MINISTERO DELLO SVILUPPO ECONOMICO
DIREZIONE GENERALE PER LA LOTTA ALLA CONTRAFFAZIONE
UFFICIO ITALIANO BREVETTI E MARCHI

DOMANDA NUMERO	**102007901508066**
Data Deposito	**27/03/2007**
Data Pubblicazione	**27/09/2008**

Sezione	Classe	Sottoclasse	Gruppo	Sottogruppo
G	06	F		

Titolo

METODO E FORMALISMO PER INVIARE ISTRUZIONI A DATABASE DISTRIBUITI REALIZZATO MEDIANTE PROGRAMMA PER COMPUTER.

Descrizione dell'invenzione industriale dal titolo:

"Metodo e formalismo per inviare istruzioni a DataBase distribuiti realizzato mediante programma per computer"

a nome di: UNIVERSITA' DEL SALENTO

con sede in: Piazza Tancredi, 7 - 73100 Lecce

Inventori designati: FIORE Sandro Luigi, CAFARO Massimo, ALOISIO Giovanni.

* * * * * * * * * * *

La presente invenzione si riferisce ad un metodo e formalismo per inviare istruzioni a DataBase distribuiti realizzato mediante programma per computer.

Nel campo dell'informatica esiste da diversi anni un certo grado di standardizzazione per quanto riguarda il formalismo da utilizzarsi per estrarre od inserire dati in DataBase Relazionali (DB), siano essi commerciali od Open Source. Più delicata risulta l'estrazione di informazioni da banche dati non strutturati per le quali risulta indispensabile l'utilizzo di "wrapper", ovvero di programmi ad hoc per l'interpretazione stringhe testuali.

Un problema estremamente serio attiene l'estrazione e l'inserimento di dati da DB remoti, specie quando la mole di dati diviene importante. In tal caso ogni singola richiesta e ciascuna relativa risposta viaggiano sulla rete internet, determinando la congestione della rete stessa ed un generale rallentamento della rappresentazione del risultato delle operazioni.

Un altro problema altrettanto importante attiene la disomogeneità di caratteristiche funzionali tra i DB forniti dalla tecnica nota.

Un ulteriore problema attiene al fatto che l'operatore debba, da remoto, inviare le singole istruzioni ai DB attendendo l'esito di ciascuna istruzione spendendo una grande quantità di tempo/uomo. Tale modo di operare, male si presta alla parallelizzazione di operazioni e soprattutto, non consente di sfruttare le ottimizzazioni di schedulazione che dipendono dal carico operativo dei diversi Server/DBMS e dalla natura medesima dell'architettura di ciascun DB.

Un ultimo problema è relativo alla conversione di un DB in un altro, dovendosi scrivere programmi ad hoc per estrarre dati dall'uno e popolare il secondo.

Pertanto scopo della presente invenzione è quello di superare tutti gli inconvenienti suddetti e di indicare un metodo per inviare istruzioni a DB distribuiti realizzato mediante programma per computer.

Uno scopo fondamentale dell'invenzione è quello, dunque, di ridurre drasticamente il carico di dati che viaggia sulla rete internet, avvantaggiandosi di una minore influenza sullo stato di congestione della stessa.

Un altro scopo della presente invenzione è quello di fornire un esempio di formalismo secondo cui codificare ed interpretare le direttive, ovvero istruzioni complesse di alto livello, dell'operatore e più in generale del Data Producer.

Un ulteriore scopo dell'invenzione consiste nell'elevare il livello di astrazione nella fase di codifica delle direttive, in modo da non patire le diversità tra i diversi DB, proprietari, OpenSource, nuovi od obsoleti, discen-

dendone che anche la conversione di un DB in un altro diventa immediata. Analogamente, ininfluente diviene la asincronia delle transazioni parallelizzate in relazione alle schedulazione effettuate dai singoli DB.

Un altro scopo della presente invenzione consiste nel rendere possibile la parallelizzazione delle transazioni su più DB, o nell'ambito del medesimo DB, o nell'ambito della medesima tabella, garantendosi la coerenza delle successione delle transazioni elementari, tipicamente in SQL.

Un altro scopo ancora è quello di demandare la supervisione delle transazioni elementari ad un computer munito di apposito software, ottenendosi un notevole risparmio tempo/uomo.

E' oggetto della presente invenzione un metodo secondo il quale ridistribuire i compiti di elaborazione e supervisione delle transazioni operate su DB e reti di DB remote, garantendo un elevato grado di astrazione per il Data Producer e fornendo un esempio di formalismo (di seguito indicato con l'acronimo GXBL sulla scorta dei termini Grid relational catalog, Xml e Bulk Load) con cui scrivere/interpretare direttive mediante il quale sia possibile realizzare tale metodo.

E' particolare oggetto della presente invenzione un metodo per inviare istruzioni a DataBase distribuiti che prevede una fase in cui le direttive vengano formattate in un documento che rispetti un opportuno formalismo; una fase in cui il documento venga spedito; una in cui il documento venga interpretato e tradotto in istruzioni elementari ed, in fine, una in cui dette istruzioni vengano eseguite sotto la supervisione di un apposito supervisore. Inoltre è particolare oggetto della presente invenzione un formalismo secondo cui formattare i documenti da spedire. In fine, è

particolare oggetto della presente invenzione una infrastruttura informatica che consente di realizzare tale metodo, così come prescritto nelle rivendicazioni che fanno parte integrante della presente descrizione.

Ulteriori scopi e vantaggi della presente invenzione risulteranno chiari dalla descrizione particolareggiata che segue di un esempio di realizzazione della stessa (e di sue varianti), dai disegni annessi e dai listati del formalismo dati a puro titolo esplicativo e non limitativo, in cui:

nella figura 1 è rappresentato un possibile scenario di utilizzo del metodo;

nella figura 2 è riportata una rappresentazione grafica del di detto formalismo GXBL;

nella figura 3 è riportata una rappresentazione grafica sella sottosezione TABLE di cui alla figura 2 con in evidenza i singoli componenti DCL, DDL e DML di detto formalismo GXBL;

nella figura 4 è riportata una rappresentazione grafica dell'espansione del componente DML di cui alla figura 3;

nella figura 5 è riportata graficamente il diagramma relazionale tra due tabelle;

nelle figure 6 e 7 si riporta schematicamente il flusso dei dati parallelizzato a livello di DB ed a livello si Tabella;

nella figura 8 si riporta schematicamente l'interruzione di un processo di elaborazione a seguito della perdita di un documento, formattato secondo il formalismo GXBL (documento GXBL), facente parte della successione ordinata di un processo comprendente più documenti GXBL;

nelle figura 9.1 e 9.2 è mostrato un esempio di listato di DTD del formalismo GXBL;

nelle figure 10.1, 10.2, 10.3 e 10.4 è riportato un esempio di listato XSD che rappresenta lo schema del generico documento GXBL;

nella figura 11 è riportato, a titolo di esempio, il listato corrispondente ad una istanza GXBL in cui si effettua un'operazione di INSERT (DML).

Il metodo oggetto della presente invenzione prevede che a livello di Data Producer 1 (Producer) le direttive vengano formattate mediante un traduttore residente nel Data Producer secondo un formalismo, per es. GXBL, ed inviate come uno o più documenti formattati 2, attraverso una generica rete, per es. internet (WAN), all'interprete GXBL 3 (GXBL Translator), il quale supervisiona l'esecuzione delle transazioni elementari, generalmente in formato SQL 4, da parte dei diversi DB (relational DB).

