MGI Schema Design

Author: Lori Corbani

Created: September 24, 2004

Last Modified: September 23, 2005 09:34

1 Purpose of Document

Schema design for MGI.

2 Standards

All tables:

1. Contain these attributes:
   ```
   _CreatedBy_key            int not null
   _ModifiedBy_key           int not null
   ```
   These attributes are foreign keys to *MGI_User._User_key.*
2. Contain these attributes:
   ```
   creation_date           datetime not null
   modification_date       datetime not null
   ```
3. Bind *current_date* to *creation_date* and *modification_date*.
4. Have a nonclustered, nonunique index on *modification_date*.
5. All primary keys have a unique clustered index.
6. All foreign keys have a non-unique, nonclustered index (unless otherwise indicated).

3 References

3.1 BIB_Refs

A record in this table represents a journal article, book or personal communication.

```
_Refs_key           int not null
_ReviewStatus_key   int not null
refType             char(4) not null
authors             varchar(255) null
authors2            varchar(255) null
_primary            varchar(60) null
title               varchar(255) null
title2              varchar(255) null
journal             varchar(100) null
vol                 varchar(20) null
issue               varchar(25) null
date                varchar(30) null
```

```
year                int null
pgs                 varchar(30) null
NLMstatus           char(1) not null
abstract            text null
isReviewArticle     bit not null
```

1. Primary key is *_Refs_key*.
2. Foreign key on *_ReviewStatus_key* to *BIB_ReviewStatus*.
3. Valid values for *refType* are *ART*, *BOOK*.
4. Valid values for *NLMstatus* (NLM = National Library of Medicine) are:
   - N = No, the journal of this reference is not in NLM.
   - Y = Yes, the journal of this reference is in NLM.
   - X = Never, the journal of this reference will never be in NLM (examples are *Mouse Genome* and *Mouse News Lett*).
5. If *isReviewArticle* = 1, then the reference refers to a Review Article.

3.2 BIB_Books

A record in this table represents the Book attributes of a *BIB_Refs* record. This table is only populated if *BIB_Refs.refType = BOOK*.

```
_Refs_key           int not null
book_au             varchar(160) null
book_title          varchar(200) null
place               varchar(50) null
publisher           varchar(50) null
series_ed           varchar(50) null
```

1. Primary key is *_Refs_key*.
2. Foreign key on *_Refs_key* to *BIB_Refs*.

3.3 BIB_Notes

A record in this table represents a 255-character segment of a Reference's note.

```
_Refs_key           int not null
sequenceNum         int not null
note                char(255) not null
```

1. Primary key is *_Refs_key* + *sequenceNum*.
2. Non-unique index on *sequenceNum*.
3. Foreign key on *_Refs_key* to *BIB_Refs*.
4. To retrieve the entire note for a Reference, search for all *BIB_Notes.note* where *_Refs_key* = your Reference, order by *sequenceNum* and concatenate all of the *note* values.

3.4 BIB_DataSet

A record in this table represents a MGI DataSet. Each DataSet corresponds to a curatorial area within MGI and enables us to index References to particular curatorial areas.

```
_DataSet_key         int not null
dataSet              varchar(255) not null
abbreviation         varchar(15) not null
inMGIprocedure       varchar(255) null
sequenceNum          int not null
isObsolete           bit not null
```

1. Primary key is *_DataSet_key*.
2. Data Sets (examples):
   - Molecular Segments
   - Mapping
   - Gene Ontology
3. *inMGIprocedure*, if non-null, is the name of a stored procedure which is used to determine if a given Reference (_Refs_key) is indexed to this DataSet.

3.5 BIB_DataSet_Assoc

A record in this table represents the association between a DataSet and a Reference.

```
_Assoc_key           int not null
_Refs_key            int not null
_DataSet_key         int not null
isNeverUsed          bit not null
```

1. Primary key is *_Assoc_key*.
2. Unique index on *_Refs_key* + *_DataSet_key*.
3. Foreign key on *_Refs_key* to *BIB_Refs*.
4. Foreign key on *_DataSet_key* to *BIB_DataSet*.
5. If *isNeverUsed* = 1, then the reference is indexed to the DataSet but the reference wound up not being used to curate data in the area of MGI.

