Extensiones gramaticales del algebra relacional para el soporte de integridad en sistemas relacionales

Beatriz Bernábe Loranca[1], Ma. del Rocío Boone Rojas[1], Ramiro López Sales[2]

[1] Area de Bases de Datos
Facultad de Ciencias de la Computación, BUAP
Calle 14 sur y Avenida San Claudio, Colonia San Manuel, Puebla, México
rboone@cs.buap.mx, beti@cs.buap.mx
www.cs.buap.mx/~bety/Investigacion.html

[2]Facultad de Ciencias de la Computación, BUAP
Calle 14 sur y Avenida San Claudio, Colonia San Manuel, Puebla, México
ramiro@cs.buap.mx

Resumen. Actualmente la gran mayoría de los Sistemas Administradores de B.D. Relacionales ofrecen un soporte limitado para la representación y comprobación de restricciones de integridad, de tal manera que es necesario proporcionar algunas alternativas de solución para esta situación.

Con el propósito de ofrecer un conjunto de reglas gramaticales que permitan la definición y el tratamiento de restricciones de integridad a nivel de un DBMS, en este trabajo se presenta una gramática extendida basada en un intérprete para el álgebra relacional, en un lenguaje de integridad y en especificaciones complementarias para la regla de integridad referencial del modelo relacional.

1 Introducción

Para asegurar que nuestra información dentro de una BD sea correcta y por ende de calidad, debemos de garantizar que nuestro DBMS proporcione un soporte satisfactorio para la representación y comprobación de restricciones de integridad. El modelo relacional y por ende los DBMS basados en él, contemplan restricciones de integridad generales asociadas a las llaves primarias y a las llaves ajenas, actualmente algunos DBMS soportan estas restricciones de integridad. Sin embargo la gran mayoria se encuentran lejos de proporcionar un soporte satisfactorio para el tratamiento de integridad, como lo hemos mostrado en los trabajos [5,9]. De tal forma que se deben plantear alternativas de solución para esta situación.

El problema de integridad en bases de datos, se identifica como uno de los problemas clásicos en Bases de Datos, cuya investigación ha dado lugar a diferentes contribuciones para su tratamiento tanto en el nivel lógico, como es el caso de las especificaciones complementarias para el componente de integridad del modelo relacional y el lenguaje de integridad propuestos en [2, 3], como en el nivel conceptual, como es el caso del análisis y extensión del modelo entidad relación, que se presenta en [11].

© A. Gelbukh, H. Calvo (Eds.)
Titulo del Libro
Research on Computing Science 13, 2005, pp. 191-202

Por otra parte, se ha analizado en [10] la pertinencia de dar tratamiento a las restricciones de integridad a nivel del DBMS, de tal manera que sea nuestro DBMS el que se encargue de verificar automáticamente las restricciones cada vez que se actualicen los datos.

Dentro del marco del tratamiento de integridad a nivel lógico de una base de datos y con el propósito de ofrecer un conjunto de reglas gramaticales que permitan la definición y el tratamiento de restricciones de integridad en principio, bajo control del DBMS, en este trabajo se presenta una gramática extendida basada en referencias tales como un intérprete para el álgebra relacional[7], un lenguaje de integridad [2,3] y en especificaciones complementarias para la regla de integridad referencial del modelo relacional [2,3].

El trabajo se organiza como sigue: en la sección dos se incluyen aspectos fundamentales del concepto de integridad y del componente de integridad del modelo relacional, la sección tres proporciona las especificaciones complementarias que se han propuesto para la regla de integridad referencial, la siguiente sección presenta una breve descripción de un lenguaje de integridad, mediante el cual, en particular se formulan las reglas de integridad relacionales, en la sección cinco se describen las ideas básicas relacionadas con el intérprete del álgebra relacional que proporciona la base para el presente trabajo, en la sección seis, se presentan las extensiones de la gramática desarrollada para el intérprete del álgebra relacional. Finalmente se presentan las conclusiones y perspectivas de este trabajo.