L'interprete GXBL assolve al compito di tradurre i documenti GXBL 2 in SQL 4 e di sovrintendere alla corretta esecuzione delle transazioni in termini di esito e di rispetto dell'ordine logico di esecuzione nel rispetto dei parametri e attributi di cui il documento si compone che saranno illustrati nel seguito.

Vantaggiosamente, il Data Producer, dopo aver definito le direttive può dedicarsi ad altre attività, mentre un software che rispetti il formalismo, per es. GXBL, oggetto della presente invenzione assolve alla funzione di traduttore/sender ed un altro assolva alla funzione di receiver/interprete/supervisore.

In particolare, detto traduttore/sender, generalmente risiede nel Data Producer, che fisicamente può coincidere con uno od una rete di computer, mentre detto receiver/interprete/supervisore è un software che risiede su almeno un computer posto, per es., al bordo di una rete locale a cui sono connessi diversi e differenti DB.

Inoltre, il Data Producer non è più costretto a tener conto dei link fisici ai diversi DB e relative Tabelle nell'impartire al traduttore le direttive.

Nelle figura 9.1 e 9.2 è riportato il DOCUMENT TYPE DEFINITION (DTD) di una realizzazione preferita del formalismo GXBL, che risulta fondamentale per convalidare un documento XML creato secondo tale formalismo.

Nelle figure 10.1, 10.2, 10.3 e 10.4 è riportato il listato XSD che rappresenta lo schema del generico documento GXBL.

Ne consegue che i documenti GXBL XML sono semplicemente istanze o oggetti relativi a tale schema.

Nella figura 2 si evidenzia il parametro:

- PARALLEL, quello indicato più in alto e riferito al documento GXBL, attiene alla possibilità di parallelizzare le operazioni sugli n DB coinvolti nelle sezioni DATABASE;
- IDENTIFIER, il quale rappresenta un numero di una sequenza se una successione di documenti devono essere eseguiti secondo un preciso ordine, in particolare, tale attributo può assumere valore 0..+∞, da intendere in questo modo: (i) 0 per documenti indipendenti, (ii) [1.. +∞] per documenti dipendenti e da interpretare secondo valori strettamente crescenti dell'attributo IDENTIFIER);

- PRIORITY, il quale è diverso da zero se IDENTIFIER è posto a zero e serve a privilegiare l'elaborazione di un documento rispetto ad altri.

A livello di DB :

- è possibile definire gli utenti che possono accedervi ed i relativi privilegi. Tali informazioni sono contenute nella sottosezione INSTANCES_DCL;
- PARALLEL attiene all'esecuzione parallela di più sottosezioni contenute nel medesimo documento e riferite, dunque, al medesimo DB (DB_NAME)
- DB_OPER può assumere i valori CREATE, UPDATE, DELETE in accordo col precedente DTD.

Nella figura 3 si nota come nella sezione TABLE sia possibile specificare le tre parti INSTANCES_DML, INSTANCES_DDL ed INSTANCES_DCL che sono state descritte precedentemente. La parte INSTANCES_DCL si aggiunge a quella del livello superiore, ovvero di DB, sovrascrivendo definizioni già presenti precedentemente e aggiungendone di nuove. La sezione INSTANCES_DDL, in particolare nella sottosezione ATTRIBUTES_DDL, si descrive la struttura della tabella, i nomi degli attributi, il tipo etc., mentre nella sottosezione FOREIGN_KEYS_DDL si descrivono le relazioni (constraints) di chiave esterna.

La sezione INSTANCES_DML contiene *N* sottosezioni RECORD_DML, ovvero, una per ogni operazione di manipolazione dei dati che si intende effettuare.

Anche in questo caso, come per i livelli gerarchici superiori (documento GXBL e DATABASE) per l'elemento TABLE è definito l'attributo PARALLEL. E' possibile impostare tale attributo a TRUE o FALSE per indicare che l'interpretazione a *"livello di records"* della tabella può avvenire in parallelo, ovvero l'interpretazione può andare in parallelo su *N* records. Nel caso di documento GXBL con un'unica sezione RECORD, impostare l'attributo PARALLEL nella sottosezione TABLE a TRUE o FALSE è del tutto ininfluente.

La figura 4 espande la generica sottosezione RECORD_DML della figura precedente, ovvero del precedente e superiore livello gerarchico in cui si specificava la tabella su cui operare. REC_OPER può assumere i valori (INSERT, DELETE, UPDATE, FORCED_UPDATE) in accordo col DTD. All'interno di questa sezione è possibile individuare due sottosezioni:

- ATTRIBUTES_DML in cui vengono inseriti gli attributi col relativo valore, dei campi dati (ovvero quelli che non rappresentano chiavi esterne);
- FOREIGN_KEYS_DML in cui vengono inserite le informazioni relative ai riferimenti logici, ovvero quelle informazioni a partire dalle quali è possibile ottenere il valore della foreign key effettivo.

E' importante notare che la sezione FOREIGN_KEYS_DML contiene al suo interno una sezione ATTRIBUTE, ma anche una sezione FOREIGN_KEY_DML che permette di risolvere situazioni di chiavi esterne

annidate a più livelli (questa caratteristica è definita Nested_Foreign_Keys).

In accordo con quanto detto precedentemente, i riferimenti fisici ai DB e relative tabelle vengono risolti all'atto dell'interpretazione da parte dell'interprete GXBL 2.

Di seguito è possibile analizzare un esempio di traduzione del riferimento logico in riferimento fisico.

Si supponga di avere due tabelle *table1* e *table2* all'interno di una database relazionale legate da una relazione 1 a N, e si supponga di voler effettuare un'operazione di inserimento all'interno della tabella *table2*. Tale esempio è illustrato in figura 5.

Nel modello relazionale la tabella *table2* contiene il campo *idref1* che è la chiave esterna al campo *id1* della tabella *table1* e che nello schema non è evidenziato. Un'operazione di INSERT (DML) nella tabella *table2* è pertanto rappresentata tramite la seguente istanza GXBL, come mostrato in figura 11.

Il listato GXBL di figura 11 viene tradotto nella seguente query SQL:

insert into table2(key2,satellitedata2,idref1) values(key2value, satdata2, ***idref1****);*

dove **idref1** viene ricavato preventivamente mediante la sezione FOREIGN_KEYS_DML dalla quale viene ricavata la seguente query:

"select id1 from table1 where key1= key1value;"

In figura 6 è riportato il Run Time Execution Model dell'interpretazione di un documento GXBL con tre sezioni TABLE (definite come SEQUEN-

TIAL, ovvero con l'attributo PARALLEL impostato a FALSE) ed una sezione DATABASE con attributo PARALLEL impostato a TRUE.

Come si evince dalla figura, sul documento GXBL in questione possono essere avviate in parallelo le interpretazioni delle tre sezioni TABLE. Per ognuna di esse il flusso è comunque sequenziale per via dell'impostazione dell'attributo PARALLEL nella sottosezione TABLE a FALSE.

Analogamente anche a livello TABLE (Figura 7) è possibile inserire sezioni parallele suddividendo N operazioni sui records in blocchi di dimensione opportuna, con l'obiettivo di ripartire il carico di lavoro su più processori.

Il formalismo GXBL supporta pertanto la parallelizzazione dei vari blocchi mentre in generale gli interpreti GXBL potrebbero non implementare questa caratteristica in relazione al tipo di licenza concesso.

Un'altra caratteristica estremamente importante del presente formalismo è la possibilità di ordinare l'interpretazione dei documenti GXBL per singoli documenti (tramite numeri di sequenza). L'ordinamento è reso possibile dalla presenza dell'attributo IDENTIFIER del tag GXBL dei documenti.

