ISSN No. 0976-5697

Volume 4, No. 4, March-April 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Incremental Schema Upgrade Methodology for Microsoft SQL Database

Yogesh S Patil *
Department of Computer Science & Engineering
Student (M.E), S. S. G. B. C. O. E. & Technology,
Bhusawal , India
yogeshe146@gmail.com

Priti Subramanium
Department of Computer Science & Engineering
Asst. Professor, S. S. G. B. C. O. E. & Technology,
Bhusawal , India
pritikanna559@gmail.com

Abstract: There are many problems associated, when we upgrade our product database schema to next version. Many times it is found that there is data corruption or we find that data integrity and consistency is lost. There are many solutions/methods, but each method has its own drawback to overcome these problems. We have suggested an incremental methodology to upgrade a database schema from one version to next version having no data corruption and maintaining the database consistency and integrity.

Keywords: schema upgrade, schema migration, Schema Evolution, Object database, schema integration.

I. INTRODUCTION

A simple method of performing a schema migration is to execute SQL script files before or after the software upgrade [1]. It is painstaking aspects of to updating a deployed application and to transforming existing data [2].The life cycle of any product consists of many releases. For e.g. at the beginning version 1.0 of a product with limited or targeted functionality is released to the customer. While customer start evaluating/using the product, the developers start working on enhancing, add new functionality and fixing issues of version 1.0 of product to develop next version of the product. This cycle goes on and newer version of product are released to the customer like (version 2.0, 3.0 etc). This creates the need for upgrade of product from lower version to later versions.

With this approach of development there are following challenges to be address:

a. Upgrade of code files (class files for Java/ dlls for C# .net).
b. Upgrade of database without the loss or corruption of data.

This methodology provides a complete guidance about how upgrade of database can be achieved.

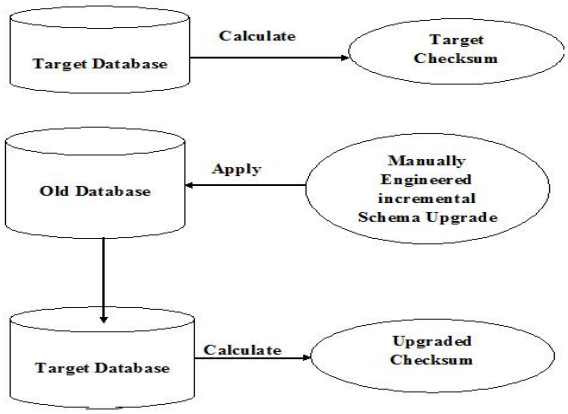
II. DATABASE SCHEMA

A problem in building and maintaining large, long-lived application systems is to cope with all the changes that inevitably will be imposed on the systems over time. Numbers of large application systems are centred on a database [3]. Schema evolution is one of the most important aspects of the use of a database, because once the database has been constructed, users' requirements tend to change [4].In the simplest definition, a database is a collection of data. Software applications that are designed to help create, maintain and use databases are called database management systems (DBMS). DBMSs provide abstractions by which the logical model of the data can be separated from the way it is physically stored. A metadata evolution technology has been use in which they realize the mapping from the old data schema to new data schema smoothly during schema upgrade. So the schema upgrade can be completed without data migration [5].

A Schema extraction and evolution method has been used when they release any history of an application, they start extracts the database schemas embedded in the application and compare the schemas, and produce a tally of schema evolution results [6]. Any update to be executed we have to check all integrity constraints with efficient integrity methods [7].The dependencies between attributes within tables are measure using an information-theoretic .Then they construct a dependency graph for each table capturing the dependencies among attributes after then they find matching node pairs across the dependency graphs by running a graph-matching algorithm [8].

III. IMPLEMENTATION

Upgrade from source schema version to target schema version should work smoothly. First take out the checksum of target database schema and name it as target checksum. Then manually write upgrade scripts once we complete writing the upgrade script then apply this upgrade scripts sequentially on the old database schema and calculate its checksum and name it as upgrade checksum. If target checksum value is equal to upgrade checksum value then we can say that the database schema is successfully upgraded to target version. Figures 1 show overview of Incremental Schema Upgrade Methodology.



