Transformaciones entre modelos temporales de evolución

Javier Pastorino Regina Motz
Facultad de Ingeniería – Universidad ORT Uruguay
Montevideo, Uruguay
javierpg@adinet.com.uy, motz@athenea.ort.edu.uy

Resumen

Las bases de datos temporales almacenan la evolución de la información en el transcurso del tiempo. Dicha evolución la podemos clasificar en evolución del esquema y en evolución de la extensión del mismo. Esto nos permite clasificar los sistemas temporales en cuatro tipos en función de sus capacidades para manipular diferentes dimensiones de evolución temporal. El objetivo del presente estudio es la definición de un modelo que permita intercambiar la información almacenada en estos tipos de bases de datos brindando metodologías para la conversión entre los mismos. Concluiremos que es posible, sin perder significado semántico de la información ni de la evolución que proveen estos sistemas, transformar la información que almacenan dichos sistemas a un modelo Bitemporal y a partir de éste poder generar esquemas que brinden información equivalente a la almacenada en el sistema original.

Palabras Clave: bases de datos temporales, transformación entre modelos temporales

1. Introducción

Las bases de datos temporales almacenan la evolución de la información a través de la asociación de marcas de tiempo a la misma [1, 4, 5]. Estos cambios pueden presentarse tanto en los datos como en la estructura que almacena la definición de los mismos. Por consiguiente podemos catalogar dichos cambios en: evolución del esquema (Schema Evolution) y en evolución de los datos (Extensional Data Evolution).

La evolución en el esquema nos permite modelar la evolución que sufre los metadatos que definen la estructura que tendrá la base de datos para almacenar la información representada. La capacidad brindada por el sistema para almacenar la evolución del esquema esta ligada al tipo de base de datos que se este utilizando. Mientras que la evolución de la información expresa los cambios que sufrirá la extensión de dicho esquema. Se puede observar que los cambios que sufre el esquema son de significante consideración como se muestra en el estudio realizado sobre la evolución del esquema de un sistema de salud en [2]. Esto nos hace concientizarnos de cuan frecuentes son los cambios en el universo de discurso, y no solo en la información que obtenemos del mismo sino también de la estructura que utilizamos para almacenarla. En este entorno, se puede considerar almacenar dichas evoluciones utilizando como plataforma un modelo relacional e implementar manualmente toda la metodología de manejo del cambio. Sin embargo, la intención de las bases de datos temporales es la de definir un entorno de trabajo transparente para los usuarios, de modo tal que el manejo de los cambios no implique un costo asociado al diseño de la base de datos.

Considerando la capacidad de manejar distintas dimensiones en la evolución, tanto a nivel de esquema como de extensión, en [6] se identifican cuatro categorías de bases de datos temporales: “Snapshot Databases”, “Transaction time Databases”, “Valid time Databases” y “Bitemporal Databases” las cuales permiten almacenar los cambios de diferentes formas, utilizando variadas marcas de tiempo en cada una. La existencia de estos diferentes modelos dificulta el intercambio y la integración de la información debido a las distintas dimensiones temporales que almacenan. Por

ejemplo podemos intentar intercambiar información entre un sistema Snapshot (que no posee dimensionamiento temporal) con un sistema de tiempo de transacción (que posee una dimensión temporal). En este sentido el aporte de este trabajo es la definición de un modelo que permita realizar el intercambio de información entre modelos temporales con distintas dimensiones, tanto para el esquema como para los datos.

El primer paso para lograr nuestro objetivo es estudiar las cualidades que posee cada modelo para identificar las características del modelo de intercambio. Del estudio efectuado surge que el modelo de base de datos bitemporal permite representar todas las cualidades que poseen los otros tres modelos. Sobre este modelo de intercambio definimos las transformaciones que nos permiten asegurar la no pérdida de información. Presentamos entonces un modelo de conversión de la información desde los modelos Snapshot (que no utilizan versionado), Valid Time (que utilizan el tiempo de validez) y Transaction Time (que utilizan el tiempo de transacción) hacia el modelo Bitemporal y viceversa.