Utilizzando l'attributo IDENTIFIER l'esecuzione dei documenti avviene rigorosamente in ordine secondo l'ordine indicato nel medesimo parametro come in figura 8.

La figura evidenzia come il processo di interpretazione si fermi attendendo l'arrivo del documento con identificativo pari a 5 (nonostante quelli con identificativo 6 e 7 siano già disponibili). L'attributo IDENTI-

FIER rappresenta un numero di sequenza il cui scopo è mantenere traccia dell'ultimo documento GXBL analizzato.

L'attributo IDENTIFIER ha senso laddove determinate operazioni (come la creazione di una tabella) debbano essere eseguite prima di altre (ingestion della tabella precedentemente creata), arrivando a bloccare, se necessario il processo di interpretazione fino all'arrivo del documento mancante.

Nelle tre sottosezioni che seguono si riportano alcune informazioni di più basso livello relativamente alla parte GXBL-DDL, DML e DCL

FORMALISMO GXBL-DDL (DATA DEFINITION LANGUAGE)

Attributi *DATABASE*
DB_NAME = stringa di caratteri
DB_OPER = tipo enum {CREATE, *default=UPDATE*, DELETE}

Attributi *TABLE*
TB_NAME = stringa di caratteri
TB_OPER = tipo enum { CREATE, *default=UPDATE*, DELETE}

Attributi *ATTRIBUTE*
AT_NAME= stringa di caratteri
AT_TYPE= tipo enum {default = INT, CHAR, VAR_CHAR, SMALL_INT, FLOAT, DOUBLE, DATE,TIME, TIMESTAMP}
AT_DIM = numero di elementi (utile per i var_char)
AT_PREC = tipo enum { default = SINGLE, DOUBLE}
AT_ NULL = tipo enum { default = FALSE, TRUE}
AT_PRIMARY_KEY = tipo enum { default = FALSE, TRUE}
AT_OPER = tipo enum { default = ADD, DELETE, ALTER}
AT_ UNIQUE = tipo enum { default = FALSE, TRUE}

La notazione (-) indica che il valore dell'attributo non ha importanza perché non cambia la semantica dell'operazione. Ciò significa che quell'attributo non verrà valutato.

OPERATION	DB_OPER	TB_OPER	AT_OPER
DB CREATION	CREATE	-	-
DB_DELETE	DELETE	-	-
TABLE CREATION	UPDATE	CREATE	-
TABLE DELETE	UPDATE	DELETE	-

TABLE UPDATE	UPDATE	UPDATE	ADD
TABLE UPDATE	UPDATE	UPDATE	DELETE
TABLE UPDATE	UPDATE	UPDATE	UPDATE

FORMALISMO GXBL-DML (DATA MANIPULATION LANGUAGE)

Attributi *DATABASE:*
DB_NAME = stringa di caratteri
DB_OPER = tipo enum { *UPDATE* }

Attributi *TABLE*
TB_NAME = stringa di caratteri
TB_OPER = tipo enum { *UPDATE* }

Attributi *RECORD*
REC_OPER = tipo enum {default= INSERT, UPDATE, DELETE, FORCED_UPDATE}

Attributi *ATTRIBUTE:*
AT_NAME= stringa di caratteri
AT_WHERE = tipo enum { default=FALSE, TRUE }

La notazione (-) indica che il valore dell'attributo non ha importanza perché non cambia la semantica dell'operazione. Ciò significa che quell'attributo non verrà valutato.

Tra gli attributi descritti, degno di nota è nel tipo enumerato REC_OPER il valore FORCED_UPDATE la cui interpretazione è la seguente:

- aggiornamento della tupla negli attributi non chiave se la tupla stessa è già stata inserita;
- inserimento della tupla descritta nella sezione RECORD, se la tupla in questione non esiste.

OPERATION	**REC_OPER**	**AT_WHERE**
INSERT QUERY	INSERT	-
UPDATE QUERY	UPDATE	FALSE
UPDATE QUERY	UPDATE	TRUE
DELETE QUERY	DELETE	FALSE
DELETE QUERY	DELETE	TRUE
FORCED_UPDATE QUERY	FORCED_UPDATE	TRUE
FORCED_UPDATE QUERY	FORCED_UPDATE	FALSE

FORMALISMO GXBL-DCL (DATA CONTROL LANGUAGE)

Attributi *DATABASE:*

DB_NAME = stringa di caratteri
DB_OPER = tipo enum { *UPDATE* }

Attributi *USER*
US_NAME = stringa di caratteri
US_OPER = tipo enum {default= INSERT, UPDATE, DELETE}
US_PRIVS = tipo enum {SELECT, INSERT, DELETE, UPDATE, MNG_USER, MNG_DB, DB_ROOT}

La notazione (-) indica che il valore dell'attributo non ha importanza perché non cambia la semantica dell'operazione. Ciò significa che quell'attributo non verrà valutato.

La notazione (*) indica la possibilità di avere uno qualsiasi dei valori ammessi dall'attributo assegnando una semantica differente e legata al valore dell'attributo.

OPERATION	**US_OPER**	**US_PRIVS**
USER INSERT	INSERT	*
USER UPDATE	UPDATE	*
USER DELETE	DELETE	*

Sono possibili varianti realizzative all'esempio non limitativo descritto, senza per altro uscire dall'ambito di protezione della presente invenzione, comprendendo tutte le realizzazioni equivalenti per un tecnico del ramo.

Risulta chiaro che il metodo oggetto della presente invenzione consente di demandare all'interprete GXBL la decodifica in operazioni elementari SQL, offrendo un alto livello di astrazione al Data Producer, essendo i collegamenti fisici introdotti dall'interprete medesimo, e diventando immediata la conversione di un DB in un altro.

Secondo tale metodo il traffico sulla rete internet risulta drasticamente ridotto e si può ottenere un altissimo grado di parallelizzazione delle operazioni, demandando all'interprete GXBL il compito di sovrintendere

all'esecuzione delle transazioni che debbono rispettare un preciso ordine logico, sequenziale o prioritario.

La presente invenzione può essere vantaggiosamente realizzata tramite un programma per computer che comprende mezzi di codifica per la realizzazione di uno o più passi del metodo, quando questo programma è fatto girare su di un computer. Pertanto si intende che l'ambito di protezione si estende a detto programma per computer ed inoltre a mezzi leggibili da computer che comprendono un messaggio registrato, detti mezzi leggibili da computer comprendendo mezzi di codifica di programma per la realizzazione di uno o più passi del metodo, quando detto programma è fatto girare su di un computer.

Sono possibili varianti realizzative all'esempio non limitativo descritto, senza per altro uscire dall'ambito di protezione della presente invenzione, comprendendo tutte le realizzazioni equivalenti per un tecnico del ramo.

Dalla descrizione sopra riportata il tecnico del ramo è in grado di realizzare l'oggetto dell'invenzione senza introdurre ulteriori dettagli costruttivi.

RIVENDICAZIONI:

1. Metodo per inviare istruzioni a DataBase distribuiti comprendente le seguenti fasi:
   a. Formattazione di direttive complesse di alto livello generate da un Data Producer (1) secondo un formalismo per ottenere un documento formattato;
   b. Spedizione di detto documento formattato (2);
   c. Ricezione, interpretazione di detto documento formattato da parte di almeno un interprete (3) per ottenere istruzioni elementari;
   d. Esecuzione e supervisione di transazioni relative alle istruzioni elementari (4) su almeno un DB o tabella o record.
2. Metodo, secondo la rivendicazione 1, in cui l'esecuzione e supervisione delle transazioni elementari nel rispetto dell'ordine logico di esecuzione e di parametri e attributi contenuti nel documento formattato.
3. Metodo, secondo la rivendicazione 1, in cui i riferimenti fisici ai DB siano inseriti da detto interprete durante l'elaborazione.
4. Metodo, secondo la rivendicazione 1, in cui il documento formattato contenga una o più sottosezioni DataBase.
5. Formalismo per formattare i documenti, secondo la rivendicazione 1, in cui il documento formattato contenga un attributo PARALLEL, di valore TRUE se è richiesta la parallelizzazione delle sottosezioni.
6. Formalismo, secondo la rivendicazione 5, in cui il documento for-

mattato possegga l'attributo sequenziale IDENTIFIER, di valore zero, se il documento non appartiene ad una successione ordinata di documenti, diverso da zero altrimenti.

