Towards a New Incremental Implementation Approach for Distributed Databases

Bouraoui Marwa
Université Tunis El Manar
SITI, ENIT
Tunisia
e-mail: bourawimarwa@gmail.com

Hassen Fadoua
Université Tunis El Manar
LIPAH, FST
Tunisia
e-mail: hassen.fadoua@gmail.com

Grissa touzi Amel
Université Tunis El Manar
ENIT, LIPAH, FST
Tunisia
e-mail: amel.touzi@enit.rnu.tn

Abstract—In this paper, we propose a new incremental implementation approach of a Distributed Database (DDB). On one hand, the frequent need to add and/or delete number of sites in the geographical distribution of a DDB has become a requirement of the user. On the other hand, we find that the already existing Distributed Database Management System (DDBMS) does not even offer an automatic implementation for a Distributed Database, which has been initially allocated to a predefined number of sites. In this approach, we propose a weak coupling with any existing DDBMS. This consists in garnishing all DDBMS with an intelligent layer that offers: 1) a convivial interface for an incremental definition of the different sites in the DDB, 2) an incremental design approach to DDB while knowing fragmentation attributes and 3) an automatic update of the scripts in the different sites. To validate our approach, we have used Oracle as an example of DDBMS.

Keywords-Distributed Database; Fragmentation; Allocation; Integrity Constraint; Distributed Updates.

I. INTRODUCTION

The design and implementation of a Distributed Database (DDB) has always been a challenge for the designers of this type of Database especially with: 1) the size of the original model, 2) the frequent need to add and/or suppress sites in the geographical distribution of DDB and 3) the limits of existing Distributed Database Management System (DDBMS).

Several studies have been conducted in this context. As examples, we can cite the work of Abdalla [1] and Moussa [2] who have proposed a support system for the design of DDB. We can also mention the work of Hassen and Grissa [3], who has proposed a new aid approach for the DDB implementation. This approach has been validated by the design and implementation of a tool that provides a graphical interface which guides the user through the DB fragmentation and validates his choices. The final product was a set of Structured Query Language (SQL) scripts automatically generated for each site from the initial configuration.

Unfortunately, these proposals remain static and do not take into account the evolution of the number of sites that occurs throughout the Database (DB) lifecycle.

In this paper, we propose a new incremental implementation approach of DDB taking into account the evolution of the number of the DB sites. This approach should allow 1) an incremental design of the DB taking into consideration the addition and/or deletion of a site, the fragmentation concepts and the duplication of data, 2) the updating of the various links between the sites, and 3) an automatic generation of DDL script for each site as well as Procedural Language (PL) / SQL procedures and the necessary triggers for the update and the verification of the DDB integrity constraints. This approach has been validated by an implementation of an intelligent layer under the Oracle DDBMS.

This paper is organized as follows: Section 2 presents some basic concepts of DDB. Section 3 presents an example of a DDB implementation, thus illustrating the problems and limitations of already existing DDBMS. Section 4 presents our motivation for this work. Section 5 presents a description of our proposed approach. Section 6 presents our implemented tool, Intelligent-Incremental-DDB. We end with a conclusion and some perspectives.

II. BASIC CONCEPTS

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network [4].

A Distributed Database Management System (DDBMS) is a software system that manages a set of databases which are physically distributed but logically connected and which provides the necessary means of access to ensure a transparent distribution [5].

In particular, a DDBMS must ensure a continuous functioning. Indeed, the need for a planned system shutdown should never be felt, even for some operations such as site adding or site deleting, or else the dynamic creation or deletion of fragments in one or more sites.

We cite as examples: Oracle 11g [6], Cassandra [7], Informix [8], INGRES [9], among the top DDBMS ranked in the market.

Distributed database design must take into account the number of sites on the distribution. It is based especially on the following concepts [10]:

- **Fragmentation:** Fragmentation is the process of the decomposition of a database into a set of sub databases. This decomposition should be with no loss of information [11]. We distinguish three types of fragmentation: 1) *horizontal fragmentation*: It consists of dividing the relations into sub relations obtained through the selection of tuples in a table according to a specific criterion. The reconstruction of the relations is defined by the union of the fragments. 2) *Vertical Fragmentation*: Each fragment represents a subset of relationship attributes. The primary key must be maintained in each fragment. The reconstruction of the relations is defined by join. 3) *Mixed fragmentation*: It results from the successive application of horizontal and vertical fragmentation operations on a global relation.

- **Replication:** the replication of a database is the reproduction of a subset of the main database on remote sites.
- **Data allocation:** is the allocation of fragments to different sites depending on the origin of the queries that have been used during the fragmentation process.

