CROSSBASE
An Opensource Database Data Migration Tool

Vamsi Krishna Myalapalli
California State University,
Sacramento, CA
Vamsikrishna.vasu@gmail.com

***Abstract*— In the contemporary world, there is high necessity for the engineers to handle unstructured or semi-structured data. To handle this situation, companies are exploring ways to make their software faster and flexible by migrating their data from traditional database to NoSQL database.**

Hence the companies or standalone users need an application or tool which migrates their data from traditional SQL systems (e.g. Oracle) to NoSQL systems (e.g. Mongo database). However, the current market holds extremely few commercial tools to perform data migration from RDBMS Databases to Mongo database. Furthermore, these tools expect the user to pre-process into specific format (such as JSON) and then migrate the data onto target database. Such processes are not straight-forward could be error-prone.

CROSSBASE is the first open source, handy and fully automated tool that can perform data migration from Oracle database to Mongo database. User do not need to pre-process the data other than providing minimal inputs. This migrator migrates data while preserving relationships between data. Apart from migration this tool offers several other features like data migration from XML databases to Mongo database, generating dump file for future use etc. Compared to the two existing commercial tools Safe Software and Studio3T, CROSSBASE is fully automated and offers several additional features.

***Index Terms* - CROSSBASE; Mongo Database Migration; Oracle-Mongo DB Migration; RDBMS to MongoDB Migration.**

I. INTRODUCTION

The increasing magnitude of semi/unstructured data and unpredictable data might necessitate companies to make use of NoSQL databases. The necessity to develop agile system or scoping for changing requirement(s) or to function at any scale or any other requirement [14] demands the use of NoSQL systems.

The NoSQL Mongo database is an open-source, platform-independent document-oriented database. MongoDB uses JSON documents (key value pairs) with schemas. Fig 2.1 explains the terminology translation between traditional database and MongoDB [1][2]. MongoDB is used to store data for various high-performance applications.

NoSQL databases can easily handle unstructured data, hence vendors who want to handle unstructured data in real time, or to increase scalability might want to migrate to NoSQL database like MongoDB.

CROSSBASE provides an easier and efficient way of migrating data from traditional RDBMS systems to NoSQL Databases. CROSSBASE is a JAVA based application that uses JDBC to perform read and write operations on SQL and NoSQL databases. The operations like reading data from source database, migrating data to target data structures, writing data to target databases are completely taken care by CROSSBASE.

Further content in the paper is framed as follows. Section 2 explains the related work and background. Section 3 deals with implemented model. Section 4 demonstrates how to use the tool. Section 5 explains comparative analysis among tools and at the end section 6 concludes the paper.

II. BACKGROUND AND RELATED WORK

Motivation:
Even today millions of people are googling for an open source data migrator or an equivalent program to perform data migration from traditional SQL database systems to MongoDB or other NoSQL systems.

There are extremely few commercial tools (3 at most) available in the market to perform this data migration [3] [13]. Unfortunately, there are no open source tools as well to perform this data migration. This reason motivates an open source tool to be developed.

RDBMS	MongoDB
Database	Database
Table	Collection
Record/Row	Document
Index	Index
JOIN	Embedded document, document references or $lookup to combine data from different Collections
Foreign Key	Embedded/Nested Document

Fig. 2.1: Terminology Translation

The above figure describes the differences in terminology among traditional database systems and NoSQL systems (specifically Mongo database).

Foreign Key vs. Embedding:
Traditional databases offer referential constraint mechanism known as Foreign Key. However, there is no concept of foreign key in NoSQL databases like MongoDB. No relation exists between table-table (i.e. collection to collection) in NoSQL systems. Foreign keys are represented in the form of embedding in NoSQL systems. Embedding is the mechanism in which child entry or dependent row is embedded into the respective parent row. This is explained in the below diagram by comparing relations in SQL system and collections in NoSQL system i.e. Mongo database.

