la base de datos que la describe, el tipo de datos, el tipo de nivel, la columna que contiene la descripción, etc. Esto lo realizaremos para cada uno de los niveles de la jerarquía. El orden con el que se van creando va a determinar la estructura de la jerarquía.

Podemos tener tantas jerarquías como sea necesario dentro de la dimensión. Luego podremos utilizar la que deseemos a la hora de realizar los análisis (la primera será la jerarquía por defecto). Una vez concluido el diseño de todas las dimensiones con sus correspondientes jerarquías, ya podemos proceder a la creación de los cubos.

Creación de Cubos

Al crear el Cubo, le indicaremos un nombre y una descripción, y podemos elegir además las opciones cache (para que Mondrian trabaje con cache en este cubo) y la opción enabled (para que el cubo sea visible).



A continuación, seleccionaremos la Tabla de Hechos del cubo (a partir de la cual podremos calcular las medidas o indicadores). Antes de proceder a crear las medidas, seleccionaremos las dimensiones que queremos incluir en el cubo, con la opción Add Dimension Usage.

Debemos incluir todas las dimensiones necesarias (de las compartidas que hemos creado antes). El cubo hereda todas las características que hayamos incluido en la dimensión, incluyendo todas las jerarquías y sus correspondientes atributos.

Finalmente, necesitamos definir las Medidas, que van a ser los valores de análisis. Tenemos aquéllas que se calculan directamente con campos de la base de datos, y los Miembros Calculados, que son fórmulas en las que utilizamos otras medidas.

Los atributos que podemos indicar para las Medidas son su nombre, descripción, función de agregación (suma, media, valor máximo, valor mínimo, contador, etc.), la columna que genera la medida, si es visible o no (en el caso de campos intermedios que se utilizan para otras medidas puede convenir que no se vean), tipo de datos, formato y "caption" (el nombre que aparecerá cuando lo utilicemos).

En lo referente a los Miembros Calculados, los atributos están más limitados:

Podremos indicar su nombre, descripción, "caption" (el nombre que aparecerá en los análisis), si es visible o no, y el atributo más importante: la fórmula que genera el valor de la medida calculada.

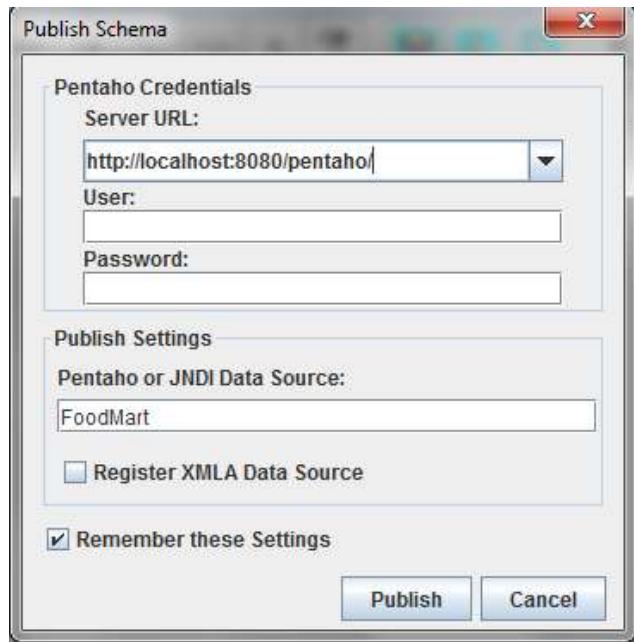
Para indicar los valores de otras medidas, utilizamos la notación [dimension_medidas].[medida]. Por ejemplo, para calcular el importe bruto de ventas, la fórmula, siguiendo esta notación será la siguiente: [Measures].[Unidades]*[Measures].[Precio_Bruto]

