Diseño e implementación de una base de datos relacional para la gestión de un videojuego

Diego Saavedra Jaén

MEMORIA

Segundo ciclo de Ingeniería en Informática

Consultor Juan

Índice de contenido

1. Introducción.

Este documento, pretende mostrar la memoria del proyecto fin de carrera de la ingeniería superior en informática para la especialidad en Bases de datos. Así mismo, se mostrarán algunos apartados del Plan de Trabajo (PT) realizado durante el proyecto , con su planificación diseño e implementación de una base de datos relacional.
Básicamente se pretende desarrollar la construcción de una base de datos que gestione todos los procesos de un Videojuegode un videojuego y la creación de un Datawarehause DWH que pueda analizar los datos almacenados.

Nuestro videojuego estará disponible tanto en plataformas de móvil como webs y será una plataforma multiusuarios.

Objetivos

Se consideran los siguientes objetivos generales del proyecto.

- Detectar necesidades básicas del sistema a implementar para la gestión de la base de datos.
- Detectar posibles funcionalidades a añadir que aporten valor al proyecto.
- Proponer un diseño adecuado y ajustado fielmente a las necesidades y requerimientos.
- Implementar un sistema que encapsule las funciones de acceso a los datos y sea un sistema escalable abierto y flexible a futuras ampliaciones o mejoras.

Alcance

En el apartado de requerimientos se realiza un análisis mas detallado del alcance del proyecto.
Es muy importante definir un buen alcance cerrado en el que todas las partes estén de acuerdo, se intentará definir qué entra y qué no entra en el alcance.

- Diseño Base de Datos relacional(Conceptual y E/R).
- Instalación Modelo de datos del diseño relacional.
- Desarrollo/Instalación de procesos necesarios para la gestión del Videojuego.
- Diseño/desarrollo/Instalación de almacén de datos para obtención de Estadísticas data warehouse(DWH).
- Desarrollo/instalación procesos de log, recuperación, backups y estadísticas.
- Instalación y Gestión SW Oracle/BI.

Metodología a seguir

Se ha optado finalmente por usar una metodología con ciclo de vida en cascada que se intentará adaptar al manual de buenas prácticas PMBOK(Project Management Body of Knowledge).
Esta metodología se desarrolla en varias fases y los productos conseguidos en una fase forman parte imprescindible para que la fase siguiente continúe su desarrollo.

Toma de Requisitos y Análisis de los mismos: Se analiza exhaustivamente los requerimientos y necesidades descritas por el cliente para realizar un estudio detallado de los requerimientos. Se ha de involucrar en lo máximo posible al cliente ya que es crucial la obtención de información de parte del mismo y así poder realizar unos buenos requisitos para poder definir un mejor alcance del proyecto y sus necesidades.

Análisis funcional y Diseño Técnico**:** Una vez están claros y cerrado los requisitos, se pasa a realizar un análisis funcional donde se definen nuevamente los requisitos a desarrollar y el alcance del proyecto. También se realiza un diseño técnico donde se podrá documentar con mas detalle un diagrama conceptual, E/R, lógico y físico de la Base de datos a implementar y sus procesos.

Instalación SW e Implementación del producto**:** En esta fase primero se instala el SW necesario para la ejecución del proyecto tales como el Sistema Generador de Bases de Datos (SGDB), alguna aplicación de Business Intelligence (BI) y se implementa el código a desarrollar bien sea código PL/SQL, scripts, C, ProC o JAVA.

Testing y prueba**:** Se define un plan de pruebas y se realizan pruebas unitarias, de volumen. Se testean todos los casos posibles susceptibles de causas de errores. Esta fase permitirá realizar cambios en el código bien para su optimización o corrección de errores detectados en la fase de pruebas.

Mantenimiento**:** Una vez la aplicación ha pasado a estado "Live" o producción, se hacen los cambios necesarios para corregir errores o para adaptarlo a nuevas necesidades. Es de vital importancia que en la fase de diseño se haya pensado en un producto escalable que pueda crecer, mejorar sin tener que realizar muchos cambios en la estructura principal.

2. Planificación.

2.1 Hitos y Objetivos del proyecto.

La siguiente tabla muestra los hitos y entregas mas importantes definidos en el proyecto.

Hitos	Fin
PAC1_PT_PLANIFICACION	**lun 07/10/13**
PAC2 Ejecución, Seguimiento y Control	**lun 11/11/13**
PAC3 Ejecución, Seguimiento y Control	**mié 11/12/13**
Entrega Final	**lun 13/01/14**
Preparación Tribunal Fin de PFC	vie 24/01/14

- Dentro del primer hito "Planificación y PT" destacamos como objetivo importante realizar la planificación o project que se irá modificando a lo largo del proyecto.
- El segundo hito será la entrega de la PAC2 donde se destacan como objetivos importantes la toma de requerimientos y su análisis (Muy importante para definir el alcance), el diseño técnico y conceptual de la BD y los sistemas

satélites, La instalación del SW necesario y el desarrollo de los primeros procesos del proyecto.

- Como tercer hito se tiene la PAC3 donde se puede destacar como objetivos tangibles que aportan valor al proyecto, los distintos desarrollos y procesos de la BD, el diseño de procesos ETL y el data-mart de cara a gestionar el almacén de datos del DWH y la Realización de análisis OLAP y por último la realización de un plan de pruebas y su ejecución final.
- La preparación de la Memoria para la entrega final será el hito mas importante donde el objetivo es llegar a este punto con un proyecto maduro donde el SW funcione correctamente, se hayan realizado las pruebas diseñadas en el plan de prueba exitosamente de todos los procesos desarrollados. Los objetivos por tanto una vez se considera un producto maduro, serán acabar la documentación del proyecto en la memoria requerida, documentar y preparar la presentación.
- Por último, tenemos el hito de tribunal final.

2.2 Tareas PAC1

Nombre de tarea	Duración	Comienzo	Fin
Lectura Documentación PFC	1 día	mar 01/10/13	mar 01/10/13
Descarga Material/Repaso Documentación	1 día	mié 02/10/13	mié 02/10/13
Elaborar Planificación	**2 días**	**jue 03/10/13**	**vie 04/10/13**
Definición de Alcance y Objetivos	1 día	sáb 05/10/13	sáb 05/10/13
Búsqueda de Metodología a desarrollar	1 día	dom 06/10/13	dom 06/10/13
Análisis de Riesgos	1 día	lun 07/10/13	lun 07/10/13
Recursos y Presupuesto	1 día	lun 07/10/13	lun 07/10/13

Subtareas "Elaborar Planificación"

Nombre de tarea	Duración	Comienzo	Fin
Identificación de Tareas	1 día	jue 03/10/13	jue 03/10/13
Diagrama de Gant	1 día	vie 04/10/13	vie 04/10/13
Hitos	1 día	vie 04/10/13	vie 04/10/13

2.3 Tareas PAC2

Nombre de tarea	Duración	Comienzo	Fin
Análisis de los Requisitos	**5 días**	**mar 08/10/13**	**sáb 12/10/13**
Análisis Funcional	4 días	lun 14/10/13	jue 17/10/13
Diseño Técnico	**8 días**	**vie 18/10/13**	**mar 29/10/13**
Instalación e Implementación	**9 días**	**mié 30/10/13**	**lun 11/11/13**

Subtareas "Análisis de los Requisitos"

Nombre de tarea	Duración	Comienzo	Fin
Análisis detallado de los requisitos iniciales	2 días	mar 08/10/13	mié 09/10/13
Resolución de Dudas con Cliente	2 días	jue 10/10/13	vie 11/10/13
Update Requisitos	1 día	sáb 12/10/13	sáb 12/10/13

Subtareas "Diseño Técnico"

Nombre de tarea	Duración	Comienzo	Fin
Diseño Diagrama E/R	4 días	vie 18/10/13	mié 23/10/13
Diseño de Procesos de BD	2 días	jue 24/10/13	vie 25/10/13
Diseño Procesos DWH	2 días	lun 28/10/13	mar 29/10/13

Subtareas "Instalación e Implementación"

Nombre de tarea	Duración	Comienzo	Fin
Instalación SW(Oracle, PLSQL/Developper, DWH)	1 día	mié 30/10/13	mié 30/10/13
Creación Base de Datos.	2 días	jue 31/10/13	vie 01/11/13
Creación Procesos DB parte 1.	5 días	lun 04/11/13	vie 08/11/13
Documentación Procesos y Modelo de datos	1 día	lun 11/11/13	lun 11/11/13

2.4 Tareas PAC3

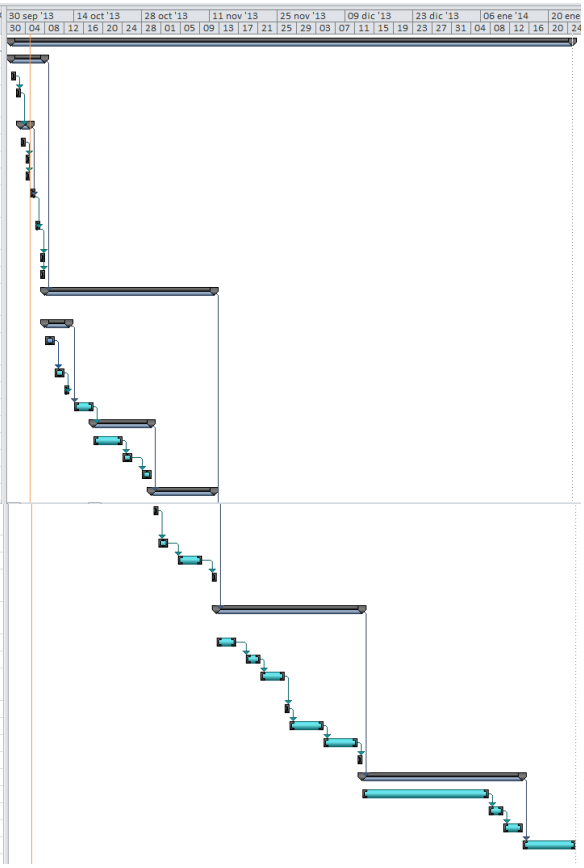
Nombre de tarea	Duración	Comienzo	Fin
Creación Procesos DB parte 2.	4 días	mar 12/11/13	vie 15/11/13
Creación Modelo datos DWH	3 días	lun 18/11/13	mié 20/11/13
Creación Procesos ETL y Análisis OLAP	3 días	jue 21/11/13	lun 25/11/13
Diseño Plan de Pruebas	1 día	mar 26/11/13	mar 26/11/13
Pruebas Unitarias	5 días	mié 27/11/13	mar 03/12/13
Pruebas Integradas	5 días	mié 04/12/13	mar 10/12/13
Documentación PAC3	1 día	mié 11/12/13	mié 11/12/13

2.5 Tareas Entrega Final

Nombre de tarea	Duración	Comienzo	Fin
Memoria	18 días	jue 12/12/13	lun 06/01/14
Presentación	3 días	mar 07/01/14	jue 09/01/14
Entrega SW	2 días	vie 10/01/14	lun 13/01/14

2.6 DIAGRAMA DE GANNT DEL PROYECTO.

Mod de	Nombre de tarea	Duración	Comienzo	Fin	Predecesi
	PFC-BASES_DE_DATOS	**84 días**	**mar 01/10/13**	**vie 24/01/14**	
	PAC1_PT_PLANIFICACION	**5 días**	**mar 01/10/13**	**lun 07/10/13**	
	Lectura Documentación PFC	1 día	mar 01/10/13	mar 01/10/13	
	Descarga Material/Repaso Documentacion	1 día	mié 02/10/13	mié 02/10/13	3
	Elaborar Planificación	**2 días**	**jue 03/10/13**	**vie 04/10/13**	**4**
	Identificación de Tareas	1 día	jue 03/10/13	jue 03/10/13	
	Diagrama de Gant	1 día	vie 04/10/13	vie 04/10/13	6
	Hitos	1 día	vie 04/10/13	vie 04/10/13	6
	Definición de Alcance y Objetivos	1 día	sáb 05/10/13	sáb 05/10/13	5
	Búsqueda de Metodología a desarrollar	1 día	dom 06/10/13	dom 06/10/13	9
	Análisis de Riesgos	1 día	lun 07/10/13	lun 07/10/13	10
	Recursos y Presupuesto	1 día	lun 07/10/13	lun 07/10/13	10
	PAC2 Ejecución, Seguimiento y Control	**25 días**	**mar 08/10/13**	**lun 11/11/13**	**2**
	Análisis de los Requisitos	**5 días**	**mar 08/10/13**	**sáb 12/10/13**	
	Análisis detallado de los requisitos iniciales	2 días	mar 08/10/13	mié 09/10/13	
	Resolución de Dudas con Client	2 días	jue 10/10/13	vie 11/10/13	15
	Update Requisitos	1 día	sáb 12/10/13	sáb 12/10/13	16
	Análisis Funcional	4 días	lun 14/10/13	jue 17/10/13	14
	Diseño Técnico	**8 días**	**vie 18/10/13**	**mar 29/10/13**	**18**
	Diseño Diagrama E/R	4 días	vie 18/10/13	mié 23/10/13	
	Diseño de Procesos de BD	2 días	jue 24/10/13	vie 25/10/13	20
	Diseño Procesos DWH	2 días	lun 28/10/13	mar 29/10/13	21
	Instalación e Implementación	**9 días**	**mié 30/10/13**	**lun 11/11/13**	**19**
	Instalación SW(Oracle, PLSQL/Developper, DWH)	1 día	mié 30/10/13	mié 30/10/13	
	Creación Base de Datos.	2 días	jue 31/10/13	vie 01/11/13	24
	Creación Procesos DB parte 1.	5 días	lun 04/11/13	vie 08/11/13	25
	Documentación Procesos y Modelo de datos	1 día	lun 11/11/13	lun 11/11/13	26
	PAC3 Ejecución, Seguimiento y Control	**22 días**	**mar 12/11/13**	**mié 11/12/13**	**13**
	Creación Procesos DB parte 2.	4 días	mar 12/11/13	vie 15/11/13	
	Creación Modelo datos DWH	3 días	lun 18/11/13	mié 20/11/13	29
	Creación Procesos ETL y Análisis OLAP	3 días	jue 21/11/13	lun 25/11/13	30
	Diseño Plan de Pruebas	1 día	mar 26/11/13	mar 26/11/13	31
	Pruebas Unitarias	5 días	mié 27/11/13	mar 03/12/13	32
	Pruebas Integradas	5 días	mié 04/12/13	mar 10/12/13	33
	Documentación PAC3	1 día	mié 11/12/13	mié 11/12/13	34
	Entrega Final	**23 días**	**jue 12/12/13**	**lun 13/01/14**	**28**
	Memoria	18 días	jue 12/12/13	lun 06/01/14	
	Presentación	3 días	mar 07/01/14	jue 09/01/14	37
	Entrega SW	2 días	vie 10/01/14	lun 13/01/14	38
	Preparación Tribunal Fin de PFC	9 días	mar 14/01/14	vie 24/01/14	36



3. Análisis y plan de contingencia de los Riesgos del proyecto.

En este apartado se explican los riesgos detectados en el proyecto y las posibles soluciones o contingencias a realizar en caso de que se produzcan.

Riesgos Técnicos: Inicialmente se pretende desarrollar el proyecto en un portátil que puede sufrir, pérdida de datos, borrado del sistema y siniestro total del PC.

Contingencia => Como contingencia a la pérdida de datos o cualquier problema técnico derivado de la misma y los riesgos definidos, se ha pensado en Varias Opciones, realizar backup diario de los datos en una carpeta de almacenamiento remota como puede ser DropBox o UbuntuOne. Adicionalmente se ha penado en crear una máquina virtual con 2 GB de RAM suficiente para la ejecución de Oracle 11 y realizar copias semanales de la máquina virtual.

Riesgos en la planificación: Se pueden dar riesgos y cambios en la planificación que pueden venir originados por viajes de trabajo, vacaciones, cambios en requisitos si no se ha cerrado bien el alcance, cambios debido a que una tarea ha tardado mas de lo planificado.

Contingencia => Como contingencia riesgos de modificación en la planificación, se ha pensado en trabajar los fines de semana "Viernes y Sábado en Middle East", llevar un trabajo constante y diario de entre 2 y 3 horas. También se tiene en cuenta realizar una planificación mas realistas en caso de retrasos y reajustar el calendario.