III. EXAMPLE OF A DDB IMPLEMENTATION

In this section, our goal is to describe, through an example, the necessary process to implement a DDB on a number of initial sites, then show the necessary changes that have to be made by the DDB designer following the addition of a new site.

Consider the database described by the following global schema:

Client (ID-cl, Name, Address, City, Business_Sales, Rate_Reduction)
Command (Num-c, Date_c, # ID-cl, Delivery)

We propose first to distribute this DB on two sites: Tunis and Sousse. This distribution is shown in Figure 1.

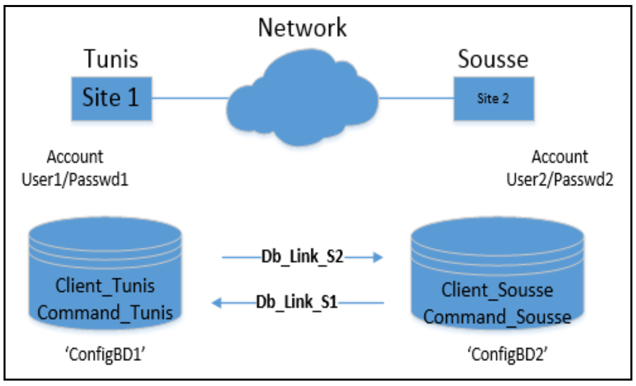


Figure 1. DB spread over two sites

Then, following the user needs, we propose to add a third site: Sfax, as it is illustrated in Figure 2.

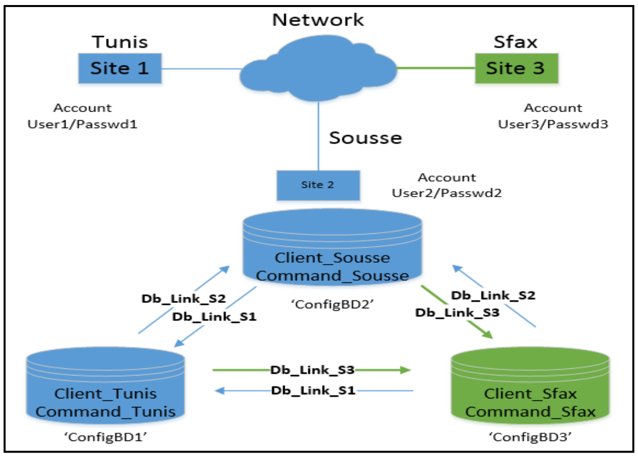


Figure 2. DB spread over three sites

In this section, we propose to use Oracle 11g as DDBMS to implement our DDB.

A. *Allocation mechanism in Oracle*

Like any commercial DDBMS, Oracle does not accept the distributed allocation mechanism, although the administrator can manually allocate DB data to produce similar results. This has the effect of shifting the responsibility under the auspices of the end user, who must know that a table has been fragmented and that he can convert this knowledge into the application. In other words, the Oracle DDBMS does not ensure transparency of the distribution, while it allows location transparency [8].

In order to ensure transparency, the designer must stick to the following steps:

- User accounts creation among sites.
- Bi-directional links creation between different sites (using oracle CREATE DBLINK command).
- Local schema implementation on each site
- Synonyms definition to ensure location transparency.
- View and/or materialized views creation and/or snapshots to ensure fragmentation independent schema. On each materialized view and snapshot definition, we have to specify update mode (asynchronous, synchronous) and refresh delay in accordance with the application need .
- Stored procedure definition, as PL/SQL script, on each update operation in a way to make data fragmentation and duplication automated and transparent.
- As a DBMS can ensure only local data integrity, the designer must define PL/SQL triggers that allow checking distributed data integrity among DDB.

B. *DB implementation at two sites*

The implementation of this database will be as follows (Figure 3):

