Proyecto del Máster Oficial de Software Libre

Desarrollo de una interfaz gráfica para la creación de sentencias SQL:

Caso base de datos PostgreSQL

Realizada por:

Luis A. Ochoa P.

lochoap@uoc.edu

Consultor:

Gregorio Robles

18 de Enero del 2008

Licencia de publicación

Attribution-ShareAlike 2.0

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.
d. **"Original Author"** means the individual or entity who created the Work.
e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.
f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
g. **"License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to create and reproduce Derivative Works;
c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;
d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.
e. For the avoidance of doubt, where the work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).

f. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.
b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.
c. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL,

PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Resumen

Este trabajo trata sobre la creación de un proyecto de software libre, llevada a cabo desde su inicio hasta su conversión en un proyecto que cuente con el apoyo de una comunidad de usuarios. Específicamente, la finalidad del proyecto de software libre generado, es la creación de una aplicación que sea capaz de guiar de forma gráfica, rápida e intuitiva al usuario a través del proceso de creación de consultas SQL para la base de datos postgreSQL.

Para llevar a cabo esto, el presente proyecto intenta bosquejar los pasos iníciales de creación de un proyecto de software libre, primero dándole un marco definitorio al mismo, mediante la asignación de objetivos por cumplir, y limitando su campo de acción al definir el problema a tratar, su estado actual, sus alcances y limitaciones, así como generando por último una posible lista de especificaciones iníciales que permitan obtener un proyecto de software libre exitoso.

Posteriormente se procede llevar a cabo una explicación de los fundamentos teóricos que soportan el software generado por el proyecto en ejecución, explicando también aquellos que son importantes para comprender el estilo de desarrollo adoptado por la aplicación, debido a la naturaleza libre de su licenciamiento.

A continuación, se explica de forma breve cuales fueron las fases de desarrollo que adoptadas por el proyecto, y como se logro que una aplicación que un inicio fue desarrollada de forma cerrada siguiendo una metodología típica del software propietario, llevase a cabo un proceso de transición sencillo que le permitió convertirse posteriormente en un proyecto de software libre que cuenta con el apoyo de una comunidad de usuarios, dispuesto a colaborar y formar parte en el mismo, haciendo énfasis para esto en lo cambios sufridos por el proyecto para lograr esta transición de forma transparente que garantice el éxito y continuidad del proyecto en el futuro. Luego, se explica la forma en la cual se llevo a cabo el diseño de la aplicación y el proceso que motivo a escoger su tipo de licenciamiento dual GPL / LGPL, así como una breve introducción a lo que consistió su proceso de implantación, mantenimiento y desenvolvimiento con la comunidad.

Por último se lleva a cabo una valoración de los resultados obtenidos, que lleva a reflexionar sobre el cumplimiento de los objetivos planteados al inicio del proyecto, explicando en algunos casos las posibles razones de su éxito o fracaso. También se expresan algunas ideas finales sobre cuál puede ser el futuro de la aplicación, y que se puede esperar de la misma en futuro, para motivar a nuevas generaciones de voluntarios a participar en la misma. Por último se presentan unas breves conclusiones acerca de lo que significo realizar este proyecto.

Nombre de la aplicación: Visual SQL Builder

Versión Actual: pre-alpha 0.3

Sitio Web del proyecto: http://sourceforge.net/projects/vsqlbuilder

Licencia: Dual GPL / LGPL v2.

Tabla de contenidos

Índice de ilustraciones

Índice de tablas

Introducción

El proceso de acceso a datos previamente almacenados en base de datos relacionales, es un proceso que requiere del aprendizaje de un lenguaje conocido como SQL (Structured Query Language), cuya principal función es la de comunicarle al software que gestiona los datos las ordenes a ejecutar, por medio del uso de este lenguaje estándar que resume en su estructura las instrucciones que le han sido enviadas a la base de datos por la persona.

Es aprendizaje y uso del lenguaje SQL, posee algunas limitaciones que no facilitan su adopción por parte de personas con escasos conocimientos técnicos de informática, siendo la más importante de estas, el hecho de que su interfaz de comunicación se encuentra constituida por una línea de comandos. Es por esto que este proyecto surge como una alternativa a esta limitación, al permitir la construcción de sentencias SQL mediante el uso de una interfaz gráfica.

Se espera que este lenguaje se coloque a disposición de las personas hacia las cuales antes estaba restringido, además de facilitar también el proceso de aprendizaje del mismo en personas que aunque con conocimientos técnicos de informática desconocen su estructura o por el contrario aunque conocen su estructura, desconocen es la organización interna de los datos que desean consultar.

Por lo que el presente proyecto se pretende, conseguir la herramienta planteada, pero mediante el uso de técnicas de desarrollo de software libre, con el fin de crear un proyecto que sea autosustentable en el tiempo y que sea capaz de crear un base de usuarios que colaboren con el mismo en la creación de una comunidad, que se encargará de mejora y mantenimiento.

Capitulo 1

1.1 Motivación

El presente proyecto tiene como tópico principal, el presentar una herramienta de software que permita la creación de una comunidad basada en los principios del software libre, la cual haga crecer en torno a la misma una herramienta que tenga como finalidad principal, el facilitar el acceso a los SGBD (Sistema Gestor de Base de Datos) relacionales y por lo tanto a los datos que estos contienen, a personas con escasos conocimientos del modelo de datos entidad relación y el lenguaje SQL (por sus siglas en ingles, Structured Query Language), o que aún teniendo los conocimientos técnicos necesarios no poseen información acerca de la estructura de base de datos creada por terceras personas que desean consultar y lo desean hacer de una forma rápida y sencilla.

Es importante recalcar que la interfaz que se pretende crear con esta aplicación va a intentar ser una metáfora visual usando una GUI (por sus siglas en ingles, Graphic User Interface) de la misma estructura que se puede plasmar mediante la interfaz de comandos para el lenguaje SQL, y por esto podrá hacer disponible el uso de SQL a una mayor audiencia, para la cual antes no estaba disponible, ya fuese por desconocimiento de su existencia o temor a su aprendizaje, lo cual permitirá a esta nueva audiencia iniciar su aprendizaje del lenguaje SQL de una forma poco traumática o llevar a cabo sus actividades con este lenguaje de forma más rápida e intuitiva.

Es destacable que el uso de una GUI para representar las sentencias SQL, se traducirá en una mayor productividad en las personas que deseen hacer uso del lenguaje SQL, posean o no conocimientos técnicos del mismo, debido a que les presentara la información necesaria para realizar sus consultas, como por ejemplo nombres de las tablas existentes, de una forma amigable que les evitara tener que memorizar la estructura de un modelo de datos, para poder posteriormente consultarlo, además de guiarlos de forma correcta entre los elementos constitutivos del lenguaje SQL, lo cual evitara que hagan uso de conceptos erróneos de su estructura sintáctica y semántica.

Para la realización del presente proyecto, se conto también con la experiencia de proyectos similares pero en el área de aplicaciones comerciales y académicas, las cuales sirvieron de base para plantear su estructura inicial, pero se diferencio de estas en un aspecto fundamental, este proyecto fue realizado en un entorno de software libre. Lo cual hizo posible la incorporación de ideas de terceros, que posteriormente lo volverán auto sustentable en el tiempo, y permitirán su continuo avance y mejora, debido a que ya no solo integrará aspectos típicos del desarrollo de software sino también involucrará aspectos humanos y sociales en su desarrollo que lo harán perdurar en el tiempo.

El software que fue necesario para su realización se encuentra disponible bajo una licencia abierta y el conocimiento técnico necesario está disponible en libros, fuentes en línea, foros y comunidades virtuales que le dieron sustento y soporte al proyecto

1.2 Planteamiento del problema

El avance de las ciencias de la computación en los últimos 20 años ha colocado a disposición de las grandes masas todo un sin fin de dispositivos electrónicos capaces de almacenar información de forma estructurada, siendo usado normalmente para su almacenamiento un software altamente especializado conocido como SGBD, debido a que este simplifica la organización, almacenamiento y recuperación de estos datos.

Aunque este momento existen muchos tipos de Sistemas Gestores de Base de Datos, uno de los más usados en este periodo de tiempo, es el conocido como el SGBD relacional, que tiene como base el uso de "tablas" para el almacenamiento de la información, y el uso de un lenguaje comúnmente conocido como SQL para la recuperación de datos y gestión del SGBD.

El lenguaje SQL, es específicamente un lenguaje de programación declarativo, por consiguiente es un lenguaje en el que se dice al computador lo que se desea, pero no como este lo puede llevar a cabo. Además este lenguaje es el estándar para el manejo, obtención de información y manipulación de base de datos relacionales, debido a que posee las acreditaciones de estándares internacionales ISO y ANSI, por lo que se puede afirmar que aunque existen extensiones propietarias del lenguaje de los SGBD, en esencia la estructura del mismo se mantiene básicamente igual para todas sus implementaciones.

Cuando una persona hace uso de un SGBD relacional, el principal proceso llevado a cabo sobre el mismo son las consultas a los datos que este gestiona, siendo estas presentadas específicamente en el lenguaje SQL, pero existen problemas que dificultan el aprendizaje de este lenguaje. Por ejemplo: Se debe tener un conocimiento adecuado de la estructura y nombres de los objetos a consultar, pueden existir conceptos erróneos en la comprensión de los elementos del lenguaje SQL y el modelo de datos relacional, los SGBD no son amigables y es difícil trabajar directamente con los mismos, debido principalmente a la incapacidad de estos al hacer frente a errores semánticos y también a su insuficiencia en la retroalimentación hacia el usuario frente a un error (Mitrovic, 1998) [1].

Por consiguiente, la redacción de sentencias SQL se ha mantenido normalmente alejado del usuario medio de computadoras, el cual ha recurrido a soluciones mas simples pero menos potentes y eficaces que SQL para solucionar sus problemas de gestión, manejo y consulta de datos e información, haciendo uso por ejemplo de hojas de cálculo para lograr su cometido.

Aunque actualmente existen muchas herramientas que permiten la gestión y creación de modelos de base de datos e igualmente facilitan la escritura de sentencias SQL para personas con conocimientos básicos/avanzados de las mismas. No existe en el mundo del software de licenciamiento libre una herramienta que apoye la escritura de sentencias del lenguaje SQL para personas con escasos conocimientos del mismo.

Por consiguiente, esta propuesta tiene como iniciativa la creación de un proyecto en el área de software libre que permita a las personas ajenas al mundo de la informática o con escaso conocimiento del lenguaje SQL, crear y mantener en el tiempo una herramienta capaz de ayudarlos en la consecución de su objetivo de forma eficaz y eficiente, que no es otro que escribir sentencias SQL y ejecutarlas en un SGBD sin muchas dificultad y mediante el uso de un software asistente de interfaz amigable.

1.3 Objetivos del proyecto

A continuación se detalla el objetivo general y los objetivos específicos del proyecto:

1.3.1 Objetivo general

Construir un software que facilite la escritura y ejecución de sentencias SQL de forma visual para la base de datos postgreSQL, en un ambiente de desarrollo de software libre.

1.3.2 Objetivos específicos.

1. Determinar la licencia apropiada para el proyecto a desarrollar.
2. Seleccionar un nombre para la aplicación a realizar.
3. Seleccionar un ambiente de desarrollo de la aplicación, propicio para la creación de una comunidad de software libre alrededor del proyecto.
4. Configurar de forma personalizada el ambiente de desarrollo seleccionado para generar los espacios de intercambios de ideas y comunicación del proyecto.
5. Promocionar el proyecto en búsqueda de la creación de una base de usuarios que sirva para brindarle una comunidad al mismo.
6. Seleccionar las funcionalidades básicas que debería brindar el software a desarrollar.
7. Seleccionar el tipo de metodología a usar para el desarrollo del proyecto.
8. Diseñar la jerarquía de clases necesarias obtener la aplicación propuesta.
9. Liberar la primera versión funcional del proyecto.
10. Generar la documentación de usuario y desarrollador del software.

1.4 Alcance y Limitaciones

El desarrollo del software para la creación y ejecución de consultas SQL, brindará a sus usuarios una plataforma sobre la cual llevar a cabo consultas sobre datos previamente existentes en una base de datos de forma rápida, sencilla y más amigable que en un entorno tradicional de un SGBD. Para lo cual se usaran técnicas de ingeniería inversa que consulten el diccionario de datos del SGBD, para obtener los metadatos de los objetos a los cuales puede tener acceso el usuario de acuerdo al esquema y contraseña usados para acceder a la Base de datos.

Técnicamente el lenguaje SQL debería ser estándar, pero en la realidad, sus implementaciones entre diferentes SGBD son inconsistentes y usualmente incompatibles, principalmente en la partes de manejo de fechas, cadenas y valores nulos [2]. Por esta razón se limitara la herramienta a desarrollar a una implementación de SGBD, específicamente la de postgreSQL, para la cual se adaptara la interfaz gráfica y se optimizara la misma.

De acuerdo con los datos acerca de los datos que se recuperen del diccionario de datos, se procederá a construir un modelo gráfico en el cual el usuario pueda representar las ideas que posea acerca de los datos que necesita visualizar. Pero solamente con esta información no se puede realizar una consulta SQL avanzada, razón por la cual se requerirá en algunas oportunidades la intervención del usuario.

Por lo que la interfaz gráfica de interacción con el usuario, solicitara también la intervención del mismo para la realización de algunas acciones, quedando limitado a las tecnologías de interacción con el usuario y manejo de gráficos existentes en la actualidad. Así mismo, la ejecución de las sentencias llevadas a cabo por el usuario estará limitada a mostrar el resultado generada por la misma y no a la manipulación de los datos recuperados mediante la sentencia ejecutada.

Debido a la complejidad de algunas actividades planteadas en el proyecto, es muy probable que el mismo no sea logrado en su totalidad en el tiempo disponible, pero si se obtendrá una base funcional o primera iteración de desarrollo sobre la cual se pueda crear un comunidad que haga avanzar el mismo y permita su expansión y crecimiento en el tiempo de forma coherente y eficaz.

Se debe también recordar que como toda interfaz gráfica de usuario (GUI, por sus siglas en ingles), la misma representará una visión metafórica de la abstracción de su autor acerca de lo que sería la escritura de una sentencia SQL y por lo tanto esta puede quedar sujeta a la visión personal que el mismo posea sobre este lenguaje y como sería su uso, por lo que su funcionalidad quedará reducida a la forma en que su creador pueda convertir las estructuras internas del software encargadas de almacenar la lógica de la sentencia en representaciones visual que hagan más cómoda la interacción con el usuario (Stephenson, 1999)[3].

1.5 Áreas de Interés

Las áreas de interés en las cuales se pueden enmarcar el presente proyecto son:

1. Interfaces Gráficas.
2. Construcción de sentencias asistido por computador.
3. Generadores de sentencias SQL.
4. Ingeniería Inversa de Base de Datos.

1.6 Marco de Desarrollo

El trabajo que se propone será desarrollado durante el último semestre del año 2007 en la UOC (Universitat Oberta de Catalunya) dentro del marco del máster oficial de software libre y de manera intrínseca deberá contar con las siguientes fases:

1.6.1. **Fase de Planificación:** esta fase consiste en llevar a cabo una planeación de la estrategia a seguir para obtener el producto de software deseado, tomando en cuenta los riesgos, antecedentes y supuestos preexistentes como base para su realización. Para esto se evalúan los recursos existentes para intentar obtener un esquema propicio para el desarrollo del proyecto, que permita obtener una aplicación gráfica para la construcción de sentencias SQL que sea útil para sus usuarios.

1.6.2. **Fase de Desarrollo:** en esta fase se desarrollara el proyecto de software propuesto y se comenzara a ejecutar la planificación elaborada, es decir se empezara a aplicar una metodología de desarrollo de software, en la cual se procederá a implementar la aplicación conceptualizada en la fase de planificación. Esta fase involucra desde el desarrollo conceptual de la aplicación hasta su desarrollo definitivo en un lenguaje de programación, incluyendo las pruebas del mismo.

1.6.3. **Fase de Distribución y Mantenimiento:** Una vez que se considere que se ha obtenido una aplicación que puede considerarse un hito en el proyecto, la misma se distribuirá para que sus posibles usuarios haga uso de la misma y descubran posibles errores en la misma o planteen nuevas funcionalidades que posiblemente se puedan incluir en la misma en un futuro.

Siendo la idea básica del proyecto, crear una aplicación para la generación visual de consultas SQL al sistema gestor de base de datos de postgreSQL, lo cual en la actualidad en ambientes de desarrollo de software libre viene a ser realizado de manera directa con el mismo mediante una aplicación de línea de comandos o mediante un entorno integrado que sirve solo como una interfaz gráfica de acceso a la línea de comandos, pero no soluciona el proceso de brindar una metáfora visual al usuario que le permita crear fácilmente sus consultas.

1.7 Estado del Arte

Desde la creación de la primeras base de datos en los años 60 ha existido siempre la necesidad de consultar los datos contenidos por las mismas, siendo esto logrado en primera instancia mediante el uso de dos modelos de datos conocidos como el modelo en red y el modelo jerárquico, con los cuales se podía acceder a la información que contenía la base de datos mediante operaciones de enlazamiento de punteros a bajo nivel, teniendo esto como principal problema que una persona debía conocer la totalidad de la estructura física de la base de datos para poder realizar consultas a la misma.

No fue sino hasta la época de los 70 cuando Edgar Frank Codd dio a conocer su teoría acerca de cómo gestionar el almacenamiento de datos, al liberar su artículo "A Relational Model of Data for Large Shared Data Banks", que se comenzó a gestar un nuevo modelo de base de datos, el modelo relacional, el cual influenció a la compañía IBM para crear un nuevo proyecto: el sistema de gestión de base de datos relacional System R, el cual posteriormente ocasiono la creación de un lenguaje conocido como **Structured English Query Language** (SEQUEL) para gestionar y manipular los datos almacenados en esta base de datos, actualmente por razones de trademark su nombre cambio a SQL (Structured Query Languague).

1.7.1 Software comercial existente:

Con el paso del tiempo el lenguaje SQL ha ido mejorando progresivamente hasta convertirse en un estándar de facto de los nuevos proyectos de software, lo cual lo llevo a obtener las certificaciones en estándar ANSI en 1986 e ISO en 1987, por lo que posteriormente han surgido múltiples herramientas cuya principal labor es facilitar la elaboración de consultas a base de datos mediante este lenguaje de consultas. Entre estas herramientas tenemos básicamente dos categorías posibles:

- Aquellas que apoyan escritura de sentencias SQL mediante un entorno de desarrollo integrado pero que requieren el conocimiento del lenguaje SQL por parte de su usuario.

- Aquellas que apoyan la escritura de sentencias SQL mediante una interfaz amigable y que no requieren conocimiento previo de SQL para uso.

Es por se procederá a tomar como antecedentes del presente proyecto solamente la segunda categoría (no necesitan conocimiento previo de SQL) de herramientas para la creación de consultas SQL, y entre estas tenemos por ejemplo:

1.7.1.1. **SQL Query Assistant:** Es una herramienta que asiste al usuario en la creación y ejecución de sentencias SQL simples, usando para esto una interfaz gráfica donde se pueden seleccionar los campos que se desean mostrar, el orden que deben poseer, criterios de selección de datos. Esta aplicación forma parte de la suite *SQL4X Manager J*, Su tipo de licenciamiento es privativo y no posee código fuente disponible.
Sitio Web: http://help.macosguru.de/sql4xj/asSQLQuery.html

1.7.1.2. **Esperant 3.0 – Query Assistant:** Es una herramienta creada por la compañía software AG, cuya principal finalidad es la de permitirle al usuario diseñar y ejecutar sentencias SQL sin tener conocimientos básicos del mismo. Esto facilita la labor de diseñar sentencias SQL que respondan a preguntas de análisis diario en un negocio, como por

ejemplo: "¿Qué porcentaje de ... tiene...?", "compare ... contra", etc., que podrían requerir niveles avanzados de conocimiento de SQL si son escritas de forma manual, pero para llevar a cabo se precisa de una cantidad de tiempo moderado, ya que la herramienta guía al usuario a través de una serie de diálogos que tienen como principal idea obtener los datos necesarios para elaborar el tipo de consulta requerida por el usuario. su licenciamiento es privativo y no posee código fuente disponible.