Riesgos por motivos personales: Un viaje imprevisto por motivos de trabajo, una enfermedad, carga de trabajo excesiva y realización de trabajos nocturnos.

Contingencia => Si se lleva un ritmo de trabajo constante, este riesgo se minimiza, pero en caso de darse de manera constante, se pediría una ampliación en la planificación de manera justificada.

4. Recursos y presupuesto.

En este apartado se sa una visión general de los recursos que necesarios para realizar el proyecto, haremos mención tanto de los recursos humanos como de los recursos técnicos.
También se incluye una valoración económica o presupuesto dando una estimación inicial del coste que puede suponer un proyecto como el que se va a desarrollar.

4.1 Recursos Humanos

Aunque el PFC se realiza de forma individual, se intenta dar una idea de los recursos que se necesitarían para realizar el proyecto que serían los siguientes:

- 1 Jefe de Proyectos o Project Manager. Realizaría las funciones de inicio del proyecto kick off, planificación, reuniones con el cliente para involucrarle en el proyecto y toma de requisitos, seguimiento y control, apoyo a los analistas en la toma de requisitos y la definición del alcance, realización de métricas para poder medir la evolución del proyecto, También se encargará de la fase de finalización del proyecto.

- 1 Analista. Realizará las funciones de análisis de los requisitos, coordinación directa con el Jefe de Proyectos, dará una estimación del coste que supone en tiempo cada tarea a desarrollar o analizar, se encargará de toda la documentación técnica y deberá coordinarse con el Analista Programador.

- 1 Analista Programador. Se encargará de realizar la programación del código SQL para los procesos a diseñar. También se encargará de realizar análisis técnicos para optimizar la BD e implementar los procesos de

estadísticas. Realizará también funciones de instalación de SW y testing de los productos desarrollados.

4.2 Recursos Hardware(HW) y Software

Actualmente se dispone de un portátil Intel Core i5-2430M CPU @ 2.40Ghz 2.40Ghz, 4GB RAM, Windows 7 64 Bit y se está planteando la posibilidad de crear una máquina virtual específica para el proyecto con Windows Vista o Linux Ubuntu.

Se considera el siguiente Software(SW) inicial para empezar el proyecto:

- Virtual Box para instalar la máquina Virtual donde se instalará windows Vista o bien Linux Ubuntu.
- Base de Datos Oracle o SGDB(Sistema gestor de Base de Datos). Se trabajará con la versión 11g XE (Express Edition) de Oracle o la versión 10 g XE en función de si se decide realizar el proyecto en Linux o Windows.
- PL/SQL Developer. Herramienta de acceso a la base de datos donde se podrá no solo programar el código de la BD sino que además ofrece una visión gráfica y visual de la BD, con todos sus componentes, triggers, procedure, function, package, package Body, queue, etc
- Magic Draw , herramienta de diseño UML que servirá para realizar el Diseño del Diagrama E/R.
- Microsoft Office 2010 (Word, Power Point, Excel y Project)

4.3 Presupuesto

Se estima realizar una media a dedicar entre 16 y 18 horas a la semana. Si consideramos las semanas laborales de 5 días de Lunes a Viernes , se obtiene una media de 3,2 horas/jornada y 3,6 horas/jornada.

Teniendo en cuenta las tareas que va a realizar cada recurso y si contamos los días que dedica de cada uno de ellos a la lista de tareas, se obtiene la siguiente tabla:

Nombre de tarea	Duración	Nombres de los recursos
Lectura Documentación PFC	1 día	**JP**
Descarga Material/Repaso Documentacion	1 día	**JP**
Identificación de Tareas	1 día	**JP**
Diagrama de Gant	1 día	**JP**
Hitos	1 día	**JP**

Definición de Alcance y Objetivos	1 día	**JP**
Búsqueda de Metodología a desarrollar	1 día	**JP**
Análisis de Riesgos	1 día	**JP**
Recursos y Presupuesto	1 día	**JP**
Análisis detallado de los requisitos iniciales	2 días	**Analista**
Resolución de Dudas con Cliente	2 días	**Analista**
Update Requisitos	1 día	**Analista**
Análisis Funcional	4 días	**Analista**
Diseño Diagrama E/R	4 días	**Analista**
Diseño de Procesos de BD	2 días	**Analista**
Diseño Procesos DWH	2 días	**Analista**
Instalación SW(Oracle, PLSQL/Developper, DWH)	1 día	**AP**
Creación Base de Datos.	2 días	**AP**
Creación Procesos DB parte 1.	5 días	**AP**
Documentación Procesos y Modelo de datos	1 día	**Analista**
Creación Procesos DB parte 2.	4 días	**AP**
Creación Modelo datos DWH	3 días	**AP**
Creación Procesos ETL y Análisis OLAP	3 días	**Analista**
Diseño Plan de Pruebas	1 día	**Analista**
Pruebas Unitarias	5 días	**AP**
Pruebas Integradas	5 días	**AP**
Documentación PAC3	1 día	**Analista**
Memoria	18 días	**JP**
Presentación	3 días	**JP**
Entrega SW	2 días	**AP**
Preparación Tribunal Fin de PFC	9 días	**JP**

Si aplicamos filtros con tablas dinámicas en una Excel, se obtiene la siguiente tabla y gráfico que nos da una visión general del número de días que dedica cada recurso al proyecto:

Etiquetas de fila	**Suma de Duración**
Analista	21
AP	30
JP	39
Total general	**90**

Por último, se muestra la tabla de presupuestos con el importe total teniendo en cuenta que cada jornada laboral serán 3,6 horas/días:

Recurso	Precio Hora	Nº Jornadas	Importe facturable
Jefe Proyecto	40	39	5616
Analista	35	21	2646
Analista Programador	28	30	3024
		Base Imponible	**11286**
		IVA	2370,06
		TOTAL	**13656,06**

5. Capítulo I. Análisis de los requisitos.

5.1. Introducción a los requisitos del proyecto.

Nuestra empresa cliente se encuentra desarrollando un videojuego y nos ha solicitado la realización de un proyecto que analice y almacene los datos de usuarios del videojuego.

El proyecto consiste básicamente a rasgos generales en la creación de una pequeña base de datos que almacene los datos de los usuarios del videojuego, en relación con el mismo y la creación de un DWH que analice los datos almacenados.

Nuestro equipo será el encargado de implementar tanto la creación de la BD como del DWH, lo cual implica realizar el análisis de los requisitos, análisis y diseño de la DB & DWH, desarrollo de los procesos y procedimientos de bases de datos, gestión de acceso, generación de logs de procesos, implementación e instalación del SW de BD.

Toda la gestión y acceso a la información se realizará a través de procedimientos de BD.

Se requiere un diseño tanto de Base de datos como de procesos que sea escalable, que permita en un futuro realizar cambios y ampliaciones tanto de funcionalidad como diseño, etc. Es decir, se requiere un diseño que permita ampliaciones, que sea flexible y esté pensado para realizar cualquier cambio o ampliación con el mínimo impacto y coste posible.

5.2. Requisitos funcionales del sistema a diseñar.

En este apartado se hará una descripción de qué datos se requieren almacenar , analizar y cómo se realizan las transacciones o el flujo de datos y procesos.

Todos los datos a almacenar y analizar estarán relacionados con los Usuarios o jugadores.

El videojuego tiene varios niveles, cada usuario debe superar un reto para pasar de nivel, luego se requerirá almacenar los datos de cada usuario relacionados con cada nivel, por lo que el sistema debe permitir saber en todo momento el número y la lista de usuarios que se encuentran en un nivel concreto o en cada nivel.

Cada usuario podrá disponer de 5 vidas a lo largo de un día, luego será necesario la creación de un proceso de actualización de vidas de todos los usuarios del sistema. Este proceso se realizará de manera automática a las 00:00 de cada día. Luego será necesario almacenar el número de vidas que va utilizando cada usuario mientras esté dado de alta en el sistema. Este proceso crea 5 vidas nuevas a cada usuario diariamente pero no acumula las vidas pendientes de usar del día anterior por lo que el proceso dará de baja o finalizará las vidas que no se han usado a excepción de la que esté usando en el caso que el jugador esté usando una vida, es decir, si un jugador está usando la vida con identificador "x" y en ese momento se ejecuta el proceso de actualización, la vida "x" no será finalizada por el proceso.

Existe la posibilidad de que un usuario se quede sin vidas y pueda adquirir mas bien a través de algún amigo dado de alta en el sistema a través de Redes Sociales o bien comprándolas directamente en el sistema.

Este requisito implicará la creación de varios procesos y datos a almacenar. Por un lado se encuentra la transferencia de vidas a través de un amigo lo cual obliga a almacenar datos de amigos y redes sociales además de procesos de transferencias de vidas.

No se encuentra en el alcance del proyecto que un usuario pueda transferir ayudas, solamente entra en el alcance que pueda transferir vidas.

Por otro lado tenemos la posibilidad de que un usuario pueda realizar compras a través de nuestro sistema, por lo que se necesita crear un sistema de cobro y facturación con todas sus tablas de bases de datos y sus procesos inherentes a la realización de una compra. Estas compras servirán tanto para comprar una vida como para comprar una ayuda de superación de un nivel en un momento determinado del juego, luego tendremos dos tipos de compra o dos productos disponibles para que los usuarios puedan comprar.

En el alcance de este proyecto, entra el diseño de las tablas necesarias para almacenar la información de compras pero no la modelización o diseño de las tablas y sus interfaces para la facturación de cada compra.

El juego está disponible para diversas plataformas como web, móvil y navegadores web. De aquí se podrá obtener información de acceso de cada usuario a través del robot o fichero log generado por el servidor del videojuego que podrá obtener información útil para el DWH donde se podrán analizar datos como de qué plataformas se tienen mas accesos, etc.

De cada usuario se requiere la siguiente información estática:

- Id_usuario.
- Lol(o localización geográfica).
- Sexo.
- Edad.
- Profesión.

- Estado (Baja, Alta).
- Número de cuenta bancaria.(Uso futuro)
- Login.
- Start_date (Fecha de inscripción en el videojuego).
- End_date (Fecha de baja, por defecto a null)

Además, se requiere la siguiente información dinámica que se irá generando a lo largo del ciclo de vida de un usuario dado de alta en el sistema:

- Vidas Intercambiadas.
- Redes sociales a través de las cuales intercambia vidas.
- Plataformas utilizadas para comprar vidas y ayudas.
- Amigos o lista de amigos con los que intercambia vidas.
- Numero de intentos utilizados para superar cada nivel determinado.
- Numero de compras realizadas para ayuda.
- Numero de compras realizadas para superar un nivel.

Se requiere un sistema de DWH(Datawarehousing) que permita realizar extracciones de datos de nuestra BD para poder así tratar y analizar los datos almacenados de usuarios del videojuego.
La creación de un DWH, implica desarrollar procesos de extracción y carga diarias desde la BD hacia el DWH. Estos procesos se podrán realizar bien de forma automática a través de colas Oracle o extracción y carga mediante ficheros planos a una hora determinada del día que podría ser las 00:00 AM.

En este proyecto se ha optado por usar el mismo esquema de base de datos tanto para la Base de datos como para el DWH. La extracción, transformación y carga se realizará a través de procesos ETL de Pentaho PDI Spoom(Aplicación de diseño de procesos ETL para Pentaho de Kettle).

Una vez extraídos y cargados los datos, se requiere obtener informes y realizar estadísticas como por ejemplo :

- Cuál es el nivel donde se compran mas vidas.
- Cuál es el nivel donde se compran mas ayudas.
- Qué sexo compra y juega mas.
- Porcentaje de usuarios que gastan mas dinero en el juego.
- Cual es la plataforma mas usada.
- Desde qué plataforma se realizan mas compras.

En el análisis funcional y diseño técnico, se describe con mas detalle los productos u objetos que entran a formar parte del DWH tales como la creación de tablas de hecho, etc.

Otro requisito del proyecto es la creación de logs por cada proceso de BD para así poder detectar futuros problemas que ayuden a una mejora del mantenimiento y posibles resoluciones de incidencias futuras.

En nuestro proyecto se opta por la creación de logs de procesos a través de tablas de bases de datos que almacenan la información de cada proceso que se ejecuta

guardando la fecha y hora de inicio, fecha y hora fin, estado(OK ó KO), nombre del proceso ejecutado, etc.

Al tener un volumen diario de almacenamiento de logs, se crearán procesos de Hausekeeping que permiten eliminar automáticamente logs antiguos que no se van a necesitar en el futuro.

El sistema se diseña para que se puedan crear estos procesos de Hausekeeping pero no entra en el alcance de este proyecto la creación de los mismos.

Adicional a los procesos de Hausekeeping, se crearán procesos de estadísticas de Bases de datos para analizar todas las tablas de nuestra BD y así el SGBD(Sistema Generador de Bases de Datos) pueda seleccionar el plan de ejecución óptimo en base a los procesos de bases de datos y consultas que se realicen. La ejecución de estadísticas sobre nuestras tablas, hará que tengamos una base de datos mas eficiente y mas rápida.

5.3. Análisis funcional de los requisitos y elementos detectados.

Los requisitos descritos anteriormente nos permite identificar cuáles son los elementos de análisis, tanto para la base de datos a diseñar(Tablas, atributos, relaciones, procesos transaccionales, etc) como para el DWH tanto a nivel de tablas de hecho, dimensiones, indicadores, métricas y atributos.

Tablas BD

Se detectan las siguientes tablas o entidades a modelar en la Base de datos:

- ***Usuarios***. Tabla que almacenará los datos de cada usuario, tales como id_usuario, login, ciudad, sexo, edad, profesión y datos de pago para realizar compras.

- ***Amigos***. Tabla que almacenará la relación de amigos a través de redes sociales de cada usuario. Almacenará datos como id_usuario, id_usuario_amigo, id_red_social a través de la que son amigos, fecha_inicio y fecha_fin por defecto a null.

- ***RedSocial***. Tabla estática que almacenará los tipos de red social permitidos para el traspaso de vidas entre amigos. Almacenará id_red_social y una descripción de la misma.

- ***TrascAyudaAmigos***. Tabla que almacenará los datos cada una de las ayudas realizadas por un usuario amigo. Almacenará datos como el nivel donde se realiza la ayuda, el usuario que transfiere la ayuda, el usuario receptor, fecha en la que se produce la transacción, red social a través de la que se realiza la transferencia de vida.

- ***Plataforma***. Tabla que almacenará los datos estáticos de los distintos tipos de plataformas desde los que se puede acceder al juego. Almacenará datos como tipo de plataforma, web, móvil, navegador.

- ***Productos***. Tabla estática que almacenará la parametrización o catálogo de productos de compras existentes tales como tipo de compra(vida, ayuda), precio.

- ***COMPRAS***. Tabla que almacenará los datos de cada compra realizada por los usuarios del videojuego. Almacenará id_compra, id_type_compra, id_user, Price, fecha_compra. Esta tabla estará relacionada con otra tabla que almacena la información de la compra, tal como productos comprados, cantidad de los mismos etc.

- ***DescLevel***. Tabla que almacenará la descripción de cada nivel existente en el juego.

- ***Dailylevel***. Tabla que almacenará los datos sobre qué usuarios se encuentran en un nivel determinado por fecha. Si un usuario pasa 3 niveles en un mismo día, esta tabla almacenará una fila por cada nivel en el que ha estado el usuario con la fecha y hora en la que entró en el nivel y la fecha y hora en la que terminó.

- ***Vidas***. Tabla que almacenará los datos de cada vida usada y gastada en el videojuego por cada usuario y nivel en el que es usada. Tendrá relación con una tabla que almacena las vidas que van siendo usadas por cada usuario en el videojuego.

- ***Ayuda***. Tabla que almacenará los datos de cada ayuda que se va usando y comprando por cada usuario en el videojuego. Las ayudas no tienen una duración, se trata de un evento único que se produce en un momento determinado pero no tiene un inicio y un fin como las vidas.

- ***PROCESOS***. Tabla estática que almacenará los datos de los diferentes procesos almacenados que tienen cabida en el videojuego y podrán ser ejecutados. Existe la excepción de los procesos de estadísticas y actualización de vidas ya que se trata de procesos internos que deben ser ejecutados por el administrador del sistema.
- ***PROCESOS_LOG***. Tabla dinámica que almacena la información de ejecución de cada proceso almacenado.

