Limitaciones de la aplicabilidad del modelo relacional en videojuegos para el aprendizaje. Caso: Videojuego SAMI

Ponencia presentada en CIINATIC 2016

Urbano E. Gómez Prada [(1)], Oscar F. Gómez Sandoval [(1)]

Facultad de Ingeniería de Sistemas e Informática, Universidad Pontificia Bolivariana (UPB) – Seccional Bucaramanga, Autopista a Piedecuesta Km 7, Edificio I, Oficina 301, Bucaramanga, Colombia (e-mail: urbano.gomez@upb.edu.co; oscar.gomezs@upb.edu.co)

Resumen

Este artículo presenta el análisis de alternativas de persistencia realizado para el proyecto SAMI, un videojuego de sistemas de producción bovina enfocado al entretenimiento y el aprendizaje en el cual los jugadores pueden contrastar los resultados de sus partidas jugadas con resultados que podrían llegar a obtenerse si se sigue un proceso óptimo de manejo de los recursos. El diseño del modelo de negocio de SAMI contiene patrones de eventos y de herencia de clases que requieren de un modelado particular bajo el modelo relacional o del uso de otros paradigmas de almacenamiento de información.

Palabras Clave: Videojuego, Aprendizaje, Base de Datos, Sincronización.

I.INTRODUCCIÓN

SAMI es un videojuego de sistemas de producción bovina enfocado al entretenimiento y el aprendizaje que brinda una interacción cercana con la realidad. El juego busca presentar a los usuarios los recursos suficientes para desarrollar actividades lúdicas que mantengan el interés y promuevan un aprendizaje autónomo para el desarrollo de habilidades cognitivas.

El videojuego opera bajo la arquitectura cliente/servidor asincrónico. En los clientes: se juega y en el servidor se ofrece un espacio para la realimentación conformado por un modelo de simulación y un portal web soportado en la estructura Modelo-Vista-Controlador (MVC) que consolida los resultados de las partidas de los jugadores y genera informes a interesados. Para realizar la realimentación es necesario migrar los datos obtenidos por el usuario en el cliente para compararlos con los estimados por el modelo de simulación en el servidor.

SAMI hace parte de un proyecto de Investigación de la Facultad de Ingeniería de Sistemas e Informática en la UPB Bucaramanga.

II. MARCO DE REFERENCIA

1.1 Descripción de SAMI

SAMI es un videojuego en el cual el jugador debe administrar una finca de producción bovina con el fin de obtener ganancias económicas. El juego presenta diversas posibilidades de acción que se puede realizar, de manera que el jugador debe tomar decisiones sobre cómo invertir sus recursos monetarios limitados para maximizar la eficiencia de su finca.

Fig. 1. Interfaz de SAMI.

La **¡Error! No se encuentra el origen de la referencia.** presenta una pantalla del videojuego, en la cual se observan los siguientes elementos:

- Tres animales en el sistema productivo
- Las acciones que el usuario puede ejecutar (parte superior). Estas acciones son las que

deben ser enviadas al terminar la partida al servidor, como se explicará más adelante.

- El detalle del animal seleccionado (lado derecho), en donde se pueden apreciar datos como el grupo etario del animal, su edad, el peso, la cantidad de agua y comida asignada y la cantidad de dinero disponible.

SAMI está siendo desarrollado como un Ambiente de Aprendizaje, que, según [3], debe tener como objetivo la generación de conocimiento, habilidades o aptitudes en una situación real que se está simulando. En este caso, el enfoque es hacia los sistemas productivos de bovinos los cuales tienen una alta complejidad administrativa que debe ser comprendida [5], debido a que para su funcionamiento requieren de diversos recursos e insumos, su administración y gestión del proceso productivo y reproductivo [4].

Para lograr su objetivo de otorgar al jugador habilidades en la administración de un sistema productivo bovino, SAMI debe almacenar la información de las partidas jugadas y compararlas con una simulación que optimice los resultados, es decir, contrastar las decisiones del jugador con decisiones que tendrían que haberse hecho.

