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. .

2 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

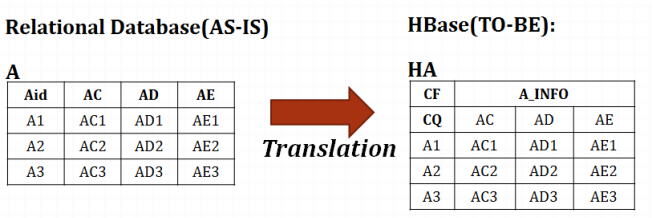

Figure 1. Entity translation

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:

Assumption:

- Let R be a relational database, that is $R = \{T_1, T_2 \cdots, T_n\}$ is a set of relational tables
- Let HR be a column-based database, that is $HR = \{HT_1, HT_2, \cdots, HT_n\}$ is a set of HTables, where HT_i is the mapping table of relational table T_i, HT_i might be null when T_i is a relational table for a weak entity
- For a table $T_i \in R$, we define that T_i *has*
  - Let pk^i be the primary key of T_i, pk^i may consists of one or more fields where are called primary key fields.
  - Assumes that $pk^i = \{pk_e^i\}, 1 \le e \le k$, be the set of primary key fields, where k is the number of primary key fields in T_i.
  - A set of non-primary key fields $f^i = \{f_d^i\}$, $k < d \le q + k$, where q is the number of non-primary key fields in T_i. We have $pk^i \cap f^i = \emptyset$
  - T_i has a set of tuples t_l^i. For any t_l^i, t_l^i has primary values $pk_l^i = \{pk_{le}^i\}$ and field values $f_l^i = \{f_{ld}^i\}$

Translation steps:

- For each table $T_i \in R$, an HTable HT_i with a set of row keys rk_l^i is constructed. The construction of HT_i includes two steps, the first step is to define the row key for HT_i and the second step is to define the column family for HT_i, which are described as follows.
  - Step 1: For each primary key $pk_l^i = \{pk_{le}^i\}$ in T^i, take $pk_{l1}^i : pk_{l2}^i : \ldots : pk_{lk}^i$ as the value of the row key rk_l^i in HT_i
  - Step 2: A column family cf^i is constructed for f^i in HT_i, such that for every field $f_d^i \in f^i$, there is a column qualifier cq_d^i in cf^i.
  - For each tuple $t_l^i \in T^i$, a key value pair KVP_{ld}^i will be constructed for each f_{ld}^i of

t_l^i, in which row key equals to t_l^i , f_d^i is the column qualifier and f_{ld}^i is the value.

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:

- For the relational table A, we define that A has
  - Let pk^A be the primary key of A, pk^A may consists of one or more fields where are called primary key fields.
  - Assumes that $pk^A = \{pk_e^A\}, 1 \le e \le k$, be the set of primary key fields, where k is the number of primary key fields in A.
  - A set of non-primary key fields $f^A = \{f_d^A\}$, $k < d \le q + k$, where q is the number of non-primary key fields in A. We have $pk^A \cap f^A = \emptyset$
  - A has a set of tuples t_l^A. For any t_l^A, t_l^A has primary values $pk_l^A = \{pk_{le}^A\}$ and field values $f_l^A = \{f_{ld}^A\}$
- For relational table B, we define that B has
  - Let pk^B be the primary key of B, pk^B may consists of one or more fields where are called primary key fields.
  - A set of non-primary key fields $f^B = \{f_c^B\}$ in B, $z < c \le y + z$, where z is the number of non-primary key fields in B. We have $pk^B \cap f^B = \emptyset$
  - A set of primary fields, $pk^B = \{pk_h^B\}, 1 \le h \le z$, be the set of primary key fields, where z is the number of primary key fields in B
  - B has a set of tuples t_k^B. For any t_k^B, t_k^B has primary values $pk_k^B = \{pk_{kh}^B\}$ and field values $f_k^B = \{f_{kc}^B\}$

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 HA with a set of row keys rk_l^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, which are described as follows.
  - Step 1: For each primary key $pk_l^A = \{pk_{le}^A\}$ in A, take $pk_{l1}^A : pk_{l2}^A : \ldots : pk_{lk}^A$ as the value of row key rk_l^A in HA
  - Step 2: A column family cf^A is constructed for f^A in HA, such that for every field $f_d^A \in f^A$, there is a column qualifier cq_d^A in cf^A.
  - For each tuple $t_l^A \in$ A, a key value pair KVP_{ld}^A will be constructed for each f_{ld}^A of t_l^A, in which row key equals to t_l^A , f_d^A is the column qualifier and f_{ld}^A is the value.