<u>For site1 :Tunis</u>

DDL_Site1.sql

```
---------------DDL FOR DATABASE LINK ------------
----------------------------------------------------------------
create public database link DB_Link_S2
connect to user2 identified by passwd2
using 'ConfigBD2';
------------------- DDL FOR TABLES  -------------------
----------------------------------------------------------------
create table Client_Tunis  ( ID-cl  number (4), Name
varchar2(10), First_Name  varchar2(10), Adress
varchar2(20), City  varchar2(10), Business_Sales
number (10,3), Rate_Reduction number (4,2),
Constraint PK11 primary key (ID-cl) );
---------------------------
create table Command_Tunis  ( Num_c  number (4),
Date_c  Date, ID_cl  number (4), Delivery char chek
(Delivery in (‘O’,’N’) ,constraint PK12 primary key
(Num_c), constraint FK1 foreign key (ID_cl)
references Client_Tunis(ID-cl));
-------------------- DDL FOR SYNONYMS --------------
----------------------------------------------------------------
create public synonym Client_Sousse  for
Client_Sousse@DB_Link_S2;
---------------------------
create public synonym Command_Sousse  for
Command_Sousse@DB_Link_S2;
---------------- DDL FOR VIEW --------------------------
----------------------------------------------------------------
create view Client
  as
 (select * from Client_Tunis)
 union
 (select * from Client_Sousse);
-------------------------
create view Command
  as
 (select * from Command_Tunis)
 union
 (select * from  Command_Sousse);
-------DDL FOR INSERT_CLIENT PROCEDURE  ---
----------------------------------------------------------------
Create or replace Procedure insert_client(idc number,
namec  varchar2, fnamec  varchar2, addressc varchar2,
cityc varchar2, bsc number, rrc number)
is
begin
   if (cityc=‘TUNIS’) then
         insert into Client_Tunis values(idc, namec,
fnamec  , addressc, cityc, bsc, rrc);
    elsif (cityc=‘SOUSSE’) then
         insert into Client_Sousse values(idc, namec,
fnamec  , addressc, cityc, bsc, rrc);
   else  DBMS_OUTPUT.put_line(‘The  client  city
must be either Tunis or Sousse’) ;
end if ;
commit ;
end ;
```

Figure 3. Part of DDL_Site1.sql

<u>For the site 2: Sousse:</u> we follow the same principle of Site1

C. *DB implementation at three sites*

Now, suppose that we add a new site 'Sfax'. Such a change will result in various modifications in the generated scripts.

To involve the new site added, several changes will be applied in different sites. For example, one of these will be the updating of the site 1 Data Definition Language (DDL) script, which includes changes at three levels (Figure 4) :

- Views
- Synonyms
- The PL / SQL procedures

```
create view Client
as
    (select * from Client_Tunis)
  union
 (select * from Client_Sousse)
 union
  (select * from Client_sfax);
----------------------------------------------------------------
create public synonym Client_Sousse  for
Client_Sousse@DB_Link_S2;
--------------------
create public synonym Client_Sfax  for
Client_Sfax@DB_Link_S3;
--------------------
create public synonym Command_Sousse  for
Command_Sousse@DB_Link_S2;
--------------------
create public synonym Command_Sfax  for
Command_Sfax@DB_Link_S3;
 ----------------------------------------------------------------
Create or replace Procedure insert_client(idc number,
namec  varchar2,  fnamec  varchar2, addressc varchar2,
cityc varchar2, bsc number, rrc number)
is
begin
   if (cityc=‘TUNIS’) then
      insert into Client_Tunis values(idc, namec,
fnamec  ,  addressc, cityc, bsc, rrc);
     elsif (cityc=‘SOUSSE’) then
      insert into Client_Sousse values(idc, namec,
fnamec  , addressc, cityc, bsc, rrc);
   elsif (cityc=‘SFAX’) then
      insert into Client_Sfax values(idc, namec,
fnamec  , addressc, cityc, bsc, rrc);
   else  DBMS_OUTPUT.put_line(‘The  client  city
must be either Tunis or Sousse or Sfax’) ;
end if ;
commit ;
end ;
```

Figure 4. Part of DDL_Site1.sql automatically updated after adding a new site

IV. MOTIVATION

As shown previously, designers are still facing issues when dealing with Distributed DBs:

- The process of implementing a DDB is still a quite tedious task and time consuming even for a fixed number of sites. The variety of scripts to generate, the size of the database, and the number of sites are factors that can rise the complexity of the exercise.

- Incremental implementation of a DDB with a variable number of sites (addition or deletion) is very delicate and error-prone mission. In fact, multiple updates must be applied in order to ensure data coherence, fragment validation and other critical constraints. In addition, such a modification will affect the initial design every time a site is added or removed.

Several studies have been conducted in this context. As examples, we can cite the work of Abdalla [1] and Moussa [2] who have proposed a support system for the design of DDB. We can also mention the work of Hassen and grissa [3], who has proposed a new aid approach for the DDB implementation. This approach has been validated by the design and implementation of a tool that provides a graphical interface which guides the user through the DB fragmentation and validates his choices.

Unfortunately, these contributions still static; they are designed for a fixed number of sites, so they do not offer a solution that supports dynamic design while adding, updating or deleting sites. They are also limited to the design and implementation of a DDB but do not offer a solution for updating this DDB through adding or deleting fragments.