El anterior proceso de contraste se puede realizar a partir de dos componentes básicos de SAMI: la interfaz de usuario, donde el jugador toma decisiones sobre su sistema productivo y observa las consecuencias de éstas, y una simulación de las decisiones ideales que podría tomar un usuario, la cual se ejecuta en un servidor cuando el jugador lo solicite. Ambos componentes operan a partir de un conjunto de ecuaciones que están soportadas en un modelo de simulación que describe parte del sistema, teniendo en cuenta algunas restricciones que decidieron considerarse como variables exógenas en el modelo, tales como ineficiencia de natalidad, diferentes tipos de alimento, diversos tipos de enfermedad, entre otras [1].

1.2 Base de Datos Relacional (BDR)

Según [6], una base de datos relacional es una colección electrónica de datos organizada por tablas y campos, la cual es gestionada por un software denominado gestor de base de datos, que gestiona los accesos a los datos.

Las BDR buscan ofrecer las siguientes características, siempre y cuando sean diseñadas e implementadas en los gestores de manera correcta [6]:

- Atomicidad: Que las transacciones que se realicen sean completas.
- Consistencia o Integridad: Ejecución de operaciones solo si mantienen la Integridad.
- Aislamiento: Aseguramiento de que una operación no puede afectar a otras.
- Durabilidad o Persistencia: Aseguramiento de que una operación persistirá en la base de datos así el sistema falle.

Además, a las BDR se les aplica un proceso denominado “normalización” que busca evitar la existencia de datos duplicados, mejorando tiempos de respuesta y facilitando la administración de la información. Sin embargo, en ciertos casos como el estudiado en este artículo, es posible considerar permitir datos redundantes para mejorar el desempeño de las aplicaciones que hagan uso de la base de datos.

1.3 Alternativas para el modelo entidad-relación cuando hay herencia en el diagrama de clases

Debido a las diferencias de los modelos relacional y orientado a objetos, existen diversas formas de modelar clases heredadas en una base de datos. En [7] se plantean alternativas de diseño para casos como estos:

1. Herencia de Tabla Única (*Single Table Inheritance*): en este tipo de herencia, todos los atributos de las clases originales son guardados en una tabla única: la tabla padre.
2. Herencia de Tablas Concretas (*Concrete Table Inheritance*): en este tipo de herencia, se crea una tabla por cada clase hija, y como datos se toman los atributos propios de cada clase, lo que incluye los atributos heredados del padre.
3. Herencia de Tablas de Clases (*Class Table Inheritance*): en este tipo de herencia, se crea una tabla por cada clase existente, incluyendo la clase padre. Las tablas sólo tendrán los atributos propios de la respectiva clase, sin incluir los atributos heredados.

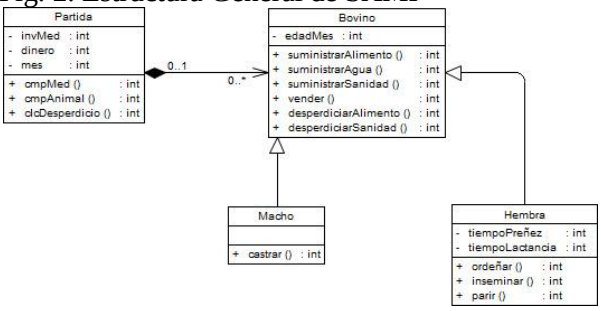
III. Discusión

En SAMI, el jugador debe comprar bovinos y criarlos para obtener beneficios. Durante el proceso, debe tomar decisiones teniendo en cuenta los recursos monetarios disponibles y el estado de cada uno de sus animales, que pueden variar de diversas formas resumidas en [2], así: aumento o pérdida de peso, producción de leche, adquisición de enfermedad, cambio de grupo etario, etc. La interacción del jugador con el sistema que administra se realiza por medio de acciones como las siguientes:

1) Comprar.
2) Suministrar Alimento.
3) Suministrar Agua.
4) Suministrar Medicamento.
5) Ordeñar.
6) Inseminar.
7) Vender.

Un esquema general de la estructura de SAMI es presentado en la Fig. 2, donde se aprecia que ciertas características de los bovinos son únicas según el genero. También se observan ciertos datos importantes que deben tenerse en cuenta para cuando llegue el momento de observar los resultados del jugador.