3.6 BIB_ReviewStatus

A record in this table represents a Review Status vocabulary term.

```
_ReviewStatus_key    int not null
name                 varchar(40) not null
```

1. Primary key is *_ReviewStatus_key*.
2. Review Status terms (examples):
   - Unreviewed
   - Reviewed by MGI Editorial Staff

- Peer Reviewed

4 Reference Associations

A generalized design that allows one to specify any number of References for a database object that has a non-composite primary key.

Implemented for:

- Nomenclature Markers
- Markers
- Alleles
- Image Annotations to Alleles and Genotypes
- Sequences
- Strains

4.1 MGI_Reference_Assoc

A record in this table repesents an association between a Reference of a specific type (*_RefAssocType_key*, e.g. *Original*) and a specific database object (*_Object_key*, e.g. *Sequence "A12345"*) of a specific object type (*_MGIType_key*, e.g., *Sequence*, *Marker/Sequence Association*).

```
_Assoc_key          int not null
_Refs_key           int not null
_Object_key         int not null
_MGIType_key        int not null
_RefAssocType_key   int not null
```

1. Primary key is *_Assoc_key*.
2. Unique index on *_Refs_key* + *_Object_key* + *_MGIType_key* + *_RefAssocType_key*.
3. Foreign key on *_Refs_key* to *BIB_Refs*.
4. Foreign key on *_MGIType_key* to *ACC_MGIType*.
5. Foreign key on *_RefAssocType_key* to *MGI_RefAssocType*.

4.2 MGI_RefAssocType

A record in this table represents a Reference Association Type (e.g. *General*, *Original*).

```
__RefAssocType_key      int not null
_MGIType_key            int null
assocType               varchar(255) not null
allowOnlyOne            bit not null
```

1. Primary key is *_RefAssocType_key*.
2. Foreign key on *_MGIType_key* to *ACC_MGIType*.

3. If *_MGIType_key = null*, then the Reference Assocation Type can be used for a Reference Assocation to any type of database object. If *_MGIType_key* is not *null*, then the Reference Assocation Type can only be used for a Reference Assocation to an object of that type.
4. If *allowOnlyOne* = true (1), then at most one Reference of this Association Type may be associated to a given object.

5 Markers

5.1 MRK_Marker

A record in this table represents a Marker.

```
_Marker_key         int not null
_Organism_key       int not null
_Marker_Status_key  int not null
_Marker_Type_key    int not null
_CurationState_key  int not null
symbol              varchar(50) not null
name                varchar(255) not null
chromosome          varchar(8) not null
cytogeneticOffset   varchar(20) null
```

1. Primary key is *_Marker_key*.
2. Non-unique index on *symbol*.
3. Non-unique index on *chromosome*.
4. Foreign key on *_Organism_key* to *MGI_Organism*.
5. Foreign key on *_Marker_Status_key* to *MRK_Status*.
6. Foreign key on *_Marker_Type_key* to *MRK_Types*.
7. Foreign key on *_CuraitonState_key* to *VOC_Term*.

5.2 MRK_History

A record in this table represents a nomenclature event in the life of a Marker. A nomenclature event is defined by the event itself (rename, merge, split, etc.), the reason for the event, date of the event, reference (J:).

```
_Marker_key              int not null
_Marker_Event_key        int not null
_Marker_EventReason_key  int not null
_History_key             int not null
_Refs_key                int not null
sequenceNum              int not null
name                     varchar(255) not null
event_date               datetime not null
```

1. Primary key is *_Marker-key, _History_key, sequenceNum*.

2. Foreign key on *_Marker_key* to *MGI_Marker*.
3. Foreign key on *_Marker_Event_key* to *MRK_Event*.
4. Foreign key on *_Marker_EventReason_key* to *MRK_EventReason*.
5. Foreign key on *_History_key* to *MRK_Marker*.
6. Foreign key on *_Refs_key* to *BIB_Refs*.

5.3 MRK_Alias

A record in this table represents the relationship between two Mouse Markers (usually used to establish a relationship between a Gene and a DNA segment).

```
_Alias_key      int not null
_Marker_key     int not null
```

1. Primary key is *_Alias_key* + *_Marker_key*
2. Foreign key on *_Alias_key* to *MRK_Marker*.
3. Foreign key on *_Marker_key* to *MRK_Marker*.