7. Formalismo, secondo la rivendicazione 5, in cui il documento formattato possegga l'attributo PRIORITY per la gestione delle priorità di esecuzione dei documenti.
8. Formalismo, secondo la rivendicazione 5, in cui l'interprete (3) effettua la schedulazione del prossimo documento in relazione ai parametri IDENTIFIER o PRIORITY.
9. Formalismo, secondo la rivendicazione 5, in cui il documento formattato contiene, a livello di sottosezione DB i parametri:
    a. DB_OPER, di valore INSERT o DELETE o UPDATE o FORCED_UPDATE;
    b. DB_NAME;
    c. DB_PARALLEL, di valore TRUE o FALSE.
10. Formalismo, secondo la rivendicazione 5, in cui il documento formattato, a livello di tabella:
    a. TB_OPER di valore INSERT o DELETE o UPDATE o FORCED_UPDATE;
    b. TB_NAME;
    c. TB_PARALLEL, di valore TRUE o FALSE.
11. Formalismo, secondo la rivendicazione 5, in cui il documento formattato, a livello di tabella abbia le seguenti sottosezioni relative ai componenti DCL, DDL, DML:
    a. INSTANCE_DCL;

b. INSTANCE _ DDL;

c. INSTANCE _ DML.

12. Formalismo, secondo la rivendicazione 5, in cui la sottosezione INSTANCE_DCL abbia almeno un parametro USER comprendente i seguenti attributi:

a. US_PRIVILEGE;

b. US_OPER;

13. Formalismo, secondo la rivendicazione 5, in cui il documento formattato nella sottosezione INSTANCE_DDL abbia almeno un parametro ATTRIBUTE_DDL comprendente i seguenti attributi:

a. AT_TYPE;

b. AT_PRIMARY_KEY :

c. AT_NULL;

d. AT_UNIQUE;

e. AT_DIM;

f. AT_PREC;

g. AT_OPER;

14. Formalismo, secondo la rivendicazione 5, in cui il documento formattato nella sottosezione INSTANCE_DDL abbia almeno un parametro FOREIGN_KEYS_DDL comprendente i seguenti attributi:

a. FK_NAME;

b. FK_REFERENCE_VALUE;

c. FK_REFERENCE_KEY;

15. Formalismo, secondo la rivendicazione 5, in cui il documento formattato nella sottosezione INSTANCE_DML abbia almeno un pa-

rametro RECORD_DML comprendente almeno una volta i seguenti attributi:

a. ATTRIBUTE_DML;
b. FOREIGN_KEY_DML;

16. Formalismo, secondo la rivendicazione 5, in cui il documento formattato nella l'attributo ATTRIBUTE_DML abbia almeno un sotto-attributo AT_NAME;
17. Formalismo, secondo la rivendicazione 5, in cui la l'attributo FOREIGN_KEY_DML abbia almeno una volta i seguenti sotto-attributi:

a. FK_NAME;
b. AT_WHERE;
c. FK_REFERENCE_TABLE;
d. FK_REFERENCE_KEY;
e. REFERENCE_FIELD_DML.

18. Formalismo, secondo la rivendicazione 5, in cui la l'attributo REFERENCE_FIELD_DML abbia almeno una volta i seguenti attributi:

a. ATTRIBUTE_DML;
b. FOREIGN_KEY_DML;

19. Formalismo, secondo la rivendicazione 5, in cui la l'attributo ATTRIBUTE_DML abbia almeno un sotto-attributo AT_NAME.
20. Formalismo, secondo la rivendicazione 5, in cui la l'attributo FOREIGN_KEY_DML abbia i seguenti sotto-parametri:

a. FK_NAME;

b. AT_WHERE;

c. FK_REFERENCE_TABLE;

d. FK_REFERENCE_KEY;

e. REFERENCE_FIELD_DML.

21. Infrastruttura informatica, secondo la rivendicazione 1, comprendente almeno un computer su cui risieda il software traduttore/sender atto ad eseguire i passi a) e b) di detto metodo ed almeno un computer su cui risieda un software receiver/interprete/supervisore atto ad eseguirne i passi c) e d).

22. Programma di computer che comprende mezzi di codifica di programma atti a realizzare i passi della rivendicazione 1, quando detto programma è fatto girare su di un computer.

23. Mezzi leggibili da computer comprendenti un programma registrato, detti mezzi leggibili da computer comprendendo mezzi di codifica di programma atti a realizzare i passi della rivendicazione 1, quando detto programma è fatto girare su di un computer.

(OFM)

Roma, 27 Marzo 2007

OF/BOR

Per UNIVERSITA' DEGLI STUDI DI LECCE

Il Mandatario

Ing. Bruno CINQUANTINI

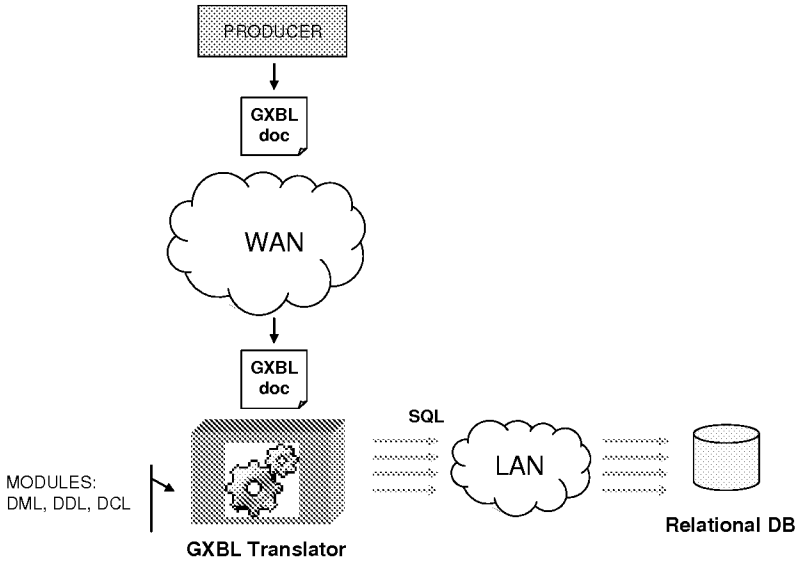
PRODUCER
GXBL doc
WAN
GXBL doc
SQL
LAN
MODULES: DML, DDL, DCL
GXBL Translator
Relational DB