2 El concepto de integridad

Él termino integridad se refiere a la exactitud o corrección y últimamente a la calidad de los datos en la base de datos, para ello es necesario incluir ciertas *reglas de integridad*, cuyo propósito es informar al DBMS de ciertas restricciones del mundo real (como la restricción de que los pesos de los artículos no pueden ser negativos) para que pueda impedir la ocurrencia de tales configuraciones imposibles de valores.
Ullman [4], establece una clasificación general de las restricciones de uso común:

a) Las llaves son atributos o conjuntos de atributos que proporcionan la identificación única de un objeto dentro de su clase o de una entidad dentro de su conjunto entidad.
b) Las restricciones de valor único son requisitos de que el valor de un papel determinado sea único.
c) Las restricciones de integridad referencial son requisitos de que el valor referenciado por algún objeto exista realmente en la base de datos.
d) Las restricciones de dominio requieren que el valor de un atributo sea extraído de un conjunto específico de valores o bien que se encuentre dentro de un intervalo determinado.
e) Las restricciones generales son afirmaciones arbitrarias que deben cumplirse en la base de datos.

En el caso del modelo relacional, el componente de integridad, incluye restricciones asociadas a las llaves primarias y ajenas respectivamente, que en términos genera-

les establecen que las llaves primarias de las relaciones base no deben contener nulos y que la base de datos no debe contener valores de llave ajena sin concordancia. Consecuentemente, los DBMS que se identifican como relacionales, deberían de ofrecer un soporte con el alcance suficiente como para dar un tratamiento adecuado a las restricciones de integridad que se pueden derivar a partir de las reglas de integridad del modelo relacional.

2.1 Reglas de integridad del modelo relacional

El componente de integridad del modelo relacional, ha permitido realizar algunas propuestas de especificación complementarias que permiten en la práctica utilizar y dar soporte a las reglas de integridad del modelo relacional, como es el caso de los trabajos de Date [2] que se citan en el presente trabajo. El modelo relacional incluye dos reglas generales de integridad, estas dos se refieren a las claves primarias y a las claves ajenas, las cuales Date [2, 3], las define a partir de los siguientes conceptos.

Una llave candidata se define de la siguiente forma. El atributo K (posiblemente compuesto) de la relación R es una llave candidata de R si y sólo si satisface las dos propiedades, independientes del tiempo: 1) Unicidad: Jamás, ningún valor válido de R contiene dos tuplas distintas con el mismo valor de K. 2) Irreductibilidad: Ningún subconjunto propio de K tiene la propiedad de unicidad.

Cabe señalar que por la propiedad de las relacionales que establece que las relaciones no contienen tuplas repetidas, toda relación tiene por lo menos una llave candidata. En la práctica, las relaciones tienden a tener una y sólo una llave candidata, pero indudablemente, es posible que tengan más. Por otro lado, del conjunto de llaves candidatas de una relación dada, se elige una y sólo una como llave primaria de esa relación, las demás, si existen, se llamarán llaves alternativas.

En términos de los conceptos anteriores se puede enunciar la primera regla de integridad de entidades: "Ningún componente de la llave primaria de una relación base puede aceptar nulos". Con "nulos" se indica información faltante por alguna razón, por ejemplo, si la propiedad no es aplicable o si el valor se desconoce etc. A veces, la regla de integridad de las entidades se expresa como: "En una base de datos relacional, nunca registraremos información acerca de algo que no podamos identificar."

De tal forma que un DBMS relacional que proporcione soporte para esta regla, deberá de verificar que los valores de llave primaria que se intenten introducir mediante las operaciones de inserción y de actualización no sean nulos y que no estén repetidos.

La segunda regla de integridad se aplica a las llaves ajenas; en términos informales, se puede definir una llave ajena de la forma siguiente: Una llave ajena es un atributo (quizá compuesto) de una relación R2 cuyos valores deben concordar con los de la clave primaria de alguna relación R1 (donde R1 y R2 no necesariamente son distintos). Un valor de clave ajena representa una referencia a la tupla donde se encuentra el valor correspondiente a la clave primaria (la tupla referida o tupla objetivo). Por tanto, el problema de garantizar que la base de datos no incluya valores no válidos de una llave ajena se conoce como el problema de integridad referencial.