Target checksum= Upgraded Checksum

Figure 1 Incremental Schema Upgrade Methodology

IV. ALGORITHM

a. Set database name and path of upgrade scripts and checksum=false.
b. Take the backup of the existing database using the new Product's script for backup and restore.
c. Sort all the upgrade scripts from the location specified in path of upgrade scripts option and start applying them sequentially on the target database.
d. Perform steps 5, 6 and 7 only if checksum flag is true
e. Calculate the checksum of the database schema after applying each script and compare it with the checksum stored into that script file.
f. If upgrade script does not have a pre-calculated checksum in it, then upgrade should fail and exit.
g. If checksum matches then it verifies that the script is applied correctly.
h. Once that is verified, add a new row should be added Product's_DBversion table with the schema version for that script (mapping and sp versions will not change in that case).
i. Repeat steps 4 and 8
j. If checksum does not match then upgrade should fail and exits by restoring the database to the original.
k. Once all the scripts are applied, all stored procedures, views and mapping files (except UDA mapping files) should be copied to target database.

V. EXPERIMENTAL RESULT

To determine whether our methodology work or not we have perform a simple experiment in which we have develop a simple software of book search system with limited functionality in first version the database name given as bss_db in this database we have create following tables Author, Subject, Book. In Author table we have following Field Name, Address, email_id, contact and author_id .In subject table we have Name, Description and subject_id field. In Book table we have Title, Book_id, Description, author_id, subject_id and Price field. After than we have modified the existing software of book search system by adding more functionality to it for example in Author table we have added age field. In Book table we have added publisher_id and stream_id field and drop the field subject_id. Rename table Subjects to Streams and added one table of Publisher with following field publisher_id, Name, Address, and Contact. We have make publisher_id as foreign key in Book table and also Rename subject_id column to stream_id. Known when we want to move our database bss_db to next version we have written upgrade scripts for every modification that we have done in first version the upgrade scripts are as follows:-

A. Upgrade script to add table Publisher:

```
CREATE TABLE [dbo].[Publisher](
        [publisher_id] [int] IDENTITY(1,1) NOT NULL,
        [Name] [nvarchar](1000) COLLATE
Latin1_General_CI_AS NULL,
        [Address] [nvarchar](1000) COLLATE
Latin1_General_CI_AS NULL,
        [Contact] [nvarchar](1000) COLLATE
Latin1_General_CI_AS NULL,
CONSTRAINT [PK_Publisher] PRIMARY KEY
CLUSTERED
(
        [publisher_id] ASC
)WITH (PAD_INDEX  = OFF,
STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  =
ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
Go
```

B. Upgrade script to add age column:

```
Alter table author
add age int;
go
```

C. Upgrade script to add Publisher_id column:

```
alter table book
add publisher_id int
go
```

D. Upgrade script to make publisher_id as foreign key:

```
alter table book
ADD CONSTRAINT fk_Publisher FOREIGN KEY
(publisher_id) REFERENCES dbo.Publisher (publisher_id)
Go
```

E. Upgrade script to Rename table Subjects to Streams:

```
sp_Rename 'Subject', 'Streams'
go
```

F. Upgrade script to Rename subject_id column to stream_id:

```
sp_rename 'streams.subject_id', 'stream_id'
go
sp_rename 'book.subject_id', 'stream_id'
go
```

Now before running this upgrade scripts when have taken backup of existing database that is bss_db in backup file and then we have calculated checksum value of database schema of both version that is bss_db for first version and bss_db2 for second version as shown in Figure 2 We found that both values are different, then we have run upgrade scripts and taken checksum value again and we found that both value are same as shown in Figure 3 We have seen that database as been successfully upgraded to next version. The Figure 4 to Figure 10 shows database table before and after upgrade scripts run.

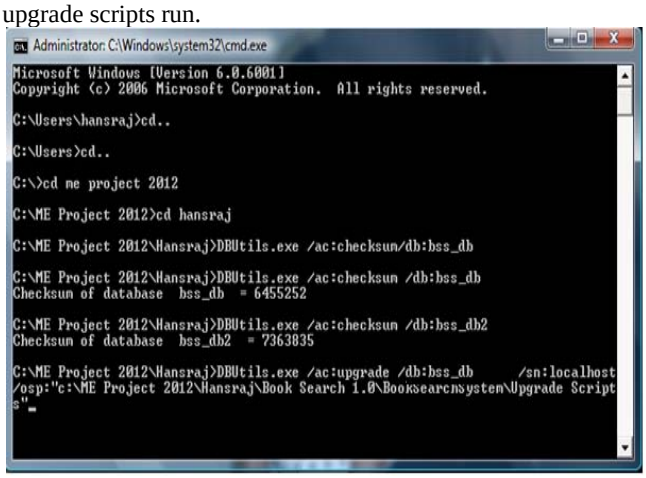


Figure 2 Checksum value of database schema before upgrade scripts run.