Fig. 1

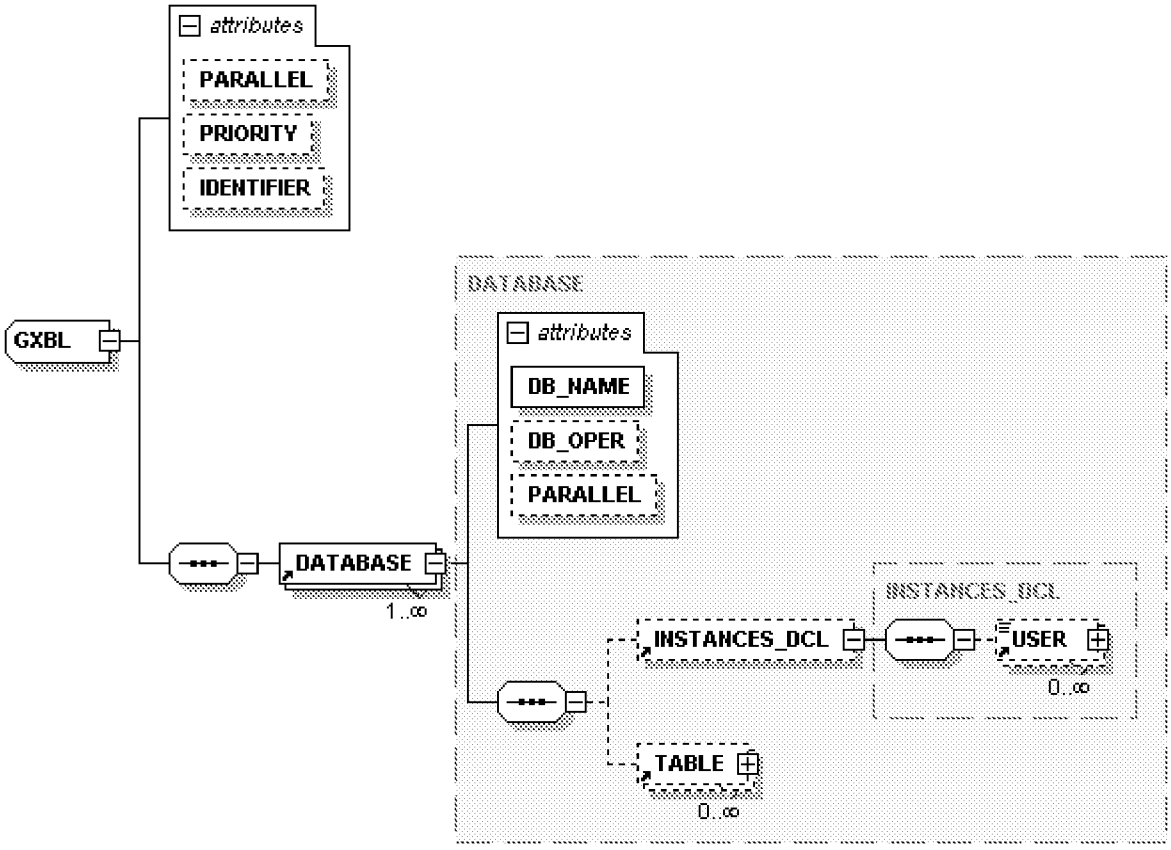
attributes
PARALLEL
PRIORITY
IDENTIFIER
GXBL
DATABASE
1..∞
DATABASE
attributes
DB_NAME
DB_OPER
PARALLEL
INSTANCES_DCL
INSTANCES_DCL
USER
0..∞
TABLE
0..∞

Fig. 2

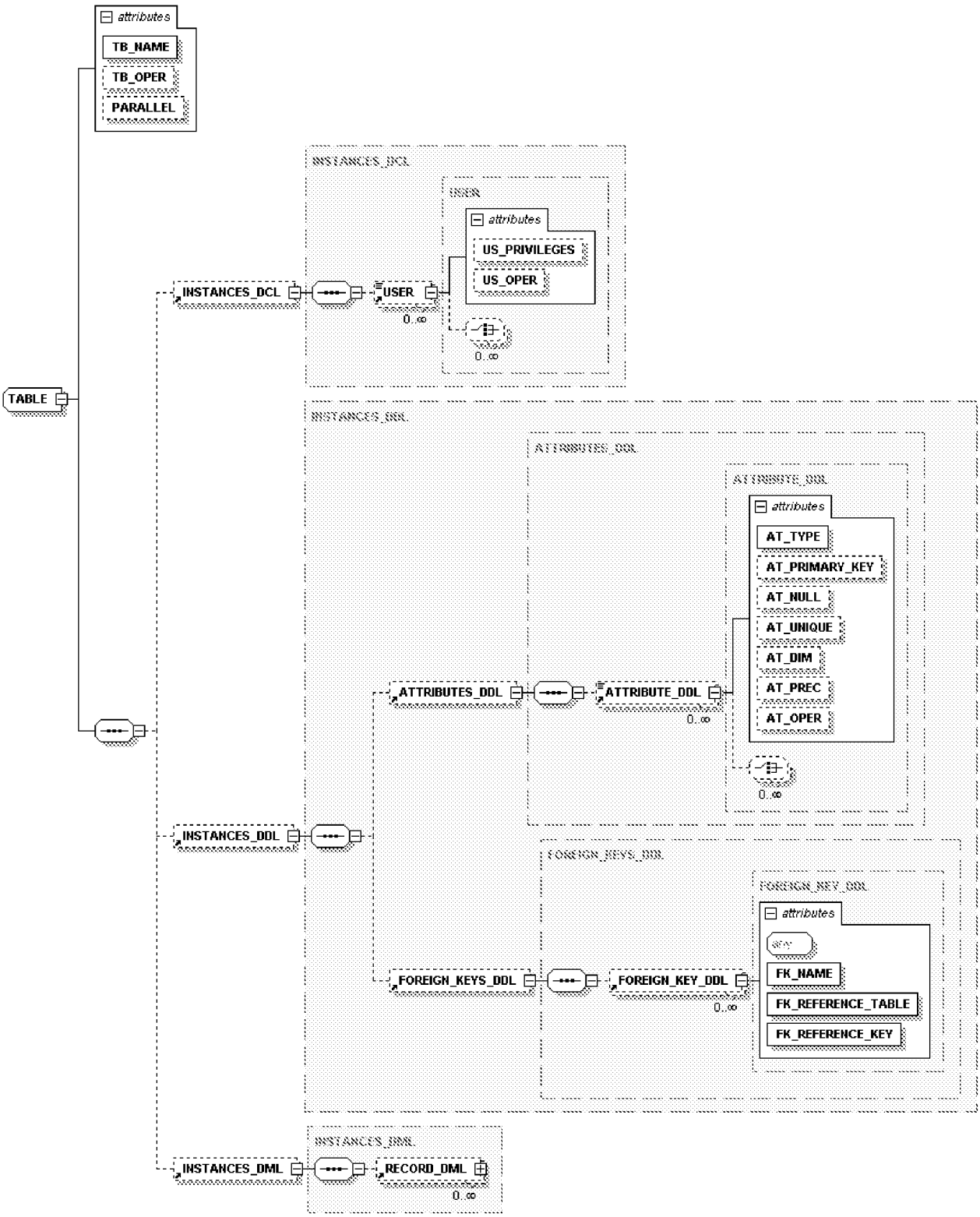
attributes
TB_NAME
TB_OPER
PARALLEL
INSTANCES_DCL
USER
attributes
US_PRIVILEGES
US_OPER
0..∞
0..∞
TABLE
INSTANCES_DDL
ATTRIBUTES_DDL
ATTRIBUTE_DDL
attributes
AT_TYPE
AT_PRIMARY_KEY
AT_NULL
AT_UNIQUE
AT_DIM
AT_PREC
AT_OPER
0..∞
0..∞
FOREIGN_KEYS_DDL
FOREIGN_KEY_DDL
attributes
FK_NAME
FK_REFERENCE_TABLE
FK_REFERENCE_KEY
0..∞
INSTANCES_DML
RECORD_DML
0..∞