La restricción según la cual los valores de una llave ajena determinada deben concordar con los valores de la llave primaria correspondiente se conoce como restricción referencial y la relación que contiene a la llave ajena se conoce como relación referencial y la relación que contiene a la clave primaria correspondiente se denomina relación referida o relación objetivo. En términos más precisos, se puede establecer la definición de una llave ajena. El atributo LF (quizá compuesto) de la relación base R2 es una clave ajena si y solo si satisface estas dos propiedades independientes del tiempo: Cada valor de LF es nulo del todo o bien no nulo del todo (con "nulo del todo" o "no nulo del todo" se quiere decir que, si LF es compuesto, todos sus componentes son nulos o bien todos sus componentes son no nulos, no una combinación.). Existe una relación base R1 con clave primaria LP tal que cada valor no nulo de LF es idéntico al valor de LP en alguna tupla de R1.

A partir de los conceptos anteriores se expresa la segunda regla de integridad del modelo relacional, la Regla de Integridad Referencial. "La Base de Datos no debe contener valores de llave ajena sin concordancia". Con el término "valores de llave ajena sin concordancia" se quiere decir aquí un valor no nulo de llave ajena para el cual no existe un valor concordante de la llave primaria en la relación objetivo pertinente. Dicho de otra manera la regla de integridad referencial dice tan solo que si B hace referencia a A, entonces A debe de existir.

3 Especificaciones complementarias para la regla de integridad referencial

De acuerdo al análisis que realiza Date [2], la regla de integridad referencial, requiere de especificaciones complementarias que permitan dar un tratamiento a la misma, que incluya por lo menos las situaciones más comunes que se presentan en la práctica. Podemos citar algunos aspectos del análisis anterior.

La regla de integridad referencial se formula en términos de estados de la base de datos. Cualquier estado de la base de datos que no satisfaga la regla será incorrecto por definición; pero, ¿cómo pueden evitarse tales estados incorrectos? La regla en sí no lo dice con exactitud. Una posibilidad, desde luego, es que el sistema rechazara cualquier operación que en caso de ejecutarse, produciría un estado ilegal. Pero en muchos casos una alternativa preferible sería que el sistema aceptara la operación pero realizara (en caso necesario) ciertas operaciones de compensación con objeto de garantizar un estado legal como resultado final.

Por tanto, para cualquier base de datos, el diseñador de la base de datos deberá de poder especificar cuáles operaciones han de rechazarse y cuáles han de aceptarse, y, en el caso de estas últimas, cuáles operaciones de compensación (si acaso) deberá de realizar el sistema. La idea fundamental es la siguiente. Para cada llave ajena, es necesario responder tres preguntas: ¿Puede aceptar nulos esa llave ajena?, ¿Qué ocurre si se intenta eliminar el objetivo de una referencia de clave ajena? Para responder esta pregunta, existen por lo menos tres posibilidades: Restringir (restricted): La operación de eliminación está "restringida" al caso en el cual no existen tuplas referenciales. Propagar (cascades): La operación de eliminación "se propaga" ("en cascada") y

elimina las tuplas referenciadas. Anular (nullifies): Se asignan nulos a la llave ajena en todas las tuplas referenciadas y en seguida se elimina el objetivo (sólo si la llave ajena puede acaptar nulos). ¿Qué deberá suceder si hay un intento de modificar la llave primaria del objetivo de una referencia de llave ajena? Nuevamente como en el caso anterior, existen por lo menos las mismas tres posibilidades: Restringir (restricted): La operación de modificación está "restringida" al caso en el cual no existen tuplas referenciales. Propagar (cascades): La operación de modificación "se propaga" ("en cascada") y modifica las tuplas referenciadas. Anular (nullifies): Se asignan nulos a la llave ajena en todas las tuplas referenciadas y en seguida se modifica el objetivo (sólo si la llave ajena puede aceptar nulos).

Para dar un tratamiento a los casos más comunes que pueden presentarse en la práctica para la regla de integridad referencial, basada en la propuesta de Date [2], proponemos como en [8], la siguiente sintaxis declarativa para una cláusula FOREIGN KEY (sintaxis de llave ajena):

```
FOREIGN KEY (clave ajena) REFERENCES objetivo
   NULLS [not] ALLOWED
   DELETE OF objetivo efecto                    (1.1)
   UPDATE OF clave-primaria-del-objetivo  efecto
```

donde el 'efecto' es RESTRICTED, CASCADES, NULLIFIES(SET NULL) o Reg_Int_Part, que es el identificador de un procedimiento que puede definir el usuario. La cual permite al diseñador de la B.D. especificar no solo el atributo o combinación de atributos que constituye una llave ajena y la relación objetivo a la cual hace referencia esa clave ajena, sino también las reglas particulares para claves ajenas –las reglas de nulos, eliminación y modificación- aplicables a esa llave ajena. O en general, satisfacer las necesidades particulares de diseñadores de bases de datos, al considerar que "efecto" en la sintaxis anterior incluya la posibilidad de invocar un procedimiento de base de datos para dar el tratamiento adecuado a la regla de integridad requerida por el diseñador.