We can mention here the work of Hsu [12], who explain the concepts of the Rensselaer Polytechnic Institute Metadatabase System while discussing a single approach to the integration problem. This contribution answers one part of the matter but it is limited for the Metadatabases. In addition, it is an integration approach of heterogeneous data application while we are interested in distribution approaches for DDBs.

What characterizes our work also is the support for fragmentation aspect, where each fragment is hosted in a remote DB. Such a feature introduces more complexity to the procedure. Not only it supports variable number of sites, but also, sychronizes scattered fragments in remote sites which are bound by some integrity rules.

In the following, we propose a new architecture of the DDBMS that supports an incremental implementation. This consists in garnishing all DDBMS with an intelligent layer that offers: 1) a graphical interface for an incremental definition of the different sites in the DDB, 2) an incremental design approach to DDB while knowing fragmentation attributes and 3) an automatic update of the scripts in the different sites. To validate our approach, we have used as an example of DDBMS, the Oracle DDBMS.

V. New Approach proposals for Oracle DDBMS

A. *The Approach Specification*

To validate our approach, the new layer must ensure the followings:

- Verification of distributed data integrity: for the centralized DB, the existing DBMS validate the integrity constraint verification, but the problem occurs in case they check them for the distributed DB.
- Dynamic design of the database: adding / removing a site entails the integration / suppression of a local schema, which leads to the modification of the global schema.
- Updating scripts of:
  - Views
  - Synonyms
  - Stored procedures
  - Triggers
  - Allocation of data in remote databases

B. *Suggested layer architecture*

The architecture of our application is illustrated in Figure 5. The graphic interface provides an easy method to interact with the users who, in turn, may interact easily with different modules:

- A validation module, which informs the user if his operation is true or false, is implemented to help the less experienced user not to make mistakes when handling complex schema. In reality, this layer is implemented on two levels:
  - **Validate fragmentation:** check if the fragmentation process respects the three criteria: Reconstruction, Completeness and Disjointness.
  - **Validate data integrity:** check distributed data integrity across the remote databases.
- The site management module includes: update bases schemas, update scripts of (views, synonyms, triggers, data allocation), update database links.
- The script generation module allows the user to consult the SQL script of all transactions that took place during the database implementation process.

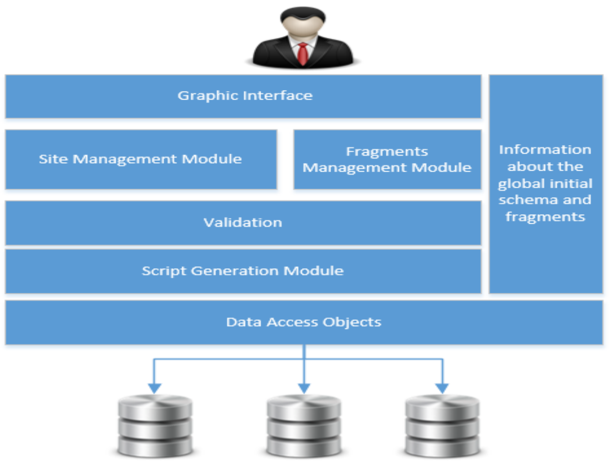


Figure 5. Layer architecture

C. Incremental implementation procedure

In the following (Figure 6), we present the procedure we have implemented for the addition of a new site:

<u>Adding site principle</u>

```
BEGIN
{
s = The site that we wish to add;
F = {f1, f2,.., fn} list of fragments //fi may be a horizontal, vertical, hybrid or dulicate fragment in a remote specific site;
1.  Enter the list of configuration parameters of this site s (database link, site name, IP address ,remote database SID, login, password);
2.  Generate scripts for creation of database links according to s;
3.  Attribute fragments to the site s;
3.1  Retrieve the list of relationships based on initial schema ;
3.2  WHILE " Complete Fragmentation " is false
{
FOR each table from the centralized database
{
// Choose the type of fragmentation
IF horizontal fragmentation
THEN {
Select column fragmentation;
Affect the value of fragmentation ; }
ELSE IF vertical fragmentation
THEN  Selected the columns of the fragment ;
ELSE IF hybrid Fragmentation
THEN Treat the hybrid fragment;
ELSE IF duplication
THEN Duplicate the table;
 Validate the fragmentation;
 Display the validation report;
IF validation is negative
THEN break;
ELSE
//Generate scripts fragments
generate scripts for creation of fragments;
generate scripts of data allocation;
}
}
4.   Run the database link scripts according to s;
5.   Generate the relational schema of this site s;
6.   Regenerate the new global schema;
7.   Add the new fragment to the list of fragments;
8.   Add this site s to the list of sites;
9.   Check distributed data integrity
9.1.  Collect some necessary information: Site list, fragment list, etc;
9.2.   Generate triggers scripts ;
9.3    Run the triggers scripts;
10.  Generate CRUD procedures;
11.  Generate views;
12.  FOR each fragment fi according to s
{
   execute script of fragment creation;
 // execute script of allocation
{  allocate data to the fragments of s;
   delete the allocated data from the centralized DB; }
}
}
END
```

Figure 6. Adding site principle

Considering the case of a site removing, it requires to delete its fragments, its database links and to update global conceptual schema and triggers. In the following (Figure 7), we present the procedure we have implemented for this operation.