5.4 MRK_Anchors

A record in this table represents a Mouse Marker which is an Anchor Marker for the specified Chromosome. All Anchor Markers for a specified Chromosome are used to build a WI Marker Detail Mini-Map for that Chromosome.

```
chromosome      varchar(8) not null
_Marker_key     int not null
```

1. Primary key is *_Marker_key*
2. Foreign key on *_Marker_key* to *MRK_Marker*.

5.5 MRK_Chromosome

A record in this table represents an ordered Chromosome for the specified Organism.

```
_Chromosome_key  int not null
_Organism_key    int not null
chromosome       varchar(8) not null
sequenceNum      int not null
```

1. Primary key is *_Chromosome_key*.
2. Non-unique index on *chromosome*.
3. Foreign key on *_Organism_key* to *MGI_Organism*.

5.6 MRK_Class

A record in this table represents a Marker Class (MLC) vocabulary term.

```
_Class_key      int not null
name            varchar(255) not null
```

1. Primary key is *_Class_key.*
2. Non-unique index on *name*.

5.7 MRK_Classes

A record in this table represents the relationship between a Marker and a MLC Marker Class.

```
_Marker_key     int not null
_Class_key      int not null
```

1. Primary key is *_Marker_key* + *_Class_key.*
2. Foreign key on *_Marker_key* to *MRK_Marker.*
3. Foreign key on *_Class_key* to *MRK_Class*.

5.8 MRK_Current

A record in this table represents the relationship between a Marker and its current Marker symbol.

```
_Current_key    int not null
_Marker_key     int not null
```

1. Primary key is *_Current_key* + *_Marker_key.*
2. Foreign key on *_Current_key* to *MRK_Marker.*
3. Foreign key on *_Marker_key* to *MRK_Marker.*

5.9 MRK_Event

A record in this table represents a Marker Nomenclature Event vocabulary term.

```
_Marker_Event_key      int not null
event                  varchar(255) not null
```

1. Primary key is *_Marker_Event_key.*
2. Non-unique index on *event*.

5.10 MRK_EventReason

A record in this table represents a Marker Nomenclature Event Reason vocabulary term.

```
_Marker_EventReason_key   int not null
eventReason               varchar(255) not null
```

1. Primary key is *_Marker_EventReason_key.*
2. Non-unique index on *eventReason*.

5.11 MRK_Label

A record in this table represents a Marker Label (current symbol, current name, synonym, orthologous symbol, etc.). This table is a cache table of all “labels” for a given Marker...representing the different ways in which a user may search for a particular Marker. This table is loaded via the *mrklabelload* product.

```
_Marker_key              int not null
_Label_Status_key        int not null
_Organism_key            int not null
_OrthologOrganism_key    int null
priority                 int not null
label                    varchar(255) not null
labelType                varchar(5) not null
labelTypeName            varchar(255) not null
```

1. Primary key is *_Marker_key* + *priority* + *_OrthologOrganism_key*, *labelType*, *label*.
2. Foreign key on *_Marker_key* to *MRK_Marker.*
3. Foreign key on *_Organism_key* to *MGI_Organism*.
4. Foreign key on *_OrthologOrganism_key* to *MGI_Organism*.
5. Values for *_Label_Status_key:*:
   - label is current.
   - label is old (a synonym or former label).
6. *_Organism_key* represents the organism of the *_Marker_key.*
7. *_OrthologOrganism_key* represents the orgranism of the *label*, if different than the organism of the Marker.
8. *priority* represents the priority of this label with regard to other labels. This is used to sort the results of a query. The lower the priority number, the higher the priority of that label result.
9. *label* is the Marker's label derived from *MRK_Marker.symbol*, *MRK_Marker.name*, *MRK_Other.name*, etc.
10. *labelType* is the type of *label*. Valid values:
    - MS = marker symbol
    - MN = marker name
    - AS = allele symbol
    - AN = allele name
    - MY = marker synonym
    - OS = ortholog symbol
    - ON = ortholog name
11. *labelTypeName* is description that gets printed in the WI summary page when a label is returned from a query. Valid values:
    - current symbol
    - current name
    - allele symbol
    - allele name

