Suggested SQL Schema based on Parts Registry DAS & FASTA output

The Parts Registry allows developers to access certain annotations via a DAS server, and FASTA files (more info here: http://partsregistry.org/Registry_API). These sources do not contain all the information available within the database (such as the name of the designer, group, accompanying comments, etc). The information that is available has been split into the following three tables: PARTS_INFO, seq_feat & feat_info.

The illustrative tables below have been populated with data extracted from BioBrick BBa_R0050.

TABLE: parts_info

PI_ID	Part_ID	Status	Part_no	Part_type	Description	Sequence	Size
1	BBa_R0050	A	188	Regulatory	Promoter (HK022 cI regulated)	TTAGGTATTGACTGTACTATCAG TTCCGTCATAATATGAACCATA AGTTCACCAC	188

PI_ID: automatically generated reference number for each row in the parts_info table.
Part_ID: unique identifier for each part in the MIT parts registry.
Status: Single letter denoting the availability of the part (more info: http://partsregistry.org/Help:Availability_and_usefulness)
Part_no: Not sure what this represents.
Part_type: Listed here: http://partsregistry.org/Part_Types
Description: Short description of the part.
Sequence: DNA sequence.
Size: Length of DNA sequence.

Candidate Keys: {PI_ID}, {Part_ID} and also, possibly {Part_no}.
Primary Key: {Part_ID}

TABLE: seq_feat

SF_ID	FI_ID	PI_ID	Feat_ID	Start	End	Orientation	Phase
1	1	1	2012	8	13	Null	-
2	2	1	2013	13	27	Null	-
3	3	1	2014	30	35	Null	-
4	4	1	2015	37	51	Null	-
5	5	1	2018	43	43	Null	-
6	6	1	7066	1	54	Null	-

SF_ID: automatically generated reference number for each row in the seq_feat table.
FI_ID: automatically generated reference number for each row in the feat_info table.
PI_ID: automatically generated reference number for each row in the parts_info table.
Feat_ID: feature reference from the MIT parts registry. The same part can have more than one Feat_ID.
Start: Start site of the feature in the part's sequence.
End: End site of the feature in the part's sequence.
Orientation: The orientation of the feature within the part. It does not always have a value.
Phase: Need to clarify the meaning of this.

Candidate Keys: {SF_ID} and {FI_ID, PI_ID}
Primary Key: {SF_ID}
Foreign Key: {PI_ID}, reference: parts_info{PI_ID}, and {FI_ID}, reference: feat_info{FI_ID}

TABLE: feat_info

FI_ID	Label	Feat_type
1	-35	promoter
2	OR2	RBS
3	-10	promoter
4	OR1	RBS
5	putative	Start
6	BBa_R0050	BioBrick

Note: this is a non-redundant table. Every row will have a unique Label/Feat_type combination.

FI_ID: automatically generated reference number for each row in the feat_info table.
Label: User-generated label for each sequence feature.
Feat_type: Standardised feature type. More info here: http://partsregistry.org/Help:Part_Features

Candidate Key: {FI_ID}, and {Feat_ID}
Primary Key: {FI_ID}

MySQL commands

1. Create toy database

```
CREATE DATABASE toy_database;
```

2. Create tables part_info and seq_feat_all

```
 use toy_database;

  create table parts_info (
  PI_ID int NOT NULL AUTO_INCREMENT,
  Part_ID varchar(20) not null,
  Status char(1) null,
  Part_no int null,
  Part_type varchar(50) null,
  Description varchar(500) null,
  Sequence varchar(10000) null,
  Size int null,
  PRIMARY KEY (PI_ID)
   );

create table seq_feat_all (
  PI_ID int not null,
  Feat_ID int not null,
  Label varchar(50) null,
  Feat_type varchar(50) null,
  Start int null,
  End int null,
  Orientation int null,
  Phase char(1) null,
  PRIMARY KEY (Feat_ID),
  FOREIGN KEY (PI_ID) REFERENCES parts_info(PI_ID)
  );
```

3. Import data from partsinfo.csv and seqfeatall.csv

```
LOAD DATA LOCAL INFILE '/path to file/partsinfo.csv'
INTO TABLE parts_info
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(Part_ID,Status,Part_no,Part_type,Description,Sequence,Size
);

LOAD DATA LOCAL INFILE '/path to file /seqfeatall.csv'
INTO TABLE seq_feat_all
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(PI_ID,Feat_ID,Label,Feat_type,Start,End,Orientation,Phase
);
```

4. Create feat_info from distinct values of seq_feat_all columns: Label & Feat_type

```
CREATE TABLE feat_info (
        FI_ID int NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (FI_ID)
        )
        AS (SELECT DISTINCT Label, Feat_type
        FROM seq_feat_all);
```

5. Combine FI_ID from feat_info, with columns from seq_feat_all to create table seq_feat, and add SF_ID.

```
CREATE TABLE seq_feat (
        SF_ID int NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (SF_ID),
        FOREIGN KEY (PI_ID) REFERENCES parts_info(PI_ID),
        FOREIGN KEY (FI_ID) REFERENCES feat_info(FI_ID)
        )
        AS (
        SELECT feat_info.FI_ID, seq_feat_all.PI_ID, seq_feat_all.Feat_ID, seq_feat_all.Start, seq_feat_all.End,
        seq_feat_all.Orientation, seq_feat_all.Phase
        FROM feat_info, seq_feat_all
        WHERE feat_info.Label = seq_feat_all.Label
        AND feat_info.Feat_type = seq_feat_all.Feat_type
        );
```

6. Delete seq_feat_all table

```
DROP TABLE seq_feat_all;
```