<u>Deleting site principle</u>

```
BEGIN
{
s = The site that we wish to delete;
F = {f1, f2,.., fn} list of fragments; //fi may be an horizontal, vertical, hybride or dulicate fragment in a remote specific site
1. FOR each fragment(fi) in the fragments lists(F)
{
IF fi belongs to the current site s
{
a. Delete fi's data
{
        IF fi does not have a primary Key
        THEN delete fi's data;
        IF fi has a primary key
        THEN activate a cascade suppression to delete fi's data(recursive procedures);
}
b. Delete fi from the fragments lists F;
}
}
2. Delete database links which refer to s;
3. Delete s from to the sites lists;
4. Generate the new global schema;
5. Check distributed data integrity
5.1. Collect information to generate scripts of distributed triggers(database link, site name, fragments lists,..);
5.2. Generate updating triggers scripts;
5.3. Execute triggers scripts;
6. Generate new scripts of views end synonyms ;
}
END
```

Figure 7. Deleting site principle

D. Performance analysis

Among the most important performance criterion that users seek today, is the response time. We are primarily interested in our application to the incremental aspect of to the number of sites, so we have analyzed the response time required for the generation of scripts when creating / deleting sites.

- First Scenario: Adding Site
- Second Scenario: Removing site

We varied the number of sites for each scenario from 1 to 60 sites. The results are described through the following figure (Figure 8):

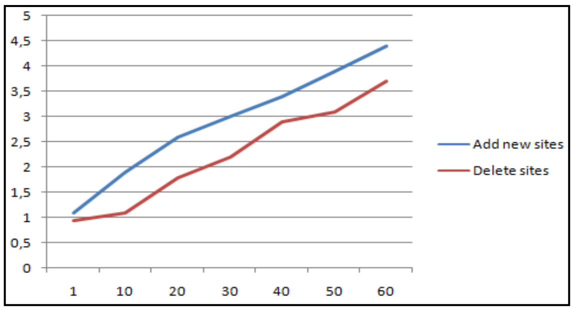


Figure 8. Performance analysis

The results presented in the Figure 8 shows that adding a site consumes more time than deleting a site. This is justified by, while adding a new site, we have more scripts generated for verifying data integrity of new integrated fragments. For each site, our application handles of updating scripts of: views, synonyms, stored procedures, triggers and allocation of data in remote databases. A cascade delete implemented by our solution can also speed up the removal process.

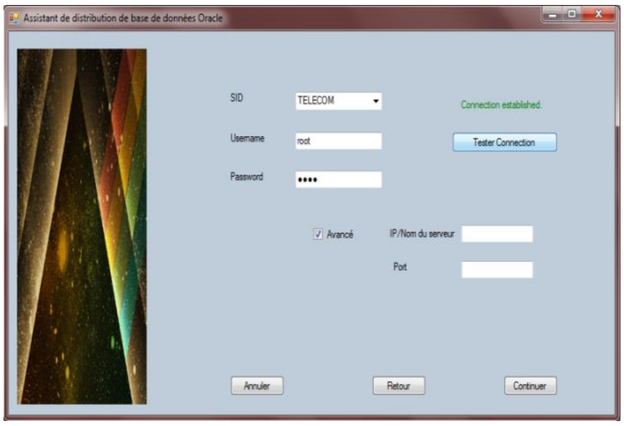
VI. INTELLIGENT-INCREMENTAL-DDB

In this section, we propose to validate our approach. For this we propose a weak coupling with Oracle as an example of DDBMS.

Thus, we used a Windows7 operating system. For the development environment, we have worked with DotNet framework 4.0 (CSharp). We installed virtual machines (Oracle Virtual Box) for the remote databases.

We detail our application operation with the most important interfaces:

Once authenticated, the user is asked to fill in the required coordinates to connect to the centralized DB. If the database is in a remote server, the user must switch to advanced mode to indicate the IP address and the port server (Figure 9).