- old symbol
- old name
- synonym
- human synonym
- related synonym
- <common organism name> + ortholog symbol
- <common organism name> + ortholog name

Table 1:

status	organism	ortholog organism	priority	labelType	labelTypeName
1	mouse		1	MS	current symbo
1	mouse		2	MN	current name
1	mouse		3	AS	allele symbol
1	mouse		4	AN	allele name
2	mouse		5	MS	old symbol
2	mouse		6	MN	old name
1	mouse		7	MY	synonym
1	mouse	human	8	MY	human synonym
1	mouse	rat	9	MY	rat synonym
1	mouse		10	MY	related synonym
1	mouse	human	11	OS	ortholog symbol
1	human		11	MS	current symbol
1	mouse	human	12	ON	ortholog name

Table 1:

status	organism	ortholog organism	priority	labelType	labelTypeName
1	human		12	MN	current name
1	mouse	rat	13	OS	ortholog symbol
1	rat		13	MS	current symbol
1	rat		13	MN	current name
1	mouse	sheep, cow, etc.	14	OS	ortholog symbol
1	sheep, cow, etc.		14	MS	current symbol
1	sheep, cow, etc.		14	MN	current name

5.12 MRK_Note

A record in this table represents a 255-character segment of a Marker's note.

```
_Marker_key          int not null
sequenceNum          int not null
note                 char(255) not null
```

1. Primary key is *_Marker_key* + *sequenceNum*.
2. Non-unique index on *sequenceNum*.
3. Foreign key on *_Marker_key* to *MRK_Marker.*
4. To retrieve the entire note for a Marker, search for all *MRK_Note.note* where *_Marker_key* = your Marker, order by *sequenceNum* and concatenate all of the *note* values.

5.13 MRK_Offset

A record in this table represents a centiMorgan (cM) position for a specific Marker from a given source: MGI (0), Chromosome Committee (1), or MIT (2). A value of -1.0 designates a syntenic Marker. A value of -999.99 designates a withdrawn Marker. Only mouse Markers have cM offsets.

```
_Marker_key          int not null
```

```
source                int not null
offset                float not null
```

1. Primary key is *_Marker_key* + *source*.
2. Non-unique index on *source*.
3. Non-unique index on *offset*.
4. Foreign key on *_Marker_key* to *MRK_Marker.*

5.14 MRK_Reference

A record in this table represents a unique Marker/Reference pair. This is table is a cache table of all Marker/Reference pairs in MGI, derived from MGI_Reference_Assoc (where *_MGIType_key* = 2), Molecular Segments, Orthology, GXD, Mapping, etc.

```
_Marker_key           int not null
_Refs_key             int not null
```

1. Primary key is *_Marker_key* + *_Refs_key.*
2. Foreign key on *_Marker_key* to *MRK_Marker.*
3. Foreign key on *_Refs_key* to *BIB_Refs*.

5.15 MRK_Status

A record in this table represents a Marker Status vocabulary term.

```
_Marker_Status_key     int not null
status                 varchar(255) not null
```

1. Primary key is *_Marker_Status_key.*
2. Non-unique index on *status*.

5.16 MRK_Types

A record in this table represents a Marker Type vocabulary term.

```
_Marker_Type_key      int not null
name                  varchar(255) not null
```

1. Primary key is *_Marker_Type_key.*
2. Non-unique index on *name*.

6 Nomenclature

6.1 NOM_Marker

A record in this table represents a specific Nomen object.

```
_Nomen_key              int not null
_Marker_Type_key        int not null
_NomenStatus_key        int not null
```

```
_Marker_Event_key        int not null
_Marker_EventReason_key  int not null
_CurationState_key       int not null
symbol                   varchar(25) not null
name                     varchar(255) not null
chromosome               varchar(8) not null
humanSymbol              varchar(25) null
statusNote               varchar(255) null
broadcast_date           datetime null
_BroadcastBy_key         int null
```

1. Primary key is *_Nomen_key*.
2. Non-unique indexes on *symbol*, *createdBy*, *broadcast_date*.
3. Foreign key on *_Marker_Type_key* to *MRK_Types*.
4. Foreign key on *_NomenNomenStatusStatus_key* to *VOC_Term*.
5. Foreign key on *_Marker_Event_key* to *MRK_Event* (table shared with *MRK_Marker*).
6. Foreign key on *_Marker_EventReason_key* to *MRK_EventReason* (table shared with *MRK_Marker*).
7. Foreign key on *_CurationState_key* to *VOC_Term*.
8. Foreign key on *_BroadcastBy_key* to *MGI_User*.

