SE305 COURSE PROJECT FINAL REPORT

Jeremy Liu, Qianyang Peng, Jingyu Cui

Contents

1 Model Design

Our models were designed with Workbench. The diagrams are much like ER diagrams but the relationships between entities are implicit. Above the basic requirements, our design provided advanced functions includes:

1. We store the label of entities and text of descriptions in **all languages**, not only in English.

2. For claims we not only store the mainsnaks but also store the **qualifiers** of them.

3. For all the snaks and qualifiers, we designed tables to store their **datavalues** for different forms of them.

Our design contains nine tables and satisfies BCNF. Moreover, we used *utf8mb4* dataset to support different language characters.

The ER Diagram is as in Figure 1. Details of the design is included in the nect section.

The programming tools we use:

1. Table design and create schema: Workbench

2. File parsing and data insert: Python3

3. User interface: Apache & PHP

2 Table Designs

We finished out table design according to our model design. Our final design contains 9 tables. Reference and aliases is not stored in our tables, while all other information in different languages are stored. The design of our tables is elaborated below:

2.1 entity

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    entity` (
  `serial_id` BIGINT(32) NOT NULL
      AUTO_INCREMENT,
  `entity_id` VARCHAR(32) NOT NULL,
  `entity_language` VARCHAR(16) NOT NULL
      ,
  `entity_type` VARCHAR(16) NULL DEFAULT
      NULL,
  `entity_text` VARCHAR(255) NULL
      DEFAULT NULL,
  PRIMARY KEY (`serial_id`),
  INDEX `EID` (`entity_id` ASC),
  INDEX `ELANG` (`entity_language` ASC),
  INDEX `ETYPE` (`entity_type` ASC),
  INDEX `ETEXT` (`entity_text` ASC))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **serial_id**: Auto incremental serial ID. As language information is introduced here and entity ID can not be used as Primary Key, so we use this serial_id here as a Primary Key.

- **entity_id**: ID of the corresponding entity. Eg. Q5, P110, Q123423, etc.

- **entity_language**: Language of the label of the corresponding entity. Eg. Zh-cn, en, etc.

- **entity_type**: The entity type identifier. ”item” for data items, and ”property” for properties.

- **entity_text**: Contains the labels in different languages. Eg. square kilometre, Kilometro quadrato, Vierkante kilometer, etc.

Indexed tables: besides indexing the Primary Key serial_id, other keys including entity_id, entity_language and entity_type are also indexed.

2.2 description

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    description` (
  `serial_id` BIGINT(32) NOT NULL
      AUTO_INCREMENT,
  `entity_id` VARCHAR(32) NOT NULL,
  `desc_language` VARCHAR(8) NULL,
  `desc_text` VARCHAR(255) NULL,
  PRIMARY KEY (`serial_id`),
  INDEX `EID` (`entity_id` ASC),
  INDEX `DLANG` (`desc_language` ASC))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **serial_id**: Same as in table ’entity’.

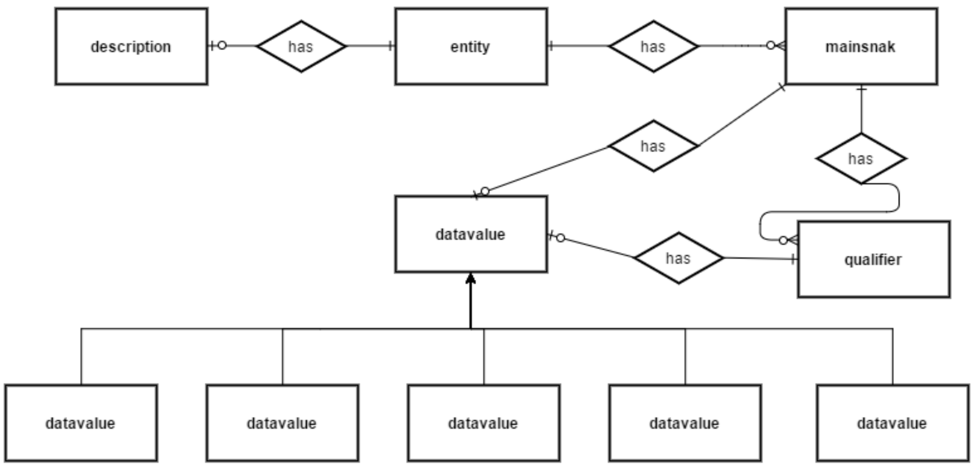

Figure 1: The ER diagram of our design

- **entity_id**: Same as in table 'entity'.
- **desc_language**: Similar to entity_language in table 'entity'.
- **desc_text**: Similar to entity_text in table 'entity'.