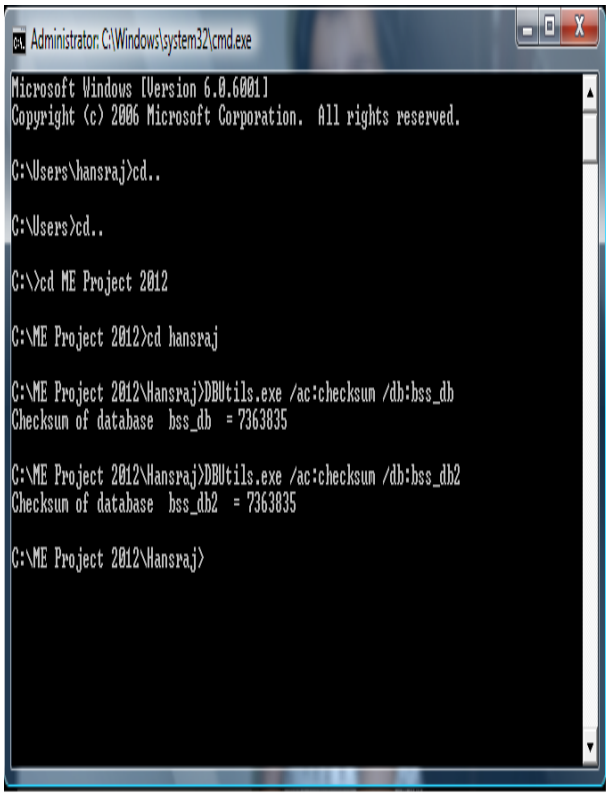


Figure 3 Checksum value of database schema after upgrade scripts run.

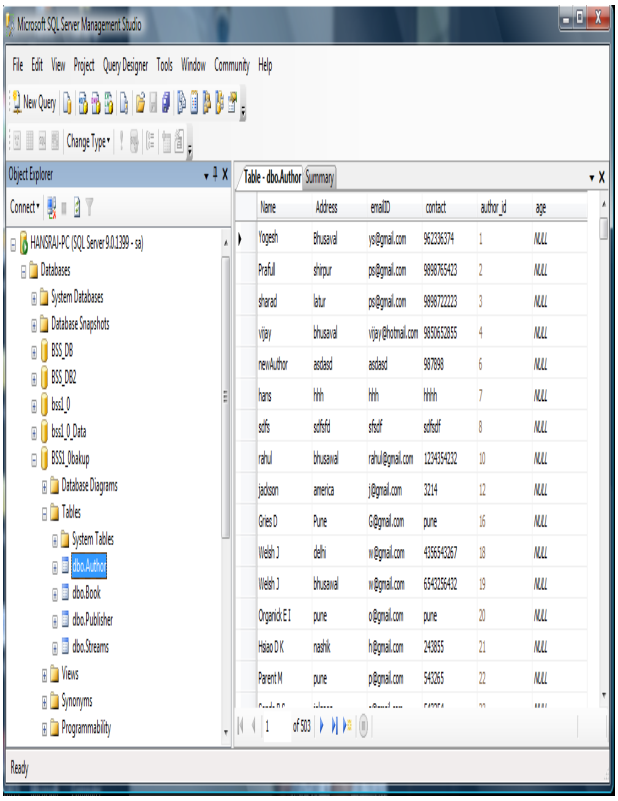


Figure: 5 Author table after upgrade scripts run.