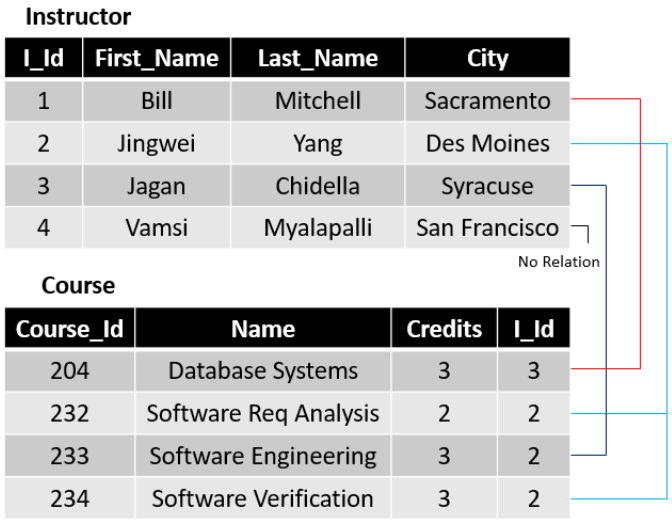

Fig. 2.2.1: Foreign Key Relation in SQL System

The above foreign key relationship in RDBMS is transformed as embedding in Mongo database.

```
{
 I-Id: 2,
 first_name: Jingwei,
 last_name: Yang,
 city: Des Moines,
 courses:
  [
   { course_id: 232,
     name: Software Req Analysis,
     credits:2, ...},
   {course_id: 234,
     name: Software Verification,
     credits: 3, ...}
  ]
}
```

Fig. 2.2.2: Embedding in Mongo Database System

The above JSON embedded entry explains that child record appears in parent record as embedded record. This is the default behavior for handling foreign key relationships. If embedding is not enabled, then parent and child tables exist as independent tables in target database.

III. IMPLEMENTED MODEL

This section introduces the implemented model, CROSSBASE challenges and features, use-case diagram and the pseudo algorithm. JAVA is used as backend programming language and does all the processing. This processing encompasses reading data from source database, transforming data to target structures and writing the transformed data to target database. The initial version of CROSSBASE fully supports data migration from Oracle Database/XML Databases/JSON databases to Mongo Database. The future versions will support more number of databases on both source and target ends.

The two JDBC modules (Oracle Thin Driver and MongoDB Driver) will exchange data between databases and complete the data transfers. The entire flow would include connection management, data validations, transforming tables to JSON Key-Value Pairs, writing data onto target database, keeping track of migrated objects (for allowing reversibility), handling logs, generating inventory etc.

Layer	**Technologies**
Backend	JAVA, JDBC
Databases	Oracle SQL, Mongo NoSQL
Application Server	Apache Tomcat (GUI Version)
Frontend	JSP (GUI Version)

Table 3.1: Technologies Used in CROSSBASE

Potential Challenges of CROSSBASE:

a) Identifying primary key/composite key in tables.
b) Preserving relationships among tables i.e. foreign keys. This issue is resolved in CROSSBASE by using JSON embedded structures [1][5][6].
d) Discrepancy in foreign key relationships or indexes or any other database object should not be seen.
d) Transforming normalized data to de-normalized data. Since referential integrity is absent in NoSQL systems, we will see data in denormalized form.

CROSSBASE Features:

a) Aimed at reducing manual effort while migrating data from traditional SQL databases to NoSQL Databases.
b) Can migrate data to multiple systems in a single go.
c) Allows user to save transformed JSON KV pairs data for future use. The tool generates a dump file and this file can be used in future.
d) Works with multiple formats like SQL Databases, JSON and XML. Crossbase can migrate several database files in case of XML or JSON databases (Multifile database).
e) Inventory report contains every operation that was performed. User can track the migrated database objects.
f) Reverting capability: User can revert the migration if he/she is no longer interested in migrating data. When migration is reverted all the objects created by CROSSBASE on target system will be deleted.

g) Exception/Status logging at every layer for tracking issues.

h) Easy to interact, fast and robust [9][10].

i) Data Injection Mode: This mode allows user to perform performance/stress testing. User can inject millions of random tables/collections into target database.

j) Debug Mode: This mode gives more information (verbose mode) to user while the tool is running.

The following two images explain the data flow in CROSSBASE and application high level flow respectively.