Indexed tables: besides indexing the Primary Key serial_id, other keys including entity_id, desc_language and are also indexed.

2.3 mainsnak

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    mainsnak` (
  `snak_id` VARCHAR(64) NOT NULL,
  `entity_id` VARCHAR(32) NOT NULL,
  `property_id` VARCHAR(32) NOT NULL,
  `serial` INT(4) NOT NULL,
  `claimtype` VARCHAR(32) NULL DEFAULT
      NULL,
  `snaktype` VARCHAR(32) NULL DEFAULT
      NULL,
  `datatype` VARCHAR(32) NULL DEFAULT
      NULL,
  `rank` VARCHAR(32) NULL DEFAULT NULL,
  PRIMARY KEY (`snak_id`),
  INDEX `EID` (`entity_id` ASC),
  INDEX `PID` (`property_id` ASC),
  INDEX `CTYPE` (`claimtype` ASC),
  INDEX `STYPE` (`snaktype` ASC),
  INDEX `DTYPE` (`datatype` ASC))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: An arbitrary identifier for the claim, used as Primary Key of this table.
- **entity_id**: ID of the entity this claim belongs to.
- **property_id**: The property this claim is describing.
- **serial**: This field is X means this is the Xth claim of this property.
- **claimtype**: the type of the claim - currently either statement or claim.
- **snaktype**: The type of the snak. Currently, this is one of value, somevalue or novalue.
- **datatype**: The datatype field indicates how the value of the Snak can be interpreted. This fields indicate the concrete table begining with ”datavalue_”
- **rank**: The rank expresses whether this value will be used in queries, and shown be visible per default on a client system. The value is either preferred, normal or deprecated.

2.4 datavalue_string

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    datavalue_string` (
  `snak_id` VARCHAR(64) NOT NULL,
  `value` VARCHAR(255) NULL DEFAULT NULL
    ,
  PRIMARY KEY (`snak_id`))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: Primary key. Refering to the table mainsnak.

- **value**: The content of the string.

2.5 datavalue_time

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    datavalue_time` (
  `snak_id` VARCHAR(64) NOT NULL,
  `time` VARCHAR(64) NULL DEFAULT NULL,
  `timezone` VARCHAR(32) NULL DEFAULT
     NULL,
  `before` VARCHAR(32) NULL DEFAULT NULL
     ,
  `after` VARCHAR(32) NULL DEFAULT NULL,
  `precision` INT(8) NULL DEFAULT NULL,
  `calendarmodel` VARCHAR(255) NULL
     DEFAULT NULL,
  PRIMARY KEY (`snak_id`))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: Primary key. Refering to the table mainsnak.
- **time**: The content of the string.
- **timezone**: Signed integer. Now unused.
- **before**: Begin of an uncertainty range, given in the unit defined by the precision field. This cannot be used to represent a duration.
- **after**: End of an uncertainty range, given in the unit defined by the precision field. This cannot be used to represent a duration.
- **precision**: To what unit is the given date/time significant.
- **calendarmodel**: A URI of a calendar model, such as gregorian or julian. Typically given as the URI of a data item on the repository.