Sitio Web: http://www.dbmsmag.com/9602d09.html

1.7.1.3. **Advanced SQL Writer:** Ayuda en la construcción de sentencias SQL de forma visual sin necesidad de escribir una línea de código. El mismo permite completar de forma fácil un marco de trabajo para la sentencia SQL a escribir, al simplemente arrastrar objetos/componentes al área de trabajo, lo que abre el camino hacia la personalización de la consulta que se desee llevar a cabo y permite a los principiantes obtener sentencias SQL de forma rápida, sin necesidad de depuración y libre de errores de sintaxis y lógicos, permitiendo de esta manera de esta manera observar el contenido de la base de datos de forma fácil y rápida. Su licenciamiento es privativo y no existe código fuente disponible.

Sitios Web: http://www.dlkj.net/en/index.htm
http://www.newfreedownloads.com/Screenshot-Advanced-SQL-Writer.html

1.7.1.4. **Advanced Query Builder:** Es un componente de software creado para las suites Delphi y C++ Builder de la compañía Borland, y su principal objetivo es permitir la construcción visual de sentencias SQL para selección, inserción, actualización y borrado. Logrando en esencia permitir a los usuarios hacer sentencias SQL de gran complejidad y tamaño sin siquiera estos tener conocimiento del lenguaje SQL. Su licenciamiento es privativo y no existe código fuente disponible.

Sitio Web: http://sqlmanager.net/en/products/tools/querybuilder

1.7.1.5. **EMS SQL Query for PostgreSQL, MySQL, Firebird, SQL Server, Oracle, DB2:** Es una herramienta utilitaria creada por la compañía de software EMS y permite crear de forma sencilla y simple consultas SQL hacia diversas base de datos, mediante el uso de una interfaz gráfica amigable al usuario, que le permite al mismo conectarse a las base de datos, seleccionar tablas y campos para una consulta, fijar un criterio de selección de filas, etc. Permitiendo al usuario inclusive trabajar con varias consultas de forma simultánea y ver los resultados de ejecución de las mismas en diferentes tipos de salida y dentro de estas llevar a cabo manipulación de los datos mostrados. Su tipo de licenciamiento es privativo y no posee código fuente disponible.

Sitio Web: http://sqlmanager.net/en/products

1.7.1.6. **Active Query Builder:** es un constructor de sentencias SQL de forma visual, que le permite a los usuarios construir de forma sencilla e intuitiva sentencias SQL complejas por intermedio de una interfaz visual amigable. Este producto lleva le permite a un usuario sin experiencia o con poca, llevar a cabo consultas que hagan uso de todas las características del lenguaje SQL para obtener consultas eficientes que

le permitan obtener de forma eficaz la información deseada, eliminando prácticamente los errores al escribir sentencias SQL y obteniendo resultados de forma instantánea al usar una interfaz similar a la de construcción de sentencias SQL de los programas Access o SQL Server. Su tipo de licenciamiento es privativo y no posee código fuente disponible.

Sitio Web: http://www.activequerybuilder.com/index.html

1.7.1.7. **OpenSqlManager:** Es una herramienta de administración de la base de datos PostgreSQL escrita en C#, cuya idea principal es llevar a cabo la creación, alteración y borrado de objetos en la base de datos, permitiendo a su vez la construcción de sentencias SQL a través de una interfaz gráfica para creación de las consultas deseadas. Su tipo de licenciamiento es GPL, posee el código fuente disponible y su estado de desarrollo es pre-alpha.

Sitio Web: http://sourceforge.net/projects/opensqlmanager/

1.7.1.8. **Open Query:** Es una herramienta creada la para la construcción, ejecución y optimización de sentencias SQL de forma visual, la misma posee una licencia de software libre y aún no posee código fuente disponible debido a que se encuentra en fase de plantación desde el año 2006, por lo cual muy probablemente no se lleve a cabo.

Sitio Web: http://sourceforge.net/projects/openquery/

1.7.2 Investigaciones previas:

En la actualidad existen diversos estudios que sustentan la afirmación sobre la cual se sustenta el presente proyecto, que no es otra que la dificultad que encuentran las personas que están comenzando a usar SQL para manejar un lenguaje altamente estructurado como este. Según A. Mitrovic estas dificultades pueden ser superadas al usar, "un sistema de aprendizaje que sea diseñado como un ambiente guiado de aprendizaje y descubrimiento, el cual ayude al estudiante a superar dificultades" [1].

Esta afirmación trae un nuevo problema, con el que se debe lidiar, ¿Cuál es la interfaz del ambiente adecuada para el aprendizaje del lenguaje SQL?, para esto debe comprenderse que las herramientas del tipos CASE (por sus siglas en ingles, Computer-aided software engineering), son herramientas que ayudan a asistir el desarrollo y mantenimiento de software y cuyo principal objetivo es el aumentar la productividad durante el desarrollo de software reduciendo a su vez el coste del software en términos de tiempo y dinero.

La idea principal es obtener una herramienta del tipo CASE, pero que a su vez sea útil para los usuarios y cuya adopción no se vea afectada por su curva de aprendizaje, ya que como lo indica C.F. Kemerer, "los modelos de curva de aprendizaje pueden cuantitativamente documentar el efecto sobre la productividad de las herramientas case, ya que se pueden usar para estimar resultados futuros con mayor precisión" [4], lo cual hace ver que sin una interfaz adecuada y práctica una herramienta CASE puede ser abandonada rápidamente y no utilizada en el futuro.

Para esto se debe entonces observar que los dos tipos principales de interfaces disponibles para la realización de un proyecto son: Gráficas y de líneas de comando, y por lo tanto se debe comparar el impacto que posee un tipo u otro de interfaz al llevar a cabo la implementación de un software. Por esto se deben comparar ambos tipos de interfaces, para esto T. Fellman, realizo una herramienta de programación denominada VR tool, para la cual procedió a realizar ambos tipos de interfaces, e implemento un estudio piloto, tras lo cual afirmo que: "una interfaz GUI es beneficiosa aún para los programadores talentosos, ya que esta le presenta a los mismos un orden lógico en lo que deben llevar a cabo" [5].

Es por esto que se acepta la interfaz gráfica como el medio ideal para llevar a cabo una herramienta CASE que permita manipular el lenguaje SQL, pero para diseñar esta interfaz, por motivos temporales se debe usar una técnica que permita su obtención de una forma eficaz y eficiente, que recurra en curvas de aprendizaje que permita afectar la productividad de un usuario de forma positiva y para esto Y. Tao, afirma que se debe hacer un estudio de usabilidad que no incremente el costo efectivo de obtener el software, para eso propone una técnica para implementar la usabilidad en la aplicación que no involucra la características físicas, espaciales o visuales, "enfocándose en el flujo de interacción entre el usuario y el sistema" [6].

Lo cual es ideal para el modelo de desarrollo del software libre, ya que el mismo es aplicable a lo planteado por E. Raymon en su obra la catedral y el bazar [7], que tiene como idea principal, el funcionamiento de este tipo de desarrollo basado en los bazares orientales, con intercambios de ideas espontáneos y no dirigidos por nadie. Compensando la falta de formalidad en el proceso de desarrollo con una gran retroalimentación por parte de los usuarios.

1.8 Metodología General

Para el desarrollo del presente proyecto la metodología seguida se encuentra dividida en tres partes: identificación y descripción de las fuentes de información útil para el proyecto, el posterior análisis de la información recolectada y finalmente la aplicación de lo obtenido en los dos puntos anteriores para la elaboración inicial de la herramienta deseada y la posterior creación de una comunidad en torno a la misma. A continuación se lleva a cabo una descripción más detallada de cada fase:

1.8.1. Identificación de los datos:

Identificar de forma apropiada los orígenes de los datos necesarios para la realización del proyecto fue de vital importancia, para elaborar una herramienta del tipo propuesto, especialmente al no poseer en el mundo del software libre una herramienta que pudiese servir como punto de partida inicial de la misma.

Es por esto que fue necesario identificar las fuentes de los datos necesarios, así como también la localización de lugares útiles para el intercambio de ideas del proyecto con personas con conocimiento del área como lo son las listas de correo y los foros temáticos de usuarios.

Tras lo cual se procedió a su estudio y descubrimiento de su utilidad, para limpiar la información no necesaria y posteriormente almacenar aquello que tuviese una significativa importancia para el proyecto.

1.8.2. Análisis de la información recolectada:

Posterior al almacenamiento de la información recopilada, la misma debió ser procesada para determinar cuál era útil para el tipo de proyecto a realizar, llevándose esto a cabo de forma experimental, al modificar el objeto de estudio (la aplicación desarrollada) y observar su reacción ante los cambios realizados basado en la información recopilada y evaluar de esta forma su resultado y determinar si lo realizado afecto al proyecto de forma positiva o negativa, teniendo siempre en cuenta la opinión de terceras personas, ya que esto le crearía a la aplicación una base de posibles usuarios que posteriormente podrían formar parte o no de la comunidad de desarrollo de la misma.

1.8.3. Elaboración de la herramienta:

Una vez recopilada y analizada la información de las fuentes necesarias para la creación del proyecto, se procedió a observar cómo se debía desarrollar el recurso humano alrededor de la misma y como se motivaría a su involucración dentro del proyecto.

Para esto se debió decidir de forma correcta el público objetivo del proyecto, las herramientas a utilizar para su desarrollo, como se crearía la comunidad a su alrededor, como sería su interacción con los usuarios y de qué forma se llevaría a cabo la comunicación entre desarrolladores.

Y se tomaron como puntos cruciales para el éxito del proyecto su división en tres fases como lo describe A. Senyard [8], siendo la primera de estas una fase cerrada desarrollado por un grupo muy pequeño de personas o desarrolladores, muy similar al desarrollo de software tradicional. Posteriormente, se debe mover ese desarrollo cerrado a un ambiente colaborativo y se debe llevar a cabo una entonces una fase de transición de un modelo de desarrollo de software tradicional a un modelo comunitario. Y por último el proyecto se convierte en una comunidad y obtiene las ventajas asociadas a este hecho.

1.9 Planificación Temporal del proyecto

Luego de definir el objetivo general y los objetivos específicos de este proyecto y su metodología, se procedió a definir una lista tentativa de tareas que permitirán realizar la planificación del proyecto propuesto, en base a una fecha de entrega fijada, que es el 15 de Enero del 2008, esta lista de tareas se puede apreciar a continuación, junto con la duración y las posibles fechas de inicio y finalización de cada tarea:

WBS	Nombre del Tarea	Inicio	Fin	Duración
1	**Determinar la licencia apropiada para el proyecto a desarrollar.**	Nov 1	Nov 5	3d
1.1	Crear una lista con las principales licencias de software libre candidatas a ser utilizadas en el proyecto.	Nov 1	Nov 1	4h
1.2	Estudiar y comprender las licencias de software libre seleccionadas.	Nov 1	Nov 2	1d
1.3	Determinar un listado de beneficios y desventajas que tendría aplicar cada licencia al proyecto.	Nov 2	Nov 2	4h
1.4	Discutir en base a los resultados obtenidos cual es la licencia que mejor se adapta al proyecto.	Nov 5	Nov 5	1d
2	**Seleccionar un nombre para la aplicación a realizar.**	Nov 6	Nov 6	7h
2.1	Crear una lista de posibles palabras claves que identifiquen la funcionalidad del proyecto.	Nov 6	Nov 6	1h
2.2	Llevar a cabo un estudio estadístico, para escoger palabras evocan de forma más intuitiva la funcionalidad del programa.	Nov 6	Nov 6	5h
2.3	Con las palabras seleccionadas confeccionar el nombre del proyecto.	Nov 6	Nov 6	1h
3	**Seleccionar un ambiente de desarrollo de la aplicación, propicio para la creación de una comunidad de s.l. entorno al proyecto.**	Nov 5	Nov 6	1d 6h
3.1	Ubicar los principales sitios de desarrollo de software para proyectos de software libre.	Nov 5	Nov 5	2h
3.2	Evaluar superficialmente las ventajas y desventajas de cada uno de ellos.	Nov 5	Nov 5	4h
3.3	Seleccionar basándose en los beneficios prestados, el sitio de desarrollo para el proyecto y dar de alta el mismo.	Nov 5	Nov 6	1d
4	**Configurar de forma personalizada el ambiente de desarrollo seleccionado para generar los espacios de intercambios de ideas y comunicación del proyecto.**	Nov 6	Nov 12	6d
4.1	Decidir cómo debe ser la interfaz del proyecto para los desarrolladores y personalizar la misma.	Nov 6	Nov 7	1d
4.2	Configurar el control de versiones del proyecto y su estructura.	Nov 7	Nov 8	1d
4.3	Definir cómo será la interfaz de la página web del proyecto mediante la creación de una plantilla.	Nov 8	Nov 9	1d
4.4	Crear la página web del proyecto.	Nov 8	Nov 9	1d
4.5	Crear un listado de los principales medios de comunicación usados en diversos proyectos de software libre.	Nov 9	Nov 12	3h
4.6	Determinar de forma empírica cuales serían las características deseables de un medio para la comunicación en el proyecto.	Nov 12	Nov 12	2h
4.7	Determinar cuáles medios de comunicación son idóneos para el proyecto de acuerdo a sus características.	Nov 12	Nov 12	3h
5	**Promocionar el proyecto en búsqueda de la creación de una base de usuarios que sirva para brindarle una comunidad al mismo.**	Nov 9	Jan 7	41d
5.1	Definir el público hacia el cual se dirigirá la estrategia de publicidad del proyecto en sus diversas etapas.	Nov 9	Nov 12	4h
5.2	Redactar un listado de posibles formas de comunicarse con el público seleccionado como target del proyecto en una etapa especifica.	Nov 12	Nov 12	4h
5.3	Seleccionar y utilizar las formas de comunicación seleccionadas.	Nov 12	Jan 7	40d
6	**Seleccionar cuales funcionalidades básicas debería brindar el software a desarrollar.**	Nov 12	Nov 13	1d 2h
6.1	Crear un listado de posibles funcionalidades del proyecto.	Nov 12	Nov 13	4h
6.2	Evaluar la importancia que tienen estas funcionalidades para el posible usuario del software.	Nov 13	Nov 13	4h
6.3	Seleccionar por orden prioritario cuales funcionalidades básicas debería brindar el proyecto.	Nov 13	Nov 13	2h
7	**Seleccionar el tipo de metodología a usar para el desarrollo del proyecto.**	Nov 14	Nov 15	2d 4h
7.1	Listar metodologías importantes usadas en ambientes de software libre.	Nov 14	Nov 14	4h
7.2	Crear una lista de características deseables de una metodología para el proyecto.	Nov 14	Nov 14	4h
7.3	Escoger en base a las características deseables cual metodología se adapta mejor a la idea de desarrollo del proyecto.	Nov 15	Nov 15	2h
8	**Especificar unas reglas mínimas que gobiernen la colaboración de la comunidad en el proyecto.**	Nov 15	Nov 19	2d 4h
8.1	Crear un listado de posibles reglas que determinen lo que se espera sea el comportamiento de cualquier colaborador en la comunidad.	Nov 15	Nov 19	2d
8.2	Decidir de mutuo acuerdo cuales reglas deberían gobernar a la comunidad.	Nov 19	Nov 19	4h
9	**Liberar la primera versión funcional del proyecto.**	Nov 19	Ene 11	103d
9.1	Comenzar a aplicar la metodología seleccionada.	Nov 19	Nov 19	

9.2	Diseñar el software deseado.	Nov 19	Nov 26	5d
9.3	Codificar.	Nov 26	Ene 7	30d
9.4	Llevar a cabo pruebas de control de calidad a medida que el proyecto avance.	Nov 27	Ene 11	33d
9.5	Habilitar un sistema de reporte de errores.	Nov 26	Nov 27	1d
9.6	Aplicar el control de versiones al desarrollo del proyecto.	Nov 26	Ene 10	33d
9.7	Seleccionar la forma de distribución del programa.	Ene 10	Ene 11	4h
9.8	Distribuir el programa en la forma escogida.	Ene 11	Ene 11	4h
10	**Generar la documentación de usuario y desarrollador del software.**	Nov 19	Ene 9	32d
10.1	Documentar cada uno de los aspectos encontrados durante el desarrollo del programa.	Nov 19	Dec 31	30d
10.2	Crear la documentación para el usuario final que le permita usar el software desarrollado.	Ene 7	Ene 8	1d
10.3	Distribuir la documentación en conjunto con el software realizado.	Ene 8	Ene 9	1d

Tabla 1 - Planificación detallada del proyecto

Igualmente se puede observar dicha planificación de forma resumida en la siguiente tabla:

WBS	Nombre del objetivo especifico	Inicio	Fin	Duración
1	Determinar la licencia apropiada para el proyecto a desarrollar.	nov 1	nov 5	3d
2	Seleccionar un nombre para la aplicación a realizar.	nov 6	nov 6	7h
3	Seleccionar un ambiente de desarrollo de la aplicación, propicio para la creación de una comunidad de software libre entorno al proyecto.	nov 5	nov 6	1d 6h
4	Configurar de forma personalizada el ambiente de desarrollo seleccionado para generar los espacios de intercambios de ideas y comunicación del proyecto.	nov 6	nov 12	6d
5	Promocionar el proyecto en búsqueda de la creación de una base de usuarios que sirva para brindarle una comunidad al mismo.	nov 9	ene 7	41d
6	Seleccionar cuales funcionalidades básicas debería brindar el software a desarrollar.	nov 12	nov 13	1d 2h
7	Seleccionar el tipo de metodología a usar para el desarrollo del proyecto.	nov 14	nov 15	1d 2h
8	Especificar unas reglas mínimas que gobiernen la colaboración de la comunidad en el proyecto.	nov 15	nov 19	2d 4h
9	Liberar la primera versión funcional del proyecto.	nov 19	ene 11	39d
10	Generar la documentación de usuario y desarrollador del software.	nov 19	ene 9	37d

Tabla 2 - Planificación resumida del proyecto

Es importante recalcar que esta planificación no es la única posible y que la misma cambio a medida que el proyecto avanzaba, y por lo tanto solo fue útil en este caso para estimar una posible división del tiempo en cada uno de los objetivos específicos planteados.

Por lo tanto las fechas que allí aparecen no se deben considerar absolutas sino relativas, ya que dependiendo de cómo avancen las actividades que preceden a una actividad, esta podrá iniciarse antes o después de su fecha planificada. También se debe considerar que el tiempo será compartido con otras actividades de carácter laboral y familiar durante el desarrollo del proyecto, lo cual puede llevar a que en el calendario propuesto se deban suplantar algunos días laborables por sábados y domingos de los cuales no se ha dispuesto para esta planificación.

Se debe tomar en consideración, que las tareas acá mencionadas para cumplir los objetivos específicos no son definitivas, sino por el contrario son un simple bosquejo de lo que se necesitaría para realizar el objetivo encomendado, y por lo tanto pueden variar durante la realización del proyecto.