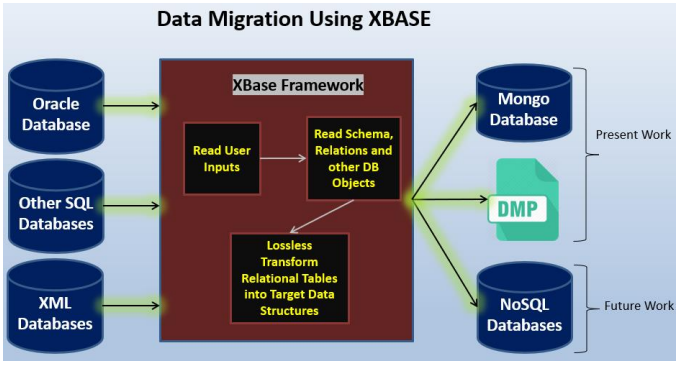

Fig. 3.2. Data Migration Using CROSSBASE

The above figure describes the block diagram of CROSSBASE framework and its components. CROSSBASE can consider any component on the left hand side as an input. After reading data from source database, data is processed and is migrated to Mongo database. It also parallely generates a dump file for future use. This dump file contains transformed data structures and can used to repeat the migration process on some other system where CROSSBASE is present.

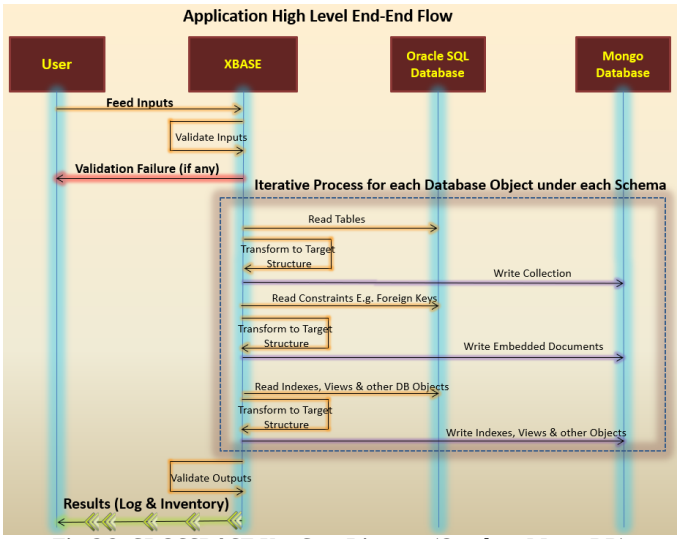

Fig. 3.2. CROSSBASE Use Case Diagram (Oracle to MongoDB)

The above diagram describes the sequential data flow between user, CROSSBASE, source and target database systems. Reading and writing is an iterative process for each table under each schema.

Pseudo Algorithm for Migration:

CROSSBASE uses the following pseudo algorithm for migration purpose. The complete source code is available online at GitHub [15].

```
determine database type; // SQL database or XML database.
initiate source and database connections;
for(each Schema in schemaList) {
    for(each table in CurrentSchema)  {
            getTableRowsFromSourceDatabase;
            transformTableintoTargetDatastructure;
            create target object using transformed structure;
            if(embeddingEnabled && foreignKeyPresent){
                embed child record into parent record;
            }
            if(indexingEnabled && indexPresent){
                read index in source and create index on target;
            }
            if(viewsEnabled && viewPresent){
                read view def in source and create view on target;
            }
      update log;
      update inventory;
      }
 }
```

The above algorithm might appear deceptively simple, but it encompasses a lot of read, conversion, store and write operations.

IV. USING CROSSBASE

In this section we will discuss, how to use CROSSBASE tool and migrate data. The primary input for CROSSBASE tool is XBASE.properties file (a standard JAVA properties file) which is built of key-value pairs. User can run this tool using the code provided or XBASE.jar file can be used. In either case XBASE.properties file is required to initiate data migration.