4 Lenguaje de integridad hipotético

Date [2], ha propuesto un lenguaje hipotético para la formulación de reglas de integridad, que permitan especificar aspectos de su comportamiento. A continuación se incluyen algunos de sus aspectos y ejemplos asociados. El Lenguaje de integridad que se propone incluye un par de proposiciones, `CREATE INTEGRITY RULE` (crear regla de integridad) y `DROP INTEGRITY RULE` (desechar regla de integridad). En general, la proposición CREATE INTEGRITY RULE (crear regla de integridad) debe especificar cuatro componentes:
1) El nombre de la regla, 2) Una o más ocasiones de verificación, 3) Una restricción que debe ser satisfecha para los estados legales de la base de datos, 4) Una respuesta de violación, para indicar qué debe hacerse si la verificación falla.
Por ejemplo, se cita el caso de la regla R1.

```
CREATE INTEGRITY RULE R1
     ON      INSERT S.SITUACIÓN,
             UPDATE S.SITUACIÓN:
```

```
      CHECK FORALL S ( S.SITUACIÓN > 0 )
ELSE REJECT;
```

En la sintaxis anterior, se especifica que el nombre de la regla es R1. Las ocasiones de verificación (ON INSERT S.SITUACIÓN, UPDATE S.SITUACIÓN), ("al insertar S.SITUACIÓN, modificar S.SITUACIÓN") en el ejemplo), para indicar al sistema cuándo debe realizar la verificación especificada en la cláusula CHECK (verificar).

La restricción, que debe ser satisfecha por todos los estados legales de la base de datos, ("FORALL S (S.SITUACIÓN > 0)" ("para toda S (S.SITUACION > 0") en este caso); La respuesta de violación, para indicar qué debe hacerse si la verificación falla ("REJECT" ("rechazar") en el ejemplo, lo cual significa que la proposición INSERT o UPDATE culpable, debe rechazarse con un código de retorno apropiado).
Para propósitos ilustrativos, se pueden realizar algunas simplificaciones al lenguaje. Las ocasiones de verificación casi siempre serán obvias; por tanto, supondremos que el DBMS es capaz de determinar por sí mismo la ocasión u ocasiones de verificación si no se especifican de manera explícita. La restricción de la cláusula CHECK casi siempre comenzará con un cuantificador universal. En este caso se omite el cuantificador y se supone que cualquier variable no cuantificada se cuantificará en forma automática con un FORALL (para toda) apropiado. También se supone que, si se omite la cláusula ELSE (si no), quedará implícita la respuesta de violación "rechazar la operación de actualización (con un código de retorno adecuado)".
A través del lenguaje de integridad que se ha citado anteriormente, a continuación se incluyen reglas de integridad relacionadas con las llaves ajenas y primarias de un sistema relacional.
Considere que el campo P# es la llave primaria de la tabla P, de tal forma que las restricciones de llaves primarias se pueden expresar en el lenguaje básico de integridad como:

```
CREATE INTEGRITY RULE INT_ENT
     BEFORE INSERT OF P FROM NUEVO_P.P#
            UPDATE OF P.P# FROM NUEVO_P.P# :
     CHECK NOT (IS_NULL (NUEVO_P.P#) )
                AND NOT EXIST PX (PX.P# = NUEVO_P.P# ) ;
```

Si se considera que el campo P# es una llave ajena en la tabla SP, y concuerda con la llave primaria de la tabla P, de acuerdo a la sintaxis (1.1) para las llaves ajenas, un efecto RESTRICTED, para el ejemplo considerado,

```
DELETE OF P RESTRICTED
UPDATE OF P.P# RESTRICTED
```

las reglas de restricción podrían representarse como:

```
CREATE INTEGRITY RULE  INT_REF_REST1
     AFTER INSERT OF SP, UPDATE OF SP.P#:
     CHECK EXIST P (P.P# = SP.P#);
CREATE INTEGRITY RULE INT_REF_RES2
    BEFORE DELETE OF P, UPDATE OF P.P# :
    CHECK NOT EXIST SP (SP.P# = P.P#);
```