- Además de estas tablas, se identifican otras como ERROR_COMPRA y ESTAR_EN_RS_SOCIAL que se dejan para uso futuro.

DWH

Se detectan los siguientes elementos para crear el DWH:

- ***Hechos***. Se detectan los hechos, compras de vidas/ayuda, transferencia de vida a través de una red Social, Agregar Amigo, alta/Baja usuario, usar vida y superación de nivel.

- ***Dimensiones***. Existen diferentes perspectivas de análisis o dimensiones. Estas son, usuarios, niveles, plataforma de acceso, red social, vida/ayuda, producto,

vida(usada), red social, amigos. En apartados siguientes se da de forma mas detallada las dimensiones diseñanas en el sistema de DWH.

-

- ***Indicadores***. Inicialmente se han de medir el nivel donde se realizan mas compras, qué sexo realiza mas compras, porcentaje de usuarios que gastan mas dinero en el juego por plataforma y sexo, plataforma donde se realizan mas compras.

- ***Atributos***. Algunas de las dimensiones facilitan mostrar la información de los atributos.

    o Nivel: codigo y nombre.
    o Plataforma: nombre y navegador.
    o Compra: código, cantidad, precio.
    o Red Social: código y nombre.
    o Producto: código, nombre, precio.
    o Data: año, mes, semana, día, hora i segundo. Estos atributos se pueden considerar en una o dos perspectivas de análisis.

Procesos de BD

Se detectan los siguientes elementos para la creación de procesos de BD:

- ***Alta Usuario.*** Proceso que inserta en la base de datos un nuevo usuario.

- ***Usar Vida.*** Cuando un usuario empieza a jugar, se activa la primera vida disponible y se almacena en la tabla dailylevel además de en la tabla USADA. El proceso recibe por parámetro el id_usuario como una orden para que el usuario almacene la información de la primera vida a usar.

- ***Finalizar Vida.*** Cuando un usuario pierde una vida en en el juego se desactiva la vida y se almacena la fecha y el nivel en el que dejó la vida. También se actualiza la información de la vida usada.

- ***Superar Nivel.*** Cuando un usuario supera un nivel, se debe actualizar las tablas que indican el nivel actual y se finaliza el nivel superado actualizando la fecha de fin del nivel.

- ***Transferir Vida.*** Proceso que realiza la transferencia de una vida por parte de un usuario a otro usuario. Recibe como parámetro el id del usuario que transfiere la vida, el id del usuario que la recibe y el id de la red_social a través de la que recibe la vida.

- ***Baja Usuario.*** Proceso que da de baja a en la base de datos un usuario. Este proceso lleva implícito la finalización de todas las vidas disponibles.

- ***Reactivación Usuario.*** Proceso que reactiva un usuario dado de baja en la BD.

- ***Agregar Amigo.*** Proceso que asigna un amigo a un usuario a través de una red social. Almacena esta información en la tabla AMIGOS.

- ***Comprar Vida.*** Proceso que realiza la compra de una vida por parte de un usuario. Recibe como parámetro, el id del usuario que desea realizar la compra y la cantidad. Un usuario podrá comprar tantas vidas como desee en una mima compra y el precio de la vida y el producto vida será calculado automáticamente en función del nivel en el que se encuentre.

- ***Comprar ayuda.*** Proceso que realiza la compra de una ayuda por parte de un usuario. Recibe como parámetro, el id del usuario que desea realizar la compra y la cantidad. Un usuario podrá comprar tantas ayudas como desee en una mima compra y el precio de la ayuda y el producto ayuda será calculado automáticamente en función del nivel en el que se encuentre.

- ***Actualizar vidas diario.*** Proceso que actualiza las vidas de cada usuario para que cada día dispongan de 5 vídas. Este proceso se ejecuta diariamente a las doce de la noche y finaliza las vidas pendientes de usar a excepción de la que se esté usando.

- ***Estadísticas.*** Proceso que realiza las estadísticas de las tablas de la BD, se ejecutará cada semana o 15 días.

Procesos de DWH

Se diseñan los procesos ETL(extracción, transformación y carga) desde la herramienta Pentaho PDI. Para ello se diseñan procesos de extracción de la información diréctamente desde la BD que se cargará en las tablas diseñadas para analizar el DWH.

- ***Extracción Info Niveles.*** Proceso que extrae de la BD información diaria de los usuarios existentes en cada nivel junto con las transacciones realizadas.

- ***Carga Info Niveles.*** Proceso que carga en el DWH información diaria de los usuarios existentes en cada nivel junto con las transacciones realizadas.

- ***Extracción Info Usuarios.*** Proceso que extrae de la BD información diaria de los usuarios existentes.

- ***Carga Info Usuarios.*** Proceso que carga en el DWH información diaria de los usuarios existentes.

- ***Extracción Info Compras.*** Proceso que extrae de la BD información diaria de las compras de vidas o ayudas realizadas.

- ***Carga Info Usuarios.*** Proceso que carga en el DWH información diaria de las compras de vidas o ayudas realizadas.

- ***Extracción Info Altas/Bajas.*** Proceso que extrae de la BD información diaria de las altas/bajas realizadas.

- ***Carga Info Altas/Bajas.*** Proceso que carga en el DWH información diaria de las altas/bajas realizadas.

Todos los procesos llevarán implícito una salida de log donde se almacena cada acción realizada y los errores detectados en cada proceso.

5.4. Anális funcional de los datos.

Base de Datos

- ***Usuarios***.

Nombre	Tipo	Descripción
Id_user	Entero	Identificador único de usuario
Sexo	String	Tipo de sexo(Masculino/Femenino)
login	String	Login o Nombre único del usuario.
Edad	Entero	Edad
Profesion	String	Profesión.
Ciudad	String	Ciudad o dirección del usuario.
Start_date	Date	Fecha de alta en el sistema
End_date	Date	Fecha de Baja en el sistema, por defecto a Null
Status	String	Estado del usuario, OK(activo), KO(Inactivo).
No_cuenta_pago	String	Número tarjeta de crédito.

- ***Amigos***.

Nombre	Tipo	Descripción
Id_amigos	Entero	Identificador relación.
Id_user	Entero	Identificador del usuario.
Id_user_amigo	Entero	Identificador del usuario amigo.
Id_redsocial	Entero	Nombre de la red social a través de la que son amigos.
Start_date	Date	Fecha de alta amistad
End_date	Date	Fecha de Baja amistad, por defecto a Null

- ***RedSocial***.

Nombre	Tipo	Descripción
Id_redsocial	Entero	Identificador red social.
Nom_redsocial	String	Identificador del primer

		usuario amigo.

- ***TrascAyudaAmigos***.

Nombre	**Tipo**	**Descripción**
Id_trans_Ayuda	Entero	Identificador transacción ayuda amigo.
Id_user_emisor	Entero	Identificador del usuario que envía la ayuda a un amigo.
Id_user_receptor	Entero	Identificador del usuario que recibe la.
Time_stamp	Date	Fecha hora en la que se produce la ayuda.
id_vida	Entero	id_vida de la vida transferida.
Id_redsocial	Entero	Id de la red social a través de la que se realiza la transferencia.
TranscRErrorLog	String	Descripción del error que se ha producido en el caso que la transacción haya fallado.
TranscCorrect	String	Información adicional en caso que se haya realizado la transacción de forma exitosa.

- ***Plataforma***.

Nombre	**Tipo**	**Descripción**
Id_platform	Entero	Identificador único de plataforma.
Tipo_platform	Entero	Tipo de plataforma.
Desc_Plataforma	String	Nombre de la plataforma, Web, Mobil, etc

- ***Productos***.

Nombre	**Tipo**	**Descripción**
Id_producto	Entero	Identificador del producto.
Id_tipo_compra	Entero	Tipo de compra(1-vida,2-ayuda).
Description_product	String	Descripción del producto a comprar, Vida o ayuda de momento pero se puede usar en futuro con otros productos.
Precio	Float	Precio de la compra.
Start_date	Date	Fecha inicio de activación

		del producto.
End_date	Date	Fecha fin de activación del producto.

- ***ERROR_COMPRA***. (Tabla de datos de uso futuro)

Nombre	**Tipo**	**Descripción**
id_error_compra	Entero	Identificador único de un error definido para una compra.
desc_error_compra	String	Descripción del error de la compra.

- ***COMPRAS***.

Nombre	**Tipo**	**Descripción**
Id_compra	Entero	Identificador único de una compra.
Id_user	Entero	Id del usuario que realiza la compra.
Fecha_compra	Date	Fecha y hora en la que se realiza la compra.
Id_platform	Entero	Identificador de la plataforma desde la que se realiza la compra.
Id_Level	Entero	Nivel del juego en el que se realiza la compra.
id_error_compra	Entero	Identificador único de un error definido para una compra en el caso que se produzca.
Estado_Compra	String	Si la compra se realizó satisfactoriamente toamrá valor "OK" en caso contrario tomará el valor "ERROR".

- ***DescLevel***..

Nombre	**Tipo**	**Descripción**
Id_level	Entero	Identificador del nivel del juego.
Nom_Level	String	Nombre del nivel del juego
Desc_level	String	Descripción del nivel del juego
Start_date	Date	Fecha inicio en la que comienza a ser activo un nivel.
End_date	Date	Fecha fin en la que el nivel deja de estar activo.

- ***Vidas***.

Nombre	Tipo	Descripción
Id_vida	Entero	Identificador de una vida usada o en uso.
Id_user	Entero	Identificador del usuario del juego con el que está relacionada la vida.
Start_date	Date	Fecha inicio en la que comienza la la vida.
End_date	Date	Fecha fin en la que termina la vida.

- ***Dailylevel***.

Nombre	Tipo	Descripción
Id_level	Entero	Identificador del nivel del juego.
Id_user	Entero	Identificador del usuario que se ha conectado al juego.
Start_date	Date	Fecha inicio en la que comienza la sesión del usuario en el nivel.
End_date	Date	Fecha fin de la sesión del usuario en el nivel.

- ***Ayudas***.

Nombre	Tipo	Descripción
Id_ayuda	Entero	Identificador de una ayuda.
Id_user	Entero	Identificador del usuario del juego con el que está relacionada la ayuda.
time_stamp	Date	Fecha en la que se adquiere la ayuda.
Id_level	Entero	Nivel en el que se solicita la ayuda.

- ***USADA***.

Nombre	Tipo	Descripción
Id_vida	Entero	Identificador de una vida usada o en uso.
Id_user	Entero	Identificador del usuario del juego con el que está relacionada la vida.

id_level	Entero	Nivel en el que se encuentra la vida en uso.
Start_date	Date	Fecha inicio en la que comienza la la vida.
End_date	Date	Fecha fin en la que termina la vida.

- ***INCLUIR_COMPRA***.

Nombre	**Tipo**	**Descripción**
Id_compra	Entero	Identificador único de una compra.
Id_user	Entero	Id del usuario que realiza la compra.
id_producto	Entero	Identificador del producto.
precio	Entero	Precio del producto comprado
cantidad	Entero	Cantidad de vidas o ayudas compradas.

- ***PROCESOS***.

Nombre	**Tipo**	**Descripción**
Id_proceso	Entero	Identificador único del proceso.
nom_proceso	STRING	Nombre del proceso .
desc_proceso	STRING	Descripción del proceso.

- ***PROCESOS_LOG***.

Nombre	**Tipo**	**Descripción**
Id_proceso	Entero	Identificador único del proceso.
id_user	Entero	Identificador del usuario con el que está relacionada la ejecución del proceso.
Status	String	Estado del proceso. OK, se ejecutó correctamente, KO hubo errores.
Start_date	Date	Fecha inicio del proceso.
End_date	Date	Fecha fin del proceso.

DWH

La base de datos nos proporciona toda la información a analizar con todas las perspectivas deseadas para el DWH. Las fuentes de datos se extraerán desde la base de datos con actualización diaria:

- ***Tabla RedSocial.*** Lista de posibles redes sociales a través de las que se realizarán transferencias de vidas. Se utilizará la misma tabla definida en la BD.

- ***Tabla Plataforma.*** Lista de posibles plataformas de acceso al videojuego. Se utilizará la misma tabla definida en la BD.

- ***Tabla DescLevel.*** Lista los distintos niveles definidos en el videojuego. Se utilizará la misma tabla definida en la BD.

- ***Tabla Productos.*** Lista de productos definidos en el videojuego para comprar por los usuarios. Se utilizará la misma tabla definida en la BD.

- ***Extracción Compras por nivel.*** Se realiza una extracción diaria de las compras que se han realizado cada día obteniendo los datos de usuario, sexo, nivel, plataforma, etc.

- ***Extracción Transferencias.*** Se realiza una extracción diaria de las transferencias de vidas y ayudas que se han realizado cada día obteniendo los datos de usuario, sexo, nivel, plataforma, red social, etc.

- ***Extracción* Altas/Bajas*.*** Se realiza una extracción diaria de las altas y bajas que se producen en el videojuego.

- ***Extracción* Superación de nivel*.*** Se realiza una extracción diaria de los nuveles que son superados en el juego por cada usuario.

- ***Extracción* Uso de vidas*.*** Se realiza una extracción diaria de las las vidas que son usadas por cada usuario y los niveles por los que ha pasado cada vida, etc.

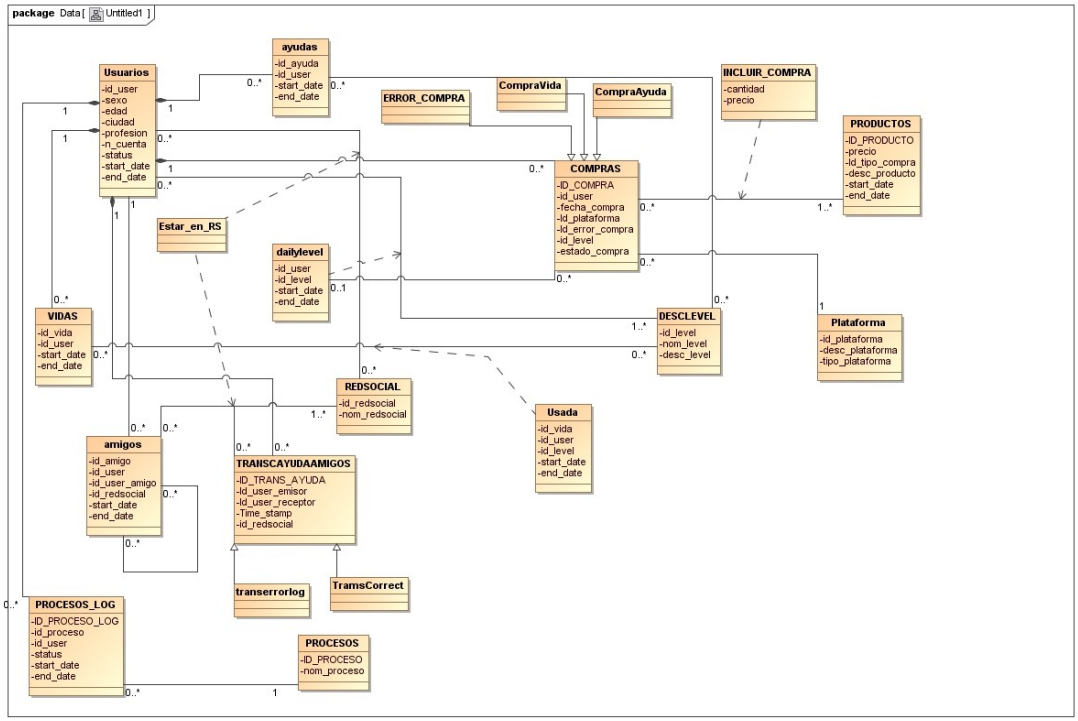
Inconsistencia de datos en DWH:

Para que no haya inconsistencia de datos cabe tener en cuenta que periódicamente se realizarán extracciones y cargas de la base de datos al DWH para tener actualizados los datos por fechas sin que se produzcan redundancias que lleven a una inconsistencia de datos.

6. Capítulo II. Análisis y Diseño Técnico.

En este apartado se entra a realizar un análisis y diseño técnico del proyecto en el que se mostrará el diseño o modelo conceptual, lógico y físico tanto de la base de datos como del DWH

6.1. Diseño Conceptual BD.



6.1.1 Reglas de Negocio BD.

Para dar respuesta a los requerimientos especificados en el apartado anterior, se han de respetar las siguientes reglas de negocio:

- Cuando se crea un nuevo usuario, tendrá como fecha "start_date" la fecha en la que se crea dicho usuario, el campo "end_date" estará a null y el campo "Status" tendrá el valor OK.