Fig. 2. Estructura General de SAMI

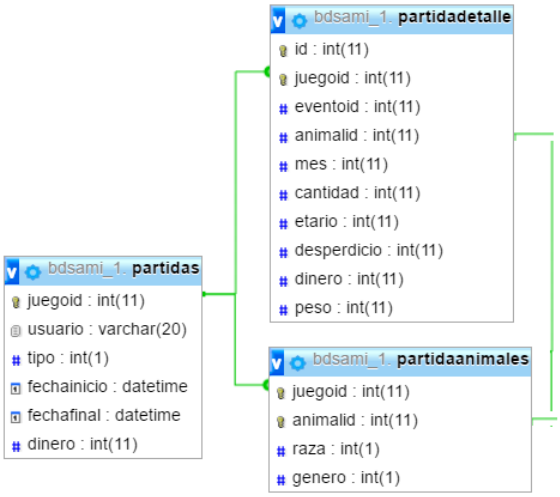


Se plantearon tres posibilidades para almacenar en el cliente los datos que genera cada jugada:

1. Guardar cada evento del mes en un registro
2. Guardar cada animal en el mes en un registro
3. Guardar en un archivo XML solo eventos de cada animal que tenían valor.

Los esquemas de solución son presentados respectivamente en Fig. 3 Fig, 4 y la Fig. 4. En la Fig. 3 se tiene la tabla "partidas", para cada una hay diferentes eventos que se almacenan en "partidadetalle" los cuales tendría asociados los animales en "partidaanimales".

Fig. 3 Entidad- Relación por Evento.



En la Fig. 4 se tiene la tabla "partidas", para cada una hay un detalle por Animal ("partidadetalle") que guarda las cantidades registradas por mes y otra de eventos ("partidahembras") para guardar lo que es particular solo a ellas.

Fig. 4 Entidad- Relación por Animal

En la **¡Error! La autoreferencia al marcador no es válida.**, se observan las etiquetas determinan el mes y animal y las decisiones que hicieron parte de cada mes junto a los valores asignados, como alimento, agua en cada mes y el evento inseminar de la vaca 2 en el mes 2.

Fig. 5. Archivo por Evento por Animal

```
<partida>
  <tiempo id="1">
    <vaca id="1">
      <alimento>20</alimento>
      <agua>10</agua>
   </vaca>
  </tiempo>
  <tiempo id="2">
    <vaca id="1">
      <alimento>20</alimento>
      <agua>10</agua>
   </vaca>
    <vaca id="2">
      <alimento>20</alimento>
      <agua>10</agua>
      <inseminar />
   </vaca>
  </tiempo>
</partida>
```

En la Fig. 6, es presentado un ejemplo de datos enviados con la alternativa uno:

- En la primera tabla (“partidas”), los datos de una partida jugada por “Pepito”.
- En la segunda tabla (“Animales por Partida”) se aprecian los tres animales con los que el usuario ha interactuado.
- En la tercera tabla (“Detalle de la partida”), cada fila corresponde a un evento, en la fila 4 se observa que se compró el animal 1 de 220 Kg de Peso y en cuanto quedo el dinero en el sistema de producción para continuar jugando, en la fila 5 se puede apreciar que se suministraron 20 Kg de alimento, en la fila 8, suministró alimento, y se presentó desperdicio (esta columna empieza a tener importancia para mostrar errores de administración).

En la

, es presentado un ejemplo de los datos enviados con la alternativa 2:

- En “Partidas” están los datos de una jugada llevada a cabo por “Pepito”.
- En la tabla “Detalle de la Partida” se observa que en el mes uno, el jugador suministró 20 Kg de Alimento, 10 Litros de Agua y no suministró Medicamento.
- En la tabla “Detalle de las Hembras” se aprecia que en el mes 2, inseminó al animal 1 y en el mes 11, la vaca parió y se ordeñó, produciendo 5 Litros de Leche.

Las dos primeras opciones requieren de una base de datos *sqllite* en donde se van agregando registros cada vez que el jugador genera un evento o se cambié de mes, posteriormente los datos son exportados al servidor en un gestor *Mysql*.

Para la tercera opción, el archivo es leído y sus datos son guardados en una BDR normalizada como la que se presenta en la
Fig. *8*. En la tabla “partidas” se guarda fecha y jugador y se agregan los “animales” y para cada uno, se asignan las cantidades de alimento, agua y medicamento que decidió el jugador (“envios”) y se guardan los “eventos” tales como inseminar u ordeñar según las condiciones del animal.