La organización de este artículo es la siguiente. En la Sección 2 se describe brevemente los distintos modelos de bases de datos temporales, en los cuales nos basaremos para la realización del presente estudio. En las Secciones 3, 4 y 5 se presentan las transformaciones entre los modelos Snapshot, Transaction Time y Valid Time y el modelo Bitemporal respectivamente. Por último en la Sección 6 presentamos las conclusiones obtenidas en el presente estudio.

2. Modelos Temporales de Evolución

Tansel, *et al*. [6] clasifican los modelos temporales dependiendo de su capacidad para manipular los cambios que se dan en el mundo real. En esta sección describiremos brevemente los cuatro modelos, analizando las características temporales que posee cada uno y los elementos que componen la estructura de los mismos.

2.1. Snapshot Database

El modelo temporal Snapshot no considera ninguna característica específica para el manejo del tiempo. En este sentido es equivalente a simular el manejo del tiempo en una base de datos relacional. La estructura que utiliza para almacenar la información del esquema consiste en dos relaciones, una para almacenar las tablas que componen el esquema y otra para almacenar los atributos que componen las mismas. La relación que describe las tablas mantiene la identificación de cada relación de la base de datos, su nombre y el tipo. La relación que describe los atributos almacena una identificación de cada atributo de las relaciones que componen la base de datos, su nombre, tipo, tamaño, identificador de tabla a la que pertenece y si pertenece o no a la clave primaria de la relación.

EJEMPLO 2.1

Considerando que tenemos la relación Doctor (nombre, especialidad, ciudad), su representación en el modelo Snapshot sería:

Table_Id	Name	Table_Type
1	Doctor	Snapshot

Attr_Id	Name	Type	Size	Tbl_Id	Key
1	Nombre	Char	30	1	Yes
2	Especialidad	Char	20	1	No
3	Ciudad	Char	20	1	No

2.2. Transaction Time Database

En el modelo Transaction Time la información es recuperada en función del tiempo en el cual la misma fue insertada en la base de datos [3]. Para esto se utiliza el tiempo de transacción en la cual se da el hecho. La principal ventaja que brindan estos sistemas es la transparencia para el usuario ya que las marcas de tiempo son automáticamente manipuladas por el motor de la base de datos [3]. El esquema en este modelo contiene tres relaciones. Una que almacena las diferentes versiones de esquema, otra relación que describe las tablas conteniendo la identificación de cada relación de la base de datos, su nombre, el tipo, la versión en la cual fue definida y las marcas de tiempo de transacción. Y una última relación que describe los atributos de las relaciones de la base de datos conteniendo una identificación de cada atributo, su nombre, tipo, tamaño, identificador de tabla a la que pertenece, si pertenece o no a la clave primaria de la relación, la versión en la cual el atributo fue definido, y las marcas de tiempo de transacción. Se puede observar que estos modelos representan una única dimensión temporal.

EJEMPLO 2.2

Considerando la misma relación que en el ejemplo 2.1 la representación dicha relación en el modelo Transaction Time es la siguiente:

Version_Id	Transaction Time
1	10/01 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time
1	Doctor	Transactional	1	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Version_Id	Trans_Time
1	Nombre	Char	30	1	Yes	1	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 - ∞
3	Ciudad	Char	20	1	No	1	10/01 - ∞

2.3. Valid Time Database

El modelo Valid Time utiliza una marca de tiempo correspondiente al lapso de validez de los datos [1,3]. Este tiempo es únicamente proporcionado por el usuario (ya que solo él conoce cuando un evento es válido en el mundo real). Este modelo al igual que el Transaction Time modela una única dimensión temporal. La estructura de su esquema es similar a la estructura del modelo Transaction Time con la diferencia de que los atributos de marca de tiempo almacenan el tiempo de validez en lugar del de transacción.

2.4. Bitemporal Database

El modelo Bitemporal manipula dos dimensiones de versionado temporal. Permiten utilizar tanto el tiempo de transacción como el tiempo de validez [1,3]. La estructura del esquema para este modelo contiene cuatro atributos para manipular las marcas de tiempo: dos para la dimensión temporal por tiempo de transacción, y dos para la de tiempo de validez.