7 Alleles

7.1 ALL_Allele

A record in this table represents an Allele of a specific Marker.

```
_Allele_key          int not null
_Marker_key          int null
_Strain_key          int not null
_Mode_key            int not null
_Allele_Type_key     int not null
_Allele_Status_key   int not null
_ESCellLine_key      int not null
_MutantESCellLine_key int not null
symbol                varchar(60) not null
name                  varchar(255) not null
nomenSymbol           varchar(50) null
isWildType            bit not null
_ApprovedBy_key       int null
approval_date         datetime null
```

1. Primay key is *_Allele_key*.
2. Foreign key on *_Marker_key* to *MRK_Marker*.
3. Foreign key on *_Strain_key* to *PRB_Strain*.
4. Foreign key on *_Mode_key* to *VOC_Term* (“Allele Inheritance Mode”)

5. Foreign key on *_Allele_Type_key* to *VOC_Term* (“Allele Type”)
6. Foreign key on *_Allele_Status_key* to *VOC_Term* (“Allele Status”)
7. Foreign key on *_ESCellLine_key* to *ALL_CellLine*.
8. Foreign key on *_MutantESCellLine_key* to *ALL_CellLine*.
9. Foreign key on *_ApprovedBy_key* to *MGI_User*.
10. *_Marker_key* specifies the Marker of the Allele. If a non-broadcast Marker symbol is used, then *_Marker_key* is null and *nomenSymbol* holds the non-broadcast Marker symbol.
11. If *isWildType* = 1, then the Allele is a wild type allele (typically A<+> or <+>).

7.2 ALL_Allele_Mutation

A record in this table represents a Molecular Mutation for a specific Allele. An Allele may have 1 or more Molecular Mutations.

```
_Allele_key          int not null
_Mutation_key        int not null
```

1. Primary key is *_Allele_key* + *_Mutation_key*.
2. Foreign key on *_Allele_key* to *ALL_Allele*.
3. Foreign key on *_Mutation_key* to *VOC_Term* (“Allele Molecular Mutation”).

7.3 ALL_CellLine

A record in this table represents a parental or a mutant ES Cell Line.

```
_CellLine_key          int not null
cellLine               varchar(255) not null
_Strain_key            not null
provider               varchar(50) null
isMutant               bit not null
```

1. Primary key is *_CellLine_key*.
2. Foreign key on *_Strain_key* to *PRB_Strain*.
3. *provider* specifies the provider of the mutant ES cell line. For example, Bay Genomics, Lexicon Genetics.
4. If *isMutant* = 1, then *provider* is not null and the ES cell line is a mutant ES cell line.

7.4 ALL_Label

A record in this table represents an Allele Label (current symbol, current name, synonym). This table is a cache table of all “labels” for a given Allele...representing the different ways in which a user may search for a particular Allele. This table is loaded via the *alllabelload* product.

```
_Allele_key          int not null
_Label_Status_key    int not null
priority             int not null
```

```
label                varchar(255) not null
labelType            varchar(5) not null
labelTypeName        varchar(255) not null
```

1. Primary key is *_Allele_key* + *priority* + *labelType* + *label*.
2. Foreign key on *_Allele_key* to *ALL_Allele*.

8 Molecular Segments

8.1 PRB_Probe

A record in this table represents a Molecular Segment or Primer.

```
name                 varchar(40) not null
derivedFrom          int null
_Source_key          int not null
_Vector_key          int not null
_SegmentType_key     int not null
primer1sequence      varchar(80) null
primer2sequence      varchar(80) null
regionCovered        varchar(255) null
insertSite           varchar(30) null
insertSize           varchar(30) null
productSize          varchar(40) null
```

1. Primary key is *_Probe_key.*
2. Non-unique index on *name*.
3. Foreign key on *_Source_key* to *PRB_Source*.
4. Foreign key on *_Vector_key* to *VOC_Term*.
5. Foreign key on *_SegmentType_key* to *VOC_Term*.
6. *name, regionCovered* are relevant for all Molecular Segments.
7. *derivedFrom*, *_Vector_key*, *_SegmentType_key*, *insertSite*, *insertSize* are only relevant for non-Primers.
8. *primer1Sequence*, *primer2sequence*, *productSize* are only relevant for Primers.
9. *_Source_key* resolves to *Not Applicable* for Primer.