The screenshot shows the EnoBI schema editor interface. The menu bar includes File, Edit, View, Tools, Windows, and Help. The toolbar contains various icons for file operations and schema management. The main window has a title bar 'Schema - EnoBI (enobi.xml)*'. On the left, there's a tree view under 'Schema' with a node 'Ventas' expanded, showing its components: 'Table: DWH_VENTAS', 'Tiempo', 'Cliente', 'Producto', 'Unidades', 'Coste_Unitario', 'Litros', 'Dto_Comercial_Unit', 'Precio_Bruto', and three measures under 'CM': 'Importe_Bruto', 'Precio_Neto', and 'Importe_Neto'. On the right, a detailed configuration panel titled 'Cube' displays the following attributes:

Attribute	Value
name	Ventas
description	Cubo Ventas Diarias
caption	Cubo Ventas Diarias
cache	<input checked="" type="checkbox"/>
enabled	<input checked="" type="checkbox"/>

Publicación del Esquema y utilización desde el portal BI

Hemos visto los elementos básicos que podemos utilizar en la definición de un esquema de Mondrian y sus cubos. Disponemos de otros elementos, como los NS (Named Set) o los UDF (User Defined Functions), así como los elementos virtuales (Cubos, Dimensiones y Medidas), en los que no vamos a entrar en detalle. Tras construir el esquema, el paso final para poder utilizarlo en los análisis del portal BI de Pentaho es su publicación.



Para ello, salvamos el cubo y seleccionamos la opción de menú File → Publish. Se nos pide la dirección de publicación del servidor, la contraseña de publicación y los datos del usuario. Se realiza la conexión con el servidor y el esquema ya está disponible para ser utilizado.

Introducción al Dahsboard de Pentaho

Cuadro de Mando Integral

El Cuadro de Mando Integral (CMI), también conocido como Balanced Scorecard (BSC) o Dashboard, es una herramienta de control empresarial que permite establecer y monitorear los indicadores (PKIs) definidos de una empresa y de sus diferentes áreas o unidades.

El Cuadro de Mando Integral se diferencia de otras herramientas de Business Intelligence, como los Sistemas de Soporte a la Decisión (DSS) o los Sistemas de Información Ejecutiva (EIS), en que está más orientado al seguimiento de indicadores (PKIs) que al análisis minucioso de información.

El dashboard, es una herramienta de control enfocada al seguimiento de variables operativas, es decir, variables pertenecientes a áreas o departamentos específicos de la empresa. La periodicidad de los dashboards puede ser diaria, semanal o mensual, y está centrada en indicadores que generalmente representan procesos, por lo que su implantación y puesta en marcha es más sencilla y rápida.



CDE de Pentaho

El Community Dashboard Editor (CDE) es un editor de cuadros de mando a través de una interfaz gráfica web. Es un proyecto independiente que se desarrolla por una empresa portuguesa liderada por Pedro Alves.

Sus antecedentes son:

- CDF (Community Dashboard Framework): Fue un framework de desarrollo que permitía hacer cuadros de mando mediante html y javascript.
- CDA (Community Data Access): Framework que permite el acceso a los datos de una forma más cómoda y elegante. Permite además almacenar consultas previamente calculadas para agilizar el desempeño y permite interactuar con diferentes orígenes de datos.
- CCC (Community Chart Component): permite de manera muy simple la visualización de nuestros datos. El Community Chart Component se basa en la librería de gráficos JavaScript Protovis.

Actualmente el CDE forma parte del paquete de instalación de la PUC (Pentaho User Console) que se ha integrado al BI Server.

Para crear un cuadro de mandos necesitamos **definir** cuáles **indicadores** (PKIs) nos interesa mostrar y cómo se desea que se visualicen.

Ejemplo

- PKI → presupuesto { Por departamento, por área, por sucursal
- PKI → eficiencia en la entrega de productos { por distribuidor
 - Rango→
 - quién se desvía por arriba (gasta más de lo presupuestado, entrega más tarde)
 - quién se desvía por abajo (gasta menos del presupuesto, entrega antes de la fecha límite)

Diseño de un dashboard

El diseño de un Dashboard en el CDE se hace **en 3 capas: Presentación, Componentes y Datos**.