Fig. 6 Ejemplo para registro por Evento

Partidas

juegoid	usuario	tipo	fechainicio	fechafinal	dinerofinal
1	Pepito	3			

Animales por partida

juegoid	animalid	raza	tipollegada	mes	genero
1	1	1	1	1	2
1	2	2	1	1	2
1	3	1	2	12	1

Detalle de la partida

id	idjuego	eventoid	animal	mes	cantidad	desperdicio	dinero	peso	
						0	3,600,000		
1	1	1	1	1	1	0	2,940,000	220	Comprar Animales
2	1	2	1	1	20	0	2,910,000	220	Suministrar Alimento
3	1	3	1	1	22	0	2,903,400	220	Suministrar Agua.
4	1	1	2	2	1	0	2,243,400	220	Comprar Animales
5	1	2	1	2	25	50	2,205,900	240	Suministrar Alimento
6	1	3	1	2	22	50	2,199,300	240	Suministrar Agua.
7	1	2	2	2	20	50	2,169,300	220	Suministrar Alimento
8	1	3	2	2	20	50	2,163,300	220	Suministrar Agua.

Fig.7 Ejemplo para registro por Animal

Partidas					
juegoid	usuario	tipo	fechainicio	fechafinal	dinerofinal
1	Pepito	3			

Detalle de la Partida					
idParEve	mes	animal	SumAlimento	SumAgua.	SumSanidad
1	1	1	20	10	0
2	2	1	20	10	0
45	11	1	20	10	0
46	11	3	0	0	50

Detalle de las Hembras						
idpareve	idanimal	mes	Ordeñar	Inseminar	Enfermar	Parir
1	1	1	0	0	0	0
2	1	2	0	1	0	0
45	1	11	5	0	0	1

Con este modelo, es posible extraer los valores para realizar los cálculos con lo que el videojuego hace las comparaciones necesarias sobre las decisiones que se debieron tomar en el sistema de producción.

Cada solución propuesta tiene su inconveniente tal y como se presentó en el marco de referencia, la

IV. CONCLUSIONES

Para sistemas de información en donde se pueda presentar comunicación asíncrona y en la que sea necesaria enviar gran cantidad de datos y guardarlos para realizar comparaciones con resultados esperados, sé debe procurar hacer el modelo tan simple como sea posible evitando almacenar datos que se puedan generar con operaciones.

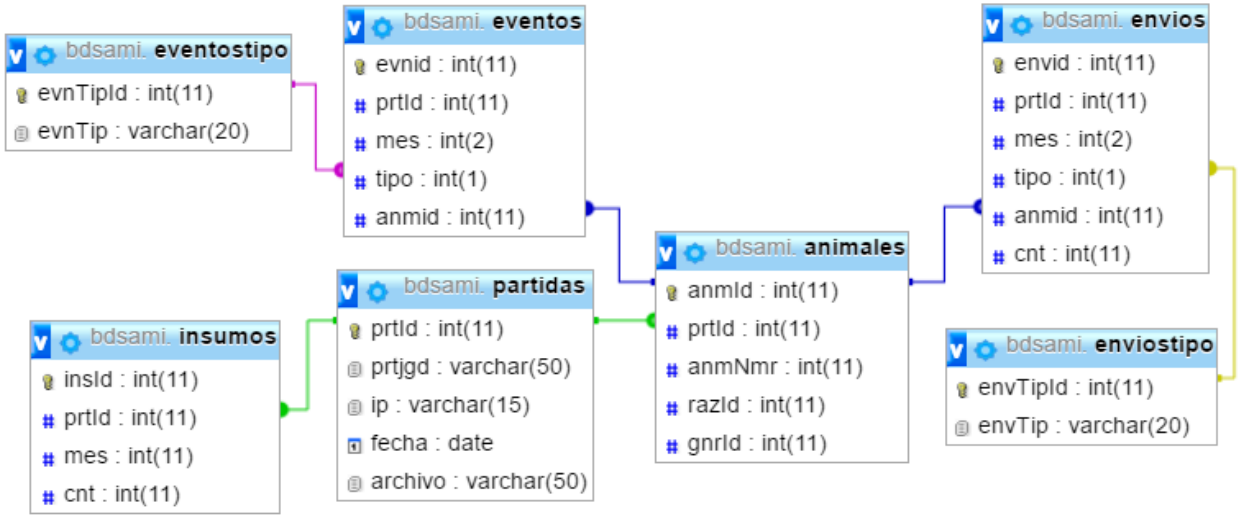
Cuando hay herencia en el diagrama de clases, el modelo entidad-relación no es suficiente para apoyar el entendimiento de las situaciones ya que la forma de captura de la información requerida, pero llegar a éste requiere, en circunstancias como las presentadas para SAMI, la elaboración de varios prototipos en la programación en donde, todos en su momento, parecen ser la mejor solución.