La planificación se puede apreciar mejor, en el siguiente gráfico de Gantt, donde

además se puede observar fácilmente la división del tiempo de las tareas dentro de los objetivos específicos, y cuáles de estas tareas serán llevadas a cabo de forma simultánea y cuales tareas se deben cumplir antes de poder comenzar a llevar a cabo otra:

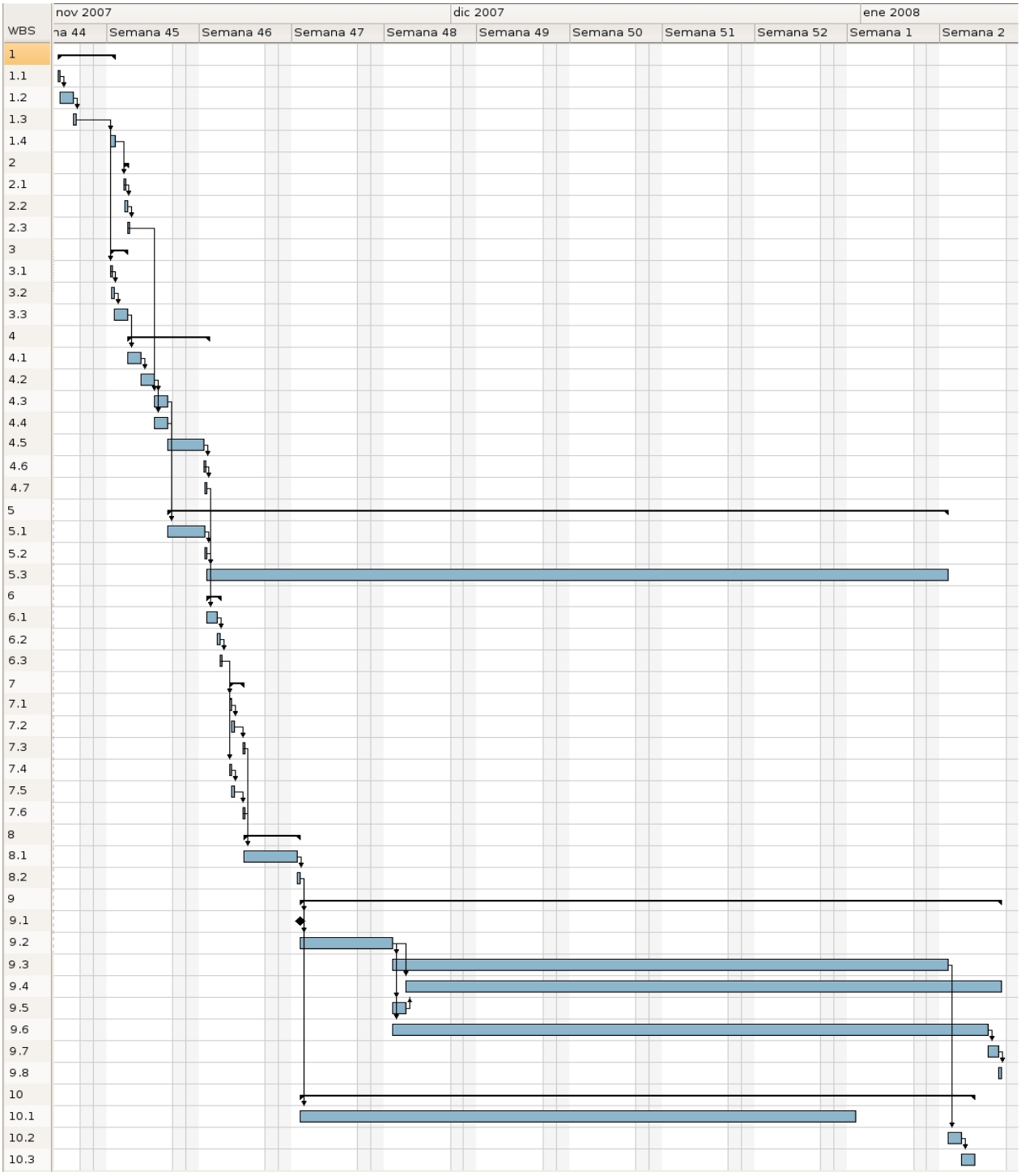


Ilustración 1 – diagrama de Gannt del proyecto

1.10 Especificaciones iníciales del software:

Determinar las especificaciones iníciales de un proyecto de software libre es una labor que no debe ser dejada al azar y para hacerlo de forma correcta se debe estudiar algunos casos de proyectos que han sido exitosos y observar con detalle que fue lo que llevaron a cabo que motivo su éxito, es por esto que según A. Senyard [8], si se procede a tomar en cuenta el modelo de bazar propuesto por Raymond [7], existen una serie de actividades que deben ser cubiertas por un prototipo o liberación inicial de un software de licenciamiento libre, para que pueda ser obtenida de forma fluida su transición de un proyecto desarrollado de forma tradicional a uno desarrollado de forma comunitaria, como se explico en la metodología general del presente proyecto.

Por lo que entonces el autor A. Senyard proporciona una lista de características que debe tener un proyecto para lograr esta transición y su importancia:

Importancia		Característica
Crucial	1	Un prototipo razonable del software tiene que haber sido creado.
	2	El diseño del prototipo debe ser modular.
	3	El código fuente del prototipo debe estar disponible y viable (Es decir, compila y ejecuta).
	4	Una comunidad de usuarios y desarrolladores deben ser atraídos a El proyecto.
	5	El autor del proyecto debe estar motivado para gestionar el proyecto o encontrar un reemplazo.
	6	Proyecto de comunicación y los mecanismos de contribución debe estar en su lugar.
Importante	7	El alcance del proyecto debe estar bien definido.
	8	Una norma de codificación de estilo o debe ser establecido.
	9	Desarrollo de las versiones de software debe tener periodos de liberación cortos, mientras que los ciclos de versiones de usuario deben ser estables y constantes.
	10	Una licencia que debe ser elegido es atractivo para los desarrolladores.
	11	Un adecuado estilo de gestión debe ser seleccionado.
Deseable	12	Una cantidad adecuada de la documentación de los proyectos debe existir.

Tabla 3 - Características que debe poseer un proyecto para pasar de desarrollo cerrado a comunitario

Por esta razón se ha decidido cumplir en el lanzamiento inicial de la aplicación Visual SQL Builder, todas las características cuyo nivel de importancia es cruciales (1-6, Tabla 3), así como algunas de las importantes (7-11, Tabla 3), y la deseable aunque se ha llevado a cabo parcialmente se le ha asignado una prioridad mucho menor a las anteriores.

Es decir, se creara una aplicación que funcione y compile, que permita generar sentencias SQL de forma visual con un nivel de complejidad "aceptable" por su usuario, cuyo código fuente sea entendible por los usuarios que deseen acceder al mismo, y además les sirva de motivación para su participación en el proyecto.

1.11 Estructura de la memoria

La estructura de esta tesis es como se presenta a continuación: el capitulo 1, es el relacionado con los objetivos que debe cumplir el proyecto propuesto, las limitaciones que conseguirá en su camino, así como una planificación y marco bajo el cual fue desarrollado. El capitulo 2 presenta los fundamentos teóricos del proyecto, en el cual se incluyen todos los aspectos teóricos básicos que se deben poseer para comprender su fin y funcionamiento. Al final del capítulo se presenta una pequeña introducción a las herramientas usadas para la elaboración del proyecto.

El capitulo 3, presenta una visión muy general de cómo fueron llevadas a cabo las fases de desarrollo del proyecto, prestando especial atención al hecho de que el proyecto inicio como un proyecto cerrado, que luego debe abrirse, de una forma correcta y acorde con la comunidad que es objetivo del mismo, para de esta manera poder garantizar su éxito futuro, existiendo entonces algo que se conoce como la transición de la catedral al bazar. Igualmente se explica el proceso de selección de la licencia del proyecto, como se distribuye el mismo, y la forma de gestionarlo.

Posteriormente durante el mismo capítulo, es presentada una muy breve explicación del diseño resultante del proyecto, el cual fue implementado, y puesto a prueba e interacción con la comunidad de usuarios. Explicando de qué forma se espera que este proceso avance en el futuro y como se ha llevado a cabo hasta el momento.

El capitulo 4, constituye un análisis a los resultados obtenidos con el presente proyecto, el cual se inicia con una pequeña revisión de los objetivos propuestos por el proyecto, junto con un autoevaluación de su cumplimiento, y algunos comentarios útiles para comprender las afirmaciones allí realizadas. A continuación, se muestra una lista de las posibles ampliaciones de las que puede ser objeto la aplicación, y que le permitirán a la misma crear un base de usuarios a su alrededor, que interactúen de forma conjunta en su desarrollo, creando una comunidad. Por último se emiten unas conclusiones con respecto al proyecto realizado y las enseñanzas que este dejo.

Capitulo 2

2.1 Fundamentos Teóricos

Para la realización de este proyecto la principal tecnología usada será la programación orientada a objetos, que permitirán generar las sentencias **SQL**, que luego el usuario podrá ejecutar de una forma intuitiva, por lo que a continuación se presentaran algunos breves fundamentos teóricos importantes dentro de este proyecto:

2.1.1 Programación Orientada a Objetos

La Programación Orientada a Objetos (OOP, por sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo, y encapsulamiento. Su uso se popularizó a principios de la década de 1990 [9].

Su eje principal son los objetos, que son entidades propias que combinan estado, comportamiento y entidad, de la siguiente forma:

- El estado está compuesto de datos, serán uno o varios atributos a los que se habrán asignado unos valores concretos.

- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, que operaciones se pueden realizar con él.

- La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identidad.

Es entonces en donde este paradigma de programación busca expresar un programa, como un conjunto de estos objetos, que colaboran entre ellos para llevar a cabo las labores las acciones que necesite el programa. Lo cual permite hacer los programas más modulares y hacerlos más fáciles de escribir, depurar y reutilizar, razón por la cual es un paradigma de programación muy usado en proyectos de software libre, ya que alientan la división del trabajo de forma colaborativa.

Otras partes esenciales de los objetos sus métodos (comportamiento) y atributos (estado), que están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. Razón por la cual, el programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a ninguno de ellos.

A continuación se definirán muy brevemente otros conceptos fundamentales de este paradigma de programación:

- **Objeto:** entidad provista de un conjunto de propiedades o atributos y de comportamiento o funcionalidad, se puede definir como la creación de una instancia a una clase.

- **Clase:** definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas, sirviendo como un molde o patrón creador de objetos.

- **Método:** algoritmo asociado a un objeto(s), cuya ejecución se desencadena tras la recepción de un mensaje. Desde el punto de vista del comportamiento, es lo que el objeto puede llevar a cabo. Un método puede producir un cambio en las propiedades del objeto, o la generación de un evento con un nuevo mensaje para otro objeto del sistema.

- **Evento:** un suceso en el sistema, el cual se maneja enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.

- **Propiedad o atributo:** contenedor de un tipo de datos asociados a un objeto, que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

2.1.1.1 Propiedades de la Programación Orientada a Objetos:

Aunque no existe una definición absoluta de las características que debe tener un lenguaje de programación orientado a objetos, a continuación se muestran las más comúnmente asociadas a estos lenguajes:

- **Abstracción:** Cada objeto en el sistema sirve como modelo de un ente abstracto que puede realizar trabajo, informar y cambiar su estado, y comunicarse con otros objetos en el sistema sin revelar cómo se implementan estas características.

- **Encapsulamiento:** Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.

- **Principio de ocultación:** Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas.

- **Polimorfismo:** comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.

- **Herencia:** las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que re-implementar su comportamiento.

2.1.2 Structured Query Language (SQL)

El lenguaje estructurado de Consultas (SQL) consiste en frases escritas en un léxico y sintaxis específicos, que permitirán gestionar y consultar bases de datos existentes, siendo este lenguaje de tipo estructurado y declarativo de alto nivel o no procedimental.

Es decir, SQL expresa el resultado deseado pero no como lograrlo, permitiendo de esta manera estandarizar el lenguaje de consulta y gestión para los sistemas gestores de base de datos relacionales, ya que debido a su base teórica y su orientación al manejo de conjuntos de registros y no de registro individuales, permite obtener resultados de forma eficiente, logrando de esta forma que una sola sentencia del mismo sea equivalente a uno o más programas que utilizaran un lenguaje de bajo nivel orientado a registros.

Pero esto no significa que las personas que hagan uso de este lenguaje no tengan que tener conocimientos básicos mínimos y conocer el léxico y sintaxis de este lenguaje para obtener sentencias semánticamente correctas, por esto se debe comprender que el lenguaje SQL se encuentra subdividido en diversos sublenguajes:

2.1.2.1 Sublenguaje de consultas:

Es la operación más común en base de datos relacionales, la cual es llevada a cabo a través de una clausula denominada SELECT, que selecciona datos de una tabla o múltiples tablas en una base de datos, pudiendo aplicar a estos datos diversas operaciones como: restringir los mismos a un criterio especifico, agruparlos de acuerdo a un criterio seleccionado, ordenarlos de alguna forma especifica. El siguiente ejemplo de sentencia SELECT retorna una lista de los productos de un almacén con superior a 200 y los ordena de forma ascendente por el valor del precio:

```
SELECT * FROM productos WHERE precio > 200 ORDER BY precio ASC;
```

Tabla 4: Ejemplo sentencia de consulta SQL

2.1.2.2 Sublenguaje de manipulación de datos:

Son las sentencias de SQL permiten manipular datos, reciben también el nombre de sentencia DML por sus siglas en ingles (Data Manipulation Language) y son básicamente las instrucciones que permiten añadir, actualizar y borrar datos, en el siguiente ejemplo se añade una fila a la tabla mitabla, luego se modifica el valor y posteriormente se elimina:

```
INSERT INTO productos (nombre,precio) VALUES ('producto1',350);
UPDATE productos SET precio=365 WHERE nombre='producto1';
DELETE FROM productos WHERE nombre='producto1';
```

Tabla 5: Ejemplo de manipulación de datos mediante el lenguaje SQL

2.1.2.3 Sublenguaje de control de transacciones:

Las transacciones son eventos atómicos que no pueden ser divididos, y si están disponibles pueden usarse para envolver las operaciones DML emitidas, existiendo básicamente tres órdenes: Iniciar la transacción la cual se representa normalmente con la palabra START TRANSACTION, validar los cambios realizados con las sentencias DML, lo que se representa normalmente con la palabra COMMIT y por último rechazar los cambios realizados mediante sentencias DML durante una transacción y devolver los datos a su estado original, representándose esto con el comando ROLLBACK, aunque estos nombres de comandos son los más comunes, los mismos pueden variar de acuerdo con la implementación de la base de datos. Por ejemplo, se actualizara el precio de un producto en una transacción y luego se hará este cambio permanente:

```
START TRANSACTION;
UPDATE productos SET precio=365 WHERE nombre='producto1';
COMMIT;
```

Tabla 6: Ejemplo de control de transacciones mediante el lenguaje SQL

2.1.2.4 Sublenguaje de definición de datos:

Es el encargado definir las estructuras de almacenamiento de datos que tendrá una base de datos, en otras palabras se puede decir, que permite la creación de las tablas y sus estructuras relacionadas, recibe también el nombre de DDL por sus siglas en ingles (Data Definition Language), y aunque este lenguaje tiene muchas extensiones propietarias de cada base de datos, las más comunes y básicas son: crear un objeto (CREATE), borrar un objeto (DROP), alterar un objeto (ALTER). Por ejemplo, se muestra la creación de la tabla productos:

```
CREATE TABLE productos(
        nombre varchar2(100),
        precio number(6,2),
        constraint pdo_pk PRIMARY KEY(nombre)
)
```

Tabla 7: Ejemplo del lenguaje de definición de datos de SQL

2.1.2.5 Sublenguaje de control de datos:

El último grupo de sentencias SQL conocidas como DCL por sus siglas en ingles (Data Control Language) son las que manejan los aspectos privilegios y control de los usuarios, determinando quien tiene acceso a que objetos de la base de datos y si los puede manipular o no, principalmente se compone de dos sentencias: otorgar el privilegio (GRANT) y revocarlo (REVOKE). En el ejemplo se observa cómo se autoriza al usuario pruebas para poder consultar y alterar los valores de la tabla productos:

```
GRANT SELECT, UPDATE ON productos TO pruebas;
```

Tabla 8: Ejemplo del lenguaje de control de datos de SQL

2.1.3 Interfaces Gráficas de Usuario

Aunque la programación orientada a objetos es la principal tecnología usada por el proyecto y SQL el principal producto del mismo, también se procederá a usar lenguajes de programación y metodologías para desarrollo de software. Todo esto con el fin de obtener una interfaz de usuario de forma gráfica que sea amigable para el usuario a la hora de redactar sentencias SQL.

Por esto se debe comprender que las interfaces gráficas en el proceso de interacción persona-computador, son un artefacto interactivo que posibilita a través del uso y la representación de un lenguaje visual, una interacción amigable con un sistema informático.

Estas también reciben el nombre de **GUI** por sus siglas en ingles (Graphical User Interface), y su funcionamiento consiste en la utilización de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz, pudiéndose realizar estas acciones mediante la manipulación directa con un dispositivo de entrada del computador como por ejemplo el ratón, para facilitar de esta forma la interacción del usuario con la computadora.

Se puede decir a grandes rasgos que las interfaces gráficas surgen como la evolución de la línea de comandos de los primeros sistemas operativos, y son usadas hoy en día de forma habitual por ejemplo para representar el escritorio del sistema operativo del cual se está haciendo uso

Por último se debe tener presente que el concepto de una interfaz gráfica amigable tiene que ver con el hecho de que las misma posea un alto grado de **usabilidad**, o de facilidad en su uso, lo cual se ve representado en el grado en que el diseño de un objeto facilita o dificulta su manejo. Si bien este criterio es de reciente aplicación se infieren a partir de su conceptualización del término llevada a cabo por la ISO: *"Usabilidad es la eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico"* , que esta posee los siguientes principios básicos:

- **Facilidad de aprendizaje:** que es básicamente la existencia de un grado de dificultad muy bajo para que los nuevos usuarios desarrollen interacciones eficientes y eficaces con el sistema desarrollado. Se puede relacionar la predicibilidad con la que se pueden llevar a cabo las cosas sin haberlas llevado a cabo antes, mediante una generalización de los conocimientos previos y consistencia en el desarrollo de los eventos.

- **Flexibilidad:** que consiste en todas las posibilidades que posee un usuario con el sistema para intercambiar información que le permita al mismo especificar lo que desea, en otras palabras se puede decir que consiste en la existencia de múltiples formas de llevar a cabo la misma tarea, pero que sea de forma similar a tareas llevadas a cabo de forma anterior.

- **Robustez:** consiste en el nivel con que el sistema apoya al usuario para llevar a cabo el cumplimiento de sus objetivos, es decir la capacidad con la que el sistema se adaptara a las tareas del usuario de forma eficaz y eficiente.

Es por esto que para lograr un buen sistema amigable al usuario se debe hacer uso de interfaces gráficas con un diseño centrado en el usuario pero no dirigido por el mismo, que proporcione un punto de vista independiente de las metas de la programación porque el papel que se de llevar a cabo a la hora de diseñar el sistema es el de identificar las necesidades funcionales del mismo y diseñar la interfaz apropiada que las soporte.

2.1.4 Software Libre o de Código abierto

El aspecto fundamental que diferencia a este proyecto de otros proyectos que poseen el mismo objetivo, es su tipo de licenciamiento, por eso debe intentar comprenderse muy brevemente el significado de este concepto.

Para comenzar se definirá una licencia de software como la autorización o permiso concedido por el titular del derecho de autor, en cualquier forma contractual, al usuario de un programa informático, para utilizar éste en una forma determinada y de conformidad con unas condiciones convenidas [10]. Por esta razón es muy importante conocer cómo y porque el licenciamiento del mismo afecta su forma de desarrollo, y como hará que una aplicación se clasifique como de software libre o no.