2.6 datavalue_globecoordinate

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    datavalue_globecoordinate` (
  `snak_id` VARCHAR(64) NOT NULL,
  `latitude` FLOAT NULL DEFAULT NULL,
  `longitude` FLOAT NULL DEFAULT NULL,
  `altitude` FLOAT NULL DEFAULT NULL,
  `precision` FLOAT NULL DEFAULT NULL,
  `globe` VARCHAR(255) NULL DEFAULT NULL
     ,
  PRIMARY KEY (`snak_id`))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: Primary key. Refering to the table mainsnak.
- **latitude**: The latitude part of the coordinate in degrees, as a float literal (or an equivalent string).
- **longitude**: The longitude part of the coordinate in degrees, as a float literal (or an equivalent string).
- **altitude**: Deprecated and no longer used. Will be dropped in the future.
- **precision**: the coordinate's precision, in (fractions of) degrees, given as a float literal (or an equivalent string).
- **precision**: To what unit is the given date/time significant.
- **globe**: the URI of a reference globe.

2.7 datavalue_quantity

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    datavalue_quantity` (
  `snak_id` VARCHAR(64) NOT NULL,
  `amount` VARCHAR(64) NULL DEFAULT NULL
     ,
  `upperBound` VARCHAR(64) NULL DEFAULT
     NULL,
  `lowerBound` VARCHAR(64) NULL DEFAULT
     NULL,
  `unit` VARCHAR(64) NULL DEFAULT NULL,
  PRIMARY KEY (`snak_id`))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: Primary key. Refering to the table mainsnak.
- **amount**: The nominal value of the quantity, as an arbitrary precision decimal string. The string always starts with a character indicating the sign of the value, either "+" or "-".
- **upperBound**: Optionally, the upper bound of the quantity's uncertainty interval, using the same notation as the amount field.

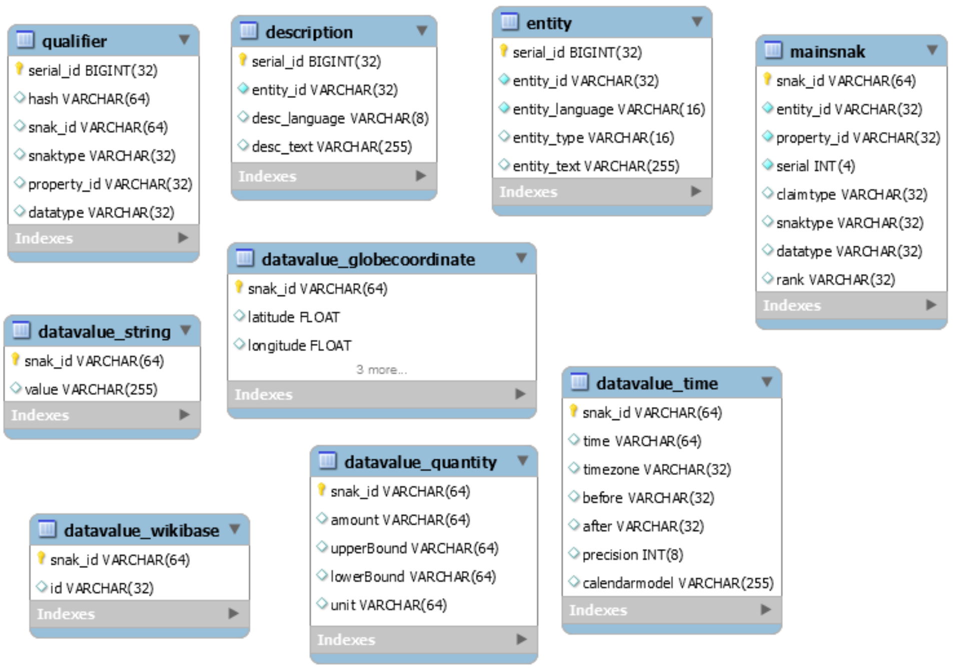

Figure 2: Details of our table design

- **lowerBound**: Optionally, the lower bound of the quantity's uncertainty interval, using the same notation as the amount field.

- **unit**: the URI of a unit (or "1" to indicate a unit-less quantity).

2.8 datavalue_wikibase

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    datavalue_wikibase` (
  `snak_id` VARCHAR(64) NOT NULL,
  `id` VARCHAR(32) NULL DEFAULT NULL,
  PRIMARY KEY (`snak_id`))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **snak_id**: Primary key. Refering to the table mainsnak.