Con las cuatro versiones de modelo ER descrito se puede evidenciar que el diagrama de clases del lenguaje UML no necesariamente corresponden con un modelo lógico de datos ER, ya que existen sistemas en donde se debe comunicar información cuya forma de organizarla iría en contra de los lineamientos de normalización de una base de datos.

El modelado ER representa las entidades, atributos y relaciones presentes en la lógica de una situación de la que se deben guardar datos, pero la forma de presentación que requiere un desarrollador lleva a -1- tener diseños que permiten la realización de cálculos que en software como los de este caso, se deben realizar fuera de la base de datos y -2- redundancias que facilitan su almacenamiento pero que sacrifican las características de Atomicidad, Consistencia, Persistencia y Aislamiento.

Con las cuatro versiones de modelo ER descrito se puede evidenciar que el diagrama de clases del lenguaje UML no necesariamente corresponden con un modelo lógico de datos ER, ya que existen sistemas en donde se debe comunicar información cuya forma de organizarla iría en contra de los lineamientos de normalización de una BD.
El modelado ER representa las entidades, atributos y relaciones presentes en la lógica de una situación de la que se deben guardar datos, pero la forma de presentación que requiere un desarrollador lleva a -1- tener diseños que permiten la realización de cálculos que en software como los de este caso, se deben realizar fuera de la base de datos y -2- redundancias que facilitan su almacenamiento pero que sacrifican las características de Atomicidad, Consistencia, Persistencia y Aislamiento.
Tabla 1 presenta un resumen del análisis.

Fig. 8 Entidad- Relación Normalizada



V. CONCLUSIONES

Para sistemas de información en donde se pueda presentar comunicación asíncrona y en la que sea necesaria enviar gran cantidad de datos y guardarlos para realizar comparaciones con resultados esperados, sé debe procurar hacer el modelo tan simple como sea posible evitando almacenar datos que se puedan generar con operaciones.

Cuando hay herencia en el diagrama de clases, el modelo entidad-relación no es suficiente para apoyar el entendimiento de las situaciones ya que la forma de captura de la información requerida, pero llegar a éste requiere, en circunstancias como las presentadas para SAMI, la elaboración de varios prototipos en la programación en donde, todos en su momento, parecen ser la mejor solución.

Con las cuatro versiones de modelo ER descrito se puede evidenciar que el diagrama de clases del lenguaje UML no necesariamente corresponden con un modelo lógico de datos ER, ya que existen sistemas en donde se debe comunicar información cuya forma de organizarla iría en contra de los lineamientos de normalización de una base de datos.

El modelado ER representa las entidades, atributos y relaciones presentes en la lógica de una situación de la que se deben guardar datos, pero la forma de presentación que requiere un desarrollador lleva a -1- tener diseños que permiten la realización de cálculos que en software como los de este caso, se deben realizar fuera de la base de datos y -2- redundancias que facilitan su almacenamiento pero que sacrifican las características de Atomicidad, Consistencia, Persistencia y Aislamiento.

Con las cuatro versiones de modelo ER descrito se puede evidenciar que el diagrama de clases del lenguaje UML no necesariamente corresponden con un modelo lógico de datos ER, ya que existen sistemas en donde se debe comunicar información cuya forma de organizarla iría en contra de los lineamientos de normalización de una BD.
El modelado ER representa las entidades, atributos y relaciones presentes en la lógica de una situación de la que se deben guardar datos, pero la forma de presentación que requiere un desarrollador lleva a -1- tener diseños que permiten la realización de cálculos que en software como los de este caso, se deben realizar fuera de la base de datos y -2- redundancias que facilitan su almacenamiento pero que sacrifican las características de Atomicidad, Consistencia, Persistencia y Aislamiento.