Fig. 3

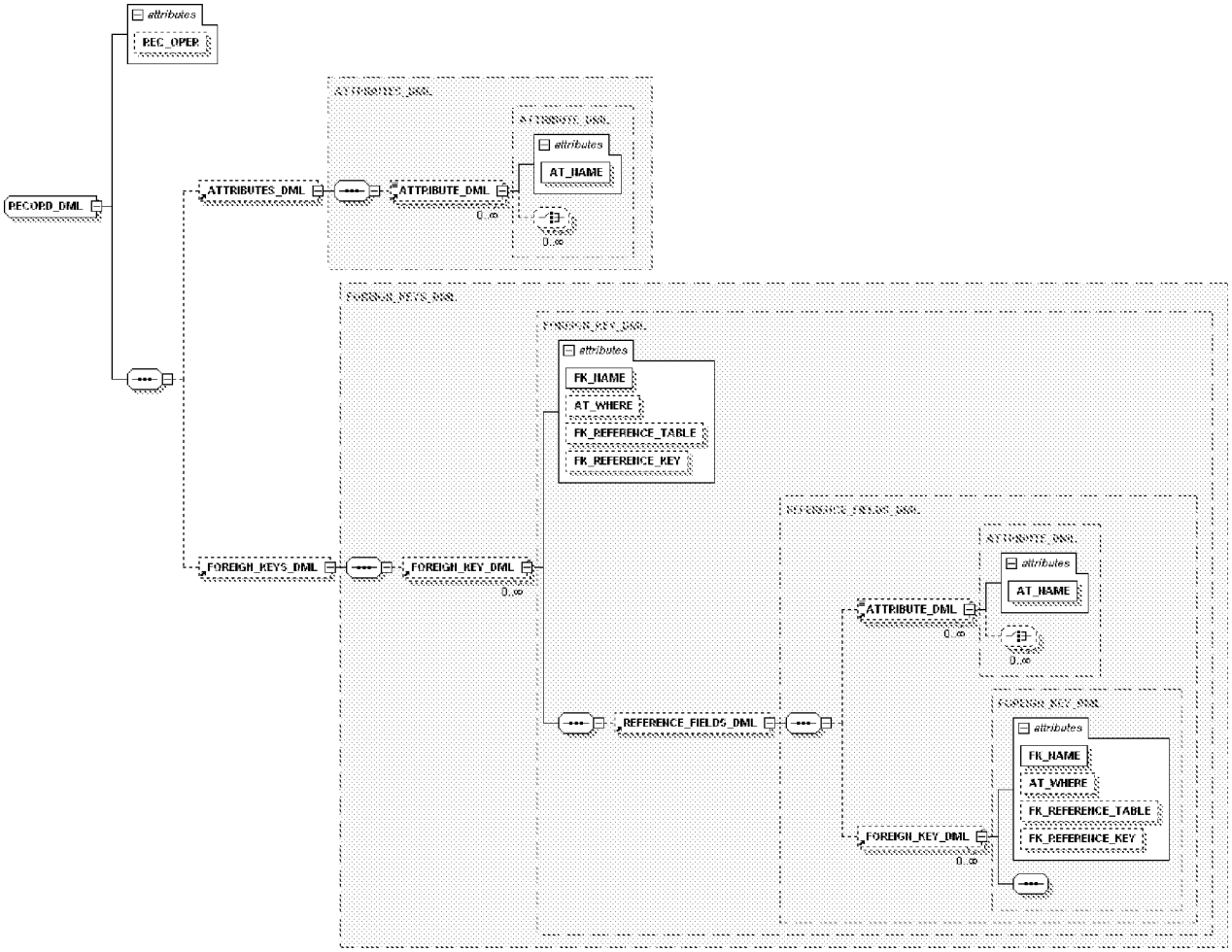
attributes
REC_OPER
RECORD_DML
ATTRIBUTES_DML
ATTRIBUTE_DML
attributes
AT_NAME
0..∞
FOREIGN_KEYS_DML
FOREIGN_KEY_DML
attributes
FK_NAME
AT_WHERE
FK_REFERENCE_TABLE
FK_REFERENCE_KEY
REFERENCE_FIELDS_DML
ATTRIBUTE_DML
attributes
AT_NAME
FOREIGN_KEY_DML
attributes
FK_NAME
AT_WHERE
FK_REFERENCE_TABLE
FK_REFERENCE_KEY

Fig. 4

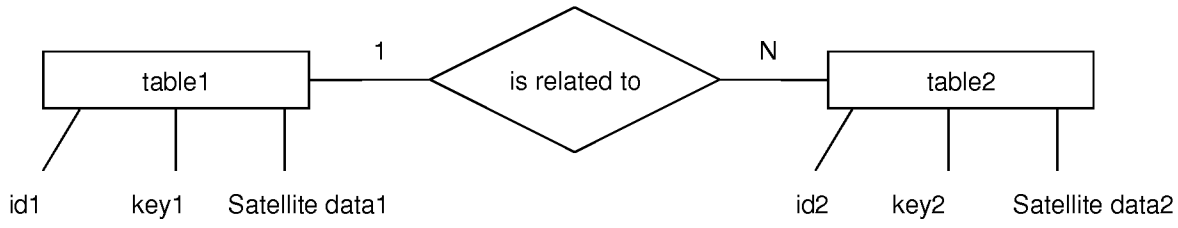
table1
1
is related to
N
table2
id1
key1
Satellite data1
id2
key2
Satellite data2