1. Presentación

Se pueden usar plantillas o podemos hacer nuestro propio diseño (Debe considerarse que el ancho máximo de la página es 24 porque está basado en el CSS BluePrint)

2. Componentes

Se elige un componente para utilizar el origen de datos, puede ser una gráfica (chart), un tacómetro, etc.

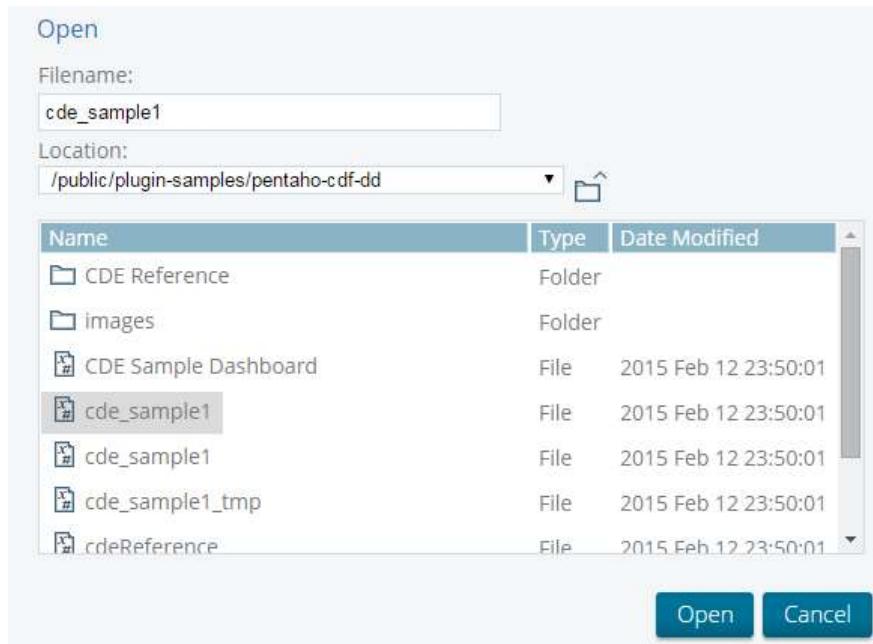
3. Datos

Se configura en la pestaña de Data Sources el origen de los datos de todos los disponibles. Puede ser una BD o un cubo OLAP (este último puede estar en formato MDX [multidimensional] generado con Mondrian). La consulta se hace en SQL.

En el BI Server se pueden consultar algunos dashboards como ejemplos dando clic en el icono



vamos a la carpeta public:



File View Tools Help...

Opened

productLineSales territorySales cde_sample1

[CDA] Community Data Access

Filename: /public/plugin-samples/pentaho-cdf-dd/cde_sample1.cda

DataAccess ID: sqlQuery 1 Exportar Query URL Enviar a caché

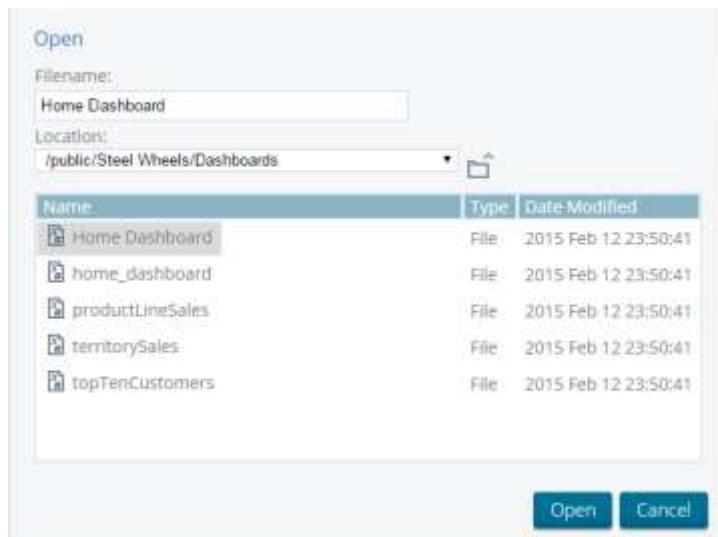
productLine: territory:

Classic Cars APAC

Show 10 elements Search

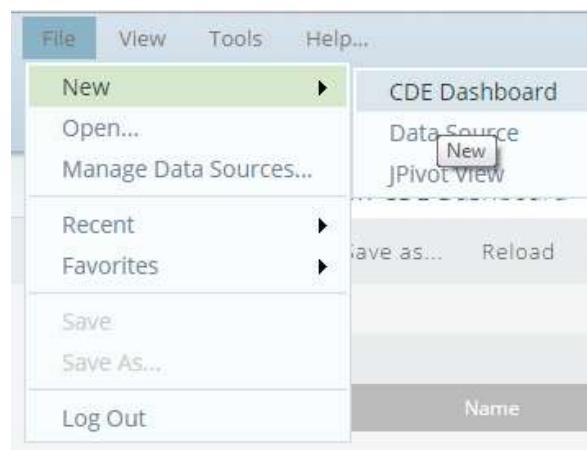
CUSTOMER	TOTAL
Annia's Decorations, Ltd.	74492.24
Australian Collectables, Ltd.	14241.32
Australian Collectors, Co.	50697.09
Australian Gift Network, Co	16973.760000000002
Down Under Souvenirs, Inc.	90489.18000000003
Extreme Desk Decorations, Ltd.	17810.829999999997
GiftsForThem.com	39359.29
Hendji Gifts & Co	51672.32999999999
Kelly's Gift Shop	15368.93
Souveniers And Things Co.	36681.13000000005

View 1 to 10 of 10 elements

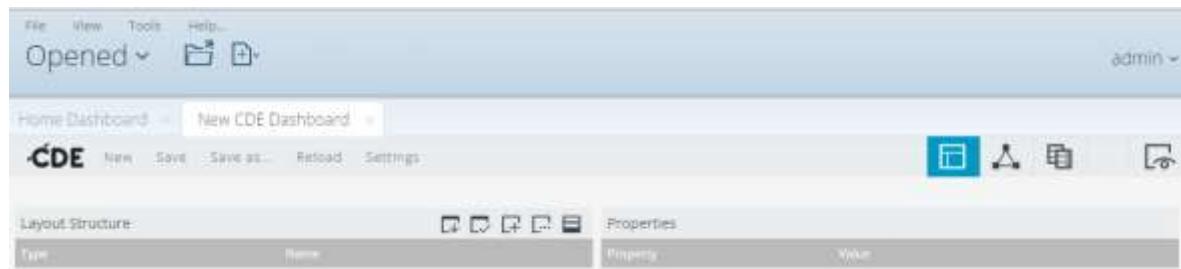




Si queremos crear un nuevo dashboard desde ceros entonces debemos dar clic en:



Con lo que se mostrará una interfaz para el diseño del dashboard:



En el editor se va a trabajar por capas de acuerdo al siguiente menú:





El primer ícono representa la parte de diseño, para lo cual en esa sección se deben cargar los templates, plantillas, archivos css.



El segundo ícono permite agregar los componentes de diseño: gráficas, tacómetros, tablas, etc.



El tercer ícono sirve para establecer los orígenes de datos.