8.2 PRB_Source

A record in this table represents a Molecular Source; the biological context from which a Molecular Segment is derived. A Molecular Segment has at most one Molecular Source.

```
_Source_key          int not null
_SegmentType_key     int not null
_Vector_key          int not null
_Organism_key        int not null
_Strain_key          int not null
_Tissue_key          int not null
```

```
_Gender_key         int not null
_CellLine_key       int not null
_Refs_key           int not null
name                varchar(255) null
description         varchar(255) null
age                 varchar(50) not null
ageMix              float not null
ageMax              float not null
isCuratorEdited     bit not null
```

1. Primary key is *_Source_key.*
2. Non-unique index on *name*.
3. Foreign key on *_Vector_key* to *VOC_Term*.
4. Foreign key on *_SegmentType_key* to *VOC_Term*.
5. Foreign key on *_Organism_key* to *MGI_Organism*.
6. Foreign key on *_Strain_key* to *PRB_Strain*.
7. Foreign key on *_Tissue_key* to *PRB_Tissue*.
8. Foreign key on *_Gender_key* to *VOC_Term*.
9. Foreign key on *_CellLine_key* to *VOC_Term*.
10. Foreign key on *_Refs_key* to *BIB_Refs*.
11. If *isCuratorEdited* is true, then an MGI Curator has edited some attribute of the Source record.
12. If *name* is null, then the Source is considered an *Anonymous* Source. Anonymous Sources can be shared across multiple Molecular Segments iff *isCuratorEdited* is false.
13. If *name* is not null, then the Source is considered a Clone Library. Clone Library sources are shared across multiple Molecular Segments.

8.3 PRB_Alias

A record in this table represents a Molecular Segment synonym froma specific Reference.

```
_Alias_key          int not null
_Reference_key      int not null
alias               varchar(30) not null
```

1. Primary key is *_Alias_key.*
2. Non-unique index on *alias*.
3. Foreign key on *_Reference_key* to *PRB_Reference*.

8.4 PRB_Notes

A record in this table represents a 255-character segment of a Molecular Segment's note.

```
_Probe_key          int not null
```

```
sequenceNum          int not null
note                 char(255) not null
```

1. Primary key is *_Probe_key* + *sequenceNum*.
2. Non-unique index on *sequenceNum*.
3. Foreign key on *_Probe_key* to *PRB_Probe*.
4. To retrieve the entire note for a Molecular Segment, search for all *PRB_Notes.note* where *_Probe_key* = your Molecular Segment, order by *sequenceNum* and concatenate all of the *note* values.

8.5 PRB_Marker

A record in this table represents the relationship between a Molecular Segment and a Marker.

```
_Probe_key           int not null
_Marker_key          int not null
_Refs_key            int not null
relationship         char(1) null
```

1. Primary key is *_Probe_key* + *_Marker_key.*
2. Non-unique index on *relationship*.
3. Foreign key on *_Marker_key* to *MRK_Marker.*
4. Foreign key on *_Refs_key* to *BIB_Refs*.
5. The *relationship* field stores these values:
   - E = encodes (manually curated or computatinally derived via *autoE*)
   - H = hybridizes (manually curated)
   - P = putative (computationally derived)
   - A = amplifies (primers only; manually curated)
   - M = MIT primers (computationally derived)

8.6 PRB_Reference

A record in this table represents a relationship between a Molecular Segment and a Reference. A Molecular Segment may have one or more Reference associations.

```
_Reference_key       int not null
_Probe_key           int not null
_Refs_key            int not null
hasRmap              bit
hasSequence          bit
```

1. Primary key is *_Reference_key.*
2. Foreign key on *_Probe_key* to *PRB_Probe*.
3. Foreign key on *_Refs_key* to *BIB_Refs*.