SID TELECOM

Username root

Password ••••

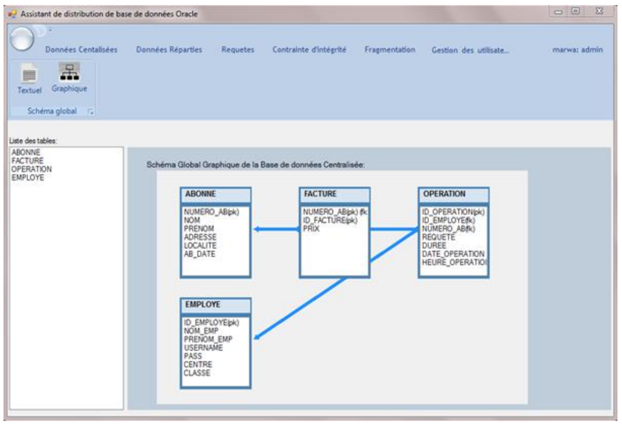
Connection established

[✓ Avancé] IP/Nom du serveur

Port

Figure 9. Connection to the centralized database

After a successful connection, the user can benefit from the functionality of the application. He can also refer to the overall relational schema of the centralized database as well as the local schemas of the remote databases (Figure 10).



Liste des tables:
ABONNE
FACTURE
OPERATION
EMPLOYE

Schéma Global Graphique de la Base de données Centralisée:

ABONNE: NUMERO_AB(pk), NOM, PRENOM, ADRESSE, LOCALITE, AB_DATE

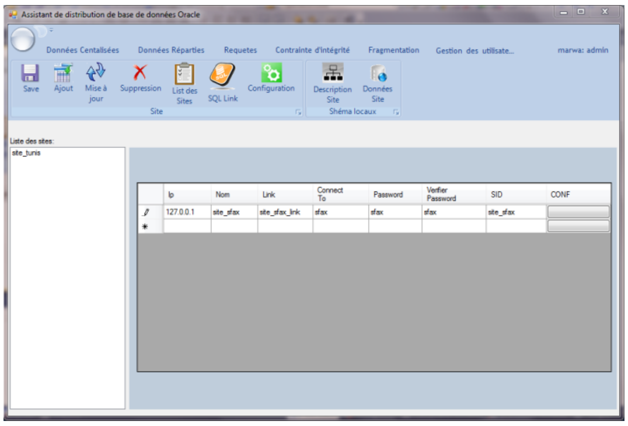
FACTURE: NUMERO_AB(pk) (fk, ID_FACTURE(pk), PRIX

OPERATION: ID_OPERATION(pk), ID_EMPLOYE(fk), NUMERO_AB(fk), REQUETE, DUREE, DATE_OPERATION, HEURE_OPERATIO

EMPLOYE: ID_EMPLOYE(pk), NOM_EMP, PRENOM_EMP, USERNAME, PASS, CENTRE, CLASSE

Figure 10. Consult relational schema

Figure 11 shows the site management interface. As an example, we present below the interface of adding a new site. Firstly, the user configures settings, which allow access to remote sites, by indicating (IP address, database link, login, password, SID).

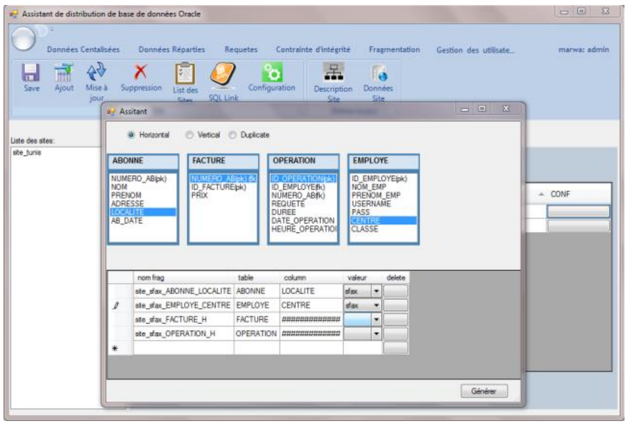


Liste des sites:
site_tunis

	Ip	Nom	Link	Connect To	Password	Verfier Password	SID	CONF
	127.0.0.1	site_sfax	site_sfax_link	sfax	sfax	sfax	site_sfax	
*								

Figure 11. Configuration of parameters access to the remote site

Secondly, the user may add fragments from the centralized tables to the current site. He can also perform a horizontal fragmentation, a vertical fragmentation, or even duplicate an existing table. Figure 12 shows an example of such an operation.