EJEMPLO 2.4

Considerando la misma relación que en el ejemplo 2.1 la representación dicha relación en el modelo Bitemporal es la siguiente:

Version_Id	Transaction Time
1	10/01 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time	Valid_Time
1	Doctor	Transactional	1	10/01 - ∞	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Version_Id	Trans_Time	Valid_Time
1	Nombre	Char	30	1	Yes	1	10/01 - ∞	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞
3	Ciudad	Char	20	1	No	1	10/01 - ∞	12/01 - ∞

2.5. Comparación entre los Modelos

La diferencia entre un sistema de tiempo de transacción o de validez es como se obtiene la fecha en la que se realiza el cambio de la versión. En los sistemas de transacción la fecha la dispone el sistema en el momento que se crea una nueva versión, mientras que en los de validez la fecha la ingresa el usuario en el momento que establece la nueva versión, permitiendo esto último generar cambios pro-activos y retroactivos en el tiempo [1]. Los sistemas tipo Snapshot al no proveer ningún mecanismo de manipulación del tiempo no almacenan historial de los cambios, por dicho motivo cualquier cambio en el esquema se traduce a reemplazar o eliminar los registros previos sin conocer a posteriori todos los estados por el cual el esquema ha pasado.

3. Conversión entre el Modelo Snapshot y el Modelo Bitemporal

El modelo Snapshot no registra ninguna especie de versionado, esto trae como consecuencia que cualquier cambio reemplaza el estado anterior de la base de datos. Para versionar dichos cambios es necesario disponer de la fecha de creación de las tablas y atributos y de las fechas en que ocurrieron cambios sobre los mismos.

Para lograr la conversión del modelo Snapshot al modelo Bitemporal observamos entonces dos alternativas:

1. Agregar a la configuración del esquema Bitemporal algún patrón que nos permita indicar que la fecha de validez y de transacción de cualquier elemento de *n* configuraciones presentes no son precisas dado que se obtuvieron de un modelo que no soporta versionado.
2. Establecer el tiempo de validez y de transacción para un hecho el tiempo en el cual el mismo se detectó.

Seguiremos la segunda opción ya que la primera requiere modificar la definición del esquema del modelo bitemporal.

Si conocemos el estado t_1 podemos inferir que cualquier cambio que modifique dicho estado lo hace obsoleto y que solo aquellos elementos de la configuración del estado t_1 son validos en el nuevo estado t_2 si siguen perteneciendo al esquema, y todos los que no pertenezcan han sido eliminados. Entendiendo por configuración del estado al conjunto de elementos que conforman al esquema de la base, o sea el conjunto de tuplas que definen el esquema de la base de datos: tabla, atributo, vista, constraint, etc., aunque en el presente estudio nos basaremos en la problemática para

la representación del esquema de las tablas y sus atributos. La desventaja de esto es que determinados cambios nunca se registran en el modelo, ya que si la velocidad con la cual se intentan leer nuevos cambios sobre el modelo Snapshot es demasiado lenta pueden haber cambios que queden ocultos, por ejemplo se agrega una tabla con *n* atributos y en un determinado instante antes de haber detectado la creación de la misma se agregan *m* atributos, como consecuencia a esto el modelo detectará una nueva tabla con *n* + *m* atributos en lugar de una con *n* y luego la adición de *m* atributos a la misma.

En concreto la transformación requiere obtener respuestas para ¿cuáles son los tiempos a considerar como tiempo de comienzo de validez y de transacción? y ¿cómo se determina el tiempo de finalización de la validez y de la transacción? Para responder esto es necesario cual es la configuración inicial y como determinar las subsecuentes configuraciones.

Para determinar la configuración inicial asumimos que el tiempo inicial de todo elemento de del esquema de la base de datos tiene tanto como tiempo de validez y de transacción el momento en el cual el área de datos (tablespace) fue creada.