XBASE.properties

```
# Welcome to XBASE. This configuration file is an input for XBASE tool. Caution: Please do not modify property names.
# MigrationMode [boolean] - true: Migrates data from source to target database. false: Does not perform any data migration. [Default:true]
MigrationMode=true

#HostName - hostname or IP address of the system. [Default value: localhost]
HostName = localhost

#SourceDatabase - Source Database from which data is read.
SourceDatabase=oracle
#SourceDatabasePort, Username and Password
SourceDatabasePort=1521
SourceDatabaseUsername=vamsi
SourceDatabasePassword=krishna
#SchemaToMigrate Source schema to be migrated
SchemaToMigrate=vamsi
#Applies only for Oracle database
MigrateSystemSchema=false

#TargetDatabase name, port and credentials
TargetDatabase=mongo
TargetDatabasePort=27017
#TargetDatabaseName. This is the name of new database into which data will be copied. [Default:SourceDatabaseName] Eg:orcl
TargetDatabaseName=newdb
TargetDatabaseUserName=admin
TargetDatabasePassword=mydatabase

#Embedding. This parameter enables embedding Foreign Key relation ships [Default:false]. Child entries are embedded into parent entries.
Embedding=true

#MigrateIndexes, true-creates indexes on tables [default-false]
MigrateIndexes=false

#Generates equivalent JSON dump of the source database [default-true-false]. If this value is false, then parameters JSONDumpFilePath,JSONDumpFileName will be ignored.
ExportJSONDump=true
JSONDumpFilePath=D:\\
JSONDumpFileName=JSONDump.txt

#InventoryFileName: File name of Inventory and its path
InventoryFilePath=D:\\
```

Fig. 4.1: CROSSBASE Properties File (XBASE.properties)

The above screenshot explains how the properties file would look like. Each key serves a specific purpose and most keys have default values.
The following are some of the keys, purpose and default value as of today.

Key	Purpose
MigrationMode	Mode in which tool operates.
Hostname	Hostname
SourceDatabase	Source Database whose data is migrated
SourceDatabase Username, Password	Credentials of Source database
SchemaToMigrate	Specifies the schema to migrate. * migrates all schemas.
Embedding	JSON records are embedded when foreign keys are encountered.
MigrateIndexes	Creates Indexes in target database.
ExportJSONDump	Exports JSON dump, so that migration can be performed on other systems or later.
InventoryFilePath	This file holds inventory such as number of tables, indexes created.
LogFilePath	Log file to track exceptions/errors
DataInjectionMode	This feature can be used as stress testing feature on target database. Injects database with specified number of tables/records.
DebugMode	Gives more details while the tool migrates data.

For demonstrating the process, data present in Oracle database will be migrated on to Mongo database. Hence the file "XBASE.properties" is configured accordingly (Fig 4.1).

A schema present in the source database named 'orcl' is migrated to mongo database. Since the target database name is 'newdb' (as per Fig. 4.1), the tables from source database are populated into newly created database named 'newdb' in mongo database.

We will walk through some tables present in source database and as a part of migration we will see how those tables are transformed and written to target database.

```
SQL> select * from emp order by id;

        ID NAME                SALARY
---------- --------------- ----------
         1 Vamsi                100000
         2 Krishna              110000
         3 Vasu                 105000
         4 Jaya                 150000
         5 Lavanya              180000
         6 Akshara              120000
         7 Vijay                130000
SQL> select * from address;

        ID ADDRESS                ZIP
---------- --------------- ----------
         1 SanFrancisco         12345
         1 NewYork              54321
         2 Norway               99999
         3 Alaska               33333
         4 Norway               99999
         5 Canada               88888
         5 Norway               99999
SQL> select * from car;

        ID NAME              YEAR
---------- ------------ ----------
         1 Aston Martin       2011
         1 Lamborghini        2010
         4 Veyron             2011
         4 Roadster           2017
         5 Chiron             2012
         5 Turbo              2011
         5 Venom              2017
SQL> select * from blood;

        ID BL
---------- --
         1 B+
```

Fig 4.2. Demonstrating User Defined Tables

Fig. 4.2 displays some user defined tables in Oracle database. Table 'EMP' is the parent table for tables 'ADDRESS', 'CAR' and 'BLOOD'. They are bound with referential integrity also known as foreign key relationship. A query is executed to demonstrate this referential integrity among tables.

