

Ingeniería inversa de bases de datos

Proceso de extracción de la estructura de la base de datos

Juan Carlos Moreno A.

UNIVERSIDAD ALFREDO PEREZ GUERRERO – Quito – Ecuador – 2010

Reingeniería de Bases de datos – Ing. Francisco Tixi



Abstracto

El objetivo de esta fase es recuperar el esquema completo de la base de datos, incluyendo todas las estructuras y constraints implícitos y explícitos. El principal problema de la fase de extracción de la estructura de datos es descubrir y hacer este descubrimiento explícito a través del proceso de refinamiento, las estructuras y los constraints que fueron implementados implícitamente o simplemente descartados durante el proceso de desarrollo. En este documento vamos a definir el concepto de construcción implícita, describiremos el proceso de extracción de código DDL y luego analizaremos los problemas y técnicas de las construcciones implícitas. Concluiremos con la aplicación de estas técnicas para recuperar claves foráneas.

Constructores implícitos vs explícitos

Un constructor implícito es un componente o propiedad de una estructura de datos que es declarada a través de una sentencia específica DDL. Un constructor implícito es un componente o una propiedad que encierra a la estructura de datos pero que no ha sido declarada explícitamente. En general, el DMS (Software Modelador de Base de datos) no está conciente de los constructores implícitos, aunque puede contribuir a su administración (a través de triggers por ejemplo). El solo análisis de las sentencias DDL deja la implícitud de los constructores indetectada. El ejemplo más popular de esto es ciertamente es una clave foránea, que voy a utilizar para explicar esta posición. Consideremos que el código de la figura 5-1 en la cual dos tablas están unidas por una clave foránea. Podemos decir que esta clave foránea es una construcción implícita, pues hemos utilizado una sentencia para declararla.

```
create table CUSTOMER(C-ID integer primary key,
                     C-DATA char(80))
create table ORDER(  O-ID integer primary key,
                   OWNER integer
                   foreign key(OWNER) references CUSTOMER)
```

Figura 5.1: Ejemplo de una clave foránea explícita.

La figura 5.2 representa un fragmento de una aplicación en el cual no se han declarado claves forneas, pero fuertemente sugiere que la columna OWNER debe tener una clave foránea. Si estamos convencidos de este comportamiento debe ser adquirido como una regla absoluta, entonces OWNER es una clave foránea implícita.

```
create table CUSTOMER(C-ID integer primary key,
                     C-DATA char(80))
create table ORDER(  O-ID integer primary key,
                   OWNER integer)
...
exec SQL select count(*) in :ERR-NBR from ORDER
       where OWNER not in (select C-ID from CUSTOMER)
end SQL
...
if ERR-NBR > 0 then
    display ERR-NBR, 'referential constraint violations';
```

Figura 5.2: Ejemplo de una (posible) clave foránea implícita.

Al examinar el poder de expresión de los DMS comparado con los formalismos de la semántica, y analizando como trabajan los programadores, podemos identificar cinco grandes fuentes de construcciones implícitas.

1. Ocultación de estructura
La ocultación de estructura concierne a los fuente de la estructura de datos o constraint S1, que puede ser implementado en el DMS. Consiste en declararlo como otra estructura de datos S2 que es más general y menos expresiva que S1. En las aplicaciones hechas con COBOL por ejemplo, un campo compuesto/multivalor o una secuencia de campos contiguos puede ser representado como un simple valor atómico. El origen de la ocultación

de estructura es siempre una decisión del programador que trata de alcanzar los requerimientos como campo de reusabilidad, genericidad, simplicidad, eficiencia así como la consistencia con los antiguos componentes de la aplicación.

2. Expresión generica
Algunos DMS ofrecen funcionalidad de propósito general para asegurar una gran variedad de constraints de datos. Por ejemplo, el actual DMS relacional propone verificaciones de predicados de columnas, vistas con opciones de revisión, triggers y procedimientos almacenados. Estas poderosas técnicas pueden ser usadas para programar validaciones y el manejo complejo de constraints de anera centralizada. Los constraints tal como la referencia de integridad pueden ser codificados de muchas formas, su elicitación puede probar que son mucho más complejas que las claves foráneas declaradas.
3. Estructuras no declarables
Las estructuras no declarables tienen un origen diferente. Son estructuras o constraints que no pueden ser declarados en un DMS objetivo, y por lo cual no pueden ser revisadas por otros medios, esternamente al DMS, como las secciones de procesos en la aplicación programa o interfaz de usuario. Más amenudola revisión de secciones no son centralizadas, pero son distribuidas y duplicadas (frecuentemente en diferentes versiones), através de las aplicaciones. Por ejemplo, los archivos estándares comúnmente incluyen claves foráneas, los DMS actuales ignoran este construct.
4. Propiedades de entorno
En algunas situaciones el el sistema de entorno garantiza que los datos externos que deben ser almacenados en la base de datos para satisfacer una propiedad en específico. Los desarrolladores no le ha encontrado el sentido de trasladar esta propiedad a las estructuras de datos, o para asegurarla através de los DMS o las técnicas de programación. Por supuesto estos constraints no pueden estar basados en una estructura de datos o análisis del programa. Por ejemplo si el contenido de un archivo secuencial viene de una fuente externa en la cual se garantiza que es única por una de su campo entonces la base de datos hedera esta propiedad, y se le puede asignar efectivamente un identificador.
5. Especificaciones perdidas.
Las especificaciones perdidas corresponden al hecho de que ha sido ignorada intencionalmente o no durante el desarrollo del sistemas. Este fenómeno corresponde a fallas en el sistema que puede ser traducido a datos corruptos. De todas formas, las especificaciones perdidas pueden ser propiedades de entorno no detectadas, en el cual los datos generalmente son válidos.