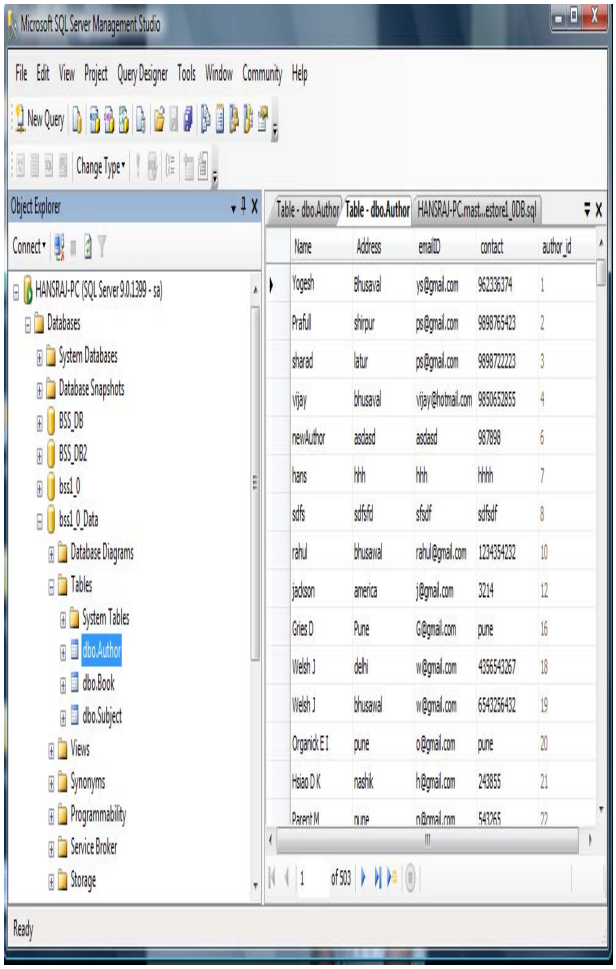


Figure: 4 Author table before upgrade scripts run.

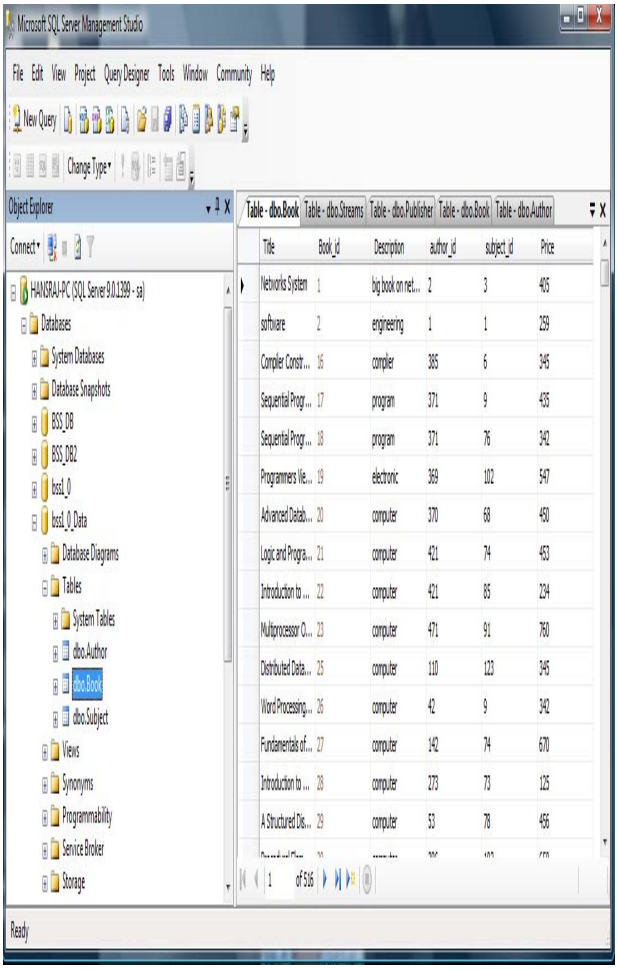


Figure: 6 Book table before upgrade scripts run.