- Cuando se crea un usuario, se le cobrará una cuota mensual a través de su número de cuenta. La facturación de esta cuota mensual queda fuera del alcance de la BD.
- Cuando un usuario se crea inicialmente, empieza en el primer nivel del juego y automáticamente se actualizan sus vidas en la base de datos con end_date a null.

- Si un usuario supera un nivel, pasa al siguiente nivel dándose de baja el nivel actual, pero si se encuentra en el último nivel(level 5), entonces el usuario pasa a empezar nuevamente en el primer nivel del juego.

- Si un usuario se da de baja en el juego, se actualizará la fecha del campo "End_date" con la fecha de baja y el campo "Status" con el valor BB.

- Si se intenta dar de alta en el juego un usuario que anteriormente estuvo dado de alta y su estado actual es que está dado de baja, se crea otra fila con el campo "Status" a OK, "start_date" a fecha actual de alta y "End_date" a null y respetando el resto de datos anteriores.

- Un usuario puede comprar una vida o una ayuda, pero además la realización de la compra puede ser errónea.

- Cuando se realiza una compra por parte de un usuario, ésta puede incluir un número de productos(vidas o de ayudas) con un precio.

- En una Plataforma se podrán realizar varias compras.

- Las compras son productos que inicialmente serán Vidas y ayudas, pero queda abierto a la parametrización de futuros y nuevos productos.

- En todo momento se puede saber qué número de usuarios está en un nivel y momento determinado además del número de intentos que realiza cada usuario para superar un nivel.

- Cada Vida de cada usuario tendrá una fecha de inicio y la fecha fin se actualizará una vez sea gastada.

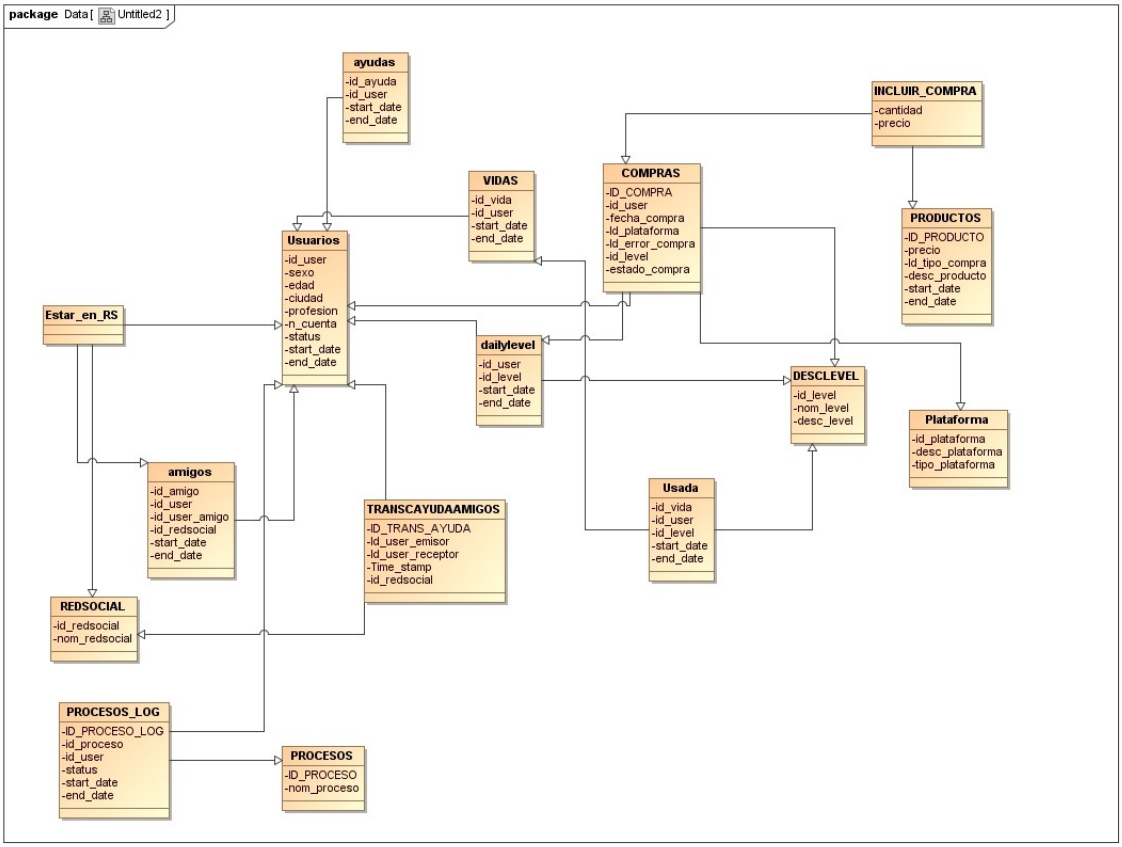
- Un usuario podrá comprar todas las vidas que quiera pero inicialmente solo dispondrá de 5 vidas iniciales diarias.

- Se podrá saber la cantidad de vidas que utiliza cada usuario en cada nivel.

- Cada vez que un usuario realiza una transferencia de vida a otro usuario a través de una red social por primera vez, se genera una entrada en la entidad Amigos donde se crea el vínculo que permite realizar la transferencia.

- Una trasferencia puede ser errónea o satisfactoria.

6.2. Diseño Lógico de BD.



6.3 Diseño Físico de BD.

A continuación se muestra un script SQL con el diseño físico que permite la creación de las tablas de la base de datos:

Como paso previo a la creación de las tablas diseñadas, se crea el esquema de usuario VIDEOADMIN que realizará todas las gestiones y transacciones del videojuego. Este usuario será utilizado además por los distintos procesos del videojuego por lo que podrá crear tablas, vistas, procedimientos, funciones, cursores, paquetes, etc.

```
-- Create the user
create user VIDEOADMIN
  identified by ""
  default tablespace USERS
  temporary tablespace TEMP
  profile DEFAULT
  quota unlimited on users;

-- Grant/Revoke system privileges
grant alter session to VIDEOADMIN;
grant create database link to VIDEOADMIN;
grant create materialized view to VIDEOADMIN;
grant create procedure to VIDEOADMIN;
grant create public synonym to VIDEOADMIN;
grant create sequence to VIDEOADMIN;
grant create session to VIDEOADMIN;
```

```
grant create synonym to VIDEOADMIN;
grant create table to VIDEOADMIN;
grant create trigger to VIDEOADMIN;
grant create type to VIDEOADMIN;
grant create view to VIDEOADMIN;
grant unlimited tablespace to VIDEOADMIN;
```

Una vez creado el usuario de base de datos que realizará la gestión, administración y ejecución de los distintos procesos del Videojuego, se crean las tablas diseñadas en le modelo lógico.

```
--PLATAFORMA
CREATE TABLE PLATAFORMA (
ID_PLATAFORMA NUMBER CONSTRAINT PK_plataforma PRIMARY KEY,
desc_plataforma VARCHAR2(30 CHAR) CONSTRAINT NN_plataforma_desc NOT
NULL,
tipo_plataforma NUMBER CONSTRAINT NN_tipo_plataforma NOT NULL
) ; -- Tipo_plataforma=1 => web si tipo_plataforma=2 => Movil
--REDSOCIAL
CREATE TABLE REDSOCIAL (
ID_REDSOCIAL NUMBER CONSTRAINT PK_rsocial PRIMARY KEY,
nom_redsocial VARCHAR2(30 CHAR) CONSTRAINT NN_rsocial NOT NULL
) ;
--DESCLEVEL
CREATE TABLE DESCLEVEL (
ID_LEVEL NUMBER CONSTRAINT PK_level PRIMARY KEY,
nom_level VARCHAR2(15 CHAR) CONSTRAINT NN_nomlevel NOT NULL,
desc_level VARCHAR2(30 CHAR) CONSTRAINT NN_desclevel NOT NULL,
start_date DATE CONSTRAINT NN_level_stDate NOT NULL,
end_date DATE,
CONSTRAINT CK_level_date CHECK (end_date IS NULL OR end_date >
start_date)
) ;
--USUARIOS
CREATE TABLE USUARIOS (
ID_USER NUMBER CONSTRAINT PK_usuarios PRIMARY KEY,
sexo VARCHAR2(20 CHAR) CONSTRAINT NN_usrsex NOT NULL,
edad NUMBER CONSTRAINT NN_usredad NOT NULL,
ciudad VARCHAR2(30 CHAR) CONSTRAINT NN_usrcity NOT NULL,
profesion VARCHAR2(30 CHAR) CONSTRAINT NN_usrprofess NOT NULL,
start_date DATE CONSTRAINT NN_usr_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_usr_date CHECK (end_date IS NULL OR end_date >
start_date),
status VARCHAR2(2 CHAR) CONSTRAINT NN_usrstatus NOT NULL,
no_cuenta_pago VARCHAR2(24 CHAR) CONSTRAINT NN_usrnocuenta NOT NULL,
LOGIN          VARCHAR2(50 CHAR) not null
) ; -- Status OK=> Activo KO=> Inactivo
--AMIGOS
CREATE TABLE AMIGOS (
ID_AMIGO NUMBER,
id_user,
id_user_amigo,
id_redsocial NUMBER CONSTRAINT NN_amigos_rsocial NOT NULL,
start_date DATE CONSTRAINT NN_amigo_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_amido_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT FK_amigos_user1 FOREIGN KEY (id_user) REFERENCES USUARIOS
(id_user),
```

```
CONSTRAINT FK_amigos_user2 FOREIGN KEY (id_user_amigo) REFERENCES
USUARIOS (id_user),
CONSTRAINT FK_amigos_rsocial FOREIGN KEY (id_redsocial) REFERENCES
REDSOCIAL (id_redsocial),
CONSTRAINT PK_amigos PRIMARY KEY (id_amigo,id_user,id_user_amigo)
) ;
--VIDAS
CREATE TABLE VIDAS (
ID_VIDA NUMBER,
id_user,
start_date DATE CONSTRAINT NN_vida_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_vida_end_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT FK_vida_user FOREIGN KEY (id_user) REFERENCES USUARIOS
(id_user),
CONSTRAINT PK_vidas PRIMARY KEY (id_vida, id_user)
) ;
--AYUDAS
CREATE TABLE AYUDA (
ID_AYUDA NUMBER,
id_user NUMBER NOT NULL,
time_stamp DATE NOT NULL,
id_level NOT NULL,
CONSTRAINT FK_ayuda_level FOREIGN KEY (id_level) REFERENCES
videoadmin.DESCLEVEL (id_level),
CONSTRAINT PK_ayuda PRIMARY KEY (id_ayuda)
) ;
--Dailylevel
CREATE TABLE DAILYLEVEL (
id_user,
id_level,
start_date DATE,
end_date DATE,
CONSTRAINT CK_dl_end_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT FK_dl_user FOREIGN KEY (id_user) REFERENCES USUARIOS
(id_user),
CONSTRAINT FK_dl_level FOREIGN KEY (id_level) REFERENCES DESCLEVEL
(id_level),
CONSTRAINT PK_daylylevel PRIMARY KEY (id_user, id_level,start_date)
) ;
--TRANSCAYUDAAMIGOS
CREATE TABLE TRANSCAYUDAAMIGOS (
ID_TRANS_AYUDA NUMBER CONSTRAINT PK_transcamigo PRIMARY KEY,
Id_user_emisor NUMBER CONSTRAINT NN_transcamigo_ue NOT NULL,
Id_user_receptor NUMBER CONSTRAINT NN_transcamigo_ur NOT NULL,
Time_stamp DATE CONSTRAINT NN_transcamigo_ts NOT NULL,
ID_VIDA NUMBER CONSTRAINT NN_vida_tra NOT NULL,
TranscRErrorLog VARCHAR2(50 CHAR) CONSTRAINT NN_transcamigo_terr NOT
NULL,
TranscCorrect VARCHAR2(50 CHAR) CONSTRAINT NN_transcamigo_tok NOT
NULL,
Id_redsocial NUMBER CONSTRAINT NN_transcamigo_rs NOT NULL,
CONSTRAINT FK_user_emisor FOREIGN KEY (Id_user_emisor) REFERENCES
USUARIOS (id_user),
CONSTRAINT FK_user_receptor FOREIGN KEY (Id_user_receptor) REFERENCES
USUARIOS (id_user),
CONSTRAINT FK_transcamigo_rsocial FOREIGN KEY (id_redsocial)
REFERENCES REDSOCIAL (id_redsocial)
) ;
```

```
--USADA
CREATE TABLE USADA (
id_vida,
id_user,
id_level,
start_date DATE CONSTRAINT NN_usada_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_usada_end_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT FK_usada_vida FOREIGN KEY (id_vida,id_user) REFERENCES
VIDAS (id_vida,id_user),
CONSTRAINT FK_usada_level FOREIGN KEY (id_level) REFERENCES DESCLEVEL
(id_level),
CONSTRAINT PK_usada PRIMARY KEY (id_vida, id_user, id_level)
) ;
--ESTAR_EN_RS ---- OJO, esta tabla no se usa finalmente
"""""""""""""""""""""""""""""""""""""""""""""""""""""""
CREATE TABLE ESTAR_EN_RS (
ID_AMIGO,
id_user,
id_user_amigo,
id_redsocial,
start_date DATE CONSTRAINT NN_estarenrs_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_estarenrs_end_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT FK_estarenrs_amigos FOREIGN KEY
(id_amigo,id_user,id_user_amigo) REFERENCES AMIGOS
(id_amigo,id_user,id_user_amigo),
CONSTRAINT FK_estarenrs_rs FOREIGN KEY (id_redsocial) REFERENCES
REDSOCIAL (id_redsocial),
CONSTRAINT PK_estarenrs PRIMARY KEY (id_amigo,
id_user,id_user_amigo,id_redsocial)
) ;
--Productos
CREATE TABLE PRODUCTOS (
ID_PRODUCTO NUMBER,
precio NUMBER,
Id_tipo_compra NUMBER CONSTRAINT NN_producto NOT NULL,
desc_producto VARCHAR2(30 CHAR) CONSTRAINT NN_producto_desc NOT NULL,
start_date DATE CONSTRAINT NN_producto_startDate NOT NULL,
end_date DATE,
CONSTRAINT CK_product_date CHECK (end_date IS NULL OR end_date >
start_date),
CONSTRAINT PK_productos PRIMARY KEY (id_producto, precio)
) ;
--ERROR_COMPRA
CREATE TABLE ERROR_COMPRA (
ID_ERROR_COMPRA NUMBER CONSTRAINT PK_errcompra PRIMARY KEY,
desc_error_compra VARCHAR2(50 CHAR) CONSTRAINT NN_errcompra NOT NULL
) ;
--COMPRAS
CREATE TABLE COMPRAS (
ID_COMPRA NUMBER,
Id_user,
fecha_compra DATE CONSTRAINT NN_fecha_compra NOT NULL,
Id_plataforma NUMBER CONSTRAINT NN_compra_platf NOT NULL,
Id_error_compra NUMBER,
id_level NUMBER CONSTRAINT NN_compra_level NOT NULL,
Estado_Compra VARCHAR2(10 CHAR) CONSTRAINT NN_status_compra NOT NULL,
```

```
CONSTRAINT FK_user_compra FOREIGN KEY (Id_user) REFERENCES USUARIOS
(id_user),
CONSTRAINT FK_error_compra FOREIGN KEY (Id_error_compra) REFERENCES
ERROR_COMPRA (Id_error_compra),
CONSTRAINT FK_platform_compra FOREIGN KEY (Id_plataforma) REFERENCES
PLATAFORMA (Id_plataforma),
CONSTRAINT PK_compras PRIMARY KEY (id_compra, id_user)
) ;
--INCLUIR_COMPRA
CREATE TABLE INCLUIR_COMPRA (
Id_compra,
Id_user,
Id_producto,
precio,
cantidad NUMBER CONSTRAINT NN_compra_cantidad NOT NULL,
CONSTRAINT FK_compras FOREIGN KEY (Id_compra,Id_user) REFERENCES
COMPRAS (Id_compra,id_user),
CONSTRAINT FK_producto_compra FOREIGN KEY (Id_producto,precio)
REFERENCES PRODUCTOS (Id_producto,precio),
CONSTRAINT PK_incluircompra PRIMARY KEY (Id_compra, id_user,
Id_producto, precio)
) ;

--PROCESOS --Tabla que almacena los distintos procesos que se ejecutan
en la BD de cara a almacenar logs de procesos.
CREATE TABLE PROCESOS (
ID_PROCESO NUMBER CONSTRAINT PK_procesos PRIMARY KEY,
NOM_proceso VARCHAR2(30 CHAR) CONSTRAINT NN_procesos_nom NOT NULL,
desc_proceso VARCHAR2(30 CHAR) CONSTRAINT NN_procesos_desc NOT NULL
) ;

--PROCESOS_LOG --Tabla que almacena información de ejecución de cada
proceso.
CREATE TABLE PROCESOS_LOG (
ID_PROCESO_LOG NUMBER CONSTRAINT PK_procesolog PRIMARY KEY,
id_proceso,
id_user NUMBER CONSTRAINT NN_usrerproceso NOT NULL,
start_date DATE CONSTRAINT NN_proceso_startDate NOT NULL,
end_date DATE,
status VARCHAR2(2 CHAR) CONSTRAINT NN_procesostatus NOT NULL,
CONSTRAINT CK_proceso_date CHECK (end_date IS NULL OR end_date >=
start_date),
CONSTRAINT FK_proceso FOREIGN KEY (id_proceso) REFERENCES PROCESOS
(id_proceso)
) ; --- status  OK -- ERR (error) -- STR(en ejecución)
```

Una vez creado el modelo físico en la BD, se pueden ejecutar los scripts sql siguientes para crear secuencias y triggers de las tablas creadas:

3_sequence_triggers_BD.sql

desde cualquier consola de sql "sqlplus videoadmin/abc1234@xe @3_sequence_triggers_BD.sql".