En principio todo software considerado libre, es el que cumple usualmente los siguientes requisitos o libertades con respecto a sus partes:

- **Libertad 0:** la libertad de ejecutar y usar el software para cualquier propósito.
- **Libertad 1:** la libertad de estudiar el programa y adaptarlo a sus necesidades.
- **Libertad 2:** la libertad de distribuir copias.
- **Libertad 3:** la libertad de modificar el programa y liberar las modificaciones al público.

Por lo que para el ejercicio de estas libertades el usuario del software debe poseer acceso a su código fuente, por lo tanto será absolutamente imprescindible que cualquier licencia de software libre contenga de manera efectiva el compromiso de proporcionar de forma el código fuente al usuario o al menos ponerlo a su disposición de alguna forma.

Aunque la libertad del software se puede ver de muchas maneras, como software libre o software de código abierto, en principio en este trabajo serán definidos dichos términos de forma conjunta, ya que pese a sus diferencias en cuanto a sus objetivos, en el fondo su significado posee el mismo fin común (más no práctico). Las consecuencias de esta forma de licenciar el software se puede cuantificar de diversas maneras, pero las principales que suelen ser el coste de desarrollo y una mayor difusión del proyecto realizado.

Por lo que, se podría terminar por definir al software libre como un software distribuido bajo un determinado tipo de licencia de uso que permite el acceso al código objeto y el código fuente del mismo, y por lo tanto su modificación y adaptación a las necesidades del usuario, y también la libre distribución y utilización de la aplicación. No debe asimilarse con software gratuito, la licencia con la que se distribuye y el uso de los derechos de autor que sus creadores realizan son su característica y diferencia principal, no su coste.

2.1.4.1 Principales Actores en el software libre:

Una vez obtenida una visión muy global del software libre, se debe intentar delimitar los actores que forman parte del mismo [11], teniendo entre ellos a:

- **Desarrolladores y distribuidores:** Son los encargados de crear y poner a disposición del usuario el software desarrollado bajo el esquema de licenciamiento libre, el cual normalmente se genera de forma cooperativa por programadores voluntarios, que trabajan de forma coordinada por un medio de comunicación global como lo es el internet.
- **Usuarios del software libre:** Es la persona(s) que lleva a cabo uso de un proyecto de software libre para cualquier actividad cotidiana, aunque en un principio esta era la propia comunidad de desarrollo y distribuidores, últimamente estos se han vuelto más diversos e incluyen a administraciones públicas, centros de enseñanza, etc.

2.1.5 Sistemas de Control de Versiones

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo [12]. Estos sistemas facilitan la administración de las distintas versiones de cada producto de software desarrollado, así como las posibles especializaciones realizadas sobre los mismos, como por ejemplo, alguna adaptación a un sistema operativo en específico.

Por lo tanto estos sistemas permiten obtener el estado presente y pasado de cualquier archivo en un proyecto de software, lo que permite en cualquier momento observar cómo se hacía algo en el pasado y compararlo con la forma con la que se está haciendo actualmente, pudiendo tener entonces un seguimiento de la evolución del proyecto.

Sin importar que sistemas de control de versiones del cual se haga uso, normalmente todos poseen las siguientes características:

- Mecanismo de almacenaje de los elementos que deba gestionar, por ejemplo archivos de texto, imágenes, documentación.
- Posibilidad de realizar cambios sobre los elementos almacenados, por ejemplo modificaciones parciales, añadir, borrar, renombrar o mover elementos.
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos, normalmente pudiendo volver o extraer un estado anterior del código fuente.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión. Actualmente existen dos principales sistemas de control de versiones usados en los proyectos de software libre y estos son: CVS y SVN.

2.1.5.1 Concurrent Versions System (CVS)

El Concurrent Versions System (CVS), también conocido como Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren, este sistema goza de gran popularidad en el mundo del software libre en parte porque su desarrollo se encuentra amparado bajo la licencia GPL [13].

Este sistema de control de versiones, hace uso de una arquitectura cliente servidor para guardar las versiones actuales del proyecto y su historial, conectándose cada cliente al servidor para obtener una copia del proyecto o actualizar la copia global con cambios realizados en su copia local del proyecto. Varios clientes pueden sacar copias de un proyecto de forma simultánea, y posteriormente cuando actualizan el proyecto con las modificaciones que realizaron en su copia local, el servidor deberá encargarse de acoplar esos cambios entre diferentes versiones, pero si no es posible hacerlo de forma automática el servidor emite una petición de resolución de un conflicto y el usuario deberá resolverlo de forma manual.

Otras características importantes del CVS es la posibilidad de obtener una copia histórica del proyecto o como se encontraba el mismo en una fecha o revisión determinada. Igualmente en el mundo del software libre es normalmente permitido el acceso anónimo de lectura al repositorio, teniendo esto como significado que los usuarios del software libre pueden tener acceso siempre a la versión más actual del software si tienen el conocimiento necesario para compilarlo.

Aunque CVS, en el pasado era el líder indiscutible del control de versiones en los proyectos de software libre, existen una serie de limitantes que han hecho buscar otras opciones a los proyectos que hacían uso del mismo, entre estas limitantes destacan: la imposibilidad de renombrar archivos, soporte limitado de archivos Unicode, manejo inadecuado de archivos binarios, etc.

2.1.5.2 Subversion (SVN)

Subversion es un software de control de versiones que ha sido diseñado para reemplazar al antiguo CVS, ya que arregla algunas de sus deficiencias. Este software es igualmente libre, pero esta licenciado bajo un tipo de licencia Apache/BSD, y se le conoce también por svn debido al nombre de su herramienta de comandos. El mismo posee como principal característica que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo [14].

Este sistema de control de versiones posee algunas ventajas con respecto a CVS:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.).

Pero igualmente sigue presentando algunas carencias como: Un manejo de los cambios de nombre de los archivos similar a CVS pero de forma automática, se deben aplicar parches repetidamente entre ramas, y no facilita la identificación de cambios trasladados.

2.1.6 Listas de Correos, Foros y wikis

Las listas de correo y los foros son el elemento principal de diseminación de la información para los proyectos de software libre, casi sin ninguna excepción cualquier proyecto de software libre posee al menos un foro o lista de correo en la cual intercambian ideas sus ideas y organizan su estructura interna.

Dependiendo de la complejidad del proyecto, y la decisión del equipo de desarrollo se pueden organizar varias listas de correos o formas (o ambos), aunque lo normal es que se habrán es que existan 3 listas de correos al menos (o foros), uno para los anuncios importantes, una para los desarrolladores y otra para dar soporte a los usuarios finales.

Por último es destacable indicar que durante los últimos años han surgido nuevos medios de comunicación y formación de relaciones entre los desarrolladores de proyectos de software libre, entre los que destacan los wikis, pero para entender cada uno de estos medios, serán explicados de forma más detallada a continuación:

2.1.6.1 Listas de correos

Las listas de correo electrónico son un uso especial del correo electrónico que permite la distribución masiva de información entre múltiples usuarios de Internet de forma simultánea. En una lista de correo se escribe un correo a la dirección de la lista, por ejemplo vsqlbuilder-news@lists.sourceforge.net y le llega masivamente a todas las personas inscritas en la lista, dependiendo de cómo esté configurada la lista de correo el usuario podrá o no tener la posibilidad de enviar correos [15].

Su funcionamiento es normalmente llevado a cabo de forma automática, mediante el uso de un gestor de listas de correo y una dirección de correo electrónico capaz de recibir mensajes de correo electrónico. Los mensajes enviados a dicha dirección son reenviados a las direcciones de correo electrónico de los suscriptores de la lista. Dependiendo del software gestor, podrían existir diversas direcciones de correo para la recepción de comandos.

Muchos servidores de listas de correo electrónico ofrecen una dirección de correo para que los suscriptores puedan enviar comandos, tales como darse de alta, de baja o cambiar sus preferencias de suscripción. Algunos servicios de listas de correo de electrónico permiten además varios modos de suscripción como: individual, digest o no correo.

2.1.6.2 Foros

Los foros son programas web, que permiten a sus visitantes interactuar con mensajes enviados al foro de forma similar a como lo harían con los hilos de su correo mediante un programa lector de correo electrónico, con la diferencia de que todo el proceso se hace directamente en una página web.

Los foros son también conocidos en internet como foros de mensajes, de opinión o foros de discusión y su labor principal es dar soporte a discusiones u opiniones en línea. Son los descendientes modernos de los sistema de noticias BBS (Bulletin Board System) y Usenet, muy populares en los años 1980 y 1990. Por lo general los foros en Internet existen como un complemento a un sitio web invitando a los usuarios a

discutir o compartir información relevante a la temática del sitio, en discusión libre e informal, con lo cual se llega a formar una comunidad en torno a un interés común [16].

Las discusiones suelen ser moderadas por un coordinador o dinamizador quien generalmente introduce el tema, formula la primera pregunta, estimula y guía, sin presionar, otorga la palabra, pide fundamentaciones y explicaciones y sintetiza lo expuesto antes de cerrar la discusión.

Un foro en Internet, comúnmente, permite que el administrador del sitio defina varios foros sobre una sola plataforma. Éstos funcionarán como contenedores de las discusiones que empezarán los usuarios; otros usuarios pueden responder en las discusiones ya comenzadas o empezar unas nuevas según lo crean conveniente.

2.1.6.3 Wikis

Los wikis constituyen uno de los medios de intercambio de ideas más recientes en los proyectos de software libre, y se puede definir como sitios web colaborativos que pueden ser editados por varios usuarios de forma simultánea, manteniendo normalmente un control de versiones sobre el texto que se ingresa en el mismo.

La principal utilidad de un wiki es que permite crear y mejorar las páginas de forma instantánea, dando una gran libertad al usuario, y por medio de una interfaz muy simple. Esto hace que más gente participe en su edición, a diferencia de los sistemas tradicionales, donde resulta más difícil que los usuarios del sitio contribuyan a mejorarlo. Dada la gran rapidez con la que se actualizan los contenidos, la palabra «wiki» adopta todo su sentido. El documento de hipertexto resultante, denominado también «wiki» lo produce típicamente una comunidad de usuarios [17].

2.1.6.4 Formas comunes de uso de estos recursos

Cada uno de los medios de comunicación mencionados, posee sus ventajas y desventajas, que lo hacen útil para una labora u otra dentro de la comunidad de los proyecto de software libre. En el caso de las listas y los foros, ambos son conceptos similares pero que se diferencian únicamente en su implementación, ya que las listas requieren del uso de un cliente de correo electrónico y los foros pueden ser usados simplemente mediante el uso de un navegador Web, de hecho debido a la similitud de conceptos incluso existen programas que transforman unos en otros.

En cuanto los wikis, aunque el fin es el mismo, compartir información entre un gran número de personas de forma normalmente bi-direccional y asíncrona, estos representan una nueva y mejorada forma de comunicación, ya que en los foros y listas de correos, no se pueden modificar los aportes de otros miembros a menos que tengas ciertos permisos especiales como moderador o administrador.

Igualmente en los proyectos de software libre se diferencian por el uso típico que se da a los mismos, siendo generalmente usadas las listas de correos y foros son usados para las discusiones y solución de problemas dentro de un proyecto, ya que cada quien puede expresar su opinión y todos la pueden observar sin que un tercero sea capaz de modificarla.

En cuanto a los wikis, por su propia naturaleza colaborativa, son más actos para documentos que sean de carácter público en el proyecto y en el cual todos deben estar de acuerdo y emitir un único documento con todas las correcciones y aportes de los participantes, por lo que son más ideales para crear las especificaciones publicas del proyecto y sus manuales por ejemplo.

2.1.7 Sistemas de seguimiento de errores

Los sistemas de seguimiento de errores son aplicaciones informáticas diseñadas para asistir a los programadores y otras personas involucradas en el desarrollo y uso de sistemas informáticos en el seguimiento de los defectos de software. El término usado en inglés es Bug Tracking System, y frecuentemente se usa el acrónimo BTS [18].

Estos sistemas de seguimiento de errores, en los proyectos de software libre permiten que los usuarios directamente den de alta las incidencias detectadas o bugs. Uno de los más conocidos es Bugzilla, el cual generalmente estos sistemas se integran con otras herramientas, como pueden ser correo electrónico, control de versiones, y otras herramientas de gestión administrativa, para crear un ambiente de trabajo ideal para el software libre conocido como un forge.

Su componente principal es la base de datos, que es el lugar donde se almacenan los hechos e historia de un fallo de software. Los hechos pueden ser una descripción detallada del fallo, la severidad del evento, forma de reproducirlo y los programadores que intervienen en su solución así como información relacionada al proceso de administración de la corrección del fallo como puede ser personal asignado, fecha probable de remedio y código que corrige el problema. Identificando de esta forma un ciclo de vida de un error, mediante el cual se le da seguimiento mediante el estado del problema desde su descubrimiento y reporte hasta su solución final.

2.1.8 Herramientas para la gestión de proyectos

El desarrollo de software libre se encuentra muy relacionado con las redes, especialmente a internet, enfocándose concretamente en aquellas herramientas de software o servicios que puedan servir de apoyo al uso correcto del internet para cumplir nuestro objetivo, que no es otro que elaborar un proyecto de software libre.

La primera elección que se debe tomar a la hora de iniciar un proyecto de software libre es como ha de crearse nuestra infraestructura de soporte del proyecto, teniendo para esto dos opciones: Elegir y configurar por cuenta propia nuestras herramientas o emplear algún servicio que brinde ya instaladas y configuradas las herramientas necesarias. En esta decisión es muy importante que se posea una idea inicial del tamaño que puede tener nuestro proyecto, ya que esto servirá como base para tomar la decisión de instalar nuestras propias herramientas o usar un servicio pre-configurado, ya que salvo en proyectos de gran tamaño no se justifica la pérdida de tiempo y recursos que significa instalar y configurar nuestra propia plataforma de desarrollo.

Sin importar la decisión que se tomé, los servicios que pueden ser útiles para la elaboración de un proyecto de software libre son relativamente los mismos: alojamiento web, servicio de compartición de archivos, tablón y foro de mensaje, listas de correos, organizador de tareas, seguimiento de fallos y solicitud nuevas características, base de datos, control de versiones, cuenta de línea de comandos, interfaz de administración (preferiblemente web).

Y es debido a esta especie de estandarización de los servicios necesarios para elaborar un proyecto de software libre, que han surgido gran cantidad de sitios que tienen como principal objetivo, brindar estos servicios a los diferentes proyectos de software libre que andan buscando una infraestructura adecuada para su desarrollo y cuyos integrantes del proyecto no desean ocuparse de las labores tediosas y poco relacionadas con el proyecto de instalar y configurar su propia infraestructura o no posean los recursos de hardware/software necesarios. Además como punto extra se desea facilidad a la hora de administrar esta infraestructura, es decir una interfaz amigable y fácil de usar generalmente en ambiente web, siendo por esto que surgieron en un principios sitios de desarrollo como sourceforge con su aplicación Alexandria, el cual debido a un cambio en su licenciamiento ha sido ramificado por otros sitios como Gforge a partir de la última versión libre de Alexandria.

Es por esto que de los diversos sitios dedicados a brindar soporte al desarrollo de proyectos de software libre, interesan solo aquellos que se encargan de esta tarea de forma profesional, poseyendo equipos dedicados de manera exclusiva a ofrecer estos servicios y contando con los conocimientos e infraestructura necesarios para esta labor. A continuación se muestran algunos de estos sitios, los cuales poseen plataformas similares, aunque sus funcionalidades y apariencia pueden variar ligeramente, entre ellos se tienen: software-libre.org, Savannah, Alioth, BerliOS, SourceForge.

Luego de escoger un sitio de desarrollo, sin importar cual sea seleccionado, el proceso de creación y configuración de los servicios necesarios para un proyecto de software libre tiende a ser el mismo, pero por labores demostrativas se escogió el sitio sourceforge.net. El primer paso consiste en obtener una cuenta en el sitio deseado, en este caso sourceforge.net, para esto se debe poseer una dirección de correo electrónico valida y dirigirnos a la página de registro del sitio y hacer clic en el enlace referente a la creación de un nuevo usuario, tras lo cual y luego de esperar un tiempo aproximado de 24 horas, se solicitara los datos necesarios y posteriormente se enviará un correo electrónico a nuestra dirección pidiéndonos confirmar el alta del usuario. Una vez confirmado nuestro usuario, se debe especificar los detalles en nuestra cuenta como el nombre de usuario y el idioma, una vez realizado este proceso se ha obtenido ya una cuenta en el sitio SourceForge.

Una vez se tenga en nuestro poder la cuenta del sitio, se podrá iniciar sesión en el mismo con nuestro nombre de usuario y contraseña, tras lo cual se mostrará nuestra página de información personal, tras lo cual se podrá hacer otras tareas como configurar nuestra identidad, nuestro perfil de desarrollador, unirnos a un proyecto existente o crear nuestro propio proyecto.

Por ejemplo, para crear nuestro propio proyecto se debe hacer clic en el enlace que permite crear un nuevo proyecto, tras lo cual se pide se brinde la información sobre el software libre a registrar (es indispensable que nuestro proyecto sea de este tipo) y se preguntará el tipo de proyecto que se desea registrar, posteriormente aparecerá una página con la explicación global de los pasos involucrados en el proceso de registro del proyecto: Información Alojamiento, Registro de Proyecto, Aceptación términos del acuerdo de uso, Escogencia de la licencia del proyecto, Descripción en detalle del proyecto y su nombre, Revisión final, Finalización del proceso, como nota importante todos estos pasos deben ser llevados a cabo en el idioma inglés.

Tras llevar a cabo estos pasos el usuario debe esperar a que los administradores de SourceForge aprueben el proyecto, lo cual puede durar varios días, y no suele ser negado normalmente. Una vez aprobado el proyecto en nuestra página personal aparecerá un enlace a la página del proyecto, en donde se podrá encontrar las herramientas/servicios que brinda el sitio para servir de soporte a dicho proyecto.

Entre las herramientas que brinda SourceForge a nuestro proyecto se tienen: la **cuenta de línea de comandos**, la cual sirve para crear/modificar nuestro sitio web y acceder al CVS como desarrollador del proyecto. Su nombre de usuario es el mismo con el que se registro el proyecto en SourceForge al igual que su contraseña, para ingresar a la misma se puede hacer uso de la herramienta ssh: *ssh **nombreusuario**@**proyecto**.sf.net*, con esto se podrá acceder a nuestro archivos en nuestro directorio de usuario y los archivos de la web de nuestro proyecto, siendo esto algo importante y de lo cual se va a hacer uso constante, por lo cual se puede automatizar con herramientas como Keychain. Para copiar archivos mediante el uso de esta cuenta se puede usar el comando scp: *scp nombrearchivo nombreusuario@proyecto.sf.net:/ruta/* o se puede copiar todo un árbol de directorios con comandos como rsync.

Otras herramientas que se incluyen entre los servicios que brinda SourceForge se tienen: el manejo de versiones a través de **CVS** (aunque SVN, el cual también es ofrecido por SourceForge está siendo más usado actualmente), con lo cual se podrá acceder a las diferentes versiones del proyecto de manera anónima (para descargar su contenido) o como desarrollador (para crear nuevas versiones). Otra opción ofrecida por SourceForge es **gestionar la descarga de nuestro proyecto**, ya sea al poner los archivos en nuestra web o usando su herramienta de descargas, la cual es la opción más acertada, ya que permitirá obtener estadísticas de descargas, historial de releases, y uso de todos los mirrors existentes de SourceForge lo cual generalmente acelera el proceso de descarga para los usuarios.