Fig. 5

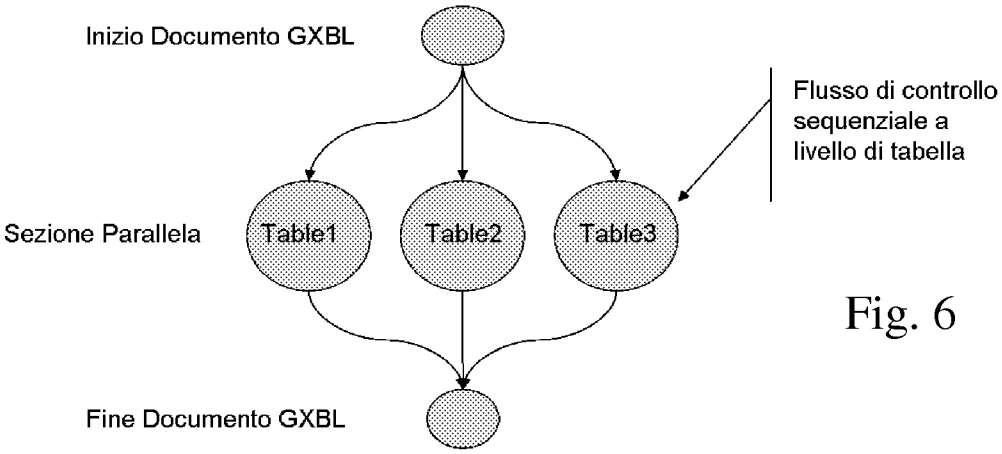

Fig. 6

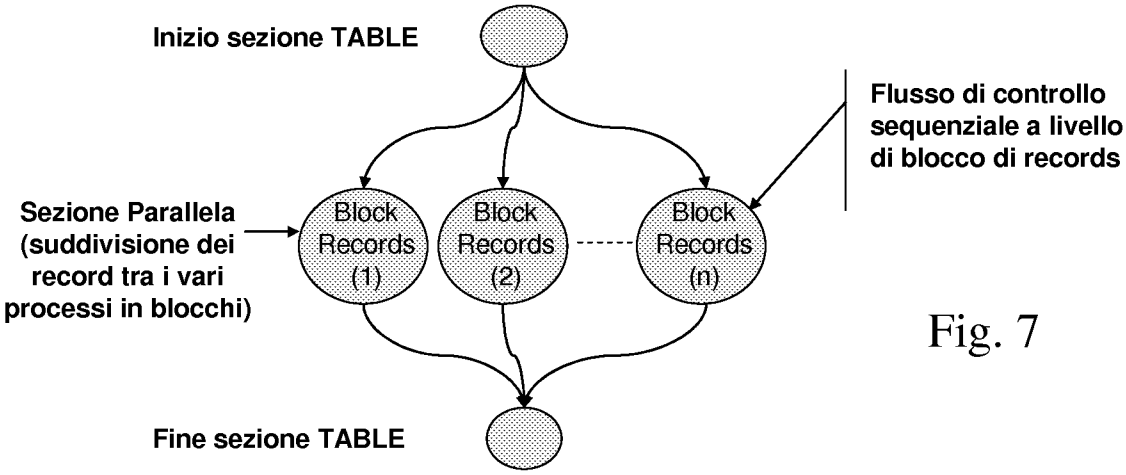

Fig. 7

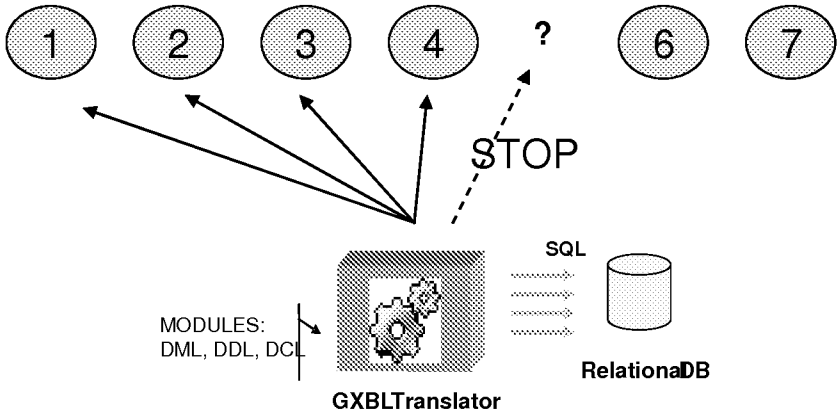

Fig. 8

Fig. 9.1

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT GXBL (DATABASE+)>
<!ELEMENT DATABASE (INSTANCES_DCL?, TABLE*)>
<!ELEMENT TABLE (INSTANCES_DCL?, INSTANCES_DDL?, INSTANCES_DML?)>
<!ELEMENT INSTANCES_DCL (USER*)>
<!ELEMENT INSTANCES_DDL (ATTRIBUTES_DDL?, FOREIGN_KEYS_DDL?)>
<!ELEMENT ATTRIBUTES_DDL (ATTRIBUTE_DDL*)>
<!ELEMENT FOREIGN_KEYS_DDL (FOREIGN_KEY_DDL*)>
<!ELEMENT INSTANCES_DML (RECORD_DML*)>
<!ELEMENT RECORD_DML (ATTRIBUTES_DML?, FOREIGN_KEYS_DML?)>
<!ELEMENT ATTRIBUTES_DML (ATTRIBUTE_DML*)>
<!ELEMENT FOREIGN_KEYS_DML (FOREIGN_KEY_DML*)>
<!ELEMENT FOREIGN_KEY_DML (REFERENCE_FIELDS_DML?)>
<!ELEMENT REFERENCE_FIELDS_DML (ATTRIBUTE_DML*, FOREIGN_KEY_DML*)>
<!ELEMENT ATTRIBUTE_DML (#PCDATA)>
<!ELEMENT ATTRIBUTE_DDL (#PCDATA)>
<!ELEMENT FOREIGN_KEY_DDL EMPTY>
<!ELEMENT USER (#PCDATA)>

<!ATTLIST DATABASE
        DB_NAME CDATA #REQUIRED
        DB_OPER (CREATE | UPDATE | DELETE) #IMPLIED
        PARALLEL (TRUE | FALSE) #IMPLIED
>

<!ATTLIST RECORD_DML
        REC_OPER (INSERT | DELETE | UPDATE | FORCED_UPDATE) #IMPLIED
>

<!ATTLIST FOREIGN_KEY_DDL
        FK_NAME CDATA #REQUIRED
        FK_REFERENCE_TABLE CDATA #REQUIRED
        FK_REFERENCE_KEY CDATA #REQUIRED
>

<!ATTLIST FOREIGN_KEY_DML
        FK_NAME CDATA #REQUIRED
        AT_WHERE (TRUE | FALSE) #IMPLIED
        FK_REFERENCE_TABLE CDATA #IMPLIED
        FK_REFERENCE_KEY CDATA #IMPLIED
>

<!ATTLIST TABLE
        TB_NAME CDATA #REQUIRED
        TB_OPER (CREATE | UPDATE | DELETE) #IMPLIED
        PARALLEL (TRUE | FALSE) #IMPLIED
>

<!ATTLIST USER
_PRIVILEGES (SELECT |  INSERT| DELETE| UPDATE| RULE| MNG_USER| MNG_DB| DB_ROOT)
#IMPLIED
_OPER (CREATE | UPDATE | DELETE) #IMPLIED
>

<!ATTLIST ATTRIBUTE_DML
        AT_NAME CDATA #REQUIRED
>
```

Fig. 9.2

```
<!ATTLIST GXBL
        PARALLEL (TRUE | FALSE) #IMPLIED
 PRIORITY CDATA #IMPLIED
 IDENTIFIER CDATA #IMPLIED
>

<!ATTLIST ATTRIBUTE_DDL
        AT_TYPE (SHORT_INT | INT | LONG_INT | CHAR | VARCHAR | FLOAT | DOUBLE | DATE |
TIME | TIMESTAMP) #REQUIRED
        AT_PRIMARY_KEY (TRUE | FALSE) #IMPLIED
        AT_NULL (TRUE | FALSE) #IMPLIED
        AT_UNIQUE (TRUE | FALSE) #IMPLIED
        AT_DIM CDATA #IMPLIED
        AT_PREC (SINGLE | DOUBLE) #IMPLIED
        AT_OPER (ADD | ALTER | DELETE) #IMPLIED>
```

Fig. 10.1

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
        <xs:complexType name="GXBL">
                <xs:sequence>
                        <xs:element ref="DATABASE" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="PARALLEL">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="FALSE"/>
                                        <xs:enumeration value="TRUE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="PRIORITY" type="xs:string"/>
                <xs:attribute name="IDENTIFIER" type="xs:string"/>
        </xs:complexType>
        <xs:element name="GXBL" type="GXBL"/>
        <xs:complexType name="DATABASE">
                <xs:sequence>
                        <xs:element ref="INSTANCES_DCL" minOccurs="0"/>
                        <xs:element ref="TABLE" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="DB_NAME" type="xs:string" use="required"/>
                <xs:attribute name="DB_OPER">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="DELETE"/>
                                        <xs:enumeration value="CREATE"/>
                                        <xs:enumeration value="UPDATE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="PARALLEL">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="FALSE"/>
                                        <xs:enumeration value="TRUE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:element name="DATABASE" type="DATABASE"/>
        <xs:complexType name="TABLE">
                <xs:sequence>
                        <xs:element ref="INSTANCES_DCL" minOccurs="0"/>
```