Tabla 1 Comparación de las Alternativas de Solución

Propuesta	Datos Necesarios	Observaciones
Cada evento	• # de Evento	• Puede tener registros que no tengan asociados un

del mes en un registro	• Animal • Mes • Cantidad • Desperdicio • Dinero • Peso	animal. • Excepción a la integridad referencial • En caso de requerir un nuevo campo a enviar y cambiaría el esquema de la BDR. • La acción de "comprar medicamento" no estaría asociado con ningún animal.
Cada animal en el mes en un registro	Decisiones en general • Suministrar Alimento • Suministrare Agua • Sanidad Decisiones de las hembras • Ordeñar • Inseminar • Enfermar • Parir	• En caso de requerir un nuevo evento cambiaría el esquema de base de datos • En las dos tablas de detalle se envían registros con datos en cero, ya que hay atributos en una tabla que corresponden con acciones de los elementos. • La acción de "comprar medicamento" no estaría asociado con ningún animal. • No se estableció como llevar el inventario de insumos ni el registro del mes de adquisición
Archivo	• Mes • Animal • Evento • Cantidad (si lo requiere)	• Flexibilidad para enviar por tiempo y animal cualquier evento que tenga valor. • Requiere lectura en el servidor y posterior guardado en la base de datos para facilitar la posterior realización de cálculos.

VI. Bibliografía

[1] U. Gómez Prada y O. Gómez Sandoval, *SAMI: Videojuego para el Aprendizaje Agropecuario soportado con Dinámica de Sistemas*, Documento Interno , 2015.

[2] U. E. Gómez P., H. Andrade y C. A. Vásquez, «Lineamientos Metodológicos para construir Ambientes de Aprendizaje en Sistemas Productivos Agropecuarios soportados en Dinámica de Sistemas,» *Información Tecnológica*, 2015.

[3] L. M. De la Torre Navarro y J. Dominguez Gómez, «Las TIC en el proceso de enseñanza aprendizaje a través de los objetos de aprendizaje,» vol.4, n.1, pp. 83-92, 2012. [En línea]. Available: http://goo.gl/sm0sSJ. [Último acceso: 29 Marzo 2106].

[4] U. Gómez Prada, O. Barragán Tarazona y H. Andrade Sosa, *Modelo de Simulación para la Investigación Integral de Sistemas de Producción de Ganadería Bovina. Un Enfoque Sistémico*, Monterrey: Memorias del Primer Congreso Latinoamericano de Dinámica de Sistemas, 2003.

[5] C. Phillips, Principios de Producción Bovina, Zaragoza: Acribia S.A, 2003.

[6] G. Powell, Beginning Database Design, Indianapolis: Wiley Publishing, Inc, 2005.

[7] D. Rice, Patterns of Enterprise Application Architechture, Martin Fowler, 2011.

Urbano Eliécer Gómez Prada es Magíster en Ingeniería, Área Informática y Ciencias de la Computación e Ingeniero de Sistemas de la Universidad Industrial de Santander. Su línea de investigación es la de Ingeniería de Software y Bases de Datos. Es profesor adscrito a la Facultad de Ingeniería de Sistemas e Informática de la Universidad Pontificia Bolivariana Seccional Bucaramanga, líder de la línea estratégica de investigación en Tecnologías de la Información y las Comunicaciones (TIC) de la misma universidad y director del Grupo de Investigación en Informática (GIINFO). Es miembro de la Association for Computing Machinery (ACM).

Oscar Fernando Gómez Sandoval es Magíster en Ingeniería de la Universidad de los Andes, Bogotá, e Ingeniero Electrónico de la Universidad Pontificia Bolivariana Seccional Bucaramanga. Sus intereses investigativos giran principalmente alrededor del desarrollo de videojuegos con aplicaciones adicionales al entretenimiento. Es profesor adscrito a la Facultad de Ingeniería de Sistemas e Informática de la Universidad Pontificia Bolivariana Seccional Bucaramanga, y coordinador del Semillero de Investigación en Ingeniería de Sistemas e Informática (SIINFO).

Es coordinador general del grupo Wikimedistas de Colombia, administrador de Wikipedia en español y miembro activo del movimiento Wikimedia. Es miembro de la IEEE Computer Society.