Una vez obtenida la primer configuración del esquema, se considera que el tiempo inicial (ya sea tiempo de validez como de transacción) para cada elemento de la configuración es el tiempo en el cual el mismo es detectado. Inicialmente todas las fechas de finalización (la de validez y la de transacción) son establecidas con el valor infinito (representadas también *inf.*) ya que se asume que toda configuración detectada originalmente prevalecerá hasta que aparezca un cambio.

Luego de haber establecido la primer configuración en el nuevo modelo, toda detección futura se comparará con la última configuración obtenida. Los posibles cambios a detectar son de: eliminación o adición de algún elemento a la configuración. Consideramos que las modificaciones a un elemento de la configuración son en realidad una “eliminación” del elemento anterior y una “adición” de un nuevo elemento ya que diferirá con el elemento existente en la configuración previa.

ELIMINACIÓN DE ALGÚN ELEMENTO

Considerando que en la configuración t_i tenemos *n* elementos y el la última revisión (llamémosle configuración t_{i+1}) se detectaron *m* elementos. Se observa que no alcanza con comparar la cardinalidad de los conjuntos ya que en un lapso de tiempo se pueden eliminar *k* elementos y a su vez adicionar otros *k* elementos. Entonces consideramos $t_i - t_{i+1} \neq \emptyset$ para poder concluir que hay elementos que estaban presentes en la configuración t_i que ya no están presentes en la configuración t_{i+1} por lo que algunos elementos fueron eliminados. Para los elementos presentes en t_i y no presentes en t_{i+1} se establece que su fecha de finalización para los tiempos de transacción y de validez es la fecha en la que se realizó la nueva revisión del esquema Snapshot.

ADICIÓN DE ALGÚN ELEMENTO

Al igual que para la eliminación de elementos de una configuración, podemos detectar que desde la ultima configuración hacia la actual se han agregado elementos de la configuración ya que partiendo de la configuración t_i y obteniendo la nueva configuración t_{i+1} podemos ver que si $t_{i+1} - t_i \neq \emptyset$ entonces implica que la nueva configuración tiene elementos nuevos. Para los nuevos elementos determinamos que la fecha de inicio de validez y de transacción de los mismos se ubica entre la fecha de la última revisión y la revisión actual, por lo tanto podemos asumir que la fecha de inicio de dichos elementos es la fecha de la revisión en curso.

EJEMPLO 3.1.1

Considerando nuevamente el Ejemplo 2.1 partiendo de que el 01-feb realizamos la primer inspección al esquema de una base de datos Snapshot creada el 10-ene, y el mismo presenta la tabla Doctor, consideramos en el modelo Bitemporal correspondiente que dicha tabla fue creada el 10-

ene. Suponiendo que en una inspección realizada el 20-feb la base de datos Snapshot presenta una nueva tabla, asumiremos que la misma fue creada efectivamente el 20-feb ya que en la última revisión realizada el 01-feb la misma no existía, por ende esta fue creada entre el 01-feb y el 20-feb pero el cambio fue detectado el 20-feb.

La configuración inicial en el modelo Bitemporal sería la siguiente:

Version_Id	Transaction Time	Valid Time
1	10/01 - ∞	10/01 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time	Valid_Time
1	Doctor	Bitemporal	1	10/01 - ∞	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Ver_Id	Trans_Tm	Val_Tm
1	Nombre	Char	30	1	Yes	1	10/01 - ∞	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞
3	Ciudad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞

En la revisión del 20-feb se obtiene la siguiente configuración:

Table_Id	Name	Table_Type
1	Doctor	Snapshot

Attr_Id	Name	Type	Size	Tbl_Id	Key
1	Nombre	Char	30	1	Yes
3	Ciudad	Char	20	1	No
4	Edad	Int.	3	1	No

Por lo que se generará la siguiente configuración en el modelo Bitemporal:

Version_Id	Transaction Time	Valid Time
1	10/01 – 09/02	10/01 – 09/02
2	10/02 - ∞	10/02 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time	Valid_Time
1	Doctor	Bitemporal	1	10/01 - ∞	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Ver_Id	Trans_Tm	Val_Tm
1	Nombre	Char	30	1	Yes	1	10/01 - ∞	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 – 09/02	10/01 – 09/02
3	Ciudad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞
4	Edad	Int.	3	1	No	2	10/02 - ∞	10/02 - ∞

Obsérvese que ya que se detecto que se había eliminado un atributo se procedió a definir la finalización de la validez del atributo (tupla 2).