Ya creadas las secuencias u triggers o disparadores, se puede realizar una carga inicial de las tablas estáticas y parametrizables como plataforma, redsocial, productos, etc. El script **4_Carga_Datos_inicial_BD.sql** contiene una información de entrada

suficiente para empezar a ejecutar procesos de Base de datos y empezar a obtener resultados. En los siguientes apartados daremos mas detalle de esta carga inicial.

6.4. Diseño Conceptual DWH.

En el modelo conceptual del DWH se identificarán los tipos de procesos y vistas de negocio que proporcionan respuesta a los requisitos del proyecto. Normalmente en esta fase se debe ser previsor y pensar mas allá de las necesidades actuales.

El análisis de estadísticas se realizará en base a la extracción de los datos de la BD del videojuego.

Nuestro objetivo es diseñar un almacén de datos o DWH que facilite el proceso de transformación de los datos.

Para mejorar el rendimiento de la carga de datos, se realizará una carga de datos previa a la staging area.

Se consideran inicialmente los hechos para análisis de Compra de Vida/ayuda, transferencia de vidas, Agregar Amigos, Altas/Bajas, uso de vida y superación de niveles.

Cada hecho a analizar nos puede dar diferentes puntos de vista que nos proporcionan las dimensiones del proceso de negocio.

En el caso de Compra de Vida:

- Dimensión Usuario: Usuario que realiza la compra.
- Dimensión Plataforma. Plataforma desde la que se realiza la compra.
- Dimensión Nivel: Nivel desde el que se realiza la compra.
- Dimensión Producto: Producto vida a comprar.
- Dimensión fecha: Fecha y hora en la que se realiza la compra.

En el caso de Compra de Ayuda:

- Dimensión Usuario: Usuario que realiza la compra de la ayuda.
- Dimensión Plataforma. Plataforma desde la que se realiza la compra de la ayuda.
- Dimensión Nivel: Nivel desde el que se realiza la compra de la ayuda.
- Dimensión Producto: Producto ayuda a comprar.
- Dimensión fecha: Fecha y hora en la que se realiza la compra.

En el caso de Transferir vida a un amigo:

- Dimensión Usuario: Usuario que realiza la transferencia.
- Dimensión Red Social: Red Social desde la que se realiza la transferencia.
- Dimensión fecha: Fecha y hora en la que se realiza la transferencia.

En el caso de Agregar amigo:

- Dimensión Usuario: Usuario que agrega a un amigo.
- Dimensión Red Social: Red Social desde la que se realiza la transferencia.
- Dimensión fecha: Fecha y hora en la que se realiza la agregación.

En el caso de Alta Usuario:

- Dimensión Usuario: Usuario que se da de alta.
- Dimensión fecha: Fecha y hora en la que se realiza el alta.

En el caso de Baja usuario.

- Dimensión Usuario: Usuario que se da de Baja.
- Dimensión Nivel: Nivel desde el que se realiza la baja.
- Dimensión fecha: Fecha y hora en la que se realiza la baja.

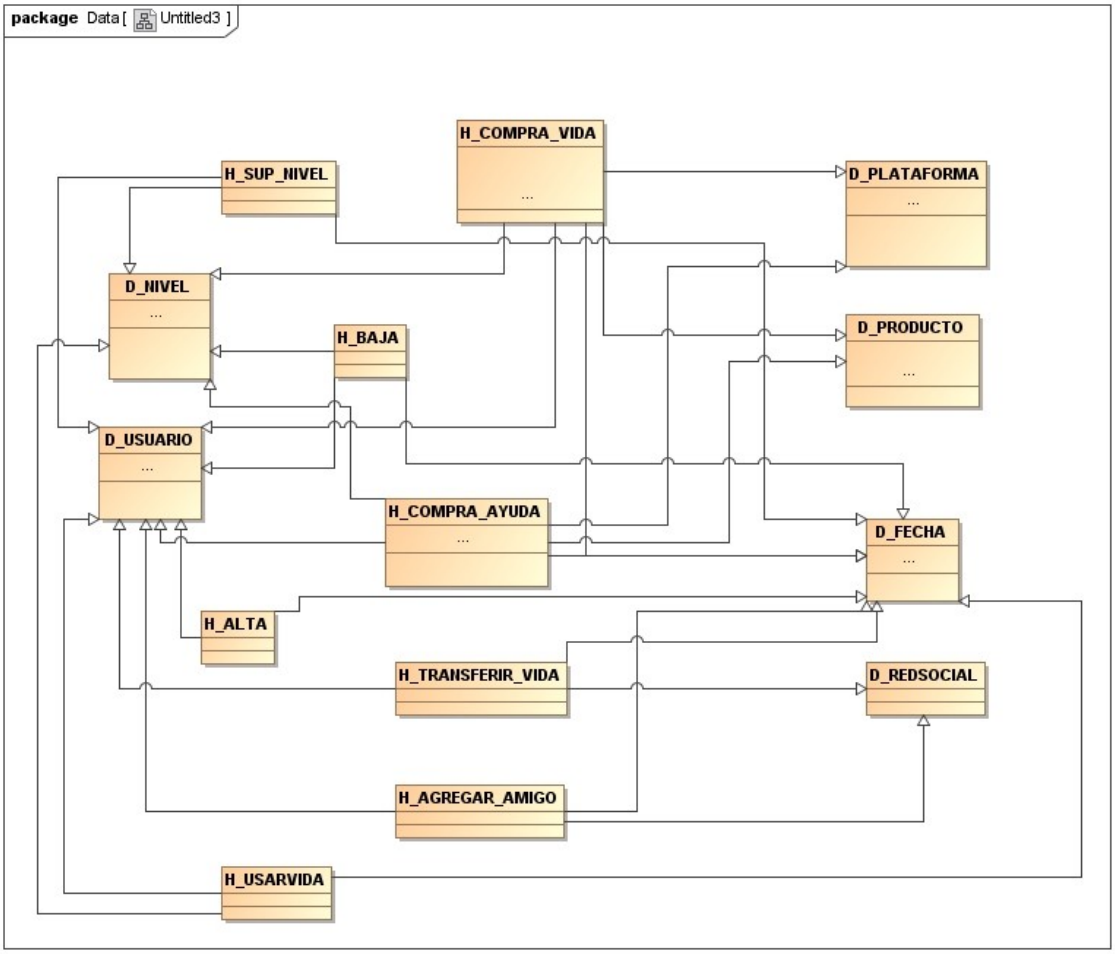
En el caso de Superar Nivel.

- Dimensión Usuario: Usuario que supera un nivel.
- Dimensión Nivel: Nivel superado.
- Dimensión fecha: Fecha y hora en la que se supera el nivel.

En el caso de Usar una vida.

- Dimensión Usuario: Usuario que hace uso de una vida disponible.
- Dimensión Nivel: Nivel en el que está usando la vida.
- Dimensión fecha: Fecha y hora en la que se supera el nivel.

En el siguiente diagrama se muestra un resumen del diagrama conceptual que en este caso se parece mucho al diagrama lógico solo que sin los atributos y campos que si detallaremos en el modelo conceptual a continuación.



Se puede apreciar que tenemos 8 tablas de hecho y 6 dimensiones.

6.5. Diseño lógico DWH.

En el modelo conceptual se han identificado las tablas de hecho y dimensiones por lo que ahora toca realizar el diseño lógico en el que se caracterizan las métricas de las tablas de hecho y sus atributos.

Las tablas de hecho tienen la clave que identifica de manera única cada registro, las claves foráneas a las dimensiones relacionadas con las tablas de hecho y las métricas. Luego a continuación se muestra cómo quedan las tablas de hecho y las dimensiones con las métricas que se han considerado.

Tabla de Hecho	Foreing Key	Métricas
h_compra_vida	id_user, id_plataforma, id_level, id_producto, id_fecha	Número de compras, precio total, cantidad.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_plataforma	id_plataforma	Desc_plataforma	Desc_plataforma
d_nivel	id_level	Nom_level	Nom_level
d_producto	id_producto	Tipo, desc_producto, Precio	Tipo → desc_producto→ Precio
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas
h_compra_Ayuda	id_user, id_plataforma, id_level, id_producto, id_fecha	Número de compras, precio total.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_plataforma	id_plataforma	Desc_plataforma	Desc_plataforma
d_nivel	id_level	Nom_level	Nom_level
d_producto	id_producto	Tipo, desc_producto, Precio	Tipo → desc_producto→ Precio
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas
h_Transferir_vida	id_user, id_redsocial, id_fecha	Número vidas transferidas

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad,	Login → sexo →

		status, login	edad → ciudad → status.
d_redsocial	id_redsocial	nom_redsocial	nom_redsocial
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

La tabla de hecho Agregar amigo se implementa a nivel de creación de tabla peso se deja para uso futuro para realizar extracción, transformación, carga y análisis.

Tabla de Hecho	Foreing Key	Métricas
h_Agregar_amigo	id_user, id_redsocial, id_fecha	Número amigos agregados.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_redsocial	id_redsocial	nom_redsocial	nom_redsocial
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas
h_Alta	id_user, id_fecha	Número altas.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas

h_Baja	id_user, id_level, id_fecha	Número de bajas.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_nivel	id_level	nom_level	nom_level
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas
h_Sup_Nivel	id_user, id_level, id_fecha	Número niveles.

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_nivel	id_level	nom_level	nom_level
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto, segundo	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→ segundo

Tabla de Hecho	Foreing Key	Métricas
h_UsarVida	id_user, id_level, id_fecha	Número de vidas usadas

Y los atributos de cada una de las dimensiones son:

Dimensión.	Primary Key.	Atributo.	Jerarquía.
d_usuario	id_user	Sexo, edad, ciudad, status, login	Login → sexo → edad → ciudad → status.
d_nivel	id_level	nom_level	nom_level
d_fecha	id_fecha	Ano, mes, desc_mes, semana_ano, dia, hora, minuto,	Ano→mes→ desc_mes→ semana_ano→ dia→ hora, minuto→