Un efecto CASCADES para el ejemplo considerado,

```
DELETE OF P CASCADES
UPDATE OF P.P# CASCADES
```

la versión "se propaga" de las reglas anteriores se puede representar como:

```
CREATE INTEGRITY RULE  INT_REF_CAS1
     BEFORE DELETE OF P:
     CHECK NOT EXIST SP ( SP.P# = P.P#)
     ELSE DELETE SP WHERE SP.P# = P.P#

CREATE INTEGRITY RULE  INT_REF_CAS2
     BEFORE UPDATE OF P.P# FROM NUEVO_P.P# :
     CHECK NOT EXIST SP ( SP.P# = P.P#)
     ELSE UPDATE SP.P# FROM NUEVO_P.P#
            WHERE SP.P# = P.P# ;
```

Como se puede observar en estas dos últimas reglas, se ilustra el hecho de que en la parte CHECK ... ELSE ... (verificar ... si no ...) de la proposición CREATE INTEGRITY RULE (crear regla de integridad) representa en realidad un procedimiento disparado, el cual debe invocarse cuando se presenta una condición de disparo especificada. En este caso, el objetivo de esos procedimientos disparados es realizar ciertas acciones de compensación (en caso necesario) para garantizar que la base de datos permanezca en un estado de integridad después de haberse ejecutado el conjunto completo de operaciones.

5 Intérprete de algebra relacional

El componente de manipulación del modelo relacional, está conformado por:

1. Un conjunto de operadores que forman la llamada Álgebra Relacional y,
2. Una operación de asignación que asigna el valor de alguna expresión arbitraria del álgebra a una relación nombrada.

En el álgebra relacional se distinguen las operaciones tradicionales de conjuntos más otras especiales de alto nivel que operan sobre relaciones. Cada uno de estos operadores toma una o dos relaciones como entrada y produce una nueva relación como resultado. Las operaciones tradicionales de conjuntos que se conservan son la unión, intersección, diferencia y producto cartesiano. Y se consideran también, las operaciones relacionales especiales restricción, proyección, reunión y división.
En esencia, dado que el resultado de cualquier operación es un objeto del mismo tipo que los operadores - todos son relaciones - el resultado de una relación puede convertirse en operando de otra, de tal forma que es posible formular expresiones relacionales anidadas.

La gramática libre de contexto original BNF (Backus Naus Form), que se desarrolló en el trabajo [7] genera un pequeño lenguaje para definir estructuras de bases de datos de acuerdo al componente estructural del Modelo Relacional, así como para manipularlas según el álgebra relacional. Este lenguaje está compuesto por un Lenguaje de Definición de Datos (DDL) y un Lenguaje de Manipulación de Datos

(DML). Tanto el DDL como el DML constituyen un intérprete en lenguaje "C". Este intérprete es la herramienta de programación que reconoce un DDL elemental como un subconjunto de instrucciones de un lenguaje anfitrión y que permite definir a las relaciones que se operan. Finalmente el DML manipula algebraicamente las relaciones definidas por el DDL.

Cabe mencionar que el intérprete desarrollado en [7], incluye su analizador sintáctico, su analizador semántico y sus respectivos generadores de errores. Las estructuras definidas por la gramática son totalmente compatibles con estructuras "X-Base" y en conjunto con las herramientas de programación reconocedores de sintaxis y semántica, se logró una exitosa integración con estructuras de CLIPPER, con lo cual podemos afirmar que la gramática extendida que presentamos en la siguiente sección produce reglas gramaticales de integridad compatibles con diferentes productos DBMS relacionales.

6 Gramática para soporte de integridad

Basado en la sintaxis propuesta por Date [2,3], en el lenguaje de integridad hipotético [2,3] considerado en este trabajo, y la gramática para álgebra relacional [7]; en esta sección, se incluye una extensión sintáctica para definir reglas de integridad asociadas a las llaves primarias y ajenas. La extensión de esta gramática no está limitada, dado que es recursiva y ya ha sido probada con productos X-Base, siendo estas las cualidades que invitan a enriquecer el marco gramatical que nos ocupa, y sobre todo, proseguir en el refinamiento de la solución a los problemas de integridad.