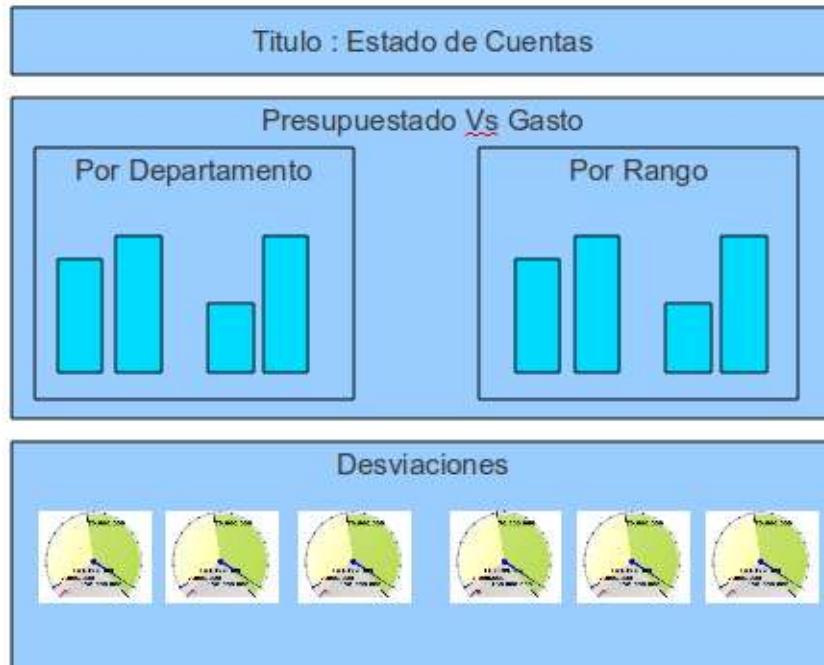
El cuarto ícono es para la visualización del dashboard.

Ejemplo

Listado de PKI

- Ejercicio del presupuesto
 - Por departamento
 - Por cargo
- Quién se desvía más del presupuesto
 - Sobre ejercicio (gasta más de lo asignado)
 - Sub ejercicio (gasta menos de lo asignado)

Boceto de visualización



Aquí debemos decidir la información a mostrar, es importante que sea lo más clara y concisa posible, de lo contrario el usuario se perderá y no comprenderá la información presentada.

Aplicamos una plantilla.

The screenshot shows the 'Layout Structure' interface. At the top, there's a toolbar with several icons: a downward arrow, a blue checkmark, a plus sign, a minus sign, and a grid icon. An orange arrow points from the text above to the blue checkmark icon. Below the toolbar is a header with 'Type' and 'Name' columns, and a 'Apply Template' button. Below this is a preview area divided into three columns. The first column is labeled 'Empty Sample' and contains a large question mark. The second column is labeled 'Two Columns Template' and has two columns. The third column is labeled 'Three Columns Template' and has three columns. Each template preview includes a small 'SampleData' watermark at the bottom right.

Una vez que se ha detallado la plantilla del cuadro de mandos, es necesario establecer el contenido que se va a mostrar, configurar el origen de datos y la presentación de las gráficas. En este caso se plantea hacer la consulta en el visor OLAP.

En particular vamos a seguir estos pasos:

- Definir un origen de datos del tipo MDX y en concreto MDX over Mondrian JNDI
- El nombre del origen de datos que vamos a utilizar es Departamento_DS
- El origen de datos es sobre el schema SampleData que usa el JNDI SampleData
- La consulta SQL que vamos a utilizar es una que nos permitirá ver el actual y el budget para toda la empresa unido al detallado para cada departamento:

```
SELECT NON EMPTY {[Measures].[Actual], [Measures].[Budget]} ON COLUMNS, NON EMPTY Hierarchize(Union({[Department].[All Departments]}, [Department].[All Departments].Children)) ON ROWS from [Quadrant Analysis]
```

