MECHANISM TO TRANSLATE RELATIONAL-BASED DATABASE TO COLUMN-BASED DATABASE

Ms.Reena[1], Ms.Renu[2]

[1,2]SRCEM, PALWAL (India)

ABSTRACT

*Lately, column-based database (CDB) has been considered as a standard solution dealing with the big data. But there is no standard translation mechanism mapping from Relational Database (RDB) to CDB. In this paper, we proposed a mechanism translating RDB to CDB. We use HBase as our CDB platform. HBase is a column-based distributed database management system built on Hadoop Distributed File System (HDFS) and has been considered as the typical column-based database on HDFS. In addition,
we build a search engine for Taiwan academic journals using our translation mechanism.*

Keywords: HBase, Hadoop, Translation Mechanism, Relational Database, Column-Based Database, ER-Model, Search Engine

I. INTRODUCTION

1.1 Background and Motivation

The digital information size explodes rapidly in nowadays environment. Chevron's CIO says hiscompany accumulates data at the rate of 2 terabytes – 17,592,000,000,000 bits – a day. While the storage capacities of hard drives have increased massively over the years, access speeds—the rate at which data can be read from drives—have not kept up. It takes a long time reading all data or searching specific data from the single disk—and writing is even slower. Therefore, distributed file systems like Google File System (GFS) [1], Hadoop Distributed File System (HDFS) [2] burst out leading the trend A distributed file system reduces the I/O time by reading from multiple disks concurrently. Working in parallel, one can reduce access time significantly. HBase[3] is a column-based distributed database system which provides manipulating methods based on HDFS. This paper aims to provide the translation mechanism from ER Model database schema to HBase column-based database schema. In order to test our translation mechanism, we take the Entity Relation Model (ERM) designed for storing academic journal papers (AJP) as input, translate the ERM into CDB Schema (CDBS) and build the CDB for AJP on HBase. This paper first simply introduces the basic concept of HBase and the translation methods between HBase and Relational Database and shows the true translation for periodical search engine at last.

1.2 HBase

HBase is a distributed column-oriented database built on top of HDFS, modeled on Google's BigTable database[4]. HBase is the Hadoop application to use when you require real-time read/write random-access to very large datasets[2]. The most basic unit for tables in HBase is a column. One or more columns form a row that is addressed uniquely by a row key. A number of rows, in turn, form a table, and there can be many of them. Each column may have multiple versions, with each distinct value contained in a separate cell. Rows are composed of columns, and those, in turn, are grouped into column families. This helps in building semantically

or topical boundaries between the data. Columns are often referenced as family:qualifier with the qualifier being any arbitrary array of bytes[3]. A table is made up of regions. Each region is defined by a start key and end key, may live on a different node, and is made up of several HDFS files and blocks, each of which is replicated by Hadoop. All table accesses are by the row key, secondary indices are possible through additional index tables. There is no SQL language in base HBase, HBase does not support complex searching functions. However, there is a Hive/HBase integration project[5] that allows Hive QL statements access to HBase tables for both reading and inserting. But the performance is not as good as expected. The better way is to design the index table and access through it.

II. TABLE TRANSLATION MECHANISM

In ERM, there are three different relationships among entities considering cardinality, i.e., one to one relationship (1-1), one to many relationship (1-m) and many to many relationship (m-n). If an entity A has a relationship with entity B and, similarly B has that with C, we defined that A has a recursive relationship (R-R) with C. In addition, we defined that the R-R length between 2 entities is the number of internal nodes on this R-R plus 1. For instance, the R-R length between A and C equals 2 (one internal node B plus 1). In our translation, we defined five types of translations from ERM to CDBS including (1) entity translation, (2) 1-1 translation, (3) 1-m translation, (4) m-n translation and (5) R-R translation. In the following, we shall discuss each of them in the following subsections.

2.1 Entity Translation
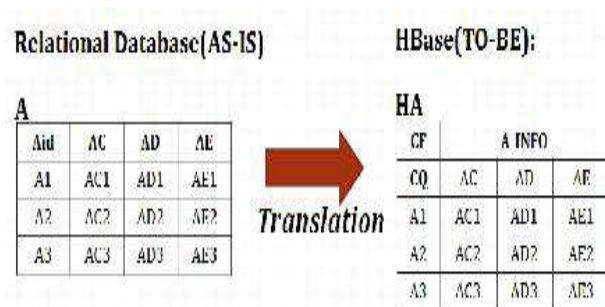
In ERM, an entity represents a relational table. A relational table contains a primary key and several non-primary key attributes. The primary key may be composed of several fields. In CDBS, a column family (CF) contains the qualifiers having the similar characteristics; each qualifier is unique in the containing column family. In relational table, a primary key value is the key identifies a particular row in relational table, and a row key in HBase table (HTable) also identifies a particular row in HTable. Thus, primary key in relational table can be considered as the row key in HTable. We combine a set of primary key values to be the unique row key value in HBase table and use notation ―:‖ to combine primary key containing two or more fields. Translation method is shown as following:

In Figure 1, we create a column family named A_INFO in *HA* to group the qualifiers and take the attributes' name as qualifiers in CDBS table. The naming of column family is user-defined. For the convenience of defining the translation, we pre-defined presentation to the features of the tables as follows:

1.For the relational table A, we define that A has

1.1 Let pk^A be the primary key of A, pk^A may consists of one or more fields where are called primary key fields.

1.2 Assumes that $pk^A = \{\ pk_e{}^A\ \}$, $1 \leq e \leq k$, *be the set of primary key* fields, where k is the number of primary key fields in A .

1.3 A set of non-primary key fields $f^A = \{fd^A\ \}$, $k < d \leq q+k$, where q is the number of nonprimary key fields in A.

2. For relational table B , we define that B

2.1 Let pk^B be the primary key of B, pk^B may consists of one or more fields where are called primary key fields.

2.2 A set of non-primary key fields $f^B\ \{\ fc^B\ \}$ in B , $z < c \leq y+z$ where z is the number of onprimary key fields in.

2.3 A set of primary fields, $pk^B = \{\ pkh^B\ \}$, $1 \leq h \leq z$, be the set of primary key fields, where z is the number of primary key fields in B.

2.4 B has a set of tuples tk^B .

Due to the entity translation from RDB to CDB for whole tables in relational database described above, we defined a translation function for single table, which is **entity(A)**, where A to be the input table for translation.

entity(A)

Assumption:

- Let A be a relational table for translation.

Translation steps:

▯ ▯

- For table *A*, an HTable with a set of row keys rkl^A is constructed. The construction for *HA* includes two steps, the first step is to define the row key for *HA* and the second step is to define the column family for *HA*.

The only way for user to retrieve data is by row key. Users can access the *HA* table by A's Primary key, but unable to do joins or other SQL queries. HBase may retrieve data by adding filters, but with the huge amount of data, the speed performs badly. Hence, in order retrieve data quickly, we have to build indexes for the entities in HBase. The building of indexes is under progress; In this paper, we only discuss the mapping of relational table into HBase.
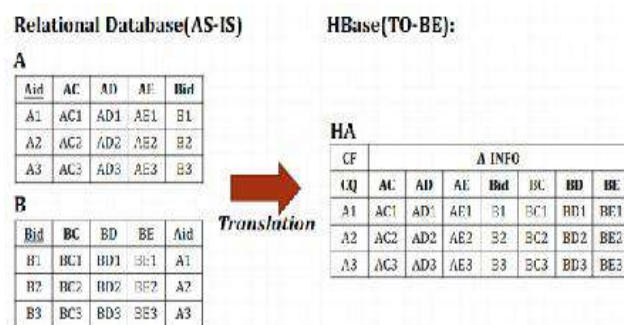
2.2 1-1 Translation

Figure.2 1-1 Translation

1-1 relationship stores the relationship of 2 entities using the Foreign Key (FK). A table uses Aid to be the FK and B table uses Bid. The two entities with 1-1 relationship actually will be stored in one table in relational database doing full outer join on two entities, taking one entity'sprimary key as its primary key. Thus, the translation is simple by mapping the actual table stored in RDB to CDB. The translation steps are shown as following:

1-1(A,B)

Assumption:

☐ Assume A entity and B entity has 1-1relationship, C is the actual table stored in the relational database taking A's primary key as C's primary key.

Translation steps

The translation includes two steps as follows:

- Step 1:
- ✓ Generate C table by A table full outer join B table on A's foreign key equals to B's primary key.
- Step 2:
- ✓ Execute entity(C) which accept C table as input and produce an HTable HC.