La gramática mencionada se describe a continuación, cada producción se acompaña de acciones semánticas que pueden ser abordadas en la codificación o en tiempo de ejecución.

Cabe subrayar que algunas producciones solo se han mencionado ya que exhibirlas explícitamente, llevan a definir otro sub-lenguaje y que precisamente estamos trabajando: (<instrucción>→IF | FOR | ...

| <instr_sql>...,<crea_r_rel>→<regla_relacional><regla_entidades>...,<condicion> →<expresion anidada>)

Especificaciones gramaticales complementarias para el soporte de integridad

G = <Vn,Vt,P,S>

Donde: Vn es el conjunto de símbolos no terminales,
Vt es el conjunto de símbolos terminales,
P es el conjunto de producciones y
S es el conjunto que consta del símbolo inicial.

Vn= {prop_alg, def_alias, asignación, alias, nombre_rel, list_espec_atr, exp_alg, espec_atr, nombre_atr, selección, proyección, exp_infija, primitiva, exp_bool, exp_alg, op_infijo, op_comp, espec_valor, comparación, oper_bool, opdo_bool, cte_bool, cte, id, cadena, alfanum, letra, dígito, cte_alfanum, cte_num, número, num1,

definición, def_campos, tamaño, defcampo, decim, definiciones, bloque, mini_relacional, define_tablas, define_reglas, define_tabla, define_regla, regla_primaria, regla_ajena, crea_regla, objetivo, efecto, claveprim_objetivo, def_claveprim, regla_integridad, borra_regla, accion1, accion, situacion, cond_ok, cond_nook, expr_bool1, expr_bool2, instrucción, nombre_regla, restriccion, condicion, resp_violacion, id_r, regla_integridad_particular, disparador, regla_primaria, regla_relacional, regla_entidades,crea_r_rel, crea_regla_a, regla_relacional_particular, especificaciones_sintacticas_para_relaciones, especificaciones_sintacticas_para_entidades}

Vt= {APODA, DONDE, UNION, INTER, MENOS, VECES, REUNION, DIVIDENTRE, [,], ;, ,, :=, (,), {, }, /, ', NOT, T, F, <, >, =, <>, <=, >=, A, ..., Z, a, ... , z, 0, ..., 9, ", +, -, ., DEFINE-TABLA CHARACTER, NUMERIC, LOGICAL, DATE, PROGRAM, BEGIN, END , : FOREIGN_KEY, REFERENCES, UPDATE OF, ON, ELSE, RESTRICTED, CASCADES, NULLFILES, PRIMARY_KEY, INSERT, UPDATE, REJET, FORSOME, DELETE, CHECK, CREATE_INTEGRITY_RULE, FORSOME, FORALL, IF, FOR, DROP_INTEGRITY_RULE, NULLS-[ALOWED]-DELETE-OF}
S = {mini_relacional}

P= {<mini_relacional> → **PROGRAM** <id>;
<definiciones> **BEGIN** <bloque> **END.**

<definiciones> → <define_tablas><define_reglas>
<define_tablas> → <define_tabla> | <define_tabla><define_tablas>
<define_reglas> → <define_reglas> | <define_regla><define_reglas>
<define_regla> → <regla_primaria> | <regla_ajena> | ><crea_regla_a>
<regla_ajena> → **FOREIGN KEY**(<id>) | **REFERENCES**<objetivo> **NULLS-[NOT]-ALOWED-DELETE-OF** <objetivo><efecto>**UPDATE-OF** <claveprim_obj><efecto>
<objetivo> → <id>
<efecto> → **RESTRICTED|CASCADES|NULLFILES** | <regla_integridad_particular>
<claveprim_obj> → <id>
<bloque> → <prog_alg>|<prop_alg><bloque>
<define_tabla> →**DEFINE TABLA**<alias>{<defcampos>}; <def_claveprim> | <def_claveajena>
<def_campos> →<def_campo> | **[NOT]NULL**<def_campo> | <def_campo><def_campos>
<def_campos> → **CHARACTER**<nombre_atr>(<tamaño>) ; | **NUMERIC** <nombre_atr>(<tamaño> , <decim>) ; | **LOGICAL** <nombre_atr> ; | **DATE** <nombre_atr> ;
<tamaño> → <num1>
<decim> → <num1>
<porp_alg> → <def_alias> | <asignación> | <regla_integridad>
<def_alias> →<alias> **APODA** <nombre_rel> ;