		segundo	segundo

6.6. Diseño físico DWH.

Al igual que se ha hecho en el modelo físico de BD, a continuación se muestra el diseño o modelo físico del DWH a crear con todas sus tablas de hecho, dimensiones, sequencias y métricas.

```
--Creación DWH tablas de hecho y dimensiones.

--Dimensión usuario
CREATE TABLE d_usuario (
ID_USER NUMBER CONSTRAINT PK_usuarios_d PRIMARY KEY,
sexo VARCHAR2(20 CHAR) CONSTRAINT NN_usrsex_d NOT NULL,
edad NUMBER CONSTRAINT NN_usredad_d NOT NULL,
ciudad VARCHAR2(30 CHAR) CONSTRAINT NN_usrcity_d NOT NULL,
profesion VARCHAR2(30 CHAR) CONSTRAINT NN_usrprofess_d NOT NULL,
status VARCHAR2(2 CHAR) CONSTRAINT NN_usrstatus_d NOT NULL
) ;
-- Status OK=> Activo KO=> Inactivo

--Dimensión PLATAFORMA
CREATE TABLE d_plataforma (
ID_PLATAFORMA NUMBER CONSTRAINT PK_plataforma_d PRIMARY KEY,
desc_plataforma VARCHAR2(30 CHAR) CONSTRAINT NN_plataforma_desc_d NOT
NULL
) ;

-- Dimension Nivel
CREATE TABLE d_nivel (
ID_LEVEL NUMBER CONSTRAINT PK_level_d PRIMARY KEY,
nom_level VARCHAR2(15 CHAR) CONSTRAINT NN_nomlevel_d NOT NULL
) ;

--Dimension REDSOCIAL
CREATE TABLE d_redsocial (
ID_REDSOCIAL NUMBER CONSTRAINT PK_rsocial_d PRIMARY KEY,
nom_redsocial VARCHAR2(30 CHAR) CONSTRAINT NN_rsocial_d NOT NULL
) ;

--Dimension Producto
CREATE TABLE d_producto (
ID_PRODUCTO  NUMBER CONSTRAINT PK_producto_d PRIMARY KEY,
precio NUMBER CONSTRAINT NN_producto_d NOT NULL,
Id_tipo_compra NUMBER CONSTRAINT NN_productoT_d NOT NULL,
desc_producto VARCHAR2(30 CHAR) CONSTRAINT NN_producto_desc_d NOT NULL
) ;

--------------------------------------------------
-------------------------FECHA
--Creación de la secuencia del id para la fecha
CREATE SEQUENCE d_fecha_id_fecha_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

--- Dimension FECHA

CREATE TABLE d_fecha (
```

```
id_fecha NUMBER(10,0) CONSTRAINT PK_fecha_d PRIMARY KEY,
anyo NUMBER(10,0) NOT NULL,
mes NUMBER(10,0) NOT NULL,
desc_mes VARCHAR2(20 CHAR) NOT NULL,
semana_anyo NUMBER(10,0) NOT NULL,
dia NUMBER(10,0) NOT NULL,
desc_dia VARCHAR2(20 CHAR) NOT NULL,
hora NUMBER(10,0) NOT NULL,
minuto NUMBER(10,0) NOT NULL,
segundo NUMBER(10,0) NOT NULL
);

-------------------------------------------------------
-----  Creacion de Las tablas de hecho
-------------------------------------------------------
-------
*********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_compra_vida.
-------
*********************************************************************
*
CREATE SEQUENCE h_compra_vida_ID_comp_vida_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_compra_vida
CREATE TABLE h_compra_vida (
ID_comp_vida NUMBER CONSTRAINT PK_cv_v PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_cvu_v NOT NULL,
ID_PLATAFORMA NUMBER CONSTRAINT NN_cvp_v NOT NULL,
ID_PRODUCTO  NUMBER CONSTRAINT NN_cvpr_v NOT NULL,
ID_LEVEL NUMBER CONSTRAINT NN_cvl_v NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_cvf_v NOT NULL,
N_comvida NUMBER(10,0) CONSTRAINT NN_comvid3 NOT NULL,
CANTIDAD NUMBER(10,0) CONSTRAINT NN_cantv3 NOT NULL,
CONSTRAINT FK_user_hcom_v FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_platf_hcom_v FOREIGN KEY (ID_PLATAFORMA) REFERENCES
d_plataforma (ID_PLATAFORMA),
CONSTRAINT FK_prod_hcom_v FOREIGN KEY (ID_PRODUCTO) REFERENCES
d_producto (ID_PRODUCTO),
CONSTRAINT FK_lev_hcom_v FOREIGN KEY (ID_LEVEL) REFERENCES d_nivel
(ID_LEVEL),
CONSTRAINT FK_fecha_hcom_v FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_vida:
CREATE INDEX D_USER_H_COMPRA_VIDA_FK_1 ON h_compra_vida(ID_USER);
CREATE INDEX D_PLATAFOFMA_H_COMPRAVIDA_FK_2 ON
h_compra_vida(ID_PLATAFORMA);
CREATE INDEX D_PRODUCTO_H_COMPRAVIDA_FK_3 ON
h_compra_vida(ID_PRODUCTO);
CREATE INDEX D_LEVEL_H_COMPRAVIDA_FK_4 ON h_compra_vida(ID_LEVEL);
CREATE INDEX D_FECHA_H_COMPRAVIDA_FK_5 ON h_compra_vida(id_fecha);

-------
*********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_compra_ayuda.
```

```
-------
**********************************************************************
*
CREATE SEQUENCE h_com_ayuda_ID_comp_ayuda_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_compra_ayuda
CREATE TABLE h_compra_ayuda (
ID_comp_ayuda NUMBER CONSTRAINT PK_ca_a PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_cu_a NOT NULL,
ID_PLATAFORMA NUMBER CONSTRAINT NN_cap_a NOT NULL,
ID_PRODUCTO  NUMBER CONSTRAINT NN_capr_a NOT NULL,
ID_LEVEL NUMBER CONSTRAINT NN_cal_a NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_caf_a NOT NULL,
N_comayuda NUMBER(10,0) CONSTRAINT NN_comayu3 NOT NULL,
CANTIDAD NUMBER(10,0) CONSTRAINT NN_cant3 NOT NULL,
CONSTRAINT FK_user_hcom_a FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_platf_hcom_a FOREIGN KEY (ID_PLATAFORMA) REFERENCES
d_plataforma (ID_PLATAFORMA),
CONSTRAINT FK_prod_hcom_a FOREIGN KEY (ID_PRODUCTO) REFERENCES
d_producto (ID_PRODUCTO),
CONSTRAINT FK_lev_hcom_a FOREIGN KEY (ID_LEVEL) REFERENCES d_nivel
(ID_LEVEL),
CONSTRAINT FK_fecha_hcom_a FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_H_COMPRA_ayuda_FK_1 ON h_compra_ayuda(ID_USER);
CREATE INDEX D_PLATFOFM_H_COMPRAayuda_FK_2 ON
h_compra_ayuda(ID_PLATAFORMA);
CREATE INDEX D_PRODUCTO_H_COMPRAayuda_FK_3 ON
h_compra_ayuda(ID_PRODUCTO);
CREATE INDEX D_LEVEL_H_COMPRAayuda_FK_4 ON h_compra_ayuda(ID_LEVEL);
CREATE INDEX D_FECHA_H_COMPRAayuda_FK_5 ON h_compra_ayuda(id_fecha);

-------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_Transferir_vida.
-------
**********************************************************************
*
CREATE SEQUENCE h_transf_vida_ID_vida_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_Transferir_vida
CREATE TABLE h_Transferir_vida (
ID_transf_vida NUMBER CONSTRAINT PK_tv PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_tv NOT NULL,
ID_REDSOCIAL NUMBER CONSTRAINT NN_rst NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_ftv NOT NULL,
N_transfe NUMBER(10,0) CONSTRAINT NN_transfe3 NOT NULL,
CONSTRAINT FK_user_ht_v FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_rs_ht_v FOREIGN KEY (ID_REDSOCIAL) REFERENCES
d_redsocial (ID_REDSOCIAL),
CONSTRAINT FK_fecha_t_v FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
```

```
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_H_TRANSF_vida_FK_1 ON h_Transferir_vida(ID_USER);
CREATE INDEX D_RS_H_TRANSF_vida_FK_4 ON
h_Transferir_vida(ID_REDSOCIAL);
CREATE INDEX D_FECHA_H_TRANSF_vida_FK_5 ON
h_Transferir_vida(id_fecha);

------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_Agregar_amigo.
-------
**********************************************************************
*
CREATE SEQUENCE h_Agregar_amigo_ID_amigo_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_Agregar_amigo
CREATE TABLE h_Agregar_amigo (
ID_agregar_amigo NUMBER CONSTRAINT PK_aa PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_aa NOT NULL,
ID_REDSOCIAL NUMBER CONSTRAINT NN_rsaa NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_faa NOT NULL,
N_agreami NUMBER(10,0) CONSTRAINT NN_agre3 NOT NULL,
CONSTRAINT FK_user_aa FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_rs_aa FOREIGN KEY (ID_REDSOCIAL) REFERENCES d_redsocial
(ID_REDSOCIAL),
CONSTRAINT FK_fecha_aa FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_H_Agregar_amigo_FK_1 ON h_Agregar_amigo(ID_USER);
CREATE INDEX D_RS_H_Agregar_amigo_FK_4 ON
h_Agregar_amigo(ID_REDSOCIAL);
CREATE INDEX D_FECHA_H_Agregar_amigo_FK_5 ON
h_Agregar_amigo(id_fecha);

-------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_Baja.
-------
**********************************************************************
*
CREATE SEQUENCE h_Baja_ID_amigo_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_Baja
CREATE TABLE h_Baja (
ID_baja NUMBER CONSTRAINT PK_baja PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_baja NOT NULL,
ID_LEVEL NUMBER CONSTRAINT NN_baja1 NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_baja2 NOT NULL,
N_BAJA NUMBER(10,0) CONSTRAINT NN_Baja3 NOT NULL,
```

```
CONSTRAINT FK_user_baja FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_level_baja FOREIGN KEY (ID_LEVEL) REFERENCES d_nivel
(ID_LEVEL),
CONSTRAINT FK_fecha_baja FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_h_Baja_FK_1 ON h_Baja(ID_USER);
CREATE INDEX D_LEVEL_h_baja_FK_1 ON h_Baja(ID_LEVEL);
CREATE INDEX D_FECHA_h_Baja_FK_1 ON h_Baja(id_fecha);

-------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_sup_nivel.
-------
**********************************************************************
*
CREATE SEQUENCE h_sup_nivel_ID_amigo_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_sup_nivel
CREATE TABLE h_sup_nivel (
ID_sup_nivel NUMBER CONSTRAINT PK_sup_nivel PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_sup_nivel NOT NULL,
ID_LEVEL NUMBER CONSTRAINT NN_sup_nivel1 NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_sup_nivel2 NOT NULL,
N_supl NUMBER(10,0) CONSTRAINT NN_supl3 NOT NULL,
CONSTRAINT FK_user_sup_nivel FOREIGN KEY (Id_user) REFERENCES
d_usuario (id_user),
CONSTRAINT FK_level_sup_nivel FOREIGN KEY (ID_LEVEL) REFERENCES
d_nivel (ID_LEVEL),
CONSTRAINT FK_fecha_sup_nivel FOREIGN KEY (id_fecha) REFERENCES
d_fecha (id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_h_sup_nivel_FK_1 ON h_sup_nivel(ID_USER);
CREATE INDEX D_LEVEL_h_sup_nivel_FK_1 ON h_sup_nivel(ID_LEVEL);
CREATE INDEX D_FECHA_h_sup_nivel_FK_1 ON h_sup_nivel(id_fecha);

-------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_UsarVida.
-------
**********************************************************************
*
CREATE SEQUENCE h_UsarVida_ID_amigo_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_UsarVida
CREATE TABLE h_UsarVida (
ID_usarvida NUMBER CONSTRAINT PK_UsarVida PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_UsarVida NOT NULL,
ID_LEVEL NUMBER CONSTRAINT NN_UsarVida1 NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_UsarVida2 NOT NULL,
```

```
N_VIDAUSADA NUMBER(10,0) CONSTRAINT NN_vusa3 NOT NULL,
CONSTRAINT FK_user_UsarVida FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_level_UsarVida FOREIGN KEY (ID_LEVEL) REFERENCES d_nivel
(ID_LEVEL),
CONSTRAINT FK_fecha_UsarVida FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_h_UsarVida_FK_1 ON h_UsarVida(ID_USER);
CREATE INDEX D_LEVEL_h_UsarVida_FK_1 ON h_UsarVida(ID_LEVEL);
CREATE INDEX D_FECHA_h_UsarVida_FK_1 ON h_UsarVida(id_fecha);

-------
**********************************************************************
*
--Creación de la secuencia de la tabla de hecho h_Alta.
-------
**********************************************************************
*
CREATE SEQUENCE h_Alta_ID_amigo_SEQ
MINVALUE 1 MAXVALUE 999999999999999999999999 INCREMENT BY 1 NOCYCLE ;

-- Tabla de Hecho h_Alta
CREATE TABLE h_Alta (
ID_alta NUMBER CONSTRAINT PK_Alta PRIMARY KEY,
ID_USER NUMBER CONSTRAINT NN_Alta NOT NULL,
id_fecha NUMBER(10,0) CONSTRAINT NN_Alta2 NOT NULL,
N_ALTAS NUMBER(10,0) CONSTRAINT NN_Alta3 NOT NULL,
CONSTRAINT FK_user_Alta FOREIGN KEY (Id_user) REFERENCES d_usuario
(id_user),
CONSTRAINT FK_fecha_Alta FOREIGN KEY (id_fecha) REFERENCES d_fecha
(id_fecha)
) ;

----Ahora se crea un índice por cada dimensión en la tabla de hechos
de comprar_ayuda:
CREATE INDEX D_USER_h_Alta_FK_1 ON h_Alta(ID_USER);
CREATE INDEX D_FECHA_h_Alta_FK_1 ON h_Alta(id_fecha);

--
**********************************************************************
*****************************************************
--*************** TRIGGERS del DWH
**********************************************************************
********************
--
**********************************************************************
*****************************************************
CREATE OR REPLACE TRIGGER d_fecha_id_fecha_TRG BEFORE INSERT ON
d_fecha
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.id_fecha IS NULL THEN
SELECT d_fecha_id_fecha_SEQ.NEXTVAL INTO v_newVal FROM DUAL;
```

```
-- If this is the first time this table have been inserted into
(sequence == 1)
IF v_newVal = 1 THEN
--get the max indentity value from the table
SELECT NVL(max(id_fecha),0) INTO v_newVal FROM d_fecha;
v_newVal := v_newVal + 1;
--set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT d_fecha_id_fecha_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
-- assign the value from the sequence to emulate the identity column
:new.id_fecha := v_newVal;
END IF;
END;

---- Trigger de h_compra_vida

CREATE OR REPLACE TRIGGER h_compra_vida_id_compvida_TRG BEFORE INSERT
ON h_compra_vida
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_comp_vida IS NULL THEN
SELECT h_compra_vida_ID_comp_vida_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_comp_vida),0) INTO v_newVal FROM h_compra_vida;
 v_newVal := v_newVal + 1;
--set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_compra_vida_ID_comp_vida_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_comp_vida := v_newVal;
END IF;
END;
---- Trigger de h_compra_ayuda
CREATE OR REPLACE TRIGGER h_comp_ayuda_id_compayuda_TRG BEFORE INSERT
ON h_compra_ayuda
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_comp_ayuda IS NULL THEN
SELECT h_com_ayuda_ID_comp_ayuda_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_comp_ayuda),0) INTO v_newVal FROM h_compra_ayuda;
 v_newVal := v_newVal + 1;
--set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_com_ayuda_ID_comp_ayuda_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
```

```
:new.ID_comp_ayuda := v_newVal;
END IF;
END;

---- Trigger de h_Transferir_vida
CREATE OR REPLACE TRIGGER h_Transferir_vida BEFORE INSERT ON
h_Transferir_vida
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_transf_vida IS NULL THEN
SELECT h_transf_vida_ID_vida_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_transf_vida),0) INTO v_newVal FROM
h_Transferir_vida;
 v_newVal := v_newVal + 1;
--set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_transf_vida_ID_vida_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_transf_vida := v_newVal;
END IF;
END;

---- Trigger de h_Agregar_amigo
CREATE OR REPLACE TRIGGER h_agregar_amigo_id BEFORE INSERT ON
h_Agregar_amigo
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_agregar_amigo IS NULL THEN
SELECT h_Agregar_amigo_ID_amigo_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_agregar_amigo),0) INTO v_newVal FROM
h_Agregar_amigo;
 v_newVal := v_newVal + 1;
 --set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_Agregar_amigo_ID_amigo_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_agregar_amigo := v_newVal;
END IF;
END;

---- Trigger de h_Baja_usuario
CREATE OR REPLACE TRIGGER h_Baja_usuario_id BEFORE INSERT ON h_Baja
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
```

```
IF INSERTING AND :new.ID_baja IS NULL THEN
SELECT h_Baja_ID_amigo_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_baja),0) INTO v_newVal FROM h_Baja;
 v_newVal := v_newVal + 1;
 --set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_Baja_ID_amigo_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_baja := v_newVal;
END IF;
END;

---- Trigger de h_Sup_Nivel
CREATE OR REPLACE TRIGGER h_superar_nivel_id BEFORE INSERT ON
h_sup_nivel
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_sup_nivel IS NULL THEN
SELECT h_sup_nivel_ID_amigo_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_sup_nivel),0) INTO v_newVal FROM h_sup_nivel;
 v_newVal := v_newVal + 1;
 --set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_sup_nivel_ID_amigo_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_sup_nivel := v_newVal;
END IF;
END;

---- Trigger de h_Usar_Vida
CREATE OR REPLACE TRIGGER h_usar_vida_id BEFORE INSERT ON h_UsarVida
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_usarvida IS NULL THEN
SELECT h_UsarVida_ID_amigo_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_usarvida),0) INTO v_newVal FROM h_UsarVida;
 v_newVal := v_newVal + 1;
 --set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_UsarVida_ID_amigo_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_usarvida := v_newVal;
END IF;
```

```
END;

---- Trigger de h_alta_id
CREATE OR REPLACE TRIGGER h_alta_id BEFORE INSERT ON h_Alta
FOR EACH ROW
DECLARE
v_newVal NUMBER(12) := 0;
v_incval NUMBER(12) := 0;
BEGIN
IF INSERTING AND :new.ID_alta IS NULL THEN
SELECT h_Alta_ID_amigo_SEQ.NEXTVAL INTO v_newVal FROM DUAL;

IF v_newVal = 1 THEN
 SELECT NVL(max(ID_alta),0) INTO v_newVal FROM h_Alta;
 v_newVal := v_newVal + 1;
 --set the sequence to that value
LOOP
EXIT WHEN v_incval>=v_newVal;
SELECT h_Alta_ID_amigo_SEQ.nextval INTO v_incval FROM dual;
END LOOP;
END IF;
:new.ID_alta := v_newVal;
END IF;
END;
```

7. Capítulo III. Implementación.

Una vez creada la estructura de la base de datos con todas sus tablas y relaciones en el modelo físico del capitulo anterior, pasamos a desarrollar e implementar los procesos almacenados que nos permitirán el acceso a los datos y la creación de los procesos ETL que realizarán la extracción y carga para el DWH.