By the translation defined above, here we get the HC as shown in Figure 2, taking A's primary key as row key and preserve all B's attributes values in the HC.
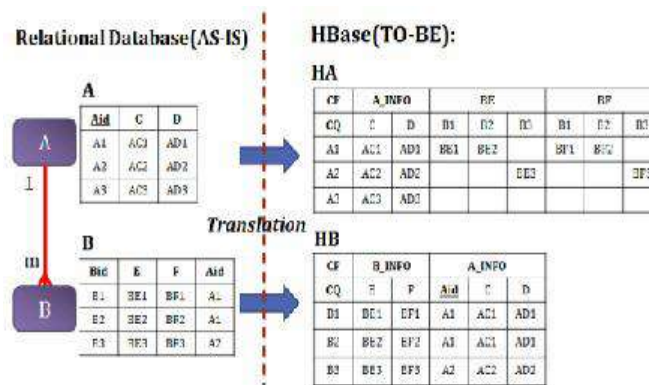
2.3 1-M Translation

Figure.3 1- M Translation

In m's perspective, 1-side is considered as 1-1 related with itself. Hence, we do 1-1 relationship translation to A and B at m side, that is, *1-1(B, A)*. We'll get HB as Figure 3 shows, recording the corresponding A primary key and it's attributes in HB. In 1's perspective, we have to store the multiple m's information. The qualifiers are the best thing to identify the multiple B's row keys. Hence, we create a column family for each attribute in B and use B's primary key as qualifier to store the attribute value for B. We call the 1's perspective translation the 1-side translation. However, B's primary key may contain several fields. Thus, we combine the B's primary key field value into one single value and defined a 1-side translation method *1-m-single(A,B)* shown as follows:

1-m-single(A,B):

Assumption:

➤ Let A and B to be the 1-m relationship table in RDB, A is at the 1-side and B is at the mside.

➤ HA and HB is constructed for the mapping HTables in CDB.

1-m (A,B)

Assumption:

➤ Assume A entity and B entity has 1-mrelationship, A is the 1-side and B is the mside.

Translation steps:

 The translation includes two steps as follows:

- Step 1:
✓ Execute 1-1(B, A), which takes B's primary key as row key generating an HTable HB.
- Step 2:

Execute 1-m-single(A,B), which generates an HTable HA preserving B's data related to A.
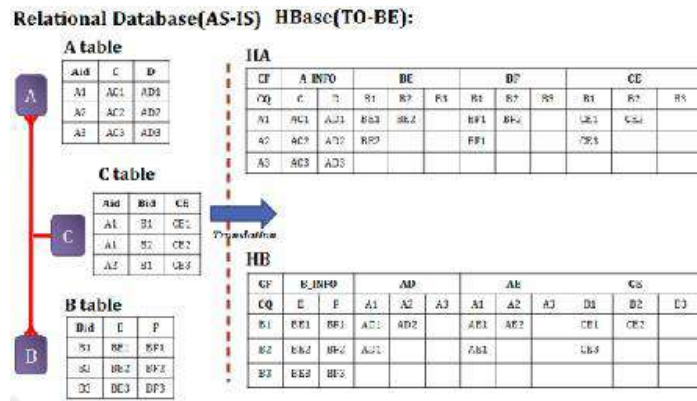
2.4 M-N Translation

Figure.4 M-N Translation

In m-n relationship, the relation is considered as two 1-m relationship. In order to make the translation, we do 1-m relationship to both mside and n-side. In Figure 4, HA preserves B's attribute column family, BE and BF and HB preserve A's attribute column family, AD and AF.

In addition, C table is the relation table generated by A and B, thus, C has 1-m relationship with A and B. Thus, we have to do the 1-m translation for the 1-side. The relation table C is not going to be preserved because C depends on A and B, so that the relationship can be stored in A and B table. For this reason, we don't have to translate the m-side relationships.

m-n(A, B, C)

Assumption:

➢ Assume A entity and B entity has m-n relationship, C is relational table generated by A and B.
   Translation steps*:*
➢ Execute 1-m(A, B), 1-m(B, A)
➢ Execute 1-m(A, C), 1-m(B, C)
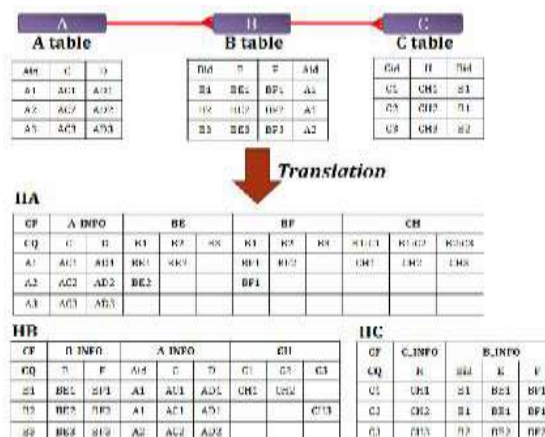
2.5 R-R Translation