<alias> → <id>
<nombre_rel> → <id>
<asignación> →<nombre_rel> [<lista_espec_atr>] := <exp_alg> ;
<lista_espec_atr>→ <espec_atr> | <espec_atr>,<lista_espec_atr>
<espec_atr> → <nombre_atr> | <nombre_rel><nombre_atr> |
<alias>.<nombre_atr>
<nombre_atr> →<id>
<def_claveprim> →**PRIMARY_KEY**<id>;<regla_primaria> |
PRIMARY_KEY <id>;
<def_claveajena>→**FOREIGN KEY**(<id>); | <def_reglaajena>
<def_reglaajena>→ **REFERENCES**<objetivo>
NULLS-[NOT]-ALOWED-DELETE-OF <objetivo><efecto>**UPDATE-OF** <claveprim_obj><efecto>
<exp_alg> →<selección> | <proyección> | <exp_infija>
<campo_llave> → <id> ;
<selección> → <primitiva> **DONDE** <exp_bool>
<primitiva> → <nombre_rel> | <alias> | (<exp_alg>)
<proyección> → <primitiva> | <primitiva> [<lista_espec_atr>]
<exp_infija> →<proyección><op_infijo><proyección>
<op_infinito> → **UNION** | **INTER** | **MENOS** | **VECES** |
REUNION | **DIVIDENTE**
<exp_bool> → <oper_bool> | <opdo_bool> | **NOT** <opdo_bool>
<oper_bool> → <opdo_bool><opdr_bool><opdo_bool>
<opdo_bool> → (<exp_bool>) | (<comparación>) | <cte_bool>
<comparación> → <espec_atr><op_comp><espec-valor>
<espec_valor> → <espec_atr> | <cte>
<cte_bool> → **T** | **F**
<op_comp> → < | > | = | <> | <= | >=
<id> → <letra> | <letra><cadena>
<cadena> →<alfanúm.> | <alanum><cadena>
<alfanum> → <letra> | <dígito>
<letra> → **A** | ... | **Z** | **a** | ... | **z**
<digito> →**0** | ... | **9**
<cte> → <cte_alfanum> | <cte_num> | <cte_bool>
| <cte_date>
<cte_alfanum> → "<cadena>"
<cte_num> → <número> | + <número> | - <número>
<número> →<num1> | <num1> . <num1>
<num1> → <dígito> | <dígito> <num1>
<cte_date> → '<num1>/<num1>/<num1>'
<regla_integridad>→ <crea_regla> | <borra_regla>
<accion> →<accion1> | <accion1>, <accion>
<accion1> →**INSERT**<situacion | **UPDATE**<situacion>
| **DELETE** <situacion>
<cond_ok> →**CHECK**<ambito><expr_bool1>
| **CHECK**<exp_bool1>

<cond_no_ok> → <expr_bool1>
<expr_bool1> →<expr_bool2> | <expr_bool2><expr_bool1>
<expr_bool2> → <expr_bool> | <instruccion>
<crea_regla> → <nombre_regla><cond_ok><cond_no_ok>
| <nombre_regla>ON <accion>;<cond_ok><cond_no_ok>
<nombre_regla> → **CREATE_INTEGRITY_RULE** <id_r>
<situacion> → <alias>.<nombre_atr>
<ambito> → **FORALL** <situacion> | **FORSOME** <situacion> | ...
<instrucción> → **IF** | **FOR** | ... | <instr_sql>...
<crea_regla_a> → <crea_regla> | <crea_r_rel>
<borra_regla> → **DROP_INTEGRITY_RULE** <id_r>
<id_r> → <id>
<regla_integridad_particular>→<disparador>

<regla_primaria> →<regla_relacional> | <regla_entidades> |
<crea_r_rel>
<crea_regla_a> →<crea_regla> | <crea_r_rel> ...
<crea_r_rel> → <regla_relacional_particular>
<regla_relacional> →<especificaciones_sintacticas_para_relaciones>
<regla_entidades> →<especificaciones_sintacticas_para_entidades>

Este último párrafo gramatical indica adentrarse en la investigación sobre definición sintáctica para relaciones y entidades. Consecuentemente, es posible extender la gramática para definir otro tipo de llaves y el tratamiento requerido de acuerdo a las necesidades de las aplicaciones de los usuarios.