8.7 PRB_Ref_Notes

A record in this table represents a 255-character segment of a Molecular Segment Reference's note.

```
_Reference_key       int not null
sequenceNum          int not null
note                 char(255) not null
```

1. Primary key is *_Reference_key* + *sequenceNum*.
2. Non-unique index on *sequenceNum*.
3. Foreign key on *_Reference_key* to *PRB_Reference*.
4. To retrieve the entire note for a Molecular Segment's Reference, search for all *PRB_Reference.note* where *_Reference_key* = your Reference record, order by *sequenceNum* and concatenate all of the *note* values.

8.8 PRB_RFLV

A record in this table represents the relationship between a Marker and an endonuclease for a specific Molecular Segment Reference. Variations exist between individuals in DNA fragment sizes cut by specific restriction enzymes; polymorphic sequences that result in RFLVs are used as markers on both physical maps and genetic linkage maps. RFLVs are usually caused by mutation at a cutting site.

```
_RFLV_key            int not null
_Reference_key       int not null
_Marker_key          int not null
endonuclease         varchar(15)null
```

1. Primary key is *_RFLV_key*.
2. Foreign key on *_Reference_key* to *PRB_Reference*.
3. Foreign key on *_Marker_key* to *MRK_Marker*.

8.9 PRB_Allele

A record in this table represents the relationship between a Marker/Endonuclease pair (*PRB_RFLV*) and an allele "symbol"/fragment size.

```
_Allele_key          int not null
_RFLV_key            int not null
allele               varchar(30) not null
fragments            varchar(255) not null
```

1. Primary key is *_Allele_key*.
2. Foreign key on *_RFLV_key* to *PRB_RFLV*.

8.10 PRB_Allele_Strain

A record in this table represents the relationship between a defined RFLV Allele (*PRB_Allele*) and a Strain. An RFLV allele may have one or more Strains associated with it.

```
_Allele_key         int not null
_Strain_key         int not null
```

1. Primary key is *_Allele_key* + *_Strain_key.*
2. Foreign key on *_Allele_key* to *PRB_Allele*.
3. Foreign key on *_Strain_key* to *PRB_Strain*.

8.11 PRB_Tissue

A record in this table represents a Tissue.

```
_Tissue_key         int not null
tissue              varchar(80) not null
standard            bit not null
```

1. Primary key is *_Tissue_key.*
2. Non-unique index on *tissue*.

9 Sequences

9.1 SEQ_Sequence

A record in this table represents a specific Sequence object.

```
_Sequence_key           int not null
_SequenceType_key       int not null
_SequenceQuality_key    int not null
_SequenceStatus_key     int not null
_SequenceProvider_key   int not null
length                  int null
description             varchar(255) null
version                 varchar(15) null
division                char(3) null
virtual                 bit
rawType                 varchar(15) null
rawLibrary              varchar(255)
rawOrganism             varchar(255)
rawStrain               varchar(255)
rawTissue               varchar(255)
rawAge                  varchar(100)
rawSex                  varchar(100)
rawCellLine             varchar(100)
numberOfOrganisms       int null
seqrecord_date          datetime
sequence_date           datetime
```

1. Primary key is *_Sequence_key.*
2. Foreign key on *_SequenceType_key* to *VOC_Term* (a term in the Sequence Type CV).
3. Foreign key on *_SequenceQuality_key* to *VOC_Term* (a term in the Sequence Quality CV).
4. Foreign key on *_SequenceStatus_key* to *VOC_Term* (a term in the Sequence Status CV).
5. Foreign key on *_SequenceProvider_key* to *VOC_Term* (a term in the Sequence Provider CV).
6. If *virtual* = false (0), then the sequence is "real".
7. *numberOfOrganisms* is the count of the number of non-mouse, non-human and non-rat organisms associated with the Sequence.
8. The *seqrecord_date* stores the date of the most recent modification by the data provider of the entire Sequence record. This may or may not indicate a change has been made to the Sequence itself.
9. The *sequence_date* stores the date of the most recent modification by the data provider of the Sequence itself (ACGT...).