- **id**: defines the id of the entity.

2.9 qualifier

Table schema:

```
CREATE TABLE IF NOT EXISTS `wikidata`.`
    qualifier` (
  `serial_id` BIGINT(32) NOT NULL
      AUTO_INCREMENT,
  `hash` VARCHAR(64) NULL DEFAULT NULL,
  `snak_id` VARCHAR(64) NULL DEFAULT
      NULL,
  `snaktype` VARCHAR(32) NULL DEFAULT
      NULL,
  `property_id` VARCHAR(32) NULL DEFAULT
      NULL,
  `datatype` VARCHAR(32) NULL DEFAULT
      NULL,
  INDEX `STYPE` (`snaktype` ASC),
  INDEX `PID` (`property_id` ASC),
  INDEX `DTYPE` (`datatype` ASC),
  PRIMARY KEY (`serial_id`),
  INDEX `SID` (`snak_id` ASC),
  INDEX `HASH` (`hash` ASC))
ENGINE = InnoDB
```

The data strored in corresponding fields:

- **serial_id**: Primary key.

- **hash**: Hash value of qualifier. Used to build connection between table qualifier and datavalue_ series.

- **snak_id**: The id of the snak this qualifier belongs to. Used to build connection between table qualifier and mainsnak.

- **snaktype**: Similar to the table mainsnak.

- **property_id**: Similar to the table mainsnak.

- **datatype**: Similar to the table mainsnak.

In Figure 2

3 DATA POPULATING

Populating the database - python script:

In order to populate the database, we have written a Python script to read the JSON file line by line and extract the information that we need. The populating process is as follows:

First we extract the entity ID and entity type, followed by the various languages supported, along with their corresponding descriptions. Once we have this information, we are able to populate both the entity and description tables. Using PyMySQL, we can connect to the MySQL database and effectively insert records. This is followed by the iteration of each property within the claims to populate the remaining tables. We extract information from the mainsnak and qualifiers to insert into the mainsnak table and qualifier table respectively.

Then, we deal with the various types of data values that we may encounter in the mainsnaks and qualifiers. This includes string, wikibase-entityid, time, globecoordinate, and quantity types. For each type, we extract the necessary fields needed to populate the corresponding table. We insert these records in the same fashion as before, and continue on to the next iteration.

4 DATABASE OVERVIEW

Due to the constraint of disk size, we only use 100k entities as the set for our experiment. The experiment on the next section is based on the subset below.

Overall statistics:

name	count
entity	999999
item	999323
property	676

Table statistics:

table	row number
datavalue_globecoordinate	201934
datavalue_quantity	204015
datavalue_string	2759499
datavalue_time	463418
datavalue_wikibase	4501342
description	6836149
entity	10430881
mainsnak	7720798
qualifier	2759499

5 QUERY DESIGN

5.1 Query 1

5.1.1 Design

5.1.2 Performance

Sampling 5 examples:

serial	entity	time
1	universe	0.016sec
2	dig	0.015sec
3	flower	0.016sec
3	president	0.016sec
3	like '%mi%'	0.172sec

5.2 Query 2

5.2.1 Design

5.2.2 Performance

Sampling 5 examples:

serial	entity	time
1	universe	0.047sec
2	apple	0.015sec
3	mud	0.016sec
3	fly	0.032sec
3	wind	0.032sec

5.3 Query 3

5.3.1 Design

5.3.2 Performance

Sampling 5 examples:

serial	entity	time
1	universe	0.031sec
2	car	0.032sec
3	noodle	0.031sec
3	dragon	0.094sec
3	china	3.171sec

5.4 Query 4

5.4.1 Design

5.4.2 Performance

Sampling 5 examples:

serial	entity	time
1	gas	0.031sec
2	kill	0.000sec
3	fix	0.000sec
3	dare	0.006sec
3	knock	0.016sec

5.5 Extra Queries

10’ bonus.ŁŁq2

5.5.1 Given an entity, return all the entities belongs to this entity.

6 USER INTERFACE DESIGN

6.1 Query interface

6.2 Q&A interface