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

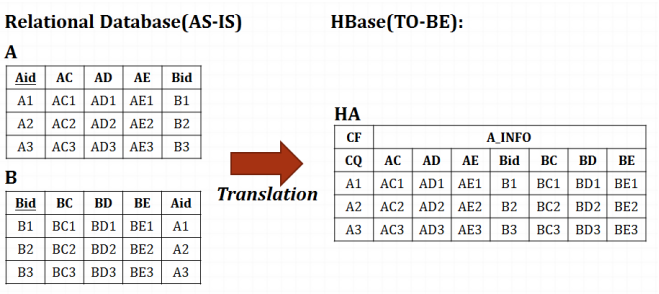

Figure 2. 1-1 translation

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's primary 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 1-1 relationship, 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

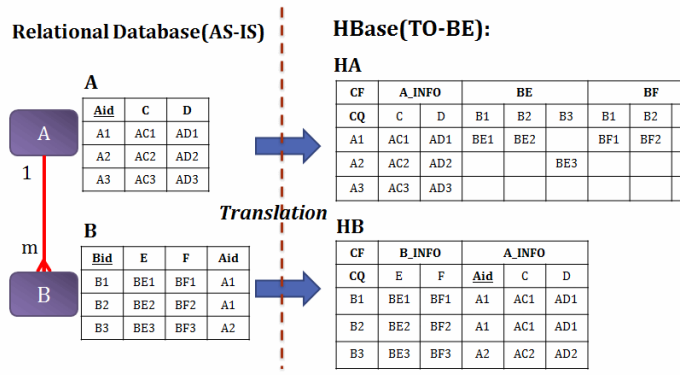

Figure 3. 1- m translation

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 m-side.
- HA and HB is constructed for the mapping HTables in CDB.

Translation steps:

- Execute entity(A) to have HA.
- In HA, a column family cf_{Bc}^{A} is constructed for each B's non-primary key field f_c^B in f^B
- Base on the 1-m relationship, a tuple t_l^A with primary key $pk_l^A = \{pk_{le}^A\}$ in A has a group of tuples $G_l^B \subset B$, such that for any tuple $t_{lk}^B \in G_l^B$, t_{lk}^B has pk_l^A as its foreign key.
  - For each non-primary key field of t_{lk}^B, a key value pair is constructed with pk_{le}^A as its row key, the primary key $pk_{l1}^B : pk_{l2}^B : \ldots : pk_{lz}^B$ as column qualifier in cf_{Bc}^A and take the value of f_c^B as the value.

1-m (A,B):

Assumption:

- Assume A entity and B entity has 1-m relationship, A is the 1-side and B is the m-side.

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

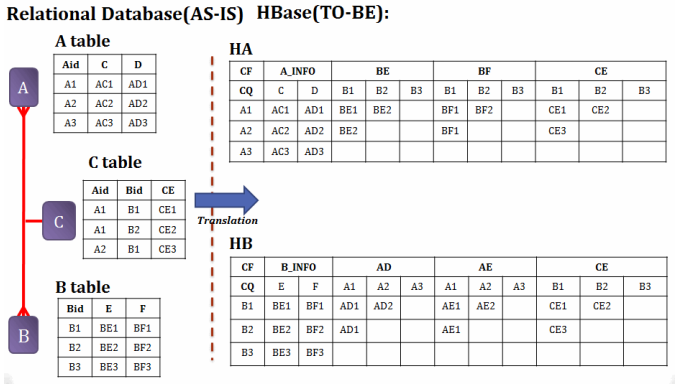

Figure 4. m-n translation

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 m-side 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