7 Conclusiones

La extensión de la gramática propuesta permite observar cómo la teoría de Lenguajes Formales se puede enlazar a la Teoría del Modelo Relacional para desarrollar algunos elementos que puedan contribuir a solucionar uno de los problemas típicos en Bases de Datos: la solución a los problemas de Integridad.

En esta etapa del trabajo, las pruebas realizadas al intérprete de álgebra relacional con la incorporación de las extensiones gramaticales que permiten la definición y el tratamiento de restricciones de integridad, han sido satisfactorias.

Se espera que las especificaciones gramaticales desarrolladas en el presente trabajo, apoyen la construcción de componentes de programación que puedan incorporarse fácilmente dentro de los DBMS que requieran y permitan la incorporación de un tratamiento complementario para restricciones de integridad basadas en las especificaciones adicionales al modelo relacional presentadas en este trabajo. Por otra parte, también se espera que proporcionen un buen fundamento para el tratamiento de restricciones de integridad a nivel de las aplicaciones, posiblemente mediante procedimientos activados. Finalmente podemos citar que se tiene en proceso un conjunto de trabajos relacionados con el desarrollo de un lenguaje de integridad extendido y el desarrollo de un componente de integridad complementario para MySQL, entre otros.

Bibliografía

1. Codd. E.F. "A Relational Model of Data for Large Shared Data Banks". CAMC 13, núm.6 (junio 1970). Publicado otra vez Milestones of Reseca -Selected Papers 1958-1982 (CAMC, ejemplar del 25 aniversario), CACM 26, núm. 1 (enero de 1983).
2. Date C.J. "Introducción a los Sistemas de Bases de Datos". Vol. 1, 7ª. Ed. 2001, Pearson-Educación de México, S.A. de C.V.
3. Date C.J. "Introducción a los Sistemas de Bases de Datos". Vol. 1, 5ª. Ed. 1998, Addison Wesley Longman de México S.A. de C.V.
4. Ullman J.D. & J. Widom, "Introducción a los Sistemas de Bases de Datos". Prentice-Hall, México, 1999.
5. Galindo-Aburto Ercilia. ,"Integridad en DBMS Relacionales. Casos de Estudio: Access, MySQL, SQL Server.". Asesor: Ma. del Rocío Boone Rojas. Tesis de Licenciatura, Julio-2004, BUAP, Fac. de Cs. de la Computación, Pue. México.
6. Boone-Rojas Ma. del Rocío. "Especificaciones, Extensiones y Ramificaciones del Modelo Relacional", Proyecto de Investigación Interno. 2004, BUAP, Fac. de Cs. de la Computación, Secretaría de Investigación y Estudios de Posgrado. Pue. México.
7. Bernabé-Loranca Beatriz. ,"Procesador para Álgebra Relacional". Asesor: Ma. del Rocío Boone Rojas. Tesis de Licenciatura 1994, BUAP, Fac. de Cs. de la Computación, Pue. México.
8. Bernabé-Loranca, M.B. & Boone-Rojas, M. R.; "Especificaciones Complementarias para el Soporte de Integridad en DBMS relacionales", CIC 2004, Congreso Internacional de Computación, IPN, México, DF, Octubre 2004.
9. Boone-Rojas Rocío, Soriano-Ulloa M.A., Bernabé-Loranca, "Marco de Evaluación para el Componente de Integridad en DBMS relacionales". II Congreso Nal. de Cs. de la Computación. Puebla, Pue. Méx. Nov. 2004.
10. Pastor-López O, Pedro Blesa P.; "Gestión de Bases de Datos", Universidad Politécnica de Valencia. Servicios de Publicaciones, Valencia 2000.
11. Pastor-López, "Análisis de Restricciones de Integridad en el Nivel Conceptual", VII Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software (IDEAS 2004), Arequipa, Perú mayo 2004.