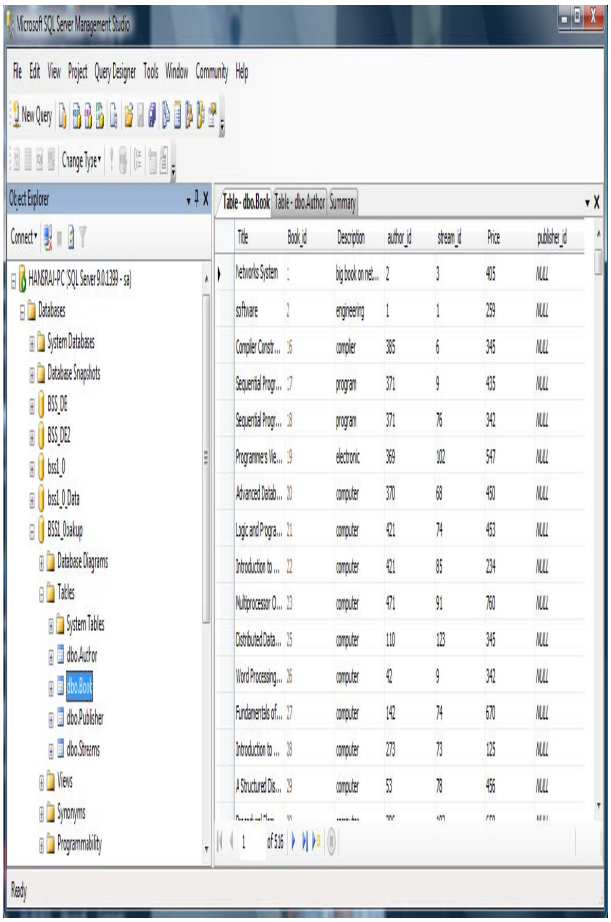


Figure: 7 Book table after upgrade scripts run.

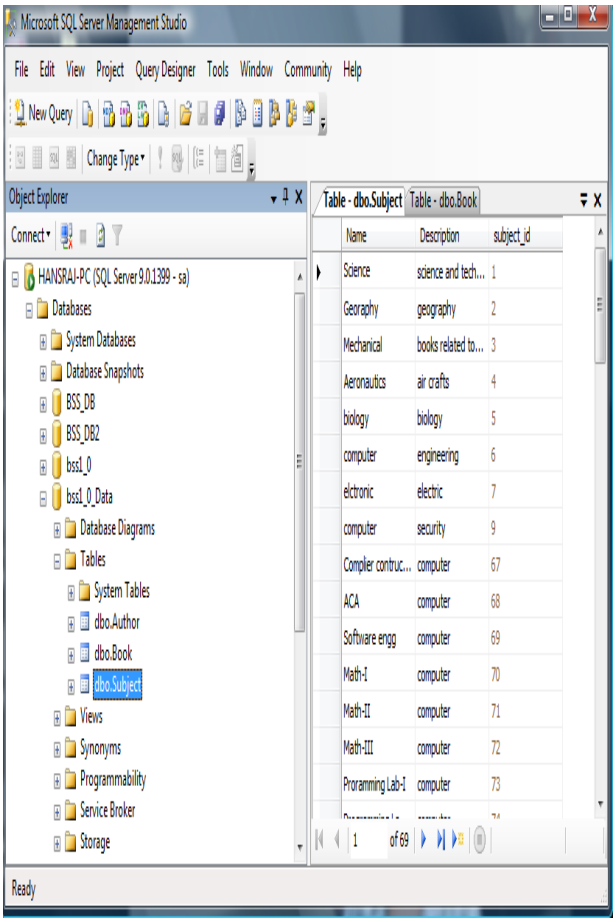


Figure: 9 Subject table before upgrade scripts run.

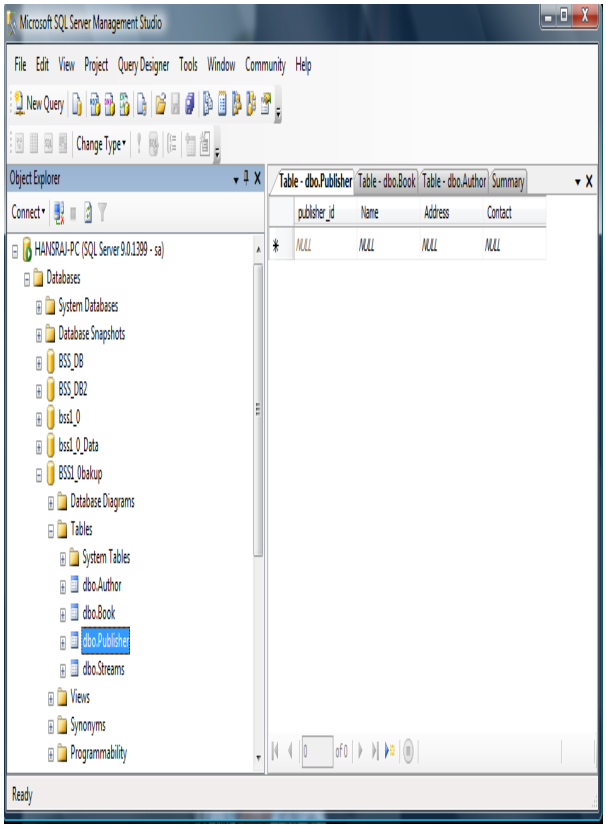


Figure: 8 Publisher table created after upgrade scripts run.