Liste des sites:
site_tunis

[✓ Horizontal] Vertical Duplicate

ABONNE: NUMERO_AB(pk), NOM, PRENOM, ADRESSE, LOCALITE, AB_DATE

FACTURE: NUMERO_AB(pk) (fk, ID_FACTURE(pk), PRIX

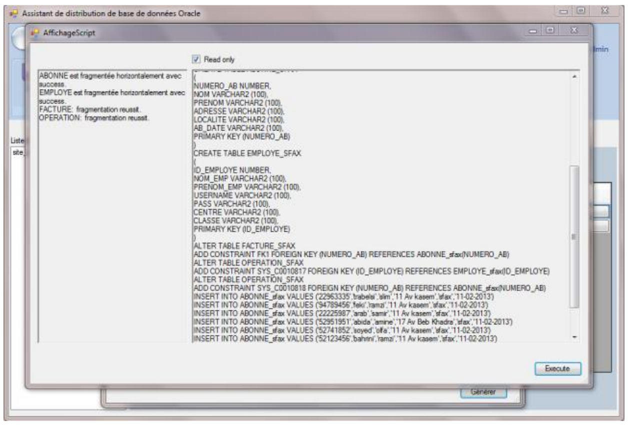
OPERATION: ID_OPERATION(pk), ID_EMPLOYE(fk), NUMERO_AB(fk), REQUETE, DUREE, DATE_OPERATION, HEURE_OPERATIO

EMPLOYE: ID_EMPLOYE(pk), NOM_EMP, PRENOM_EMP, USERNAME, PASS, CENTRE, CLASSE

	nom frag	table	column	valeur	delete
	site_sfax_ABONNE_LOCALITE	ABONNE	LOCALITE	sfax	
	site_sfax_EMPLOYE_CENTRE	EMPLOYE	CENTRE	sfax	
	site_sfax_FACTURE_H	FACTURE	##############		
	site_sfax_OPERATION_H	OPERATION	##############		
*					

Figure 12. Fragment allocation to the new site

By clicking on the "Generate" button, the script of this operation is displayed; it consists of : a validation report of fragmentation, creation script fragments and the data allocation script. For a valid fragmentation, the user can run the script (Figure 13).



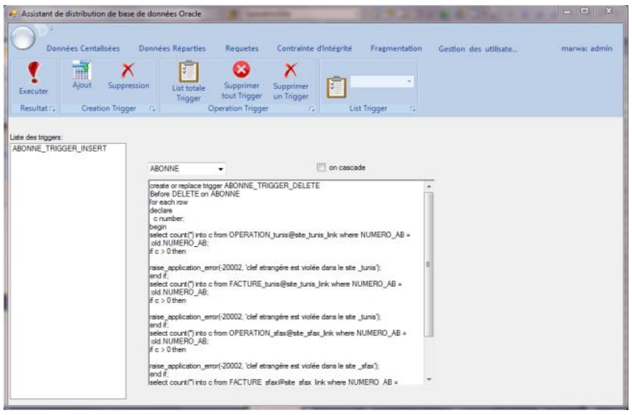
```
ABONNE est fragmentée horizontalement avec
success.
EMPLOYE est fragmentée horizontalement avec
success.
FACTURE:  fragmentation reussit.
OPERATION:  fragmentation reussit.
```

```
(
NUMERO_AB NUMBER,
NOM VARCHAR2 (100),
PRENOM VARCHAR2 (100),
ADRESSE VARCHAR2 (100),
LOCALITE VARCHAR2 (100),
AB_DATE VARCHAR2 (100),
PRIMARY KEY (NUMERO_AB)
)
CREATE TABLE EMPLOYE_SFAX
(
ID_EMPLOYE NUMBER,
NOM_EMP VARCHAR2 (100),
PRENOM_EMP VARCHAR2 (100),
USERNAME VARCHAR2 (100),
PASS VARCHAR2 (100),
CENTRE VARCHAR2 (100),
CLASSE VARCHAR2 (100),
PRIMARY KEY (ID_EMPLOYE)
)
ALTER TABLE FACTURE_SFAX
ADD CONSTRAINT FK1 FOREIGN KEY (NUMERO_AB) REFERENCES ABONNE_sfax(NUMERO_AB)
ALTER TABLE OPERATION_SFAX
ADD CONSTRAINT SYS_C0010817 FOREIGN KEY (ID_EMPLOYE) REFERENCES EMPLOYE_sfax(ID_EMPLOYE)
ALTER TABLE OPERATION_SFAX
ADD CONSTRAINT SYS_C0010818 FOREIGN KEY (NUMERO_AB) REFERENCES ABONNE_sfax(NUMERO_AB)
INSERT INTO ABONNE_sfax VALUES ('22963335','trabelsi','slim','11 Av kasem','sfax','11-02-2013')
INSERT INTO ABONNE_sfax VALUES ('94789456','feki','ramzi','11 Av kasem','sfax','11-02-2013')
INSERT INTO ABONNE_sfax VALUES ('22225987','arab','samir','11 Av kasem','sfax','11-02-2013')
INSERT INTO ABONNE_sfax VALUES ('52951951','abida','amine','17 Av Beb Khadra','sfax','11-02-2013')
INSERT INTO ABONNE_sfax VALUES ('52741852','soyed','olfa','11 Av kasem','sfax','11-02-2013')
INSERT INTO ABONNE_sfax VALUES ('52123456','bahrini','ramzi','11 Av kasem','sfax','11-02-2013')
```