Una vez conocidas las herramientas que intervienen directamente en el desarrollo de los proyectos brindadas por SourceForge, es hora de ver algunas de las herramientas de soporte al proceso de desarrollo brindadas por el sitio, entre las cuales destacan las **Listas de correo**, las cuales pueden ser creadas fácilmente a través de la interfaz administrativa del proyecto en la web de SourceForge y hacen uso del software Mailman para su gestión, el cual permitirá a través de su interfaz la configuración de las listas creadas.

Otro servicio resaltante de apoyo al desarrollador brindado es el de **Seguimiento de fallos**, o Tracker como se conoce en SourceForge, el cual permite gestionar, asignar y hacer seguimiento de diversos aspectos de nuestro proyecto como: peticiones de soporte, reportes de fallas, peticiones de nuevas funcionalidades, Parches, etc. De este servicio se debe configurar cosas como los usuarios y desarrolladores que pueden mandar fallos, respuestas automáticas a reportes, gestión de grupos, etc.

2.2 Software usado por el proyecto

Para el desarrollo del presente proyecto se procedió a hacer uso de diferentes herramientas de software que brinden la funcionalidad necesaria para cumplir con los objetivos propuestos, buscando siempre que fuese posible alternativas libres, sin embargo la lista que se mostrara a continuación enumera solo las principales herramientas de software que se utilizaron:

A) **Lenguaje de programación Java:** lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode independiente de la plataforma, lo que significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Por consiguiente con java se debe ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run everywhere" algo que traducido podría ser codifícalo una vez, ejecútalo donde quieras [19].

B) **Entorno de desarrollo integrado Eclipse:** es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido" de java, opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico. Este mecanismo de módulos le permite a Eclipse extenderse usando otros lenguajes de programación además de java o simplemente ofrecer funcionalidades extra como clientes de CVS, SVN o herramientas de modelado UML [20].

C) **Umbrello:** desarrollado por Paul Hensgen, es una herramienta libre para crear y editar diagramas UML, que ayuda en el proceso del desarrollo de software. A través de la misma puede crear un diagrama y la herramienta genera código automáticamente en los lenguajes C++, Java, PHP, Python, entre otros [21].

D) **GNU/Linux:** (GNU con Linux o GNU+Linux) es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU y por muchos otros proyectos/grupos de software. Comúnmente a este sistema operativo se le denomina, Linux [22].

E) **GIMP:** significaba originalmente «General Image Manipulation Program» («Programa general para manipulación de imágenes»); en 1997 se cambió para que significase «GNU Image Manipulation Program» («Programa de manipulación de imágenes de GNU»). Es parte oficial del proyecto GNU y sirve para procesar gráficos y fotografías digitales. [23].

F) **Inkscape:** es una herramienta libre de dibujo libre y multiplataforma para gráficos vectoriales SVG. Este programa surgió de una bifurcación del proyecto Sodipodi. Las características de SVG soportadas incluyen formas básicas, caminos, texto, canal alfa, transformaciones, gradientes, edición de nodos, exportación de SVG a PNG, agrupación de elementos, etc [24].

Capitulo 3

El presente capitulo tiene como finalidad explicar las fases de desarrollo de software usadas por el presente proyecto para su diseño e implementación, así como también los productos obtenidos en cada una de estas fases de forma general.

3.1 Fases de desarrollo y diseño del Proyecto

Al momento de iniciar cualquier proyecto de software, siempre existe la probabilidad de éxito o fracaso del mismo, si no son tomadas las decisiones correctas en el momento oportuno, por esta razón, es que si se procede a observar diferentes proyectos de software libre se conseguirán con grandes éxitos como Linux o Apache, y grandes fracasos que nunca lograron consolidarse como proyectos estables y crear una comunidad a su alrededor.

Por este motivo, se debe observar las razones de los éxitos y fracasos de los proyectos de software libre y tomar sus mejores prácticas o errores para intentar llevarlas a cabo o evitarlas respectivamente, según sea el caso. Por lo que al iniciar entonces "formalmente" el desarrollo del proyecto, se observa que el primer problema que aqueja normalmente a los proyectos de software libre, es la falta de descripciones formales de las actividades que se deben llevar a cabo dentro de cada fase de su desarrollo y que guiaran su comportamiento futuro dentro del ciclo de vida del proyecto e incluso sus propias fases son difíciles de definir formalmente.

Es por esto que para intentar comprender la situación, se decidió acudir a un ensayo que significo en su momento un gran avance en los esfuerzos por promover el software libre, el mismo es la catedral y el bazar, creado por Eric S. Raymond [7], en el cual se analiza el surgimiento de algunos proyectos de software libre y se elabora una comparación entre el desarrollo tradicional al cual se califico de rígido como una catedral y el desarrollo de software libre al cual se califico de bullicioso como un bazar.

Tras lo cual se obtuvieron ciertas razones que pueden lograr el éxito de un proyecto de software libre o no, pero la situación aún no quedaba del todo clara, porque el modelo propuesto por E. Raymond, era poco flexible y observa ambas formas de desarrollo como diametralmente opuestas. Es por eso que tras continuar buscando una forma correcta de iniciar el desarrollo del proyecto, surgieron las ideas propuestas por A. Senyard y M. Michlmayr [8], el cual hace uso de la teoría de la catedral y el bazar no como opuestas sino más bien como complementarias dentro de diversas fases durante la creación de un proyecto de software libre.

Estos autores, explican que el desarrollo de un proyecto de software libre, en sus fases iníciales no opera en el contexto de una comunidad de amplios voluntarios dispuestos a colaborar eficazmente en cada aspecto o tarea que sea necesaria realizar en el proyecto. De hecho el mismo observa incluso que los proyectos exitosos de software libre por lo general son los que hacen una "transición" exitosa de un desarrollo tradicional cerrado, a un proyecto basado en una comunidad, y de hecho el autor considera imposible generar un proyecto directamente en fase de bazar.

Es por esto que haciendo uso de este articulo, se elaboro una estrategia para desarrollo del proyecto de software libre, la cual se procedió a dividir en fases que posteriormente permitieron la ejecución y puesta en marcha del presente proyecto.

3.1.1 Fase Inicial como proyecto Cerrado

Esta fue la primera fase del proyecto, la cual tuvo varias características de un desarrollo tradicional pero mezcladas con las características propias de un proyecto de software libre, esta fase fue iniciada por un solo individuo el cual trabajando aislado de una comunidad tuvo la idea de llevar a cabo un proyecto que facilitará la creación y ejecución de sentencias SQL en la base de datos postgresql a usuarios con escasos conocimientos del funcionamiento de las base de datos.

Para esto se decidió llevar a cabo una implementación inicial del software, y se hizo uso de algunas técnicas regulares de la ingeniería del software, como lo fueron la recolección de requerimientos, el diseño, la implementación y las pruebas, pero de una forma que se ajustará al tipo de desarrollo de software libre, al llevar a cabo actividades que pudiesen fomentar algunas de las ventajas del licenciamiento usado, como por ejemplo consultando opiniones de personas en algunos foros públicos, o consultando expertos en la materia para resolver algunas dudas que fueron surgiendo durante esta fase.

Es de recalcar que este proceso de desarrollo al inicio del proyecto fue llevado a cabo de forma controlada y aislada de una comunidad, y se considero el desarrollo inicial de la aplicación deseada, el mismo se caracterizo por un desarrollo estático acompañado de una planificación centralizada. Siendo su principal característica, el seguimiento (de forma no estricta) de una metodología clásica de uso común durante el desarrollo de software orientado a objetos, conocida como RUP (Rational Unified Process)[25], la cual posee un carácter iterativo e incremental, que permite la fácil adaptación del software en desarrollo a cambios suscitados durante la ejecución del proyecto.

Para aplicar esta metodología, se dividió el proyecto en ciclos, a los cuales se le asigno un producto que se debía obtener al final de cada ciclo, a su vez cada ciclo de desarrollo se dividió fases, donde la finalización de cada fase se logra al alcanzar un hito:

3.1.1.1 Fase RUP de Concepción

En esta fase se definió el alcance del proyecto de software libre Visual SQL Builder, además de establecer sus casos de uso, para obtener de esta forma una visión general de los requerimientos primordiales del proyecto, esto se complemento con información obtenida a través de internet de manera informal. Para esto se realizaron varias actividades entre las que destacaron:

- Se crearon algunos bosquejos de casos de uso de la aplicación y se comprobó su entendimiento con personas interesadas en expresar su opinión (al azar), para comprobar su apego a lo que serian los requerimientos de la aplicación.
- Se identificaron algunos puntos importantes de la aplicación.
- Se establecieron las reglas que impone el diseño deseado para la realización de los puntos importantes de la aplicación.
- Se identificaron algunos riesgos potenciales.
- Y se obtuvo un plan de proyecto preliminar.

De forma más detallada, durante las continuas iteraciones en esta fase fue donde se definió realmente cual era el requerimiento primario que la aplicación debía satisfacer, primordialmente debido a que un proyecto de este tipo requiere de gran inversión de energía, tiempo y compromiso, por lo que para que el proyecto no pierda interés, se debe realizar algo que signifique un gran incentivo para su creador.

Pero para llevar a cabo esa búsqueda del requerimiento primordial, se procedió a responder a la pregunta, ¿Qué otros proyectos existen que puedan cumplir con el requerimiento de escribir sentencias SQL de forma visual para la base de datos postgresql?, para lo cual se identificaron una serie de proyectos del mundo del software libre o no y se observo que no existía realmente uno que cumpliese con las expectativas deseadas, ya que como lo dicta implícitamente la filosofía del software libre se debe promover siempre que sea posible la reutilización del software, evitando la duplicación de esfuerzos.

Fue luego de la etapa anterior cuando se decidió, hacer un pequeño análisis de riesgos (informal) acerca de que lograría la motivación y compromiso a largo plazo en las personas que deseen trabajar en el proyecto, y como este proyecto podría competir con otros existentes para atraer una base de usuarios y por ende posibles voluntarios. En otras palabras se decidió darle al proyecto una característica única que lo diferenciase de otros existentes como lo fue la creación del la sentencia SQL de forma visual y en una interfaz que lo ayudase a evitar los errores comunes asociados al aprendizaje del lenguaje SQL.

3.1.1.2 Fase RUP de Elaboración

Durante esta fase se expresaron con mayor claridad los requisitos de la aplicación, asignándoles sus respectivas prioridades. Igualmente se analizo el dominio del problema, estableciendo una base arquitectónica sólida y desarrollando un plan informal de proyecto.

Al final se examinan los alcances, los objetivos del software y la elección de la arquitectura, decidiendo posteriormente pasar a la fase de construcción, por lo que en esta fase se identificaron aquellos requerimientos adicionales que permitan dar mayor funcionalidad a la aplicación, para ello se realizaron las siguientes actividades:

- Se identificó y describió los principales casos de uso de la aplicación y sus posibles diagramas de secuencia.
- Se identificó los requisitos no funcionales del sistema.
- Se realizó la descripción de la Arquitectura del Software;
- Se elaborará el diagrama de clases de la aplicación
- Se elaboró del plan de desarrollo para el resto de la fase inicial del proyecto.

Durante esta fase se decidió dividir el proyecto en dos grandes áreas: la escritura de sentencias SQL y su ejecución. Siendo importante recalcar, que el área objetivo de la primera liberación pública propuesta para el proyecto, es la escritura de sentencias SQL, siendo su ejecución llevada a cabo de forma opcional y muy básica.

Posteriormente se procedió a dividir estas grandes áreas en sub-áreas que permitiesen un mejor manejo de la aplicación, esto se realizo al área de escritura de sentencias SQL (el objeto primordial de este proyecto), al dividirla en: el manejo de la interfaz grafica del usuario y el manejo de la representación lógica interna de los datos a ser manipulados, como se puede observar en la siguiente figura:

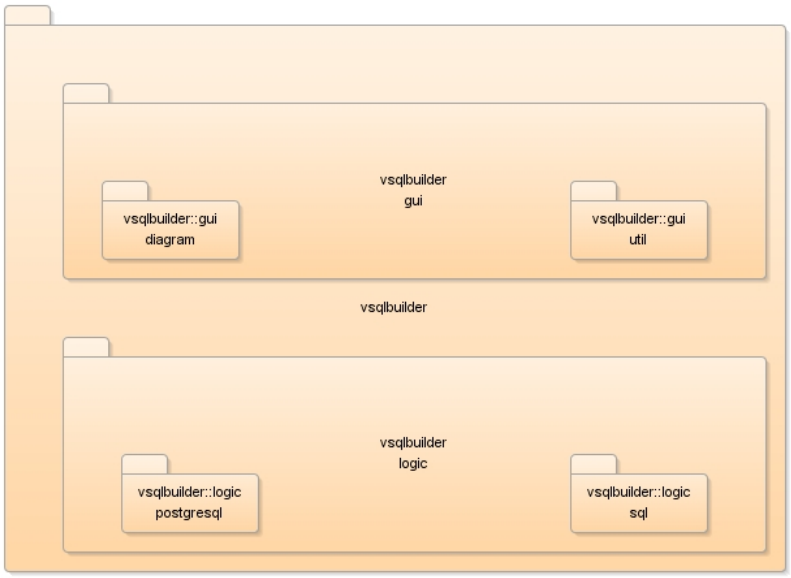


Ilustración 2 - Paquetes presentes en la aplicación Visual SQL Builder

Teniendo asignadas cada una de estas divisiones algunas de las siguientes actividades:

- **División del manejo de la interfaz gráfica de usuario:**

  Es la encargada de generar una representación gráfica del modelo creado, a partir de la manipulación del modelo lógico de la base de datos representada de forma interna en la aplicación, dicha representación gráfica debe ser capaz de interactuar con el usuario y le servirá al mismo como un punto de enlace al modelo interno de la aplicación permitiéndole introducir sus ordenes y cambios en este. En otras palabras, esta parte de la aplicación también será la encargada de responder a los eventos (cambios) ocasionados por el usuario en la representación gráfica y transmitir los mismos al modelo estructurado en memoria. Es de recalcar que esta interfaz será llevada a cabo mediante el modelo vista-controlador, buscando independencia de la visión gráfica del software con respecto a su modelo interno.

- **División del manejo de la lógica interna de la aplicación:**

  Es la encargada de gestionar las operaciones no gráficas de la redacción de sentencias SQL, es decir mantendrá un modelo estructurado en memoria, de la sentencia que se está elaborando, el cual representara a la información con la cual operara esta parte de la aplicación, es decir la lógica que añade sentido a lo que se representara posteriormente de forma visual.

  Maneja de igual forma, la parte de comunicación con la base de datos postgreSQL e ingeniería inversa, siendo la encarga de comunicar a esta parte de la aplicación con la base de datos SQL y poner a disposición de la aplicación todas las funciones de programación necesarias para intercambiar información con la misma. Más detalladamente, deberá permitir la creación de los objetos básicos existentes en el área de núcleo, es decir extraerá la información necesaria acerca de los metadatos (datos acerca de los datos) de la base de datos y con los mismos procederá a llevar a cabo una ingeniería inversa (solo en estructura de datos no visualmente) del modelo físico de datos usado para la creación de dichas tablas.

Por último la parte de generación, una vez el usuario posea un modelo coherente en memoria, la aplicación podrá recorrer el mismo y generar la sentencia SQL expresada en él para su futura ejecución, también en un futuro (por ahora lejano) optimizara las sentencias y dará consejos expertos al usuario acerca de la forma en la que está realizando su consulta.

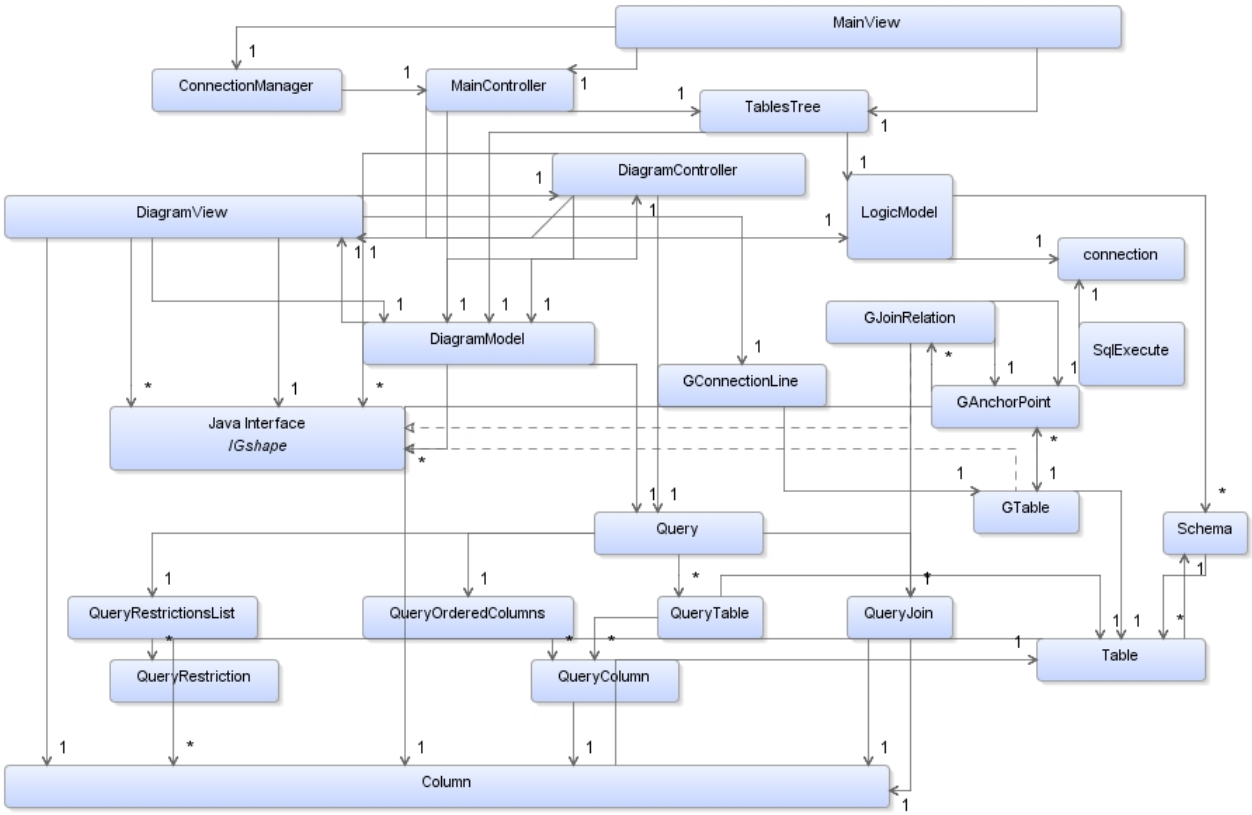


Ilustración 3 - Diagrama de Clases resumido de la aplicación Visual SQL Builder

Esta división permitirá en un futuro a los posibles voluntarios que deseen contribuir con el proyecto, hacerlo de una forma menos centralizada, ya que se poseen subsistemas que se comunican entre ellos a través de interfaces bien definidas. En esta etapa fue importante la publicación del código fuente desarrollado (a través de un CVS local) y la consulta constante a comunidades de otros proyectos existentes, en búsqueda de ideas que permitiesen solucionar problemas que iban surgiendo durante el avance de las iteraciones del proyecto.

3.1.1.3 Fase RUP de Construcción

En esta fase se desarrolló de forma iterativa e incremental el software hasta obtener el prototipo deseado, esto implicó describir los requisitos de la aplicación, refinar el diseño y especialmente revisar posibles fallas en sus criterios, describiendo y creando la interfaz gráfica de la aplicación, seguidamente se elaboro la estructura de clases necesaria y se codificaron las mismas.