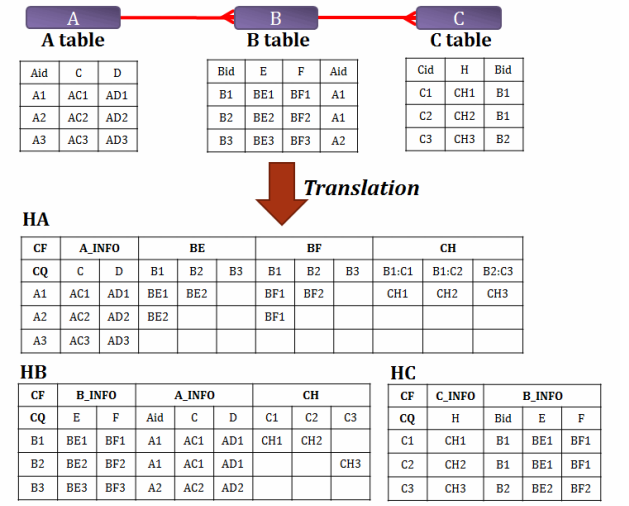

Figure 5. R-R translation

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 $RR_C^A = \{T_r\}, 1 \le r \le n-1$ 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(T_i. D)

  }

  return 1-m(A, D)

3 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.

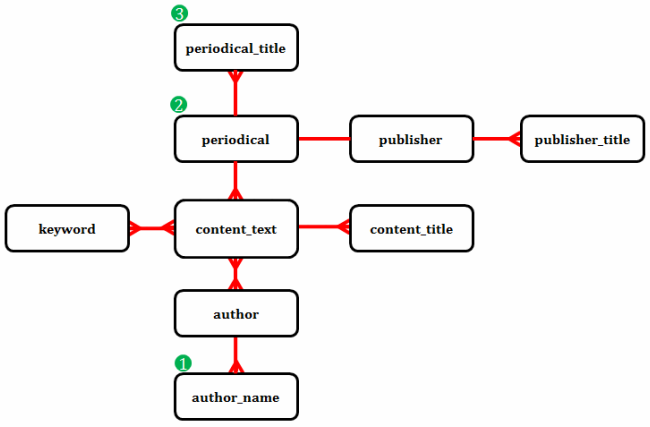

Figure 6. The AJP search engine ER Model

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.

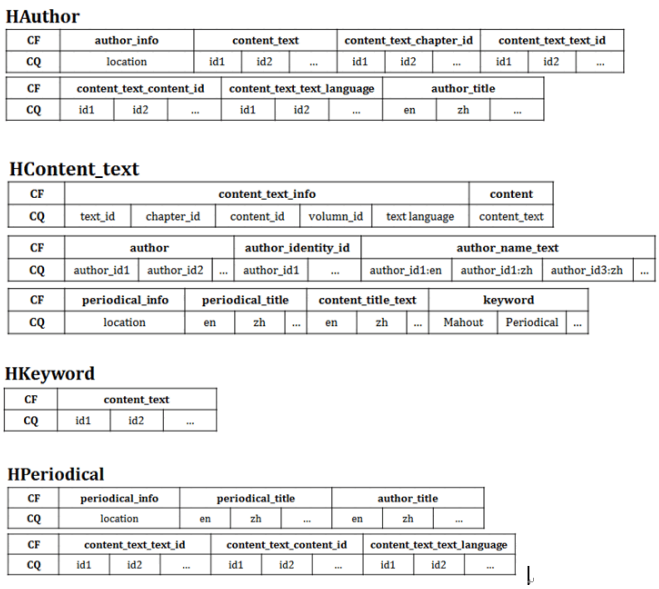

Figure 7. The AJP search engine CDBS

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.

4 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.

5 REFERENCES

1. S. 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/integrating-hive-and-hbase/
6. The Apache HBase Book, http://hbase.apache.org/book.html