Figure.5 R-R Translation

According to our translation, we might have done the translation between two entities. This makes user can easily draw the related information between two tables. However, there is R-R situation happened in ERM. One might want to draw the relative information between three or more tables. For example, there are three tables A, B and C, A and B has the 1-m relationship and B and C has the 1-m relationship, in our translation, we will preserve the B's information into A's table, but no C's information, for that C has the relationship with A. User gets C's information by accessing through A, B and C table, but this makes draw speed relatively slow. Thus, in order to draw C's information immediately, C's information should be stored in A table. We called the translation the R-R translation. R-R translation is user-decided; it is not necessary and is on the demand of user. R-R can be simplified into multiple 1-m relationship combinations, which is the A, B, C relationship described above; for the reason that other relationships are not the components of R-R relationship. m-n can be separated into 2 1-m relationships, so that we can treat it with the 1-m relationship. 1-1 is combined into one table, so that for the table having 1-1 shall be treated as one table. Hence, we discuss the R-R relationship translation for the tables composed of 1-m relationships.

In Figure 5, the translation between A and B, B and C are done respectively. We combine C's attributes into HA's column family. Follow the 1-m translation previously defined, we take C's primary key to be the qualifier. But one thing we have to notice is that C is related to A due to B, so C's primary key cannot be taken to be the qualifier. The solution is to combine B and C's primary key to be the unique qualifier. Thus, when dealing with the R-R situation, we have to combine the primary key of internal nodes in order to identify the value.

r-r translation(A, C)

Assumption:

- Assume that A and C are two relational tables having the R-R relationship with R-R length n. Let A be the 1-side and C be the m-side.
- Let $RRc^A=\{Tr\}$, $1 \leq r \leq n-1$ be the set of relational tables in the R-R relationship between A and C, excluding A and C.

Translation steps:

Construct a temporary relational table D = C.

- for (i = n-1; i < 1; i--)

  {

      D = full_outer_join(. D)

  }

    return 1-m(A, D)

III. PERIODICAL SEARCH ENGINE

In this section, we take the real implementation of AJP search engine ERM as an example to clearly describe the translation from RDB to CDBS.
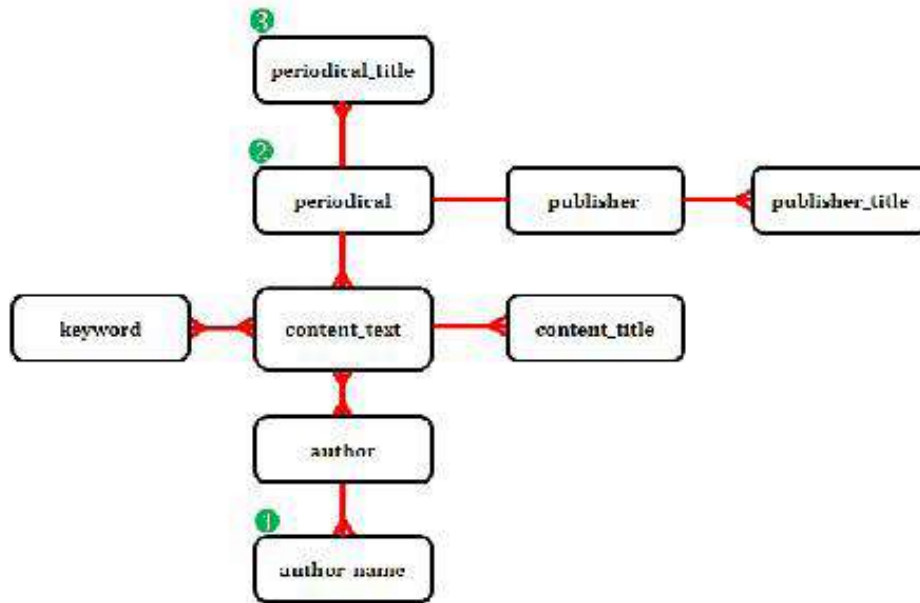
Figure.6 The AJP Search Engine ER Model

As the ER Model shown in Figure 6, the tables of each entity and its attributes is described
below (PKs are underlined)

☐ ☐

- author(author_id, author_identity_id)
- periodical(content_id, location)
- periodical_title(content_id, content_language, content_title_text)
- content_text(id, text_id, chapter_id, content_id, text_language, content_text)
- publisher(content_id, location)
- publisher_title(content_id, content_language, publisher_title)
- content_title(content_id, content_text)
- keyword(id, keyword_text)
- author_title(author_id, author_language, author_text)