Siendo el criterio de superación de esta fase, que lo elaborado fuese técnicamente adecuado y "aceptable", es decir no muy simple y lleno de fallas ya que esto alejaría los voluntarios, pero tampoco muy complejo y con casi toda su funcionalidades implementadas, porque esto igualmente alejaría a los voluntarios, debido a la dificultad de adaptación al código creado o porque que ellos desean hacer algo que ya fue implementado previamente.

La infraestructura usada en esta etapa fue elaborada de forma local, y consistió en un servidor CVS y el acceso a foros públicos y listas de correos de otros proyectos con fines similares (hacen uso de base datos o modelo gráficos), para la búsqueda código previamente elaborado (compatible con el licenciamiento escogido), siendo usado para la codificación herramientas y utilidades de licenciamiento libre.

3.1.1.4 Fase RUP de Transición

Durante esta fase el objetivo fue traspasar el software una vez desarrollado a sus posibles usuarios (compañeros o conocidos dispuestos a probarlo), pero se debe recordar que hasta este momento únicamente se tenía un pequeño bosquejo de lo que sería la aplicación, por lo cual se realizaron pruebas de validación y verificación más que todo de forma cerrada, para determinar el correcto funcionamiento de la aplicación que estaba siendo desarrollada en cada iteración, surgiendo así la posibilidad del surgimiento de nuevos elementos que implicaban nuevos ciclos de desarrollo para su obtención.

Es importante recordar que el producto obtenido por esta fase no era una aplicación final, sino por el contrario era una aplicación que cumpliese de forma creíble con la promesa realizada por el proyecto, que es la de permitir la creación de sentencias SQL de forma visual.

También se debe comprender que en esta fase, las interrogantes existentes sobre la distribución del trabajo realizado son menos importantes que en las futuras fases, por lo que el código fuente fue distribuido por medio de correo electrónico o dando acceso de lectura al CVS local a ciertas personas que decidieron contribuir a probar el software.

3.1.2 Fase de Transición de proyecto Cerrado a Bazar

Una vez completado el proceso RUP requerido para comenzar esta fase, el primer paso fue el de comprobar que la promesa del objetivo del proyecto fue alcanzada en algún porcentaje con la implementación inicial obtenida en la fase anterior, y la cual se reforzó mediante el uso de pruebas técnicas y de usuarios.

Pero antes de liberar esa implementación inicial a un universo de usuarios mucho más amplio, se comprobó igualmente que el programa se encontraba en un estado en el cual podía ser compartido con otras personas y posiblemente atraer otros desarrolladores, entonces se inicio una liberación inicial del proyecto y se comenzó la transición hacia la fase conocida como el bazar, como nota final se informa que el release inicial del proyecto fue conocido como Visual SQL Builder Pre-Alpha 0.1.

Pero una vez transcurrido un pequeño periodo de adaptación, el proyecto requirió un drástico cambio en su estructura, especialmente en la forma de desarrollo y gestión del mismo, ya que antes era llevada a cabo de forma tradicional un poco cerrada, por lo cual surgieron nuevas preguntas que debían ser contestadas, cual es el siguiente paso a llevar a cabo.

3.1.2.1 Distribución e infraestructura asociada

Luego de una breve reflexión, la primera pregunta que surgió es como se debería distribuir el código fuente generado durante el desarrollo cerrado del proyecto, se debe recordar que la idea ahora es atraer voluntarios con el prototipo creado en la fase anterior, el cual aunque funcional, está muy limitado y aún necesita ser mejorado en muchos aspectos y es aquí donde la colaboración de la comunidad es importante.

Es aquí donde una buena infraestructura de soporte y publicidad del proyecto debe entrar en juego, y para esto se tienen básicamente dos opciones: se crea nuestra propia infraestructura o se usa una de las ofrecidas por internet y que han sido ampliamente comprobadas.

En este punto se tomo la decisión de crear el proyecto en una infraestructura las pre-existentes en internet, motivado por dos razones principales:

- Evitar perder tiempo en aspectos no relevantes al desarrollo y avance el proyecto, que inicialmente pueden ser manejados mejor en sitio con amplia experiencia en ellos y que brindara su experticia y soporte al proyecto.
- Aprovechar la base de usuarios y voluntarios existentes en la comunidad, que participan en los diversos proyectos que hacen vida en el sitio, así como también verse beneficiados por los esfuerzos publicitarios de este sitio para posicionar el proyecto ante su posible y potencial audiencia, ya sean usuarios o voluntarios.

Por estas razones, el sitio escogido fue el de sourceforge.net, ya que cuenta con una amplia experiencia en el impulso y mantenimiento de la infraestructura de proyectos de software libre. Además de poseer una gran gamma de herramientas pre-configuradas que facilitaran cada uno de los aspectos que involucra el desarrollo de un proyecto de software libre.

Lo que permite a la comunidad participar en el proyecto y obtener su código fuente de una forma relativamente rápida, debido a la pre-configuración realizada por el sitio de las herramientas necesarias (SVN, Servidor de archivos, etc.), pudiendo entonces el posible usuario o voluntario, inspeccionar, compilar y ejecutar el código fuente del proyecto.

3.1.2.2 Mecanismos de comunicación

El segundo aspecto asociado a la transición del proyecto de estilo catedral a bazar, es el establecimiento de los mecanismos de comunicación, que promuevan la comunicación en el proyecto, ya que la existencia de una comunidad que permita discutir los errores y decidir el futuro del proyecto es de vital importancia para el éxito del proyecto.

Es de resaltar que existen diversos métodos de comunicación con el desarrollador como los wiki, foros, listas de correo y correos electrónicos, que permiten la interacción de la comunidad con sus desarrolladores y por ende con el proyecto. Cómo se explico anteriormente, los mecanismos de comunicación son provistos por sourceforge.net de forma pre-configurada.

Por lo que una vez establecidas las herramientas de comunicación, el siguiente punto es compartir el conocimiento acumulado por el desarrollador durante su fase cerrada, con sus comunidad.

3.1.2.3 Definición del licenciamiento

Pero antes de comenzar la distribución del código, el mismo debe ser amparado por una licencia que acepten los prospectos de voluntarios del mismo y bajo la cual ellos estén dispuestos a compartir sus parches y mejoras al proyecto.

Por el que este proceso inicio con obtención de una lista inicial de licencias de software libre se procedió a observar las páginas de las principales organizaciones encargadas de promover el software libre o de fuente abierta, como lo son: la FSF [26] y la OSI [27], el resultado directo de esta acción fue concluir que existe una gran proliferación de licencias de software libre [28], cada uno con sus particularidades pero todas con un fin común: proveer un marco legal de trabajo a las personas que hacen uso de un código fuente libre o abierto.

En el mundo del software libre o de fuente abierta, una práctica común es alentar la compartición de código fuente entre desarrolladores de software, donde cada una de las piezas de código que se desean integrar debe poseer una licencia en particular, creando esto como principal problema según S. Raymond, la posibilidad de poseer incompatibilidades entre licencias mediante la incapacidad del usuario para de forma concurrente cumplir con los términos de dos licencias al mismo tiempo [29].

Por esto para seleccionar la licencia del presente proyecto se deseo ubicar una forma de limitar el ámbito de las licencias que intervienen en la decisión a tomar, para esto se tomaron en cuenta solo aquellas licencias que fueron clasificadas como licencias populares, ampliamente usadas o con comunidades bien establecidas, según el comité de de proliferación de licencias de la OSI [30], quedando las mismas limitadas a:

- Apache License, 2.0
- New BSD license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 1.1 (MPL)
- Common Development and Distribution License
- Common Public License
- Eclipse Public License

Ahora con las posibles licencias a usar en el proyecto se observa la compatibilidad de las mismas con las dos principales tendencias existentes: software libre de las FSF o de fuente abierta de la OSI, para esto se procede elaborar un cuadro donde se muestra la compatibilidad de las mismas con cada tendencia [31]:

Nombre de la licencia	Aprobación FSF	Aprobación OSI	Permite mezclar código con diferentes licencias	Permite liberar cambios bajo una licencia diferente
Apache License, 2.0	Si [pero incompatible con GPL]	Si	Si	Si
New BSD license	Si	Si	Si	Si
GNU General Public License (GPL) v2	Si	Si	No	No
GNU Library or "Lesser" General Public License (LGPL) v2	Si [pero incompatible con BSD]	Si [pero incompatible con BSD]	Si	No
MIT license	Si	Si	Si	Si
Mozilla Public License 1.1 (MPL)	Si	Si	Si	No
Common Development and Distribution License	Si	Si	Si	No
Common Public License	Si	Si	Si	No
Eclipse Public License	Si	Si	Si	No

Tabla 9 - Breve comparación entre las licencias seleccionadas

Posteriormente se debe proceder a escoger una licencia que fuese aprobada por la OSI y la FSF, pero que permitiese mezclar código con licencias diferentes, sin comprometer la disponibilidad de los cambios realizados al código fuente original, que sean liberados posteriormente. Igualmente se debe tomar en cuenta que la meta principal del presente proyecto es obtener los beneficios del desarrollo de software libre como contribuciones de otros, arreglos de bugs, etc. por lo que la licencia debe ser lo suficientemente flexible para alentar las contribuciones de otros usuarios, pero también debe permitir usar el software a tantas personas como sea posible.

Por esta razón inicialmente se debe proceder a escoger la licencia LGPL v2, ya que esta permite mezclar el código elaborado con software no libre, aumentado de esta forma la base de usuarios y desarrolladores del software que hagan uso del código fuente elaborado, pero manteniendo este código desarrollador por el proyecto en una forma libre o de fuente abierta. Aunque, posteriormente, se añadió una licencia dual al mismo incluyendo la licencia GPL v2, y pudiendo ser usada cualquiera de ellas según se desee.

3.1.2.4 Cambio de Estilo de Gestión del proyecto

El siguiente paso es el establecimiento de un estilo de gerencia y criterios que faciliten la unión de voluntarios al proyecto, esto debido a que en la primera fase cerrada del proyecto, el desarrollador tomo sus decisiones por su propia cuenta dando simplemente valoración a las opiniones externas, pero no un peso real sobre el proceso de toma de decisiones.

Es por esto que ahora que el proyecto se ha abierto al público y se ha permitido a otras personas participar en el proyecto, se debe otorgar presencia y aceptación a las contribuciones propuestas por los voluntarios e incorporar sus ideas en el proyecto, porque de lo contrario muy probablemente se desmotiven y no vuelvan a realizar aportes al mismo.

Por eso en este proyecto, se opto por tomar como estilo de gestión del proyecto el de una leve meritocracia democrática, en la cual mientras más colabora una persona tiene mayor derecho implícito a ser escuchado, más realmente las verdaderas decisiones del proyecto deben ser tomadas de forma conjunta por la comunidad del proyecto.

Además de esto, el desarrollador principal del proyecto busca no discriminar por ninguna razón a ninguna contribución al proyecto, en base a su autoría del proyecto inicial y busca por ende en no convertirse en un dictador de su propio proyecto, al remitir las decisiones a tomar al resto de la comunidad o interesados, para luego entre todos tomar una decisión conjunta.

3.1.3 Fase del Bazar

Una vez completada la fase anterior, se comienza lo que se podría decir es la fase del bazar, en la cual se debe terminar de adoptar, esta nueva forma de desarrollo dictada por el licenciamiento del software libre, y sacar el mayor provecho posible de la misma.

Aunque es importante recalcar que lo acá mencionado se puede tomar como los planes en pleno desarrollo del presente proyecto de software libre, ya que estos acontecimientos están empezando a llevarse a cabo en el mismo, por lo que lo acá expuesto se puede tomar simplemente como un bosquejo de las actividades en desarrollo en el proyecto.

3.1.3.1 Apertura de los requerimientos

Ahora que el proyecto pertenece a una comunidad, debe ser esta la que determine que se hace o que no se hace al proyecto, cuáles son sus prioridades y cuál será su rumbo futuro, es acá entonces donde es importante escuchar a la comunidad y llevar un registro de esta actividad para posteriormente poder hacer uso de estos datos durante futuros desarrollos en el proyecto.

En el presente proyecto, fase está comenzando a implementarse en el momento de escritura del presente documento, y en una primera etapa se intento llevar a cabo mediante el uso de la herramienta tracker en su apartado de solicitud de características del sitio sourceforge.net, pero por motivos aún en estudio, la mayoría de las personas interesadas a enviado sus peticiones directamente al desarrollador, haciendo uso del correo electrónico, e ignorando esta herramienta, por esta razón, aunque las mismas se encuentran abiertas, el trabajo debe hacerse ahora es verificar cual es el problema con esta herramienta e intentar solucionarlo a futuro.

3.1.3.2 Implementación en paralelo y depuración del código fuente

Debido a que ahora cualquier persona podría técnicamente contribuir con código al proyecto, se debe tener un mecanismo que coordine estas contribuciones y evite siempre que es posible la redundancia de trabajo y para eso se debe nombrar lo que se conoce como mantenedor (maintainer en ingles) del repositorio.

En esta etapa cabe mencionar que a la hora de escribir este documento, aún no se poseía a nadie en el cargo de maintainer, pero existe una persona que ya ha manifestado querer serlo, pero debido a que no cuenta con el conocimiento necesario, aún se encuentra en fase de aprendizaje y por el momento está colaborando en la elaboración de un manual para el entendimiento del código fuente (en su estado actual), para de esta forma, ingresar en el contexto del proyecto más fácilmente a las personas interesadas en participar y prestarles una guía rápida de adaptación al mismo.

3.1.3.3 Limpieza y división en módulos del código fuente

Una vez superados los obstáculos anteriores, debido a requerimientos que han ido surgiendo entre las personas que han hecho uso de la liberación inicial, se han comenzado a observar que existen algunos problemas conceptuales en el diseño inicial que deben ser mejorados, para su posterior extensión.

Pero igualmente existen aún partes del mismo que no han sido implementadas, razón por la cual, en este momento la prioridad del software debe ser decidido por la comunidad y próximamente se observaran los resultados de estas decisiones.

3.2 Diseño, Implementación, Pruebas, Mantenimiento e interacción con la comunidad

A continuación se procederá a explicar de forma breve como el estado y los resultados de las últimas fases del proceso de creación del proyecto de software libre Visual SQL Builder.

3.2.1 Diseño

Una vez presentadas las fases de desarrollo del proyecto, el diseño resultante se puede comprender al observar una pequeña descripción de sus dos paquetes más importantes:

3.2.1.1 Paquete vsqlbuilder.logic

Es el encargado de manejar la lógica interna de la aplicación, su función abarca desde llevar a cabo la conexión a la base de datos postgresql, para extraer los metadatos de las tablas que le permitan crear un modelo en memoria de las mismas (vsqlbuilder.logic.postgresql y vsqlbuilder.logic), hasta contener los ítems que representaran la sentencia SQL creada por el usuario final de forma gráfica (vsqlbuilder.logic.sql):

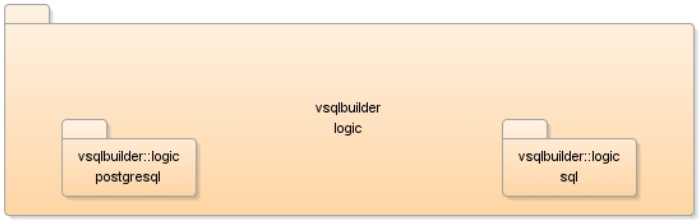


Ilustración 4 - Paquete vsqlbuilder.logic de la aplicación Visual SQL Builder

3.2.1.1 Paquete vsqlbuilder.gui

Es el encargado de interactuar con el usuario y con los modelos internos de la aplicación, contenidos en el paquete vsqlbuilder.logic, siendo su función más básica la de generar la pantalla inicial de la interfaz de la aplicación para el usuario (vsqlbuilder.gui), hasta mantener la representación gráfica de los modelos de las consultas SQL llevadas a cabo por la aplicación (vsqlbuilder.gui.diagram):

Ilustración 5 - Paquete vsqlbuilder.gui de la aplicación Visual SQL Builder

Usando en primer lugar para este fin, lo que han sido hasta la fecha las primeras tres liberación del proyecto, realizadas luego de alcanzar ciertos hitos durante el desarrollo de la aplicación.

3.2.2 Implantación, Pruebas y Mantenimiento

Como ya se menciono anteriormente en las fases del proyecto, la aplicación fue liberada a través de sourceforge.net, por lo que para poder comprender el estado actual de su implantación, se debe estudiar cómo se encuentra conformado su código fuente y la extensión del mismo la 0.1 pre-alpha , por lo que en una primera instancia se procederá a observar algunas métricas de lo que fue la primera liberación del proyecto:

Visual SQL Builder Release 0.1 Pre-Alpha
SLOC Directory SLOC-by-Language (Sorted) 2172 src-0.1 java=2172 Totals grouped by language (dominant language first): java: 2172 (100.00%)
Total Physical Source Lines of Code (SLOC) = **2,172**
Development Effort Estimate, Person-Years (Person-Months) = **0.45 (5.42)** (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months) = **0.40 (4.75)** (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule) =**1.14**
Total Estimated Cost to Develop = **$ 61,002** (average salary = $56,286/year, overhead = 2.40).

Tabla 10 - Resultado de la ejecución del Sloccount a la liberación 0.1-pre-alpha de Visual SQL Builder

Esta liberación, fue la versión 0.1 pre-alpha del proyecto, y su principal misión fue la de llevar a cabo el lanzamiento inicial del proyecto, consistiendo de un prototipo que razonablemente daba credibilidad al objetivo buscado por el proyecto, que fue realizado de forma modular, y fue probado para verificar que compilaba y se ejecuta de forma correcta (salvo algunos bugs). Su funcionalidad en esta versión, simplemente permitía la selección de datos de una tabla a la vez, sin filtrado de los mismos, siendo su principal objetivo, el de atraer colaboradores al proyecto, y permitir tener un punto de partida donde se pudiesen sentar las bases del mismo.

Posteriormente fue realizada la segunda liberación del proyecto, la 0.2 pre-alpha, la cual fue motivada por la labores de depuración de errores planteados por las personas que probaron la primera versión, y la adición de nuevas funcionalidades como la elaboración de consultas de varias tablas, pero ahora incluyendo la posibilidad de realizar uniones entre tablas (join, en ingles). A continuación se observaran algunas de sus métricas:

Visual SQL Builder Release 0.2 Pre-Alpha
SLOC Directory SLOC-by-Language (Sorted) 2833 src-0.1 java=2833 Totals grouped by language (dominant language first): java: 2833 (100.00%)
Total Physical Source Lines of Code (SLOC) = **2,833**
Development Effort Estimate, Person-Years (Person-Months) = **0.60 (7.16)** (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months) = **0.44 (5.28)** (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule) =**1.36**
Total Estimated Cost to Develop = **$ 80,631** (average salary = $56,286/year, overhead = 2.40).

Tabla 11 - Resultado de la ejecución del Sloccount a la liberación 0.2-pre-alpha de Visual SQL Builder

Tras lo cual se comenzó a realizar una promoción aún mayor del proyecto en búsqueda de una base de usuarios y voluntarios, algunos de los cuales tras probar la versión 0.2 solicitaron la inclusión del proceso de filtrado de las filas devueltas por la aplicación, lo cual se añadió a esta versión, así como también la solución de algunos bugs en la aplicación, naciendo de esta forma la versión 0.3-alpha. A continuación se observaran algunas de sus métricas:

Visual SQL Builder Release 0.3 Pre-Alpha
SLOC Directory SLOC-by-Language (Sorted) 3580 src-0.1 java=3580 Totals grouped by language (dominant language first): java: 3580 (100.00%)
Total Physical Source Lines of Code (SLOC) = **3,580**
Development Effort Estimate, Person-Years (Person-Months) = **0.76 (9.16)** (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months) = **0.48 (5.80)** (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule) =**1.58**
Total Estimated Cost to Develop = **$ 80,631** (average salary = $56,286/year, overhead = 2.40).