En la conversión inversa, o sea desde el modelo Bitemporal al modelo Snapshot, se obtiene una representación estática del modelo bitemporal. En esta representación el modelo Snapshot correspondiente está formado por la proyección del último esquema válido en la base de datos Bitemporal. Las tuplas seleccionadas serán únicamente aquellas que sean válidas en el momento en el que se solicita la conversión, independientemente de en que versión fueron definidas.

Para obtener la información almacenada en el modelo Bitemporal, sin tener pérdida sobre la misma ni generar errores, utilizamos los mismos criterios que consideramos para determinar la evolución de las configuraciones de los esquemas. Ya que la eliminación de una tupla significa la culminación de la validez de la misma, y la detección de una tupla no existente previamente en la base indica la adición de la misma con validez entre la última detección y la detección en curso.

4. Conversión entre el Modelo Transaction Time y el Modelo Bitemporal

El manejo del tiempo de transacción es idéntico en ambos modelos, por lo que podemos hacer una conversión directa entre la evolución transaccional que posee un modelo Transaction Time y el modelo Bitemporal. Respecto al manejo sobre el tiempo de validez existe la misma problemática que en la conversión desde el modelo Snapshot ya que el modelo Transaction Time carece de esta dimensión temporal. Sin embargo, en este caso existe más información que la disponible para realizar la conversión desde el modelo Snapshot. En los modelos Transaction Time todo elemento de la configuración tiene como intervalo de validez el intervalo determinado por las transacciones. Esto nos permite determinar el tiempo de validez de un elemento a través del tiempo de transacción.

A todo elemento de una configuración t_i que sea modificado en la configuración t_{i+1}, se le asigna como fecha de finalización (tanto de validez como de transacción) en la configuración t_i el valor indicado en el tiempo de finalización de transacción del mismo elemento en modelo Transaction Time. Para los nuevos elementos, se determina que los tiempos de validez en el modelo Bitemporal sean los mismos que los tiempos de transacción que en el modelo Transaction Time.

EJEMPLO 3.1.2

Considerando la primer configuración obtenida desde el modelo Transaction Time representada en el Ejemplo 2.2 se obtiene la siguiente representación en el modelo bitemporal:

Version_Id	Transaction Time	Valid Time
1	10/01 - ∞	10/01 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time	Valid_Time
1	Doctor	Bitemporal	1	10/01 - ∞	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Ver_Id	Trans_Tm	Val_Tm
1	Nombre	Char	30	1	Yes	1	10/01 - ∞	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞
3	Ciudad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞

El 20-feb se realiza una revisión a la base Transaction Time obteniéndose el siguiente esquema:

Version_Id	Transaction Time
1	10/01 – 04/02
2	05/02 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time
1	Doctor	Transactional	1	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Ver_Id	Trans_Tm
1	Nombre	Char	30	1	Yes	1	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 – 04/02
3	Ciudad	Char	20	1	No	1	10/01 - ∞
4	Edad	Int.	3	1	No	2	10/02 - ∞

Por lo que se generarán las siguientes modificaciones en el esquema de la base de datos equivalente:

Version_Id	Transaction Time	Valid Time
1	10/01 – 04/02	10/01 – 04/02
2	05/02 - ∞	05/02 - ∞

Table_Id	Name	Table_Type	Version_Id	Trans_Time	Valid_Time
1	Doctor	Bitemporal	1	10/01 - ∞	10/01 - ∞

Attr_Id	Name	Type	Size	Tbl_Id	Key	Ver_Id	Trans_Tm	Val_Tm
1	Nombre	Char	30	1	Yes	1	10/01 - ∞	10/01 - ∞
2	Especialidad	Char	20	1	No	1	10/01 – 04/02	10/01 – 04/02
3	Ciudad	Char	20	1	No	1	10/01 - ∞	10/01 - ∞
4	Edad	Int.	3	1	No	2	10/02 - ∞	10/02 - ∞

Nótese que la fecha en la que se realizó la revisión de la base original no tiene impacto en las fechas de validez, ya que estas quedan determinadas por la fecha de la transacción.