The screenshot shows the Pentaho CDE Dashboard Editor interface. On the left, there's a sidebar with a tree view under 'MDX Queries' containing items like 'denormalizedMdx over mondrianJdbc', 'denormalizedMdx over mondrianJndi', 'mdx over mondrianJdbc', and 'mdx over mondrianJndi'. The main area has tabs for 'New', 'Save', 'Save as...', 'Reload', and 'Settings'. A toolbar at the top right includes icons for 'List', 'Diagram', 'Table', and 'List with Diagram'. The central part shows a table with columns 'Type', 'Name', 'Property', and 'Value'. The 'Name' column shows 'MDX Queries' and 'Departamento_DS'. The 'Property' column includes 'Name', 'Access Level', 'Jndi', 'Mondrian schema', 'Query', 'Parameters', 'Banded Mode', 'Columns', 'Calculated Columns', 'Output Options', 'Output Mode', 'Cache Duration', and 'Cache'. The 'Value' column contains 'Departamento_DS', 'Public', 'SampleData', 'SampleData', 'select NON EMPTY {[M (...)}', '[]', 'Compact', '[]', '[]', '[]', 'Include', '3600', and 'True'. At the bottom, there are links for 'About' and 'Documentation'.

En la pestaña de componentes se pueden elegir una gran variedad de gráficos, y se les pueden añadir atributos mediante las “advanced properties”.

The screenshot shows the Pentaho CDE Dashboard Editor interface. On the left, there's a sidebar with a tree view under 'Charts' containing various chart types: 'Protovis Component', 'CCC Area Chart', 'CCC Bar Chart', 'CCC Boxplot Chart', 'CCC Bullet Chart', 'CCC Dot Chart', 'CCC Heat Grid', 'CCC Line Chart', 'CCC Metric Dot Chart', 'CCC Metric Line Chart', 'CCC 100% Stacked Bar Chart', 'CCC Pie Chart', 'CCC Stacked Area Chart', 'CCC Stacked Dot Chart', and 'CCC Stacked Line Chart'. The main area has tabs for 'New', 'Save', 'Save as...', 'Reload', and 'Settings'. A toolbar at the top right includes icons for 'List', 'Diagram', 'Table', and 'List with Diagram'. The central part shows a table with columns 'Components' and 'Properties / Advanced Properties'. The 'Components' table shows 'Type' (Group) and 'Name' (charts). The 'Properties / Advanced Properties' table shows 'Property' and 'Value'. The 'Property' column includes 'Name', 'Title', 'Listeners', 'Parameters', 'Datasource', 'Width', 'Height', 'HtmlObject', 'clickable', 'clickAction', 'compatVersion', 'crosstabMode', 'legend', 'seriesInRows', 'timeSeries', and 'timeSeriesFormat'. The 'Value' column contains 'chart', 'Ejemplo-dashboard', '[]', '[]', 'Departamento_DS', 'Departamento_DS', '300', 'Panel_1', 'False', '2', 'True', 'True', 'False', 'False', and 'My-Item-Id'.

Un tipo muy particular de gráfico que es característico de los dashboards son los llamados “tacómetros”. En este caso se utiliza un xaction (un XML con una secuencia de acciones definidas creado mediante un editor [Pentaho Design Studio] que muestra sus resultados en un HTML).

Bibliografía sugerida

Inmon, Building the Data Warehouse, (Third Edition). John Wiley & Sons, 2002

Kimball et al., The Data Warehouse Lifecycle Toolkit. 2nd Edition. New York, Wiley, 2008

Gustavo R. Rivaderra, La metodología de Kimball para el diseño de almacenes de datos (Data warehouses) <http://www.ucasal.edu.ar/htm/ingenieria/cuadernos/archivos/5-p56-rivaderra-formateado.pdf>

Calzada, Leticia y José Luis Abreu, El impacto de las herramientas de inteligencia de negocios en la toma de decisiones de los ejecutivos. Daena: International Journal of Good Conscience. 4(2): 16-52. Septiembre 2009. ISSN 1870-557X

<http://www.sinnexus.com/> Página sobre definiciones de Business Intelligence

Tutorial de PostgreSQL por El equipo de desarrollo de PostgreSQL, Editado por **Thomas Lockhart** <http://palomo.usach.cl/Docs/postgres/Postgres-Tutorial.pdf>

Cómo hacer cuadros de mando con Dashboard Editor de Pentaho. Artículo de Juanjo Ortíles.

<http://www.dataprix.com/empresa/recursos/como-hacer-cuadros-mando-dashboard-editor-pentaho>