Tabla 12 - Resultado de la ejecución del Sloccount a la liberación 0.3-pre-alpha de Visual SQL Builder

Esta última versión fue mucho más estable que las dos anteriores y carecía de errores básicos que limitaron el uso en las dos versiones anteriores, lo cual fue posible gracias a los aportes realizados por voluntarios que informaron del error. Por esta razón y luego de estudiar el desenvolvimiento del proyecto, se procedió a realizar una campaña de promoción directa a través de anuncios en fórums y listas de correo de la misma en búsqueda nuevamente de una mayor base de usuarios que sirva como testers de la misma y posteriormente si lo desean se integren a la comunidad.

Por lo cual para observar el avance de la implantación del proyecto, se procederá a mostrar algunas estadísticas útiles proporcionadas por el sitio de desarrollo sourceforge.net:

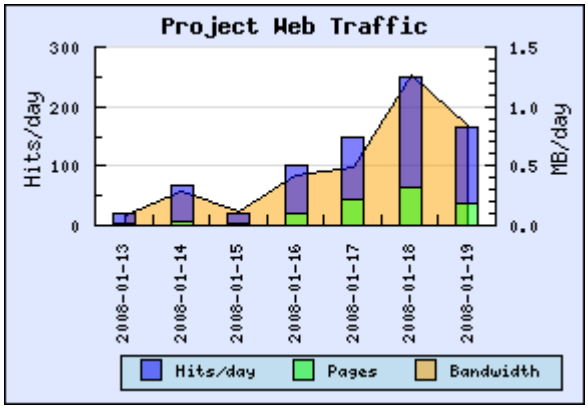


Ilustración 6 - Estadísticas de tráfico del sitio web del proyecto en sourceforge.net (19/01/2008)

En la Ilustración 6, se puede observar como ha sido el crecimiento de las personas que visitan el sitio web el proyecto en sourceforge.net durante la semana correspondiente del (13-19 de Enero del 2008). De la misma forma, se puede apreciar como ha sido la evolución de las descargas de la aplicación durante ese mismo periodo de tiempo, al observar la Ilustración 7:

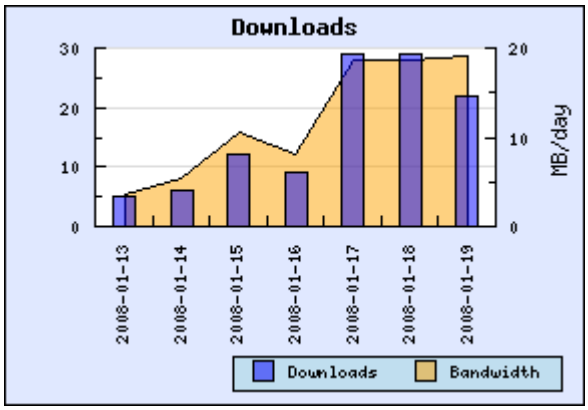


Ilustración 7 - Estadísticas de descarga de la aplicación desde sourceforge.net (19/01/2008)

Lo cual ha tenido como resultado inmediato una modificación del ranking de la aplicación dentro del propio sitio web sourceforge.net, como se observa en la Ilustración 7:

Date (UTC)	Rank	Total Pages [1]	Downloads	Project Web Hits
19 Jan 2008 *	592	325	22	164
18 Jan 2008	647	553	29	249
17 Jan 2008	898	484	29	147
16 Jan 2008	1,710	262	9	100
15 Jan 2008	1,644	121	12	21
14 Jan 2008	1,575	110	6	69
13 Jan 2008	1,538	110	5	19

Ilustración 8 - Ranking de la aplicación Visual SQL Builder en sourforge.net (19/01/2008)

Igualmente al consultar los contadores propios del proyecto dentro de la página web de la aplicación (vsqlbuilder.sourceforge.net), permite observar como la misma ha comenzado a aprovechar las ventajas ofrecidas por el sitio web sourceforge y el efecto de la promoción de la versión 0.3 alpha, por lo que ha sido visitada por personas de diferentes lugares del mundo:

Num	Perc.	Country Name	Num	Perc.	Country Name
94	43.12%	Venezuela	2	0.92%	Malaysia
18	8.26%	Peru	2	0.92%	Hungary
12	5.50%	France	2	0.92%	Belarus
11	5.05%	Germany	2	0.92%	Sweden
7	3.21%	United States	2	0.92%	Korea, Republic Of
7	3.21%	Canada	2	0.92%	Spain
6	2.75%	Unknown	1	0.46%	Romania
5	2.29%	Russian Federation	1	0.46%	Mexico
4	1.83%	Italy	1	0.46%	Serbia
4	1.83%	Belgium	1	0.46%	Poland
4	1.83%	Argentina	1	0.46%	Brazil
3	1.38%	Turkey	1	0.46%	Netherlands
3	1.38%	Czech Republic	1	0.46%	Egypt
3	1.38%	United Kingdom	1	0.46%	Tunisia
3	1.38%	Colombia	1	0.46%	Ireland
3	1.38%	Finland	1	0.46%	El Salvador
2	0.92%	Uruguay	1	0.46%	Australia
2	0.92%	Morocco	1	0.46%	New Zealand
2	0.92%	South Africa	1	0.46%	Singapore

Ilustración 9 - Estadísticas de países de origen de visitantes a la página vsqlbuilder.sourceforge.net

Aunque estos resultados muestran un 94% de personas de Venezuela, esto es debido a la promoción que se llevo a cabo en listas de correos del país, las invitaciones realizadas a personas conocidas y las visitas llevadas a cabo para pruebas, por lo que ese resultado del país no es del todo fiable.

Por estos motivos, se puede afirmar que la fase de implantación, pruebas y mantenimiento se encuentran en plena ejecución, y la labor que resta es terminar de implantar un proceso de pruebas internas que ayuden a lograr un software de mayor calidad previo a su liberación al público en general, así como también, brindar un soporte adecuado a la aplicación para que de esta forma la información que se logre capturar durante esta fase sea tomada en cuenta para futuras liberaciones de la aplicación por la comunidad.

Capitulo 4

4.1 Objetivos

A continuación se va a llevar a cabo un breve análisis de los objetivos planteados, para intentar verificar su cumplimiento o no por parte del presente proyecto, para esto se hará uso de una escala compuesta por los valores: Realizado, Parcialmente Realizado, No Realizado. Para iniciar esta autoevaluación, se procederá a revisar en primera instancia los objetivos específicos:

- **Objetivo:** Determinar la licencia apropiada para el proyecto a desarrollar.
  - **Valoración:** Realizado
  - **Comentarios:** El licenciamiento del proyecto es dual y permite de momento su uso por parte de proyectos GPL y no GPL aptos.

- **Objetivo:** Seleccionar un nombre para la aplicación a realizar
  - **Valoración:** Realizado
  - **Comentarios:** Aunque no se nombra en el presente proyecto, el mismo se cumplió mediante el análisis de posibles nombres en la aplicación web Google Trends (http://www.google.com/trends)

- **Objetivo:** Seleccionar un ambiente de desarrollo de la aplicación, propicio para la creación de una comunidad de software libre alrededor del proyecto.
  - **Valoración:** Realizado
  - **Comentarios:** El ambiente seleccionado fue sourceforge.net

- **Objetivo:** Configurar de forma personalizada el ambiente de desarrollo seleccionado para generar los espacios de intercambios de ideas y comunicación del proyecto.
  - **Valoración:** Realizado Parcialmente
  - **Comentarios:** La configuración del sitio web fue realizada de forma parcial y aún quedan muchas cosas por realizar, no tanto la configuración de los espacios que ya existen, sino ubicar la forma y manera correcta para que los usuarios hagan uso del mismo, ya que hasta el momento únicamente se han limitado a enviar correos personales.

- **Objetivo:** Promocionar el proyecto en búsqueda de la creación de una base de usuarios que sirva para brindarle una comunidad al mismo.
    - **Valoración:** Realizado
    - **Comentarios:** Esto se realizado al anunciarlo en foros, listas de correo, etc. Ya se cuenta con una cantidad de aproximadamente 8 personas que han manifestado su voluntad de colaborar en lo que sea posible (aunque por el momento no han desarrollado sino solo reportado errores) y actualmente dos de ellas se han unido al proyecto ya como miembros y se encuentran preparándose en la filosofía del software libre, para asumir tareas a ser asignadas próximamente. Igualmente se está pensando en anunciar el proyecto luego de que madure un poco más en publicidad por internet al estilo google adwords.

- **Objetivo:** Seleccionar las funcionalidades básicas que debería brindar el software a desarrollar.
    - **Valoración:** Realizado
    - **Comentarios:** Se baso en los puntos cruciales para el éxito de un proyecto de software libre, del artículo propuesto por A. Senyard y M. Michlmayr[8].

- **Objetivo:** Seleccionar el tipo de metodología a usar para el desarrollo del proyecto.
    - **Valoración:** Realizado.
    - **Comentarios:** Se baso en las fases propuestas para el éxito de un proyecto de software libre, del artículo escrito por A. Senyard y M. Michlmayr[8].

- **Objetivo:** Diseñar la jerarquía de clases necesarias obtener la aplicación propuesta.
    - **Valoración:** Realizado
    - **Comentarios:** Se comprobó al obtener la primera liberación, ya que esta cumplía con las funcionalidades básicas propuestas.

- **Objetivo:** Liberar la primera versión funcional del proyecto.
    - **Valoración:** Realizado.
    - **Comentarios:** Se libero en la página web sourceforge.net, teniendo el siguiente enlace: http://sourceforge.net/projects/vsqlbuilder. Actualmente acaba de ser aprobado otro sitio más especializado en el área, específicamente en proyectos para la base de datos postgreSQL, que será usado como espejo de sourceforge.net, para captar voluntarios: http://vsqlbuilder.projects.postgresql.org/

- **Objetivo:** Generar la documentación de usuario y desarrollador del software.
    - **Valoración:** Realizado parcialmente
    - **Comentarios:** Solo se documento parcialmente el código fuente (dentro del mismo con comentarios), tomando como base para dejar un poco a un lado este proceso el hecho de que este punto no era considerado un factor determinante para conseguir el éxito de un proyecto de software libre, según el artículo escrito por A. Senyard y M. Michlmayr[8].

Por lo que una vez discutidos los objetivos específicos, se puede analizar el objetivo general:

❖ **Objetivo General:** Construir un software que facilite la escritura y ejecución de sentencias SQL de forma Visual para la base de datos postgreSQL en un ambiente de desarrollo de software libre.

- **Valoración:** Realizado

- **Comentarios:** Esto fue posible solo después de comprender que partiendo de un código cerrado se puede obtener un código abierto que dé lugar a un proyecto de software libre exitoso, porque de lo contrario con el poco tiempo con que se contaba para la realizar el presente proyecto no hubiese sido posible llevar a cabo el mismo, tal cual y como estaba planteado.

4.2 Futuras ampliaciones

Una vez finalizado el presente proyecto, solo queda por observar el futuro del mismo y hacia donde se dirigirá el esfuerzo de la comunidad en las fechas por venir, pero como toda estimación de algo que aún no se conoce, esta puede contener errores o no incluir algunos importantes, pero obviando esto, se podría resumir lo que se llevará a cabo en varios puntos concretos:

- Se terminará de implementar todas las funcionalidades aún no implementadas en el software liberado.

- Se mantendrá el mantenimiento en el proyecto, así como las pruebas para obtener cada con el paso del tiempo un software de gran calidad.

- Se mejorará la comunidad entorno al software y se llevará a cabo un desarrollo más comunitario en un futuro próximo.

- Se incorporaran las ideas propuestas por algunas personas, como la inclusión de la edición de los datos recuperados por la base de datos, o posibilidad de crear tablas dentro de la misma aplicación.

- Es probable que en un futuro se amplié la gama de base de datos a las cuales se pueda conectar el software para incluir otras como mysql, oracle, etc.

4.2 Conclusiones

Durante la realización de este proyecto se logro la consolidación un proyecto de software libre, que probablemente tenga un éxito mediano en un futuro, y luego de analizar este proceso desde la perspectiva de una persona que no había tenido experiencia real en la elaboración de proyectos de software libre, se puede afirmar con propiedad que si bien no existe una receta mágica que ayude a crear un proyecto exitoso de software libre, existen recomendaciones y objetivos implícitos dentro de este proceso, que deben ser cubiertos para que se pueda lograr el objetivo final deseado de crear un proyecto que al menos motive la participación de personas a su alrededor.

El desarrollo de este proyecto implico la realización de un gran esfuerzo en la parte de diseño, y sobre todo de dedicación y motivación para conseguir respuestas a todas las dudas que fueron surgiendo a medida que el proyecto avanzaba, pero la existencia listas de correos, foros y wikis en otros proyectos, hicieron posible consultar a personas con experiencia en el área, las cuales siempre guiaron de forma correcta el proyecto, y que aunque nunca participaron en el mismo directamente, el resultado que se observa depende directamente de sus ideas y aportaciones.

Igualmente a medida que el proyecto avanza, la aparición del apoyo, por parte de una comunidad de personas dispuestas a colaborar con el proyecto, crea una motivación que no existe al inicio del mismo, ya que hace ver que la herramienta desarrollada tiene un significado de pertenencia no solamente para la persona que la desarrolla, sino también para aquellos otros que se han interesado en la misma y que desean colaborar con ella.

Referencias bibliográficas

[1] Mitrovic, Antonija. (1998). **Learning SQL with a computerized tutor**. *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education.* Páginas 307 – 311. Atlanta, Georgia, Estados Unidos.

[2] Doll, Shelley. (2002). **Is SQL a standard anymore?**. (HTML), Visitado el 30-11-2007. <http://articles.techrepublic.com.com/5100-22-1046268.html>

[3] Stephenson, Neal. (1999). **In the Beginning...Was the Command Line**. HarperCollins, Canada.

[4] C.F. Kemer, (1992). **Now the learning curve affects CASE tool adoption**. Software IEEE, Volume 9, Issue 3. Páginas 23-28.

[5] T. Fellmann, M. Kavakli (2007). **A Command Line Interface versus a Graphical User Interface in Coding VR Systems**. Human-Computer Interaction HCI 2007. Chamonix, Francia.

[6] Y. Tao (2005). **Developing Usable GUI Applications with Early Usability Evaluation Software Engineering**, Peter Kokol. Insbruck. Austria

[7] Eric S. Raymond. **La catedral y el bazar.** (HTML) Visitado el 03-01-2008. <http://catb.org/~esr/writings/cathedral-bazaar/>

[8] A. Senyard, M. Michlmayr (2004). **How to Have a Successful Free Software Project.** Proceedings of the 11th Asia-Pacific Software Engineering Conference. Melbourne, Victoria, Australia.

[9] Wikipedia. (2003). **Programación Orientada a objetos.** (HTML), Visitado 05-01-2008. <http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos>

[10] Wikipedia. (2003). **Licencia de Software**. (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Licencia_de_software

[11] M. Bain, M. Gallego, M. Martinez, J. Rius. (2004). **Aspectos Legales y de explotación del software libre.** UOC, Barcelona España.

[12] Wikipedia. (2003). **Control de Versiones**. (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Control_de_versiones

[13] Wikipedia. (2003). **CVS**.(HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/CVS

[14] Wikipedia. (2003). **Subversions**. (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Subversions

[15] Wikipedia. (2003). **Lista de Correo.** (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Lista_de_correo

[16] Wikipedia. (2003). **Foro de Internet.** (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Foro_(Internet)

[17] Wikipedia. (2003). **Wiki.** (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Wiki

[18] Wikipedia. (2003). **Sistema de seguimiento de errores.** (HTML), Visitado 05-01-2008. http://es.wikipedia.org/wiki/Sistema_de_seguimiento_de_errores

[19] Wikipedia. (2003). **Lenguaje de Programación Java.** (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java>

[20] Wikipedia. (2004). **Eclipse (Software)**. (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/Eclipse_(software)>

[21] Wikipedia. (2007). **Umbrello**. (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/Umbrello>

[22] Wikipedia. (2003). **GNU/Linux**. (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/GNU/Linux>

[23] Wikipedia. (2002). **GIMP**. (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/GIMP>

[24] Wikipedia. (2004). **Inkscape**. (HTML), Visitado el 04-12-2007. <http://es.wikipedia.org/wiki/Inkscape>

[25] P. Krutchten .(2000). **The Rational Unified Process: An Introduction (2nd Edition).** Addison-Wesley Professional;

[26]Free Software Foundation (2007). **Licenses Lists**. (HTML). Visitado 13-10-2007. <http://www.gnu.org/licenses/license-list.html>

[27] Open Source Initiative (2007). **Licenses Alphabetical.** (HTML). Visitado 14-10-2007. < http://www.opensource.org/licenses/alphabetical>

[28] Wikipedia. (2004). **License Proliferation**. (HTML), Visitado el 17-10-2007. http://en.wikipedia.org/wiki/License_proliferation

[29]ipinfoblog (2005). **Licensing law issues, open source license proliferation a broader view.** (HTML). Visitado 23-10-2007. <http://www.ipinfoblog.com/archives/licensing-law-issues-open-source-license-proliferation-a-broader-view.htm>

[30] Open Source Initiative (2006). **Report of License Proliferation Committee.** (HTML). Visitado 19-10-2007. <http://www.opensource.org/lpc>

[31] Wikipedia. (2004). **Comparison of free software licences**. (HTML), Visitado el 17-10-2007. http://en.wikipedia.org/wiki/Comparison_of_free_software_licences

Anexos

1. Requisitos de la aplicación

Para la realización del presente proyecto de desarrollo de un software capaz de ayudar en la construcción de sentencias SQL de manera Visual, se pueden tener como posibles condiciones del mismo:

1. **Requisitos del usuario:** Las necesidades que han sido captadas inicialmente para que un usuario haga uso del software que será desarrollado por el presente proyecto son: un computador personal computador personal con al menos 64 MB de memoria RAM, instalado con un sistema operativo que permita correr el entorno de ejecución de aplicaciones java (jre, por sus siglas en ingles) en su versión 1.5 o posterior de forma gráfica y que cuente con los dispositivos de entrada de datos apropiados, aunque esta configuración puede ser dependiente de la arquitectura y sistema operativo seleccionado.
2. **Requisitos de Software y Hardware:** Para la realización del proyecto se necesita contar con una plataforma de hardware y software apropiada para la ejecución del software seleccionado (jre 1.5 o superior, eclipse 3 o superior, etc.) como lo es un computador personal con 1 GB de memoria RAM al menos, la cual ejecutara el sistema operativo Ubuntu 7.10, donde se desarrollara la aplicación interactuando con la forja seleccionada para el proyecto, por lo que también será de vital importancia el contar con una conexión a Internet confiable y de velocidad aceptable.

2. Glosario

- ***Metadatos:*** El término «metadatos» no tiene una definición única. Según la definición más difundida metadatos son «datos sobre datos». También hay muchas declaraciones como «informaciones sobre datos» , «datos sobre informaciones» e «informaciones sobre informaciones».

- ***GNU:*** es un acrónimo recursivo que significa GNU No es Unix (GNU is Not Unix). Puesto que en inglés "gnu" (en español "ñu") se pronuncia igual que "new", Richard Stallman recomienda pronunciarlo "guh-noo". En español, se recomienda pronunciarlo fonéticamente; por ello, el término mayoritariamente se deletrea (G-N-U).