En la conversión inversa, o sea desde el modelo Bitemporal al modelo Transaction Time, alcanza con proyectar sobra todos los atributos menos las marcas de tiempo de transacción, obteniéndose así una vista que permite acceder al esquema como si se tratase de un modelo Transaction Time.

Para poder obtener información del modelo Transaction Time, utilizamos el mismo criterio que utilizamos para obtener el esquema. De igual forma realizamos la conversión inversa, o sea leer datos desde el modelo Bitemporal como si se tratase de un modelo Transaction Time, simplemente proyectamos todos los atributos excepto los que manejan las marcas de tiempo de validez.

5. Conversión entre el Modelo Valid Time y el Modelo Bitemporal

Para la conversión del modelo Valid Time al modelo Bitemporal, debemos realizar consideraciones similares a las que realizamos para la conversión desde el modelo Transaction Time al modelo Bitemporal. Las mismas consideraciones realizadas antes para el tiempo de validez deben hacerse para el tiempo de transacción. En este caso se puede considerar que el tiempo de transacción es el mismo que el tiempo de validez.

En la conversión inversa, o sea desde el modelo bitemporal al modelo Valid Time, alcanza con proyectar sobre todos los atributos menos los que manejan el tiempo de transacción, definiendo una vista que permite acceder al esquema como si fuese un modelo Valid Time.

Para poder obtener información del modelo Valid Time, utilizamos el mismo criterio que utilizamos para obtener el esquema. De igual forma realizamos la conversión inversa, o sea leer datos desde el modelo Bitemporal como si se tratase de un modelo Valid Time, simplemente proyectamos todos los atributos excepto los que manejan las marcas de tiempo de transacción.

6. Conclusiones

En este trabajo definimos cuales son las conversiones necesarias para poder transformar el esquema de cualquier modelo de evolución temporal (Snapshot, Transaction Time y Valid Time) al modelo Bitemporal de evolución. Indicamos también la forma realizar las transformaciones inversas, o sea convertir del modelo Bitemporal a otro modelo (Snapshot, Transaction Time y Valid Time).

Sin embargo se observa que no es posible obtener la misma base de datos con la cual se construyó la base de datos Bitemporal debido a las distintas transformaciones realizadas. Para ejemplificar esto, podemos considerar la conversión de una base de datos a lo largo de *n* revisiones. Cuando se convierte este resultado Bitemporal a cualquiera de los otros modelos, se obtiene un número mayor de tuplas respecto a las originales, esto se debe a la registración de evolución que se obtiene en el modelo Bitemporal. Estas mismas consideraciones son válidas al establecer una marca de tiempo a partir de un modelo que no manipula dicha dimensión temporal.

En este trabajo mostramos también cómo a partir de la especificación de las conversiones a nivel de esquemas de los modelos Snapshot, Transaction Time y Valid Time hacia el Bitemporal se derivan las conversiones que se deben realizar a nivel de instancias. La conversión de datos desde una base Bitemporal a una base de datos Transaction Time o Valid Time puede realizarse estableciendo una vista que proyecte los atributos y excluya las marcas de tiempo que no son necesarias para dicho modelo, y si la base de datos a la cual se quieren migrar los datos es del modelo Snapshot alcanza con seleccionar los datos cuya validez corresponda con la fecha en la que se realiza la consulta. Estas transformaciones pueden observarse en la Figura 6.1, en la cual se describe como podemos tanto de una base de datos que utiliza un modelo Snapshot, Transaction Time o Valid Time convertir la información almacenada en la misma (ya sea a nivel de esquema como de su extensión) a una base de datos que utiliza un modelo Bitemporal aplicando las transformaciones necesarias en cada caso. También observamos que es posible obtener vistas sobre esa base de datos Bitemporal que permitan manipularla como si fuese una base de datos de los otros tres modelos.