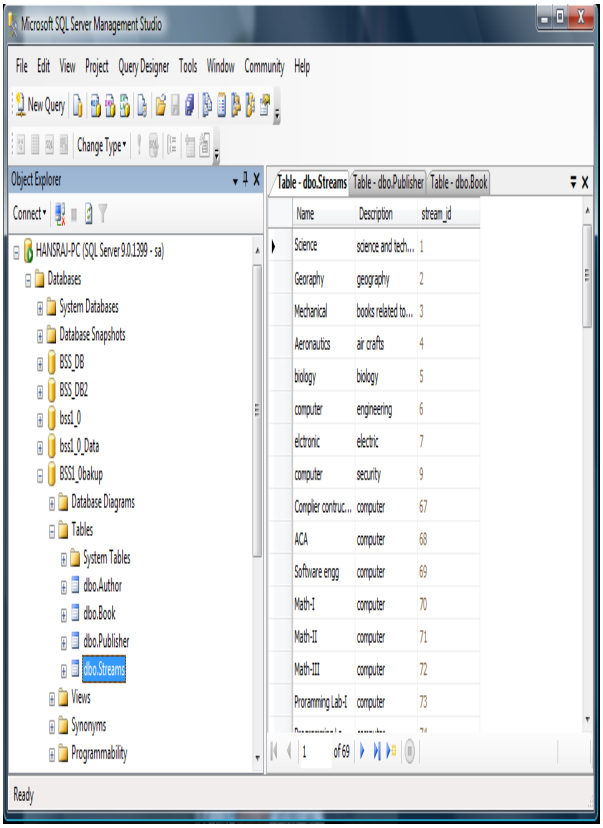


Figure: 10 Subject table became streams table after upgrade scripts run.

VI. ADVANTAGES

a. Maintain relationships among attributes, and the conditions for maintaining consistent views of programs.
b. Each schema version is expressed as a (database) view which constitutes actual interface to the application programs.
c. Schema changes are made transparent to the application programs.
d. No need of extra data cleanup.
e. Easy to find out if any schema written wrong.

VII. CONCLUSION

Finally we conclude that our incremental methodology to upgrade database schema solve the problem associated with relational database upgrade to enterprise software. In future we can work on analyzing different methodology including my methodology that are available to upgrade data base schema and try to make such software that will directly write SQL script for us.

VIII. REFERENCES

[1]. Robert M. Marks," A Metadata Driven Approach to Performing Multi-Vendor Database Schema Upgrades",' 19th IEEE International Conference and Workshops on Engineering of Computer-Based Systems', 2012, PP 108-116.

[2]. Praveena Mandapati , U. Tulasi ," Extended Schema Extraction and Evolution",' IEEE International Journal of Emerging Technology and Advanced Engineering', 2012, Vol. No. 2, PP 220-224.

[3]. Dag.Sjøberg, "Quantifying schema evolution," in Information and Software Technology, vol. 35, no. 1, January 1993, PP 35–44.

[4]. Junichi Takahashi, "Hybrid Relations for Database Schema Evolution",' Fourteenth Annual International IEEE International Conference on Computer Software and Applications', 1990, PP 465-470.

[5]. Hui Liu, Shengqi Wu ," Data Storage Schema Upgrade via Metadata Evolution in SaaS ",' IEEE International Conference on Consumer Electronics, Communications & Networks (CECNet)', 2012,PP 3148-3151.

[6]. Shengfeng Wu, Iulian Neamtiu," Schema Evolution Analysis for Embedded Databases ",' IEEE International Conference on Data Engineering Workshops (ICDEW)', 2011, PP 151-156.

[7]. Hendrik Decker, Davide Martinenghi, " Inconsistency-Tolerant Integrity Checking ",' IEEE Transactions on Knowledge and Data Engineering ', Vol. 23, No. 2, February 2011.PP 218-234.

[8]. Jaewoo Kang ,Jeffrey F. Naughton," Schema Matching Using Interattribute Dependencies ",' IEEE Transactions on Knowledge and Data Engineering ', Vol. 20, No. 10, October 2008. PP 1393-1407.