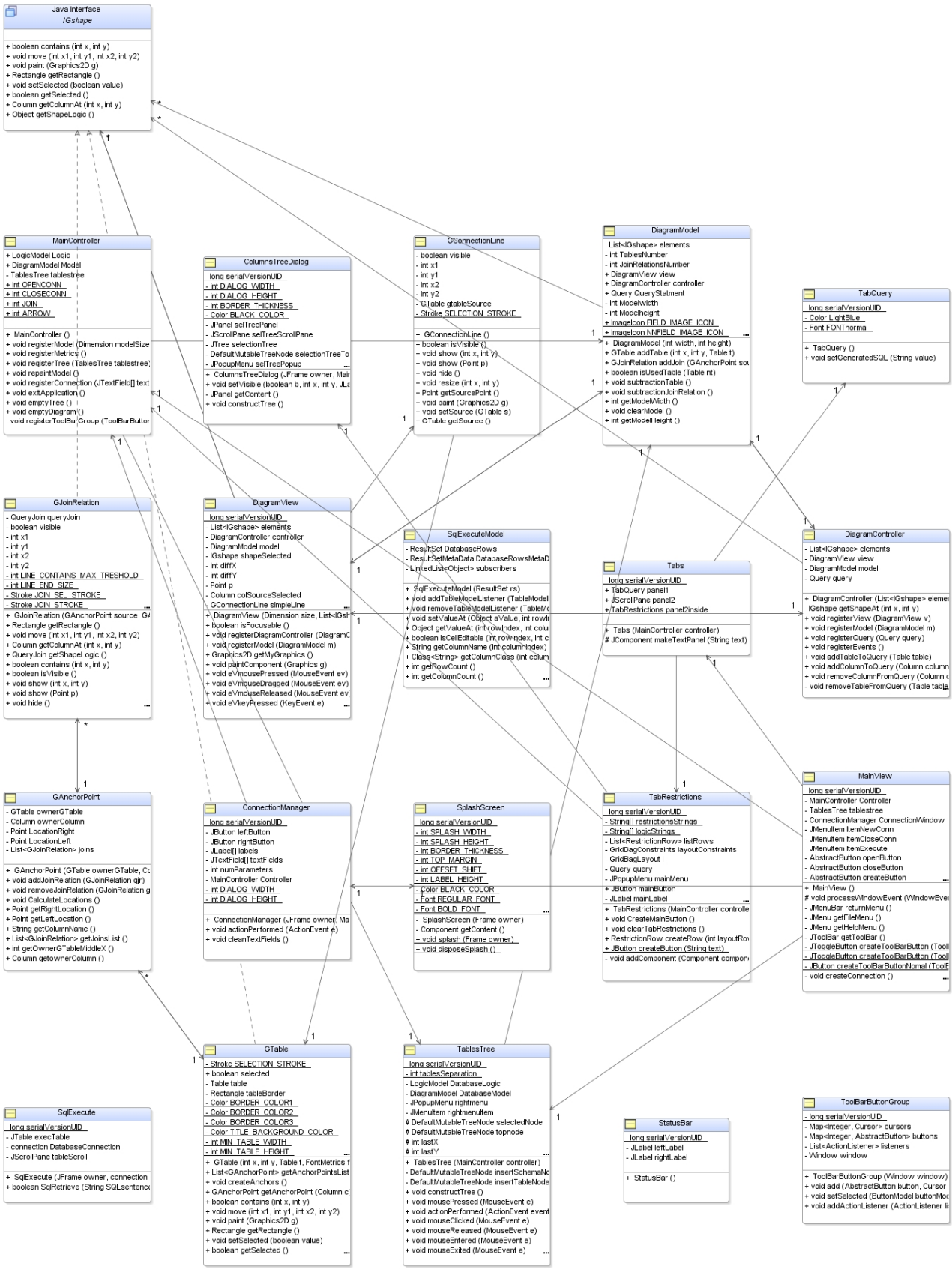
- ***Linux:*** se refiere estrictamente al núcleo Linux, pero es comúnmente utilizado para describir al sistema operativo tipo Unix (que implementa el estándar POSIX), que utiliza primordialmente filosofía y metodologías libres (también conocido como GNU/Linux) y que está formado mediante la combinación del núcleo Linux con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software (libre o no libre).

- ***Kernel:*** En informática, el núcleo (también conocido en español con el anglicismo kernel, de raíces germánicas como kern) es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

- ***Forja:*** La **forja** es el arte y el lugar de trabajo del forjador o herrero, cuyo trabajo consiste en dar forma al metal por medio del fuego y del martillo. En términos informáticos es el lugar donde se da origen a un software.

3. Diagramas de clases detallados de la aplicación

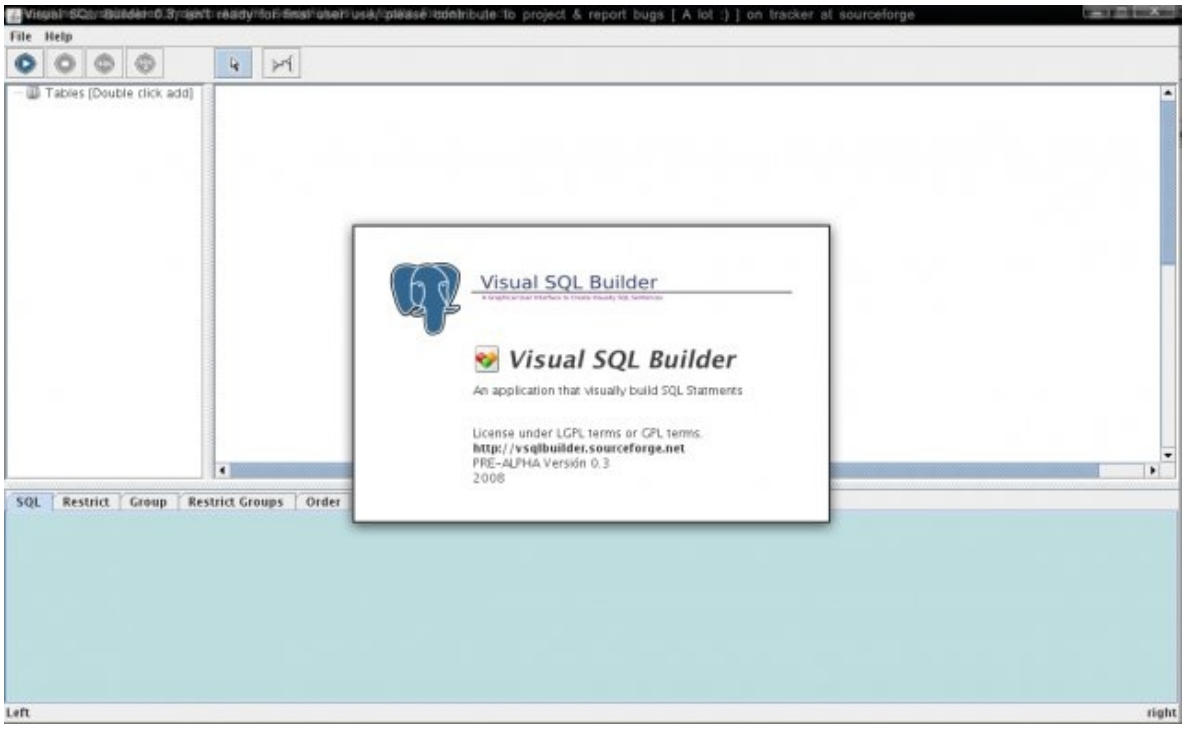
3.1 Paquete vsqlbuilder.gui



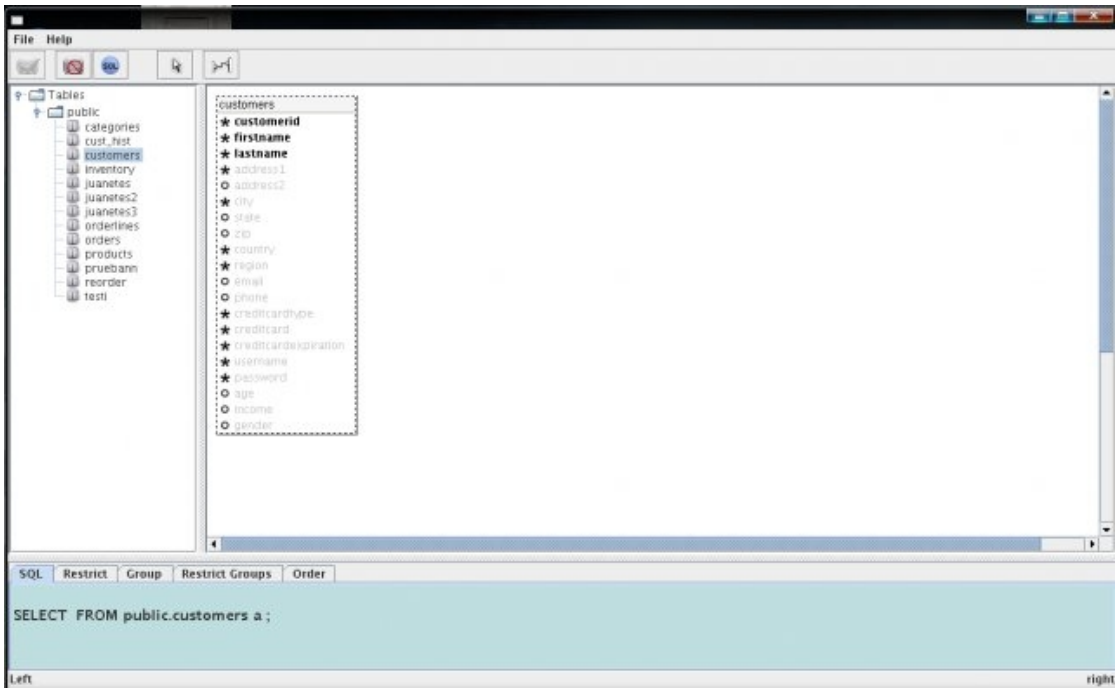
3.1 Paquete vsqlbuilder.logic

4. Capturas de pantallas de la aplicación Visual SQL Builder en funcionamiento

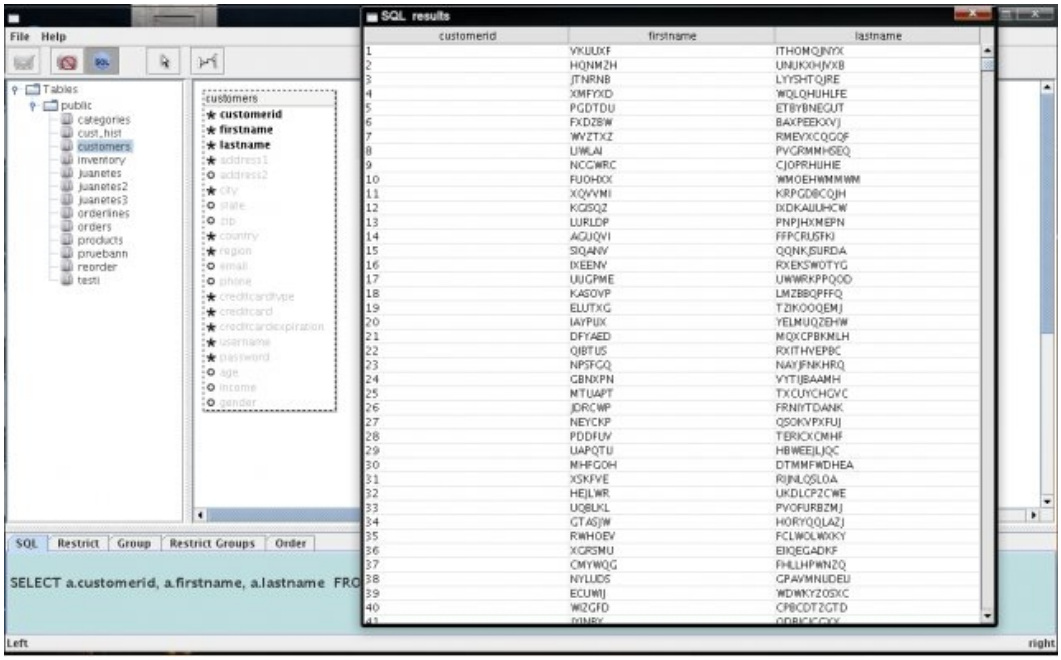
4.1 Pantalla de Inicio de la aplicación



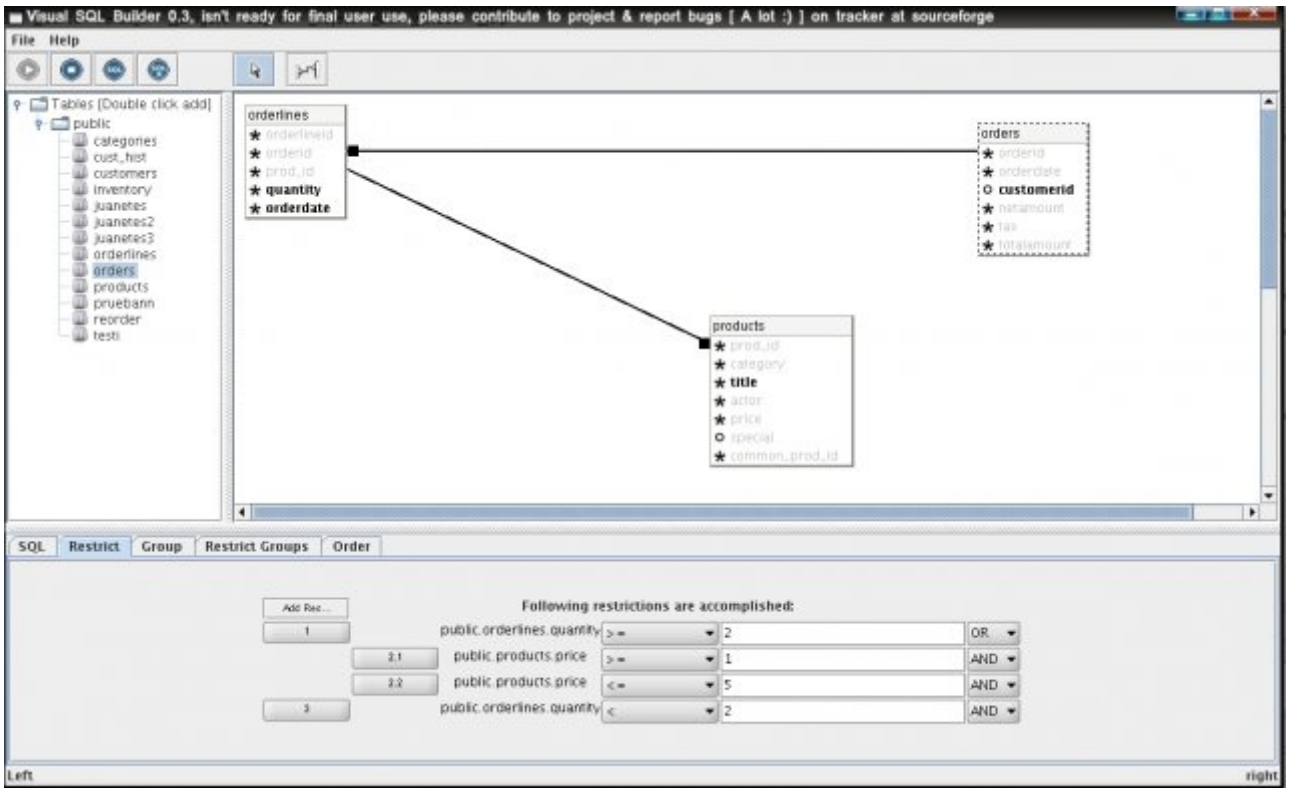
4.2 Selección de tres columnas en la tabla customers



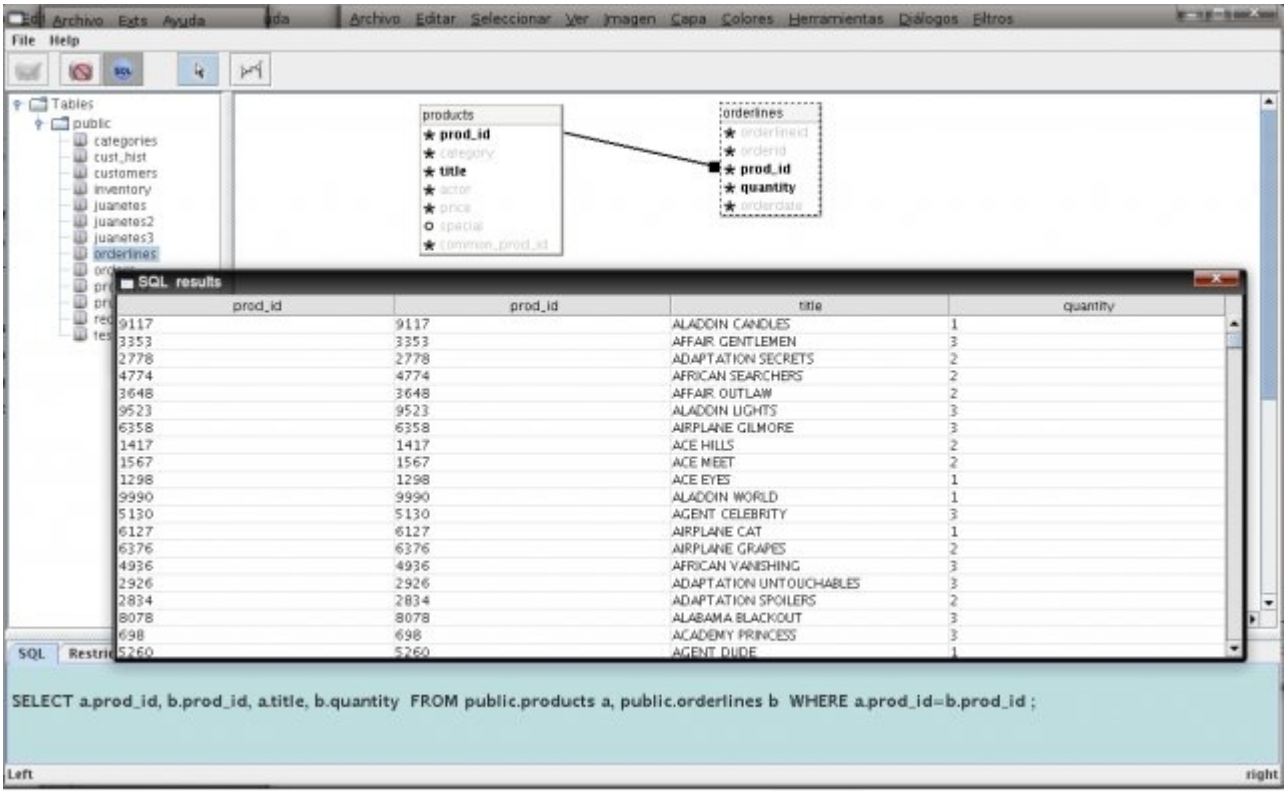
4.3 Ejecución de sentencia SQL Simple



4.4 Sentencia SQL con datos provenientes de varias tablas y filtrado de filas por valores



4.5 Consulta de unión (join) entre dos tablas y su resultado



5. LICENCIA GPL v2

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it,

either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable

source code, which must be distributed under the terms of Sections
1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three
years, to give any third party, for a charge no more than your
cost of physically performing source distribution, a complete
machine-readable copy of the corresponding source code, to be
distributed under the terms of Sections 1 and 2 above on a medium
customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer
to distribute corresponding source code. (This alternative is
allowed only for noncommercial distribution and only if you
received the program in object code or executable form with such
an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it. For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable. However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License. Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

5. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Program or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions. You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all. For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

6. LICENCIA LGPL v2

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser

General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated

interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work

during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or

distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE

LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!