The following screenshot describes how the foreign key relations exist in source database. Query in the below screen shot displays the child tables (ADDRESS, CAR, BLOOD) for parent table (EMP).

```
SQL> SELECT FK.TABLE_NAME, COL.COLUMN_NAME FROM USER_CONSTRAINTS PK
  2  JOIN USER_CONSTRAINTS FK ON PK.CONSTRAINT_NAME=FK.R_CONSTRAINT_NAME
  3  AND FK.CONSTRAINT_TYPE = 'R' JOIN ALL_CONS_COLUMNS COL
  4  ON FK.CONSTRAINT_NAME=COL.CONSTRAINT_NAME WHERE PK.TABLE_NAME='EMP'
  5  AND PK.CONSTRAINT_TYPE = 'P';

TABLE_NAME   COLUMN_NAME
------------ ------------
ADDRESS      ID
BLOOD        ID
CAR          ID
```

Fig 4.3. Referential Integrity in Oracle for Table EMP

After performing migration, the above foreign key relations are transformed using the method called 'Embedding'.
The same can be seen in the below screenshot i.e. we displayed the content of EMP table, and the tables 'ADDRESS' and 'CAR' are embedded into parent table i.e. EMP table. If embedding is not enabled in "XBASE.properties" file, then every table becomes an independent table in target database.

C:\WINDOWS\system32\cmd.exe - mongo

```
> db.EMP.find().pretty()
{
        "_id" : ObjectId("5c00faf25fed738fc4e58a8e"),
        "SALARY" : "120000",
        "CAR" : [ ],
        "ADDRESS" : [ ],
        "ID" : "6",
        "NAME" : "Akshara",
        "BLOOD" : [ ]
}
{
        "_id" : ObjectId("5c00faf25fed738fc4e58a8f"),
        "SALARY" : "130000",
        "CAR" : [ ],
        "ADDRESS" : [ ],
        "ID" : "7",
        "NAME" : "Vijay",
        "BLOOD" : [ ]
}
{
        "_id" : ObjectId("5c00faf25fed738fc4e58a90"),
        "SALARY" : "100000",
        "CAR" : [
                {
                        "YEAR" : "2011",
                        "NAME" : "Aston Martin"
                },
                {
                        "YEAR" : "2010",
                        "NAME" : "Lamborghini"
                }
        ],
        "ADDRESS" : [
                {
                        "ZIP" : "12345",
                        "ADDRESS" : "SanFrancisco"
                },
                {
                        "ZIP" : "54321",
                        "ADDRESS" : "NewYork"
                }
        ],
        "ID" : "1",
        "NAME" : "Vamsi",
        "BLOOD" : [
                {
                        "BLOODGROUP" : "B+"
                }
        ]
}
```

Fig 4.4. Embedding Child Record(s) into Parent Record in Mongo Database

The following figure 4.5 shows inventory when embedding is enabled during migration.

In this way all the tables from the source database are read, transformed and written to target database, with respect to the inputs present in XBASE.properties file.

Parallelly a dump file with JSON entries will be generated for future use i.e. user might not want to migrate at that moment, but he/she might be interested in doing migration later. Then this file becomes handy for them.

XBASEInventory.txt

```
        XBASE Migration Report

*************************************************
Data Migration Started:  2018-11-30 13:54:15.88
-------------------------------------------------
Databases Created: 1
    newdb
Schemas Created: 0
Collections Created: 8
    TESTS
    T2
    DML_LOG
    STUDENT
    EMP
    USER_TABLE
    MYTABLE
    TEST
Collections Embedded: 3
    EMP.CAR
    EMP.ADDRESS
    EMP.BLOOD
Indexes Created: 0

Collections Deleted: 0
Indexes Deleted: 0
Schemas Deleted: 0
Databases Deleted: 0

-------------------------------------------------
Data Migration Completed: 2018-11-30 13:54:20.371
Elapsed Time: 0 minute(s) 4 seconds
*************************************************
```

Fig 4.5. Inventory Report (Embedding Enabled)

The following figure 4.6 shows inventory when embedding is disabled during migration.