La realización de este trabajo fue orientado a brindar un aporte que permita la continuación del estudio de estos modelos hacia la definición e implementación de una extensión del modelo relacional que permita soportar estos modelos y que permita la realización de las mencionadas transformaciones. Entre los trabajos futuros se encuentra la realización de un prototipo en este sentido.

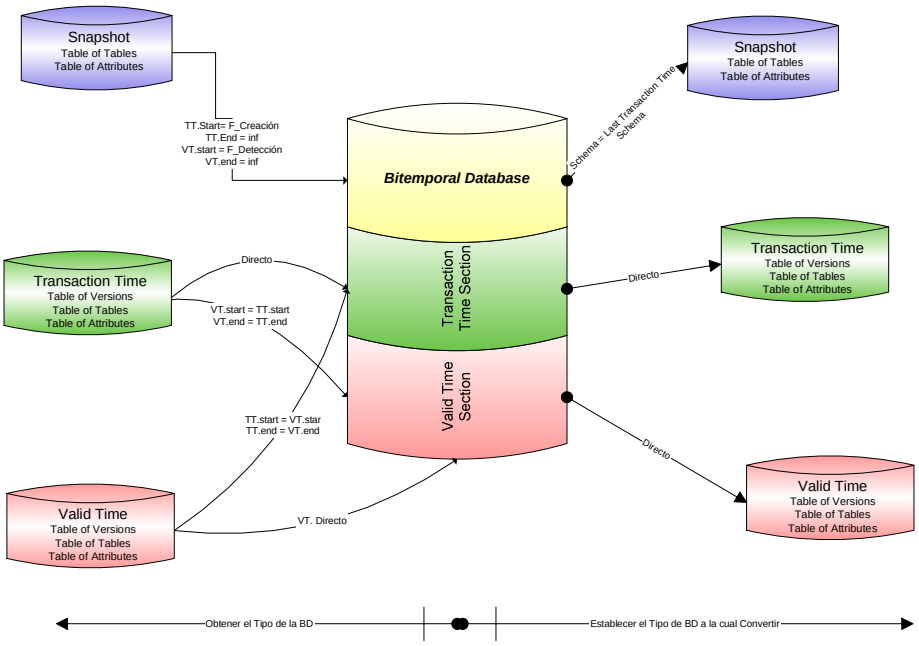

Figura 6.1: Transformaciones entre los distintos modelos y el modelo Bitemporal

7. Referencias

[1]. Moreira, Viviane; Edelweiss, Nina. Queries To Temporal Databases Supporting Schema Versioning. Universidade Federal do Rio Grande do Sul, Curso de Pós-graduaçao em Ciencia da Computaçao.
XIV Simpósio Brasileiro de Banco de Dados – SBBD'99, Florianópolis, SC, 11-13 outubro, 1999.

[2]. Sjøberg, Dag. Quantifying Schema Evolution. University of Glasgow. Published in *Information and Software Technology*, Vol. 35, No. 1, pp. 35-44, January 1993

[3]. Nogueira, Patrícia; Edelweiss, Nina. Implementing a Temporal Database on Top of a Conventional Database: Mapping of the Data Model and Data Definition Management.
XV Simpósio Brasileiro de Banco de Dados – SBBD'2000, 2-4 outubro, João Pessoa, Paraíba.

[4]. Jensen, C.; Clifford, J.; Gadia, S.; Segev, A.; Snodgrass, R. A Glossary of Temporal Database Concepts. Appeared in SIGMOD Record, Vol. 21, No. 3, September 1992.

[5]. McKenzie, N.G.; Snodgrass, R. Schema Evolution and the Relational Algebra. Information Systems, V. 15, N. 2, P. 207-232, 1990

[6]. Tansel, A.U.; Clifford, J.; Gadia, S.; Jajodia, S.; Segev, A.; Snodgrass, R. Temporal Databases – Theory, Design and Implementation. Redwood City: Benjamin/Cummings, 1993. 633p