In Figure 6, we see many R-R situations, for instance, periodical_title to author_name. Actually, not all of the R-
R should be translated; it depends on the user requirements. For instance, we don't require author_name's
information when we access periodical_title table. Thus, it is not necessary to deal with such relationship. In our
AJP search engine, we only do the R-Rs translation containing content_text, the only R-R to deal with is the one
whose nodes starts or ends at content_text. Consequently, there are 3 R-Rs been translated, which are (1)
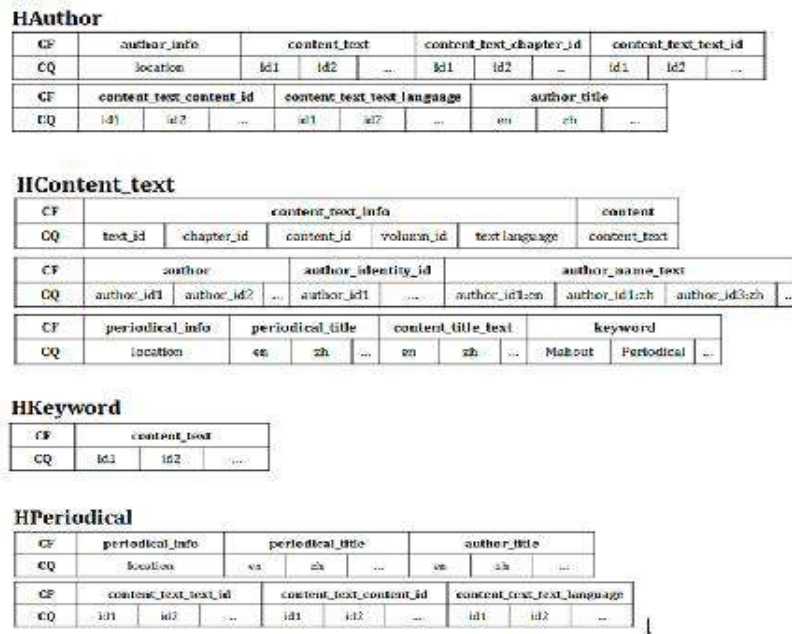content_text →author_name (2) content_text→periodical_title (3) periodical→publisher.

Figure.7 The AJP Search Engine CDBS

As shown in Figure 7, we get (1) HContent_text (2) HKeyword (3) HAuthor (4) HPeriodical in HBase finally. In the translation from ERM to CDBS, we don't preserve the following tables (1) HAuthor_name (2) HContent_Title (3) HPublisher_Title. The reason is that in the concept of the AJP search engine, some of the tables are created only to preserve the 1-m relationship with other tables. For example, the author_name table preserves the relationship with Author. Thus, these kinds of tables are created only if we have requirements on it or we have to access it. However, in the above translation we translate the relational tables into multiple corresponding HBase tables, due to the concept of Google Bigtable, the whole translated tables can be integrated into one Bigtable, distinguished by column families. But the HBase currently does not do well with anything above two or three column families because of compaction and flushing mechanism might increase the unnecessary I/O loading[6]. Hence, we translate the relational table into multiple corresponding HTables.

IV. CONCLUSION

This paper proposes a translation mechanism from RDB to CDB and builds an AJP search engine to verify the translation. According to the mechanism proposed, one can easily translate their system database into the CDB, not limited to HBase. This paper contributes to providing the approach to build the standard translation methods and stimulating the development of Object Relationship Mapping (ORM) in the future. This in-progress research will keep an eye on the translation in the more detailed translations, like the Business Object (BO) translation, or the sub-type, super sub-type translations. Furthermore, we will pay more attention on the AJP search engine we developed, try to create the indexing mechanism for it and test the performance and stability of the CDB.

REFERENCES

[1]. Ghemawat, H. Gobioff, and S.-T. Leung.: The Google file system. SIGOPS Oper. Syst. Rev., vol. 37, pp. 29-43(2003).

[2]. White, T.: Hadoop: The definitive guide. Yahoo Press (2010).

[3]. George, L.: HBase: The Definitive Guide. O'Reilly Media, Inc (2011).

[4]. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber.: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), vol. 26, p. 4, (2008).

[5]. Integrating Hive and HBase, http://www.cloudera.com/blog/2010/06/integratinghive- and-hbase/

[6]. The Apache HBase Book, http://hbase.apache.org/book.html