XBASEInventory.txt

```
        XBASE Migration Report

*************************************************
Data Migration Started:  2018-11-30 19:31:54.415
-------------------------------------------------
Databases Created: 1
    newdb
Schemas Created: 0
Collections Created: 11
    ADDRESS
    TESTS
    T2
    BLOOD
    DML_LOG
    STUDENT
    EMP
    CAR
    USER_TABLE
    MYTABLE
    TEST
Collections Embedded: 0
Indexes Created: 0

Collections Deleted: 0
Indexes Deleted: 0
Schemas Deleted: 0
Databases Deleted: 0

-------------------------------------------------
Data Migration Completed: 2018-11-30 19:32:00.614
Elapsed Time: 0 minute(s) 6 seconds
*************************************************
```

Fig 4.6. Inventory Report (Embedding Disabled)

V. CROSSBASE VS. COMMERCIAL TOOLS

The following table explains the differences between CROSSBASE and Studio3T [3] one of the commercial tool available in the market.

Feature	**CROSSBASE**	**Studio3T**
Application Type	Open Source	Commercial
Data Import	Imports from a) Oracle DB b) Mongo DB c) JSON d) XML	Import formats- a) JSON b) Mongo DB
Data Export	JSON, XML	JSON
Inventory	Reported	Not Reported
Customized Migration (E.g. Tables only)	Supported	Not supported (Will import all objects)
Interaction	UI, Command Line	UI
Primary Purpose of Application	Data Migration between Oracle and MongoDB	Easy Interaction with Mongo DB
Data Injection Support	Yes. To perform stress testing, this feature was developed.	Not supported.
Migration Revertability	Yes	No

In case of other tools user must feed JSON key value pairs to inject data into NoSQL database. In case of CROSSBASE, user need not deal with all these intermediate phases, thus obviating intermediate user interactions.

VI. CONCLUSION AND FUTURE WORK

This open source tool helps lot of users who are looking for an open source tool to migrate their database data. CROSSBASE obviates the user overhead of writing JSON entries for target database and importing them manually.

Future Work: We would like to extend CROSSBASE functionality by performing data migration from all SQL Databases to all other No SQL databases like Oracle NoSQL (KV Database), Neo4j (Graph Database) etc.

VII. REFERENCES

[1] Mat Keep, RDBMS to MongoDB Migration, MongoDB Inc, "https://www.slideshare.net/matkeep/migrating-from-relational-databases-to-mongodb", referred on 11/28/2018.

[2] Alexandru Boicea et al, "MongoDB vs Oracle - Database Comparison", IEEE Third International Conference on Emerging Intelligent Data and Web Technologies, 2012.

[3] Application Studio3T, "https://studio3t.com//", referred on 11/21/2019.

[4] Anju R et al, "MongoDB vs. Oracle: A Comparative Study Using Classification Algorithms", IEEE International Conference on Computational Intelligence and Computing Research, 2016.

[5] Anuradha Kanade, "A Study of Normalization and Embedding in MongoDB", IEEE International Advance Computing Conference, 2014.

[6] Gansen Zhao1 et al, "Modeling MongoDB with Relational Model", Fourth International Conference on Emerging Intelligent Data and Web Technologies, 2013.

[7] Gokul Prabagaren, "Systematic Approach for validating JAVA-MongoDB Schema", IEEE International Conference on Information Communication and Embedded Systems, 2014.

[8] Mongo database official documentation or web page, "https://docs.mongodb.com/manual/reference/mongo-shell/", referred on 11/28/2018.

[9] Vamsi Krishna Myalapalli "An Appraisal to Optimize SQL Queries", IEEE International Conference on Pervasive Computing, India.

[10] Vamsi Krishna Myalapalli, "High Performance SQL", 11th IEEE International Conference on Emerging Trends & Innovation in Technology, India.

[11] MongoDB Inc, Official Documentation or WebPage, "https://docs.mongodb.com/manual/reference/sql-comparison/".

[12] Webpage, "https://db-engines.com/en/system/MongoDB%3BOracle", referred on 11/28/2018.

[13] Safe Software, "https://www.safe.com/convert/oracle/mongodb/", referred on 11/28/2018.

[14] Web page, "https://www.couchbase.com/resources/why-nosql", referred on 11/28/2018.

[15] GitHub website, "https://github.com/Vamsicse/CROSSBASE", CROSSBASE source code.