7.1. Sistema Operacional de la BD.

- Se crean secuencias y triggers o disparadores necesarios en cada tabla para generar las claves artificiales (Las secuencias, triggers y disparadores del DWH se encuentran incluidas en el modelo físico del apartado anterior):

```
--Sequence & trigger PLATAFORMA
CREATE SEQUENCE seq_plataforma INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_plataforma
BEFORE INSERT ON PLATAFORMA
FOR EACH ROW
BEGIN
SELECT seq_plataforma.NEXTVAL INTO :NEW.id_plataforma
FROM DUAL;
END insertar_plataforma;
--Sequence & trigger REDSOCIAL
CREATE SEQUENCE seq_redsocial INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_redsocial
BEFORE INSERT ON REDSOCIAL
FOR EACH ROW
BEGIN
SELECT seq_redsocial.NEXTVAL INTO :NEW.id_redsocial
FROM DUAL;
END insertar_redsocial;
```

```
--Sequence & trigger DESCLEVEL
CREATE SEQUENCE seq_desclevel INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_desclevel
BEFORE INSERT ON DESCLEVEL
FOR EACH ROW
BEGIN
SELECT seq_desclevel.NEXTVAL INTO :NEW.id_level
FROM DUAL;
END insertar_desclevel;
--Sequence & trigger USUARIOS
CREATE SEQUENCE seq_user INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_user
BEFORE INSERT ON USUARIOS
FOR EACH ROW
BEGIN
SELECT seq_user.NEXTVAL INTO :NEW.id_user
FROM DUAL;
END insertar_user;
--Sequence & trigger AMIGOS
CREATE SEQUENCE seq_amigo INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_amigo
BEFORE INSERT ON AMIGOS
FOR EACH ROW
BEGIN
SELECT seq_amigo.NEXTVAL INTO :NEW.id_amigo
FROM DUAL;
END insertar_amigo;
--Sequence & trigger VIDAS
CREATE SEQUENCE seq_vida INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_vida
BEFORE INSERT ON VIDAS
FOR EACH ROW
BEGIN
SELECT seq_vida.NEXTVAL INTO :NEW.id_vida
FROM DUAL;
END insertar_vida;

--Sequence & trigger TRANSCAYUDAAMIGOS
CREATE SEQUENCE seq_trans_ayuda INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_trans_ayuda
BEFORE INSERT ON TRANSCAYUDAAMIGOS
FOR EACH ROW
BEGIN
SELECT seq_trans_ayuda.NEXTVAL INTO :NEW.id_trans_ayuda
FROM DUAL;
END insertar_trans_ayuda;

--Sequence & trigger PRODUCTOS
CREATE SEQUENCE seq_producto INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_producto
BEFORE INSERT ON PRODUCTOS
FOR EACH ROW
BEGIN
SELECT seq_producto.NEXTVAL INTO :NEW.id_producto
FROM DUAL;
END insertar_producto;

--Sequence & trigger ERROR_COMPRA
CREATE SEQUENCE seq_error_compra INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_error_compra
BEFORE INSERT ON ERROR_COMPRA
```

```
FOR EACH ROW
BEGIN
SELECT seq_error_compra.NEXTVAL INTO :NEW.id_error_compra
FROM DUAL;
END insertar_error_compra;

--Sequence & trigger COMPRAS
CREATE SEQUENCE seq_compra INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_compra
BEFORE INSERT ON COMPRAS
FOR EACH ROW
BEGIN
SELECT seq_compra.NEXTVAL INTO :NEW.id_compra
FROM DUAL;
END insertar_compra;


--Sequence & trigger AYUDA
CREATE SEQUENCE seq_ayuda INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_ayuda
BEFORE INSERT ON AYUDA
FOR EACH ROW
BEGIN
SELECT seq_ayuda.NEXTVAL INTO :NEW.id_ayuda
FROM DUAL;
END insertar_ayuda;


--Sequence & trigger PROCESOS_LOG
CREATE SEQUENCE seq_proceso_log INCREMENT BY 1 START WITH 1;
CREATE OR REPLACE TRIGGER insertar_proceso_log
BEFORE INSERT ON PROCESOS_LOG
FOR EACH ROW
BEGIN
SELECT seq_proceso_log.NEXTVAL INTO :NEW.id_proceso_log
FROM DUAL;
END insertar_proceso_log;
```

- Se cargan los siguientes datos iniciales de las tablas estáticas que se encuentran en el fichero sql "4_Carga_Datos_inicial_BD":

PLATAFORMA

ID_PLATAFORMA	DESC_PLATAFORMA	TIPO_PLATAFORMA
1	WEB	1
2	MOVIL	2

REDSOCIAL

ID_REDSOCIAL	NOM_REDSOCIAL
1	Facebook
2	Google plus
3	Twitter
4	Tuenty

DESCLEVEL

ID_LEVEL	NOM_LEVEL	DESC_LEVEL	START_DATE	END_DATE

1	Nivel 1	Cazar a Barcenas.	SYSDATE	NULL
2	Nivel 2	Ahora toca Aznar.	SYSDATE	NULL
3	Nivel 3	¿qué es de la ESPE?.	SYSDATE	NULL
4	Nivel 4	Lobies Fuera.	SYSDATE	NULL
5	Nivel 5	Por fin sin monarquía.	SYSDATE	NULL

PRODUCTOS

ID_PRODUCTO	PRECIO	ID_TIPO_COMPRA (1-vida;2-ayuda)	DESC_PRODUCTO	START_DATE	END_DATE
1	0.2	2	Ayuda nivel 1	SYSDATE	NULL
2	0.40	2	Ayuda nivel 2	SYSDATE	NULL
3	0.60	2	Ayuda nivel 3	SYSDATE	NULL
4	0.80	2	Ayuda nivel 4	SYSDATE	NULL
5	1	2	Ayuda nivel 5	SYSDATE	NULL
6	1	1	Vida nivel 1	SYSDATE	NULL
7	1,5	1	Vida nivel 2	SYSDATE	NULL
8	2	1	Vida nivel 3	SYSDATE	NULL
9	2,5	1	Vida nivel 4	SYSDATE	NULL
10	3	1	Vida nivel 5	SYSDATE	NULL

ERROR_COMPRA

ID_ERROR_COMPRA	DESC_ERROR_COMPRA
1	Cuenta de pago incorrecta.
2	Hubo un problema en la transacción.
3	Producto caducado.
4	Ayuda Caducada.

PROCESOS

ID_PROCESO	NOM_PROCESO	DESC_PROCESO
1	AGREGAR_AMIGO	Agraga un amigo a su red.
2	ALTA_USUARIO	Alta de un usuario
3	FINALIZAR_VIDA	Usuario agota una vida
4	SUPERAR_NIVEL	Usuario supera un nivel
5	TRANSFERIR_VIDA	Usuario transfiere vida
6	USAR_VIDA	Usuario utiliza vida
7	COMPRAR_VIDA_AYUDA	Compra vida o ayuda
8	BAJA_USUARIO	Baja de un usuario
9	REACTIVAR_USUARIO	Reactivación de un usuario.

– **PROCESOS ALMACENADOS.**

Aunque se podría haber optado por diseñar un flujo de procesos que incluya funciones, triggers y procedures, se ha optado por la creación de varios procedures que utilizan todas las técnicas del lenguaje de programación PL/SQL tales como cursores, etc.
A continuación se muestra un listado de todos los procesos almacenados creados en el sistema para el funcionamiento del videojuego con sus características.

PROCESOS_ALTA_USUARIO

Parámetros de Entrada	DESCRIPCION_PROCESO
Sexo, edad, ciudad, profesión, número de cuenta y login	Se trata de un procedure que da de alta a a un usuario nuevo en el sistema recibiendo los datos de entrada, los inserta en la tabla USUARIOS, crea 5 entradas en la tabla VIDAS(una por cada vida que dispone en un día) y finalmente incluye una entrada en la tabla Dailylevel indicando que el usuario se encuentra en el nivel 1

PROCESOS_BAJA_USUARIO

Parámetros de Entrada	DESCRIPCION_PROCESO
id_user	Proceso que recibe como parámetro el id del usuario, finaliza todas las filas de las tablas dailylevel, usada, vidas, users y amigos actualizando la fecha end_date con la fecha actual de baja.

PROCESOS_ACTUALIZAR_VIDA_DIARIO

Parámetros de Entrada	DESCRIPCION_PROCESO
No recibe parámetros.	Se trata de un proceso almacenado que se debe planificado para ejecutarse diariamnte a las doce de la madrugada. Si un usuario está usando una vida, ésta no se toca y se le permite siga jugando y disfrutando de la misma, pero el resto de vidas que tenga sin usar serán dadas de baja para dar a cada usuario 5 vidas pertenecientes al día actual.

PROCESOS_AGREGAR_AMIGO

Parámetros de Entrada	DESCRIPCION_PROCESO
id_user, id_user del amigo, id_redsocial	EL proceso agrega un amigo en la tabla de Amigos. Tanto el usuario como el amigo a agregar deben existir en la tabla de USUARIOS.

PROCESOS_COMPRAR_VIDA_AYUDA

Parámetros de Entrada	DESCRIPCION_PROCESO
id_user, id_plataforma, id_tipocompra, cantidad	El proceso recibe por parámetro el id del usuario que realiza la compra, el tipo de compra que quiere realizar, si se trata de ayuda o la compra de una vida y la cantidad. En función de los parámetros de entrada, el procedure calcula el nivel en el que se encuentra el usuario y una vez calculado, se obtiene el ptoducto que necesita, bien sea una vida del nivel 3 si se encuentra en tal nivel o bien una compra de una vida. Posteriormente se actualiza la información en la tabla compras, vidas, ayuda e incluir_compras.

PROCESOS_ESTADISTICAS

Parámetros de Entrada	DESCRIPCION_PROCESO
No recibe parámetros	Proceso de estadísticas, interno para el administrador de

	sistemas que ejecuta el paquete de oracle "DBMS_STATS.GATHER_SCHEMA_STATS " para todo el esquema creado y así poder mejorar el rendimiento de la base de datos y los planes de ejecución de cada consulta.

PROCESOS_FINALIZAR_VIDA

Parámetros de Entrada	DESCRIPCION_PROCESO
Id_user	Proceso que finaliza la vida que esté usando el usuario con id_user recibido como parámetro. Actualiza el end_date de la vida en las tablas VIDAS y USADA.

PROCESOS_REACTIVAR_VIDA

Parámetros de Entrada	DESCRIPCION_PROCESO
Id_user	Proceso que reactiva un usuario que estaba dado de baja para que vuelva a estar activo y pueda participar en el juego. Es parecido a una alta nueva solo que el usuario ya existía.

PROCESOS_USAR_VIDA

Parámetros de Entrada	DESCRIPCION_PROCESO
Id_user	Proceso que recibe por parámetro de entrada el id de un usuario que al disponerse a jugar o comenzar una partida, se actualiza la información de la primera vida disponible. Actualiza la tabla Usada donde se almacena la fecha/hora en la que comienza a usar la vida y la fecha. También actualiza la información en las tablas dailylevel y vidas.

PROCESOS_SUPERAR_NIVEL

Parámetros de Entrada	DESCRIPCION_PROCESO
Id_user	Proceso que actualiza las tablas relacionadas con la superación de un nivel tales como dailylevel. Este proceso recibe como parámetro de entrada el usuario que supera un nivel.

PROCESOS_TRANSFERIR_NIVEL

Parámetros de Entrada	DESCRIPCION_PROCESO
Id_user (emisor), id_user(receptor), id_redsocial a través de la cual se realiza la transferencia de una vida.	Proceso que realiza la transferencia de una vida de un usuario a otro al que está conectado a través de una red social. El usuario debe disponer de vidas para poder realizar la transferencia y debe estar conectado como amigo a través de una red social.

Para cargar y compilar los procesos almacenados en la Base de datos, se ha de ejecutar desde cualquier consola sql el script sql "7_Procesos_Almacenados_Procedures".

Sqlplus videoadmin/abc1234@xe @7_Procesos_Almacenados_Procedures.sql

7.1. Sistema Operacional de la DWH.

- Una vez implantada la base de datos, pasamos a explicar la implantación del DWH. Previamente se debería haber ejecutado y cargado en la BD el modelo de datos del DWH que se encuentra en el fichero sql "5_Creación_Tablas_ModeloDatos_DWH.sql"
- Ahora pasamos a explicar los procesos ETL diseñados para cargar los datos en el DWH.
- Debido problemas de recursos, se ha optado por instalar el DWH dentro de la misma BD oracle y el mismo esquema. Así pues se diseñan los siguientes procesos ETL que cargan los datos desde las tablas de la base de datos del videojuego hasta la dimensiones y tablas de hecho del DWH.

7.1.1. Diseño de procesos ETL.

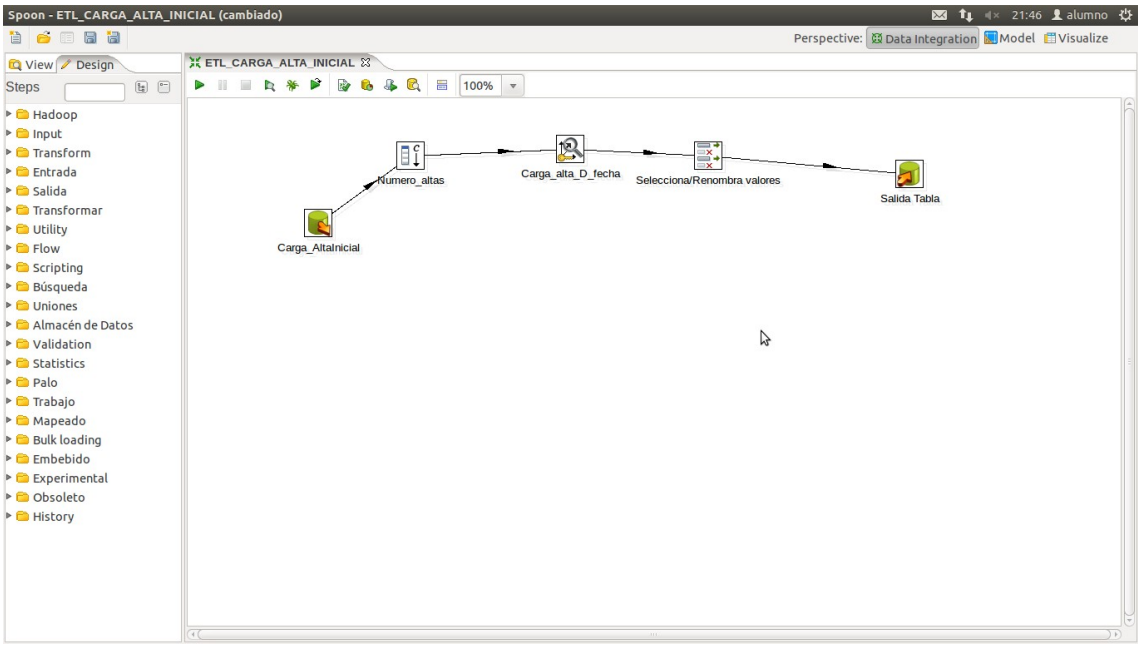
Como ya se ha comentado anteriormente, usaremos el sistema de DWH de Pentaho que usa la aplicación SPOOM de Ketel para diseñar procesos ETL de extracción, transformación y carga.

Los primeros procesos que se han diseñado son los que cargan los datos de las tablas estáticas en las dimensiones de d_plataforma, d_nivel, d_producto y d_redsocial. Finalmente se diseñan los procesos que cargan la información en el resto de dimensiones tales como d_fecha ,d_usuario y las tablas de hecho h_compras_vida, h_compras_ayuda, H_transferencias, h_agregar_amigos, h_altas, h_baja, h_sup_nivel y h_usarvida.

A continuación se muestran las pantallas de los distintos procesos ETL qu ese han diseñano:

ETL_CARGA_ALTA_INICIAL

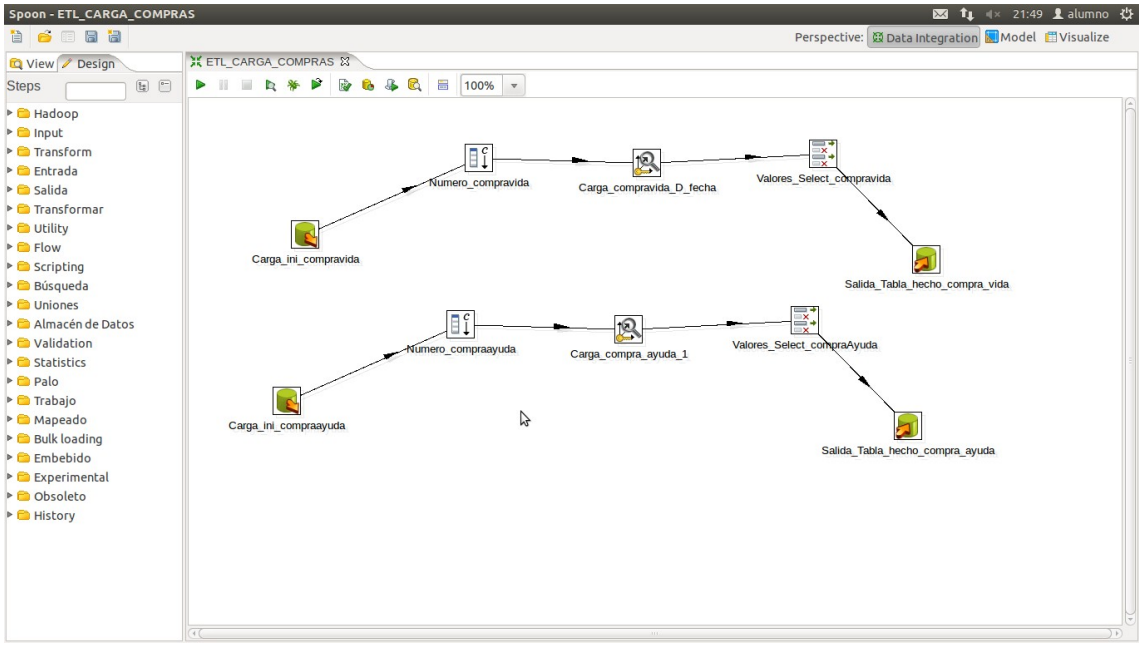
Obtiene los datos de las tablas de la BD Usuarios y a través de la fecha de alta carga la información de fecha en la dimensión d_fecha. Finalmente carga la información de las dimensiones d_fecha y d_usuarios en la tabla de hecho h_alta.