Extracción explícita de los constructores

Este proceso claramente es el más fácil en la ingeniería inversa de bases de datos. Consiste en asociar las abstracciones físicas con cada constructor DDL. Este set de reglas son fáciles de establecer en la mayoría de DMS, proveen un blanco del

modelo de abstracción física que incluye un rico set de características. Debemos recordar que cada DDL, inclusive en los más modernos DMS incluye clausulas destinadas a declarar conceptos físico (indexes, clusteres), conceptos lógicos (tipos de registro, campos, claves foráneas) así como conceptos conceptuales (identificadores). Distribuir el modelo de datos a veces puede resultar confuso. La tabla 1 y tabla 2 muestran las reglas principales de abstracción principal para convertir código CO-BOL and SQL-2 en estructuras físicas abstractas.

COBOL statement	Physical abstraction
select S assign to P	define storage S
record key is F	define a primary identifier with field F; define an access key with field F.
alternate record key is F	define a secondary identifier with field F; define an access key with field F.
alternate record key is F with duplicates	define an access key with field F.
fd S 01 R	define record type R within storage S.
05 F pic 9(n)	define numeric field F with size n, associated with its parent structure (record type or compound field).
05 F pic X(n)	define alphanumeric field F with size n, associated with its parent structure (record type or compound field).
05 F1. 10 F2 ...	define compound field F1, with components F2, etc.
05 F ... occurs n times	define multivalued field F, with cardinality n.

Tabla1: Abstracción principal de reglas COBOL para estructura de archivos.

Tabla2: Abstracción principal para estructuras relaciones.

SQL statement	Physical abstraction
create dbspace S ...	define storage S
create table T (...) in S	define record type T within storage S.
name numeric(n)	define numeric field F with size n.
name char(n)	define character field F with size n.
... not null	define the current field as mandatory
primary key (F)	define a primary identifier with field(s) F
... unique (F)	define a secondary identifier with field(s) F
foreign key (F) references T	define field(s) F a foreign key referencing record type T.
create index ...	define an access key with field(s) F.
create unique index on T(F)	define a secondary identifier with field(s) F; define an access key with field(s) F.

Casi todas las herramientas CASE proponen algun tipo de extractor DLL para los más populares sistemas de manejo de base de datos. También nos sirve como diccionario de datos

rHhÖœ@éöœf√KÄ}çÖçv...Ë°û\$xØk_a[äg

~F

rlëFH"âãõÓío\@Zö/#Iú':éCv

ßñE

ö"æ4çé"3Sâvxç øœ%ÅÖ4^FvtvªΩ-œ-çZx

""

°"™<<

CEìÆà'WÔ"\$v¥XPÀôàBÓÿÊèSÇ

-îÚÉÁÉ

2)L6AÅªÜõ

çJd·ú@bçq^%oø|»g#Gø

Z/i≤;œÂ"ë_ÛÖú0Açm%ooμ!^°çCüh™ç-ë(yüçç<ûÄf)‡...çÍ´ç

P—B9çç2πnö•• NÅΣæ

©≥P≠∅.∫y™ÿ İLvÔ/"iŪVπK

Ê∫^Ö!eœ∫D√DR±ζÑ√{

Py8'(w4Ie»√22ê`Äjâ

@/P`@!Ér_p#auÿ@`%'»â-@" &Aô[]5Xë;~l4h@ ÇÅv

'lÔ-~@xÒfl@eXü#\$4çQÂ<Ú;ãW·@t>ÿgÄyπi;Tá~' @fi&, Ñ¥Úá-
NäÂ'fip'W#Å@P#"8;™xÄ (û#Åw`≤çDÖP

<ë#AA™}pAÉ<}∞Ä~-ëêù. J7-ë÷#÷€Ä6ce}Ä0-ê[]vÄx@EYü-ñ@;d
ûÉM Êh†Cìç0≈s&ÅA#xÿb~ò@â\CEIÇá†]!vÁ#È/*Ô!ãseP-
i;cn@a_äGbfii«"â]n\$ôáB❁iêí&Y4q»s,ò€q∞ÿãfiG
d;@ÅÛ&@j†,ùH ÓHÈì 1@CK≤F¶

"ê†ì8x»Ê, Ná†~∅-÷gÉÅX]º+-êÅ~¶{®~£]+Ç¶ç.
oøË:-.¶+û'º¶\$,†ø*6x¶. æç¶\0o≤

Ü §ø%f©]ô‡~Ë≈klGnyflµ`à è;îÄÄä`€Ä@ä¶k+"pAÕ%

(ÙÄ¬≈º...ð<lÙ“L∞;PG-1‘TWm1’XgM5ÃzĪÃ¿∂`á-^ydóm^Yhã
t”lÁ-Ö’ÆÆΣΣÿÚO IPÖZ/ ...

8"ÄÜÄ,3`

QÄðc@h%í;ÁÂ—!μq±πCEP7ëæ d)°;Cçs-Lt+HÎ
\$ÛP§<¿;a¿*ΣãP#Π'†-Ä!;Û<... @zs†°E°æ°Áüè9Ò3çΔÈTA%TT\$F
ÄgCÄi¿O^¿;»@~X1≠x'ÉJ Ä<çÄ<Dñãİ-Ä÷AZ¿@

0É

ò“â”∞\$∞é0—ìM ß

I+_P%o\$ß±¶ÿÛ°Ã\$4

C

≤ 2uÑ°xhúzGΠ•áUPL;