Figure 13. SQL script for the creation of the new site Generation

To check distributed data integrity, our application provides the user with the opportunity to run triggers that take into account the database distribution. He can also consult their SQL script. These triggers are updated automatically when adding, changing, or deleting a site.

In Figure 14, we can visualize an example of an automatically generating script for a deleting trigger (the user just mentions the centralized table and the system detects the tables' fragments and generate the distributed trigger script). The user can choose to activate a cascade delete.



Liste des triggers:
ABONNE_TRIGGER_INSERT

ABONNE

```
create or replace trigger ABONNE_TRIGGER_DELETE
Before DELETE on ABONNE
for each row
declare
  c number;
begin
select count(*) into c from OPERATION_tunis@site_tunis_link where NUMERO_AB =
:old.NUMERO_AB;
if c > 0 then

raise_application_error(-20002, 'clef etrangère est violée dans le site _tunis');
end if;
select count(*) into c from FACTURE_tunis@site_tunis_link where NUMERO_AB =
:old.NUMERO_AB;
if c > 0 then

raise_application_error(-20002, 'clef etrangère est violée dans le site _tunis');
end if;
select count(*) into c from OPERATION_sfax@site_sfax_link where NUMERO_AB =
:old.NUMERO_AB;
if c > 0 then

raise_application_error(-20002, 'clef etrangère est violée dans le site _sfax');
end if;
select count(*) into c from FACTURE_sfax@site_sfax_link where NUMERO_AB =
```

Figure 14. Triggers script generation

In this section, we have detailed our implemented tool with the most important interfaces.

VII. CONCLUSION

In this paper, we proposed an incremental solution for the implementation of DDB in the Oracle DBMS, that takes into account the addition or deletion of a site or the modification of a site by adding or removing fragments. Our approach has been validated by providing a graphical tool, which takes charge of the necessary updates that ensure distributed data integrity, remote access, transparency, allowance, etc.

However, our implementation still needs several improvements such as: 1) the refinement of the script generation algorithms 2) the expansion of the environment supporting our solution, which is currently limited to Oracle technology, by generalizing it to the other DBMS.

REFERENCES

[1] H.I. Abdalla, "A New Data Re-Allocation Model for Distributed Database Systems", International Journal of Database Theory and Application, 2012, Vol. 5, No. 2, pp. 45-60.

[2] R. Moussa, "DDB Expert: A Recommender for Distributed Databases Design", Database and Expert Systems Applications (DEXA), 2011, pp. 534-538.

[3] F. Hassen and A. Grissa Touzi, "Toward a new approach of distributed databases design and implementation assistance", DBKDA, 2014, pp. 104-110.

[4] S. Gupta, K. Saroha, and K. Bhawna, "Fundamental Research of Distributed Database", International Journal of Computer Science and Management Studies, 2011, Vol. 11, Issue 02, pp. 138-146.

[5] M. T. Özsu and P. Valduriez, "Principles of distributed database systems", New York, Springer, 2013.

[6] F. Bouzaiene, "Oracle Golden Gate", Oracle Technologie Day Tunis, March. 2013.

[7] E. Hewitt, "Cassandra: The definitive guide", O'Reilly, 2010.

[8] J. Shute, M. Oancea, S. Ellner, B. Handy, E. Rollins, B.Samwel, R. Vingralek, C. Whipkey, X. Chen, B. Jegerlehner,K. Littlefield, P. Tong, "F1: The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business", SIGMOD conference, 2012, pp.777-778.

[9] M. Stonebraker, "The INGRES Papers: natomy of a relational database system", Addison-Wesley Publishing Compay, 1986.

[10] H. Madi, "Design and implementation of a distributed database under Oracle: For accommodation of university residences", Faculty of Exact Sciences in Algeria, 2009.

[11] J. Guignard, "Design method for DDB", 2004.

[12] C. Hsu, A.Rubenstein , L. Yee, G. Babin, N. Lawson, W. Hofmann., "What Is Rensselaer's Metadatabase System? ", Proc. 3rd International Conference on computer integrated manufacturing, IEEE Computer Society, pp.424-433,1992.