Para el ETL que realiza la carga de las bajas en la tabla de hechos de h_bajas, se realiza la misma operación solo que teniendo en cuenta la fecha end_date de la tabla usuarios.

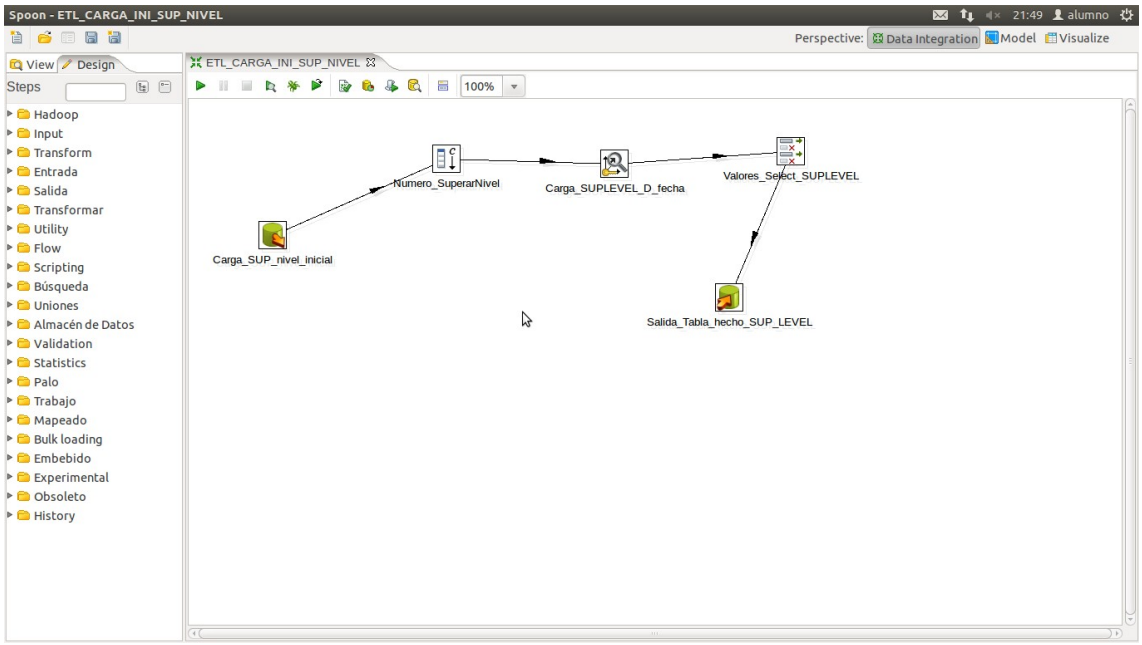
ETL_CARGA_COMPRAS

Este etl realiza la carga tanto para la tabla de hechos h_compra_vida como para la tabla de hechos H_compra_ayuda.



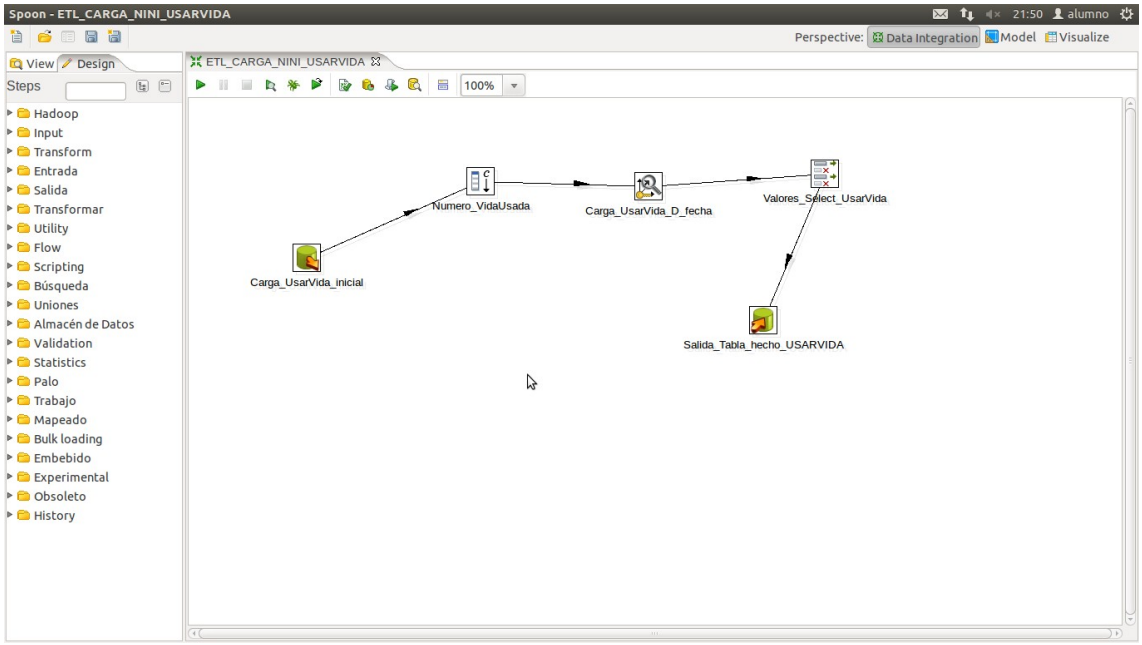
ETL_CARGA_INI_SUP_NIVEL

ETL que carga la información de la BD a través de las dimensiones correspondientes en la tabla de hechos h_sup_nivel.



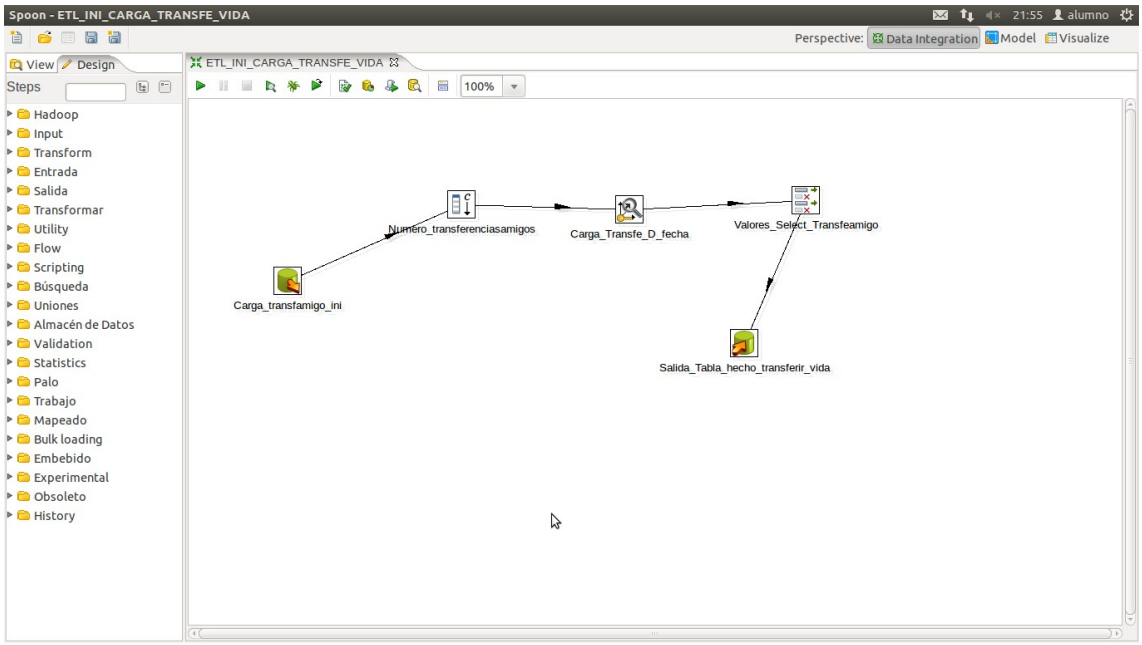
ETL_CARGA_INI_USARVIDA

ETL que carga la información de la BD a través de las dimensiones correspondientes en la tabla de hechos h_usarvida.



ETL_INI_CARGA_TRANSFE_VIDA

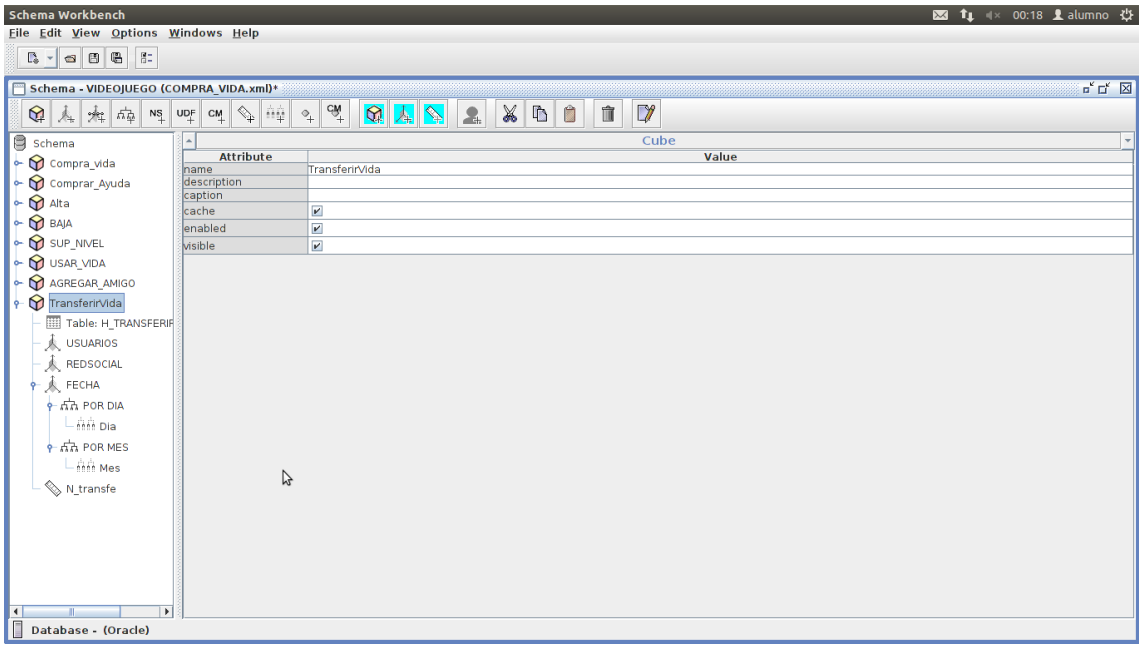
ETL que carga la información de la BD a través de las dimensiones correspondientes en la tabla de hechos h_transfvida.



Para el resto de ETLs se realiza la carga de dimensiones de manera sencilla. En el SW a entregar en el proyecto se incluyen los ETL necesarios para examinar y ver internamente.

7.1.2. Diseño Esquema OLAP.

Se utiliza la aplicación Pentaho Schema WorkBench para crear los cubos con sus dimensiones, tablas de hecho, Jerarquías y niveles necesarios. La siguiente pantalla muestra los cubos creados para nuestro DWH.



EL siguiente XML contiene toda la configuración a cargar en Pentaho Schema WorkBench con todos cubos, dimensiones y análisis OLAP necesarios para analizar los

datos de la BD en el DWH. Esteb XML se ha obtenido de la imagen anterior que muestra todos los cubos creados.


```
<Schema name="VIDEOJUEGO">
  <Cube name="Compra_vida" visible="true" cache="true" enabled="true">
    <Table name="H_COMPRA_VIDA" schema="VIDEOADMIN">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_PLATAFORMA"
highCardinality="false" name="Plataforma">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" highCardinality="false"
name="USUARIOS">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_LEVEL" highCardinality="false"
name="NIVEL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_PRODUCTO"
highCardinality="false" name="PRODUCTO">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" highCardinality="false"
name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MESES" visible="true" hasAll="true" allMemberName="Meses"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES"
nameColumn="DESC_MES" ordinalColumn="MES" type="String" uniqueMembers="false"
levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
      <Hierarchy name="fecha" visible="true" hasAll="true" allMemberName="fecha"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="fecha" visible="true" table="D_FECHA" column="ID_FECHA"
nameColumn="ID_FECHA" parentColumn="ID_FECHA" type="String" uniqueMembers="false"
levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="COMPRAS" column="CANTIDAD" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
  <Cube name="Comprar_Ayuda" visible="true" cache="true" enabled="true">
    <Table name="H_COMPRA_AYUDA" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="false" foreignKey="ID_PLATAFORMA"
name="Plataforma">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_LEVEL" name="NIVEL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_PRODUCTO"
name="PRODUCTO">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
```

```
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="COMPRAAyuda" column="CANTIDAD" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
  <Cube name="Alta" visible="true" cache="true" enabled="true">
   <Table name="H_ALTA" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_ALTAS" column="N_ALTAS" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
  <Cube name="BAJA" visible="true" cache="true" enabled="true">
   <Table name="H_BAJA" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
 <Dimension type="StandardDimension" visible="true" foreignKey="ID_LEVEL" name="NIVEL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_BAJA" column="N_BAJA" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
 <Cube name="SUP_NIVEL" visible="true" cache="true" enabled="true">
```

```
   <Table name="h_sup_nivel" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
 <Dimension type="StandardDimension" visible="true" foreignKey="ID_LEVEL" name="NIVEL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_SUPL" column="N_SUPL" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
<Cube name="USAR_VIDA" visible="true" cache="true" enabled="true">
   <Table name="h_UsarVida" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
 <Dimension type="StandardDimension" visible="true" foreignKey="ID_LEVEL" name="NIVEL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_VIDAUSADA" column="N_VIDAUSADA" aggregator="distinct-count"
visible="true">
    </Measure>
  </Cube>
<Cube name="AGREGAR_AMIGO" visible="true" cache="true" enabled="true">
   <Table name="h_Agregar_amigo" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
 <Dimension type="StandardDimension" visible="true" foreignKey="ID_REDSOCIAL"
name="REDSOCIAL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
```

```
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_agreami" column="N_agreami" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
<Cube name="TransferirVida" visible="true" cache="true" enabled="true">
   <Table name="h_Transferir_vida" schema="VIDEOADMIN" alias="">
    </Table>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_USER" name="USUARIOS">
    </Dimension>
 <Dimension type="StandardDimension" visible="true" foreignKey="ID_REDSOCIAL"
name="REDSOCIAL">
    </Dimension>
    <Dimension type="StandardDimension" visible="true" foreignKey="ID_FECHA" name="FECHA">
      <Hierarchy name="POR DIA" visible="true" hasAll="true" allMemberName="Dias"
primaryKey="ID_FECHA" primaryKeyTable="ID_FECHA">
        <Level name="Dia" visible="true" table="D_FECHA" column="DESC_DIA"
nameColumn="DESC_DIA" type="String" uniqueMembers="false" levelType="Regular">
        </Level>
      </Hierarchy>
      <Hierarchy name="POR MES" visible="true" hasAll="true" primaryKey="ID_FECHA"
primaryKeyTable="ID_FECHA">
        <Level name="Mes" visible="true" table="D_FECHA" column="DESC_MES" ordinalColumn="MES"
parentColumn="DESC_MES" type="String" uniqueMembers="false" levelType="Regular"
hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="N_transfe" column="N_transfe" aggregator="distinct-count" visible="true">
    </Measure>
  </Cube>
</Schema>
```


8. Referencias.

Las referencias utilizadas hasta el momento son las siguientes:

www.oracle.com

www.proyectosfindecarrera.com

www.wikipedia.org

www.uoc.edu

www.slideshare.net/rclariso/orientacions-generals-per-al-treball-final-a-la-uoc