Fig. 10.2

```
                <xs:element ref="INSTANCES_DDL" minOccurs="0"/>
                <xs:element ref="INSTANCES_DML" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="TB_NAME" type="xs:string" use="required"/>
        <xs:attribute name="TB_OPER">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="DELETE"/>
                                <xs:enumeration value="CREATE"/>
                                <xs:enumeration value="UPDATE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="PARALLEL">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FALSE"/>
                                <xs:enumeration value="TRUE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
</xs:complexType>
<xs:element name="TABLE" type="TABLE"/>
<xs:complexType name="INSTANCES_DCL">
        <xs:sequence>
                <xs:element ref="USER" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="INSTANCES_DCL" type="INSTANCES_DCL"/>
<xs:complexType name="INSTANCES_DDL">
        <xs:sequence>
                <xs:element ref="ATTRIBUTES_DDL" minOccurs="0"/>
                <xs:element ref="FOREIGN_KEYS_DDL" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="INSTANCES_DDL" type="INSTANCES_DDL"/>
<xs:complexType name="ATTRIBUTES_DDL">
        <xs:sequence>
                <xs:element ref="ATTRIBUTE_DDL" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="ATTRIBUTES_DDL" type="ATTRIBUTES_DDL"/>
<xs:complexType name="FOREIGN_KEYS_DDL">
        <xs:sequence>
                <xs:element ref="FOREIGN_KEY_DDL" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="FOREIGN_KEYS_DDL" type="FOREIGN_KEYS_DDL"/>
<xs:complexType name="INSTANCES_DML">
        <xs:sequence>
                <xs:element ref="RECORD_DML" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="INSTANCES_DML" type="INSTANCES_DML"/>
<xs:complexType name="RECORD_DML">
        <xs:sequence>
                <xs:element ref="ATTRIBUTES_DML" minOccurs="0"/>
                <xs:element ref="FOREIGN_KEYS_DML" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="REC_OPER">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FORCED_UPDATE"/>
                                <xs:enumeration value="DELETE"/>
                                <xs:enumeration value="INSERT"/>
                                <xs:enumeration value="UPDATE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
</xs:complexType>
<xs:element name="RECORD_DML" type="RECORD_DML"/>
<xs:complexType name="ATTRIBUTES_DML">
        <xs:sequence>
                <xs:element ref="ATTRIBUTE_DML" minOccurs="0" maxOccurs="unbounded"/>
```

Fig. 10.3

```
        </xs:sequence>
</xs:complexType>
<xs:element name="ATTRIBUTES_DML" type="ATTRIBUTES_DML"/>
<xs:complexType name="FOREIGN_KEYS_DML">
        <xs:sequence>
                <xs:element ref="FOREIGN_KEY_DML" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="FOREIGN_KEYS_DML" type="FOREIGN_KEYS_DML"/>
<xs:complexType name="FOREIGN_KEY_DML">
        <xs:sequence>
                <xs:element ref="REFERENCE_FIELDS_DML" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="FK_NAME" type="xs:string" use="required"/>
        <xs:attribute name="AT_WHERE">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FALSE"/>
                                <xs:enumeration value="TRUE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="FK_REFERENCE_TABLE" type="xs:string"/>
        <xs:attribute name="FK_REFERENCE_KEY" type="xs:string"/>
</xs:complexType>
<xs:element name="FOREIGN_KEY_DML" type="FOREIGN_KEY_DML"/>
<xs:complexType name="REFERENCE_FIELDS_DML">
        <xs:sequence>
                <xs:element ref="ATTRIBUTE_DML" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="FOREIGN_KEY_DML" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="REFERENCE_FIELDS_DML" type="REFERENCE_FIELDS_DML"/>
<xs:complexType name="ATTRIBUTE_DML" mixed="true">
        <xs:choice minOccurs="0" maxOccurs="unbounded"/>
        <xs:attribute name="AT_NAME" type="xs:string" use="required"/>
</xs:complexType>
<xs:element name="ATTRIBUTE_DML" type="ATTRIBUTE_DML"/>
<xs:complexType name="ATTRIBUTE_DDL" mixed="true">
        <xs:choice minOccurs="0" maxOccurs="unbounded"/>
        <xs:attribute name="AT_TYPE" use="required">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FLOAT"/>
                                <xs:enumeration value="VARCHAR"/>
                                <xs:enumeration value="DOUBLE"/>
                                <xs:enumeration value="CHAR"/>
                                <xs:enumeration value="TIME"/>
                                <xs:enumeration value="SHORT_INT"/>
                                <xs:enumeration value="LONG_INT"/>
                                <xs:enumeration value="DATE"/>
                                <xs:enumeration value="INT"/>
                                <xs:enumeration value="TIMESTAMP"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="AT_PRIMARY_KEY">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FALSE"/>
                                <xs:enumeration value="TRUE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="AT_NULL">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="FALSE"/>
                                <xs:enumeration value="TRUE"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="AT_UNIQUE">
                <xs:simpleType>
```

Fig. 10.4

```
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="FALSE"/>
                                        <xs:enumeration value="TRUE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="AT_DIM" type="xs:string"/>
                <xs:attribute name="AT_PREC">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="DOUBLE"/>
                                        <xs:enumeration value="SINGLE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="AT_OPER">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="ALTER"/>
                                        <xs:enumeration value="ADD"/>
                                        <xs:enumeration value="DELETE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:element name="ATTRIBUTE_DDL" type="ATTRIBUTE_DDL"/>
        <xs:complexType name="FOREIGN_KEY_DDL">
                <xs:complexContent>
                        <xs:restriction base="xs:anyType">
                                <xs:attribute name="FK_NAME" type="xs:string" use="required"/>
                                <xs:attribute name="FK_REFERENCE_TABLE" type="xs:string" use="required"/>
                                <xs:attribute name="FK_REFERENCE_KEY" type="xs:string" use="required"/>
                        </xs:restriction>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="FOREIGN_KEY_DDL" type="FOREIGN_KEY_DDL"/>
        <xs:complexType name="USER" mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded"/>
                <xs:attribute name="US_PRIVILEGES">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="DB_ROOT"/>
                                        <xs:enumeration value="DELETE"/>
                                        <xs:enumeration value="SELECT"/>
                                        <xs:enumeration value="MNG_USER"/>
                                        <xs:enumeration value="MNG_DB"/>
                                        <xs:enumeration value="RULE"/>
                                        <xs:enumeration value="INSERT"/>
                                        <xs:enumeration value="UPDATE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="US_OPER">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="DELETE"/>
                                        <xs:enumeration value="CREATE"/>
                                        <xs:enumeration value="UPDATE"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:element name="USER" type="USER"/>
</xs:schema>
```

Fig. 11

```
<?xml version="1.0" encoding="UTF-8"?>
<GXBL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="gxbl.xsd">
        <DATABASE DB_NAME="DBEsempio" DB_OPER="UPDATE">
                <TABLE TB_OPER="UPDATE" TB_NAME="table2">
                        <INSTANCES_DML>
                                <RECORD_DML REC_OPER="INSERT">
                                        <ATTRIBUTES_DML>
                                                <ATTRIBUTE_DML
AT_NAME="key2">key2value</ATTRIBUTE_DML>
                                                <ATTRIBUTE_DML
AT_NAME="satellitedata2">satdata2</ATTRIBUTE_DML>
                                        </ATTRIBUTES_DML>
                                        <FOREIGN_KEYS_DML>
                                                <FOREIGN_KEY_DML FK_REFERENCE_KEY="id1"
FK_REFERENCE_TABLE="table1" FK_NAME="idref1">
                                                        <REFERENCE_FIELDS_DML>
                                                                <ATTRIBUTE_DML
AT_NAME="key1">key1value</ATTRIBUTE_DML>
                                                        </REFERENCE_FIELDS_DML>
                                                </FOREIGN_KEY_DML>
                                        </FOREIGN_KEYS_DML>
                                </RECORD_DML>
                        </INSTANCES_DML>
                </TABLE>
        </DATABASE>
</GXBL>
```