9.2 SEQ_Source_Assoc

A record in this table represents the relationship between a Sequence and a Molecular (Probe) Source. This enables us to represent Sequences which have multiple Organisms by associating the Sequence with more than one Molecular Source (which, in turn, have different Organisms).

```
_Assoc_key            int not null
_Sequence_key         int not null
_Source_key           int not null
```

1. Primary key is *_Assoc_key.*
2. Unique index on *_Sequence_key* + *_Source_key.*
3. Foreign key on *_Sequence_key* to *SEQ_Sequence*.
4. Foreign key on *_Source_key* to *PRB_Source*.

9.3 SEQ_Coord_Cache

The purpose of this table is to provide the WI with one table from which it can retrieve the information necessary for the Marker detail page and the Sequence detail page.

A record in this table represents the map coordinates of a Sequence from a specific genomic sequence provider data set (Map). The CVS product *seqcacheload* re-loads this table on a regular basis (for example, whenever a new genomic sequence provider data set is loaded).

```
_Map_key              int not null
_Sequence_key         int not null
chromosome            varchar(8) null
startCoordinate       float not null
```

```
endCoordinate       float not null
strand              char(1) not null
mapUnits            varchar(50) not null
provider            varchar(255) not null
version             varchar(255) null
```

1. Primary key is *_Map_key*, *_Sequence_key*.
2. Unique index on *_Map_key*, *_Sequence_key*.
3. Non-unique indexes on *_Map_key*, *_Sequence_key*.
4. Foreign key on *_Map_key* to *MAP_Coordinate*.
5. Foreign key on *_Sequence_key* to *SEQ_Sequence*.
6. *chromosome* is derived from *MAP_Coordinate._Object_key* where *_MGIType_key* = Chromosome.
7. *startCoordinate* is *MAP_Coord_Feature.startCoordinate*.
8. *endCoordinate* is *MAP_Coord_Feature.endCoordinate*.
9. *strand* is *MAP_Coord_Feature.strand*.
10. *mapUnits* is *MAP_Coordinate._Units_key*.
11. *provider* is *SEQ_Sequence._SequenceProvider_key*.
12. *version* is *MAP_Coordinate.version*; the version of the Assembly Build.

9.4 SEQ_Description_Cache

A record in this table represents the description of a Sequence that is annotated to a mouse Marker or a mouse Molecular Segment. This table is loaded via the *seqcacheload* product.

```
_Sequence_key           int not null
description             varchar(255) not null
```

1. Primary key is *_Sequence_key*.
2. Non-unique index on *description*.
3. Foreign key on *_Sequence_key* to *SEQ_Sequence*.

9.5 SEQ_Marker_Cache

A record in this table represents a qualified association between a Sequence and a Marker, the association's Reference and annotation date. The CVS product *seqcacheload* re-loads this table on a nightly basis (note that manual curation of Sequence/Marker associations and Sequence/Marker association loads both affect the contents of this table). The Qualifier is either automatically derived or manually curated.

```
_Sequence_key         int not null
_Marker_key           int not null
_Organism_key         int not null
_Refs_key             int not null
_SequenceType_key     int not null
```

```
_SequenceProvider_key int not null
_LogicalDB_key        int not null
_Qualifier_key        int not null
annotation_date       datetime not null
```

1. Primary key is *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key.*
2. Unique index on *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key.*
3. Non-unique indexes on *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key.*
4. Foreign key on *_Sequence_key* to *SEQ_Sequence*.
5. Foreign key on *_Marker_key* to *MRK_Marker.*
6. Foreign key on *_Organism_key* to *MGI_Organism*.
7. Foreign key on *_Refs_key* to *BIB_Refs*.
8. Foreign key on *_SequenceType_key* to *VOC_Term*.
9. Foreign key on *_SequenceProvider_key* to *VOC_Term*.
10. Foreign key on *_LogicalDB_key* to *ACC_LogicalDB*.
11. Foreign key on *_Qualifier_key* to *VOC_Term*.
12. *annotation_date* is the date of the Sequence/Marker association (*ACC_Accession.modification_date*).
13. *_CreatedBy_key* is the user who created the annotation (*ACC_Accession._CreatedBy_key*).
14. *_ModifiedBy_key* is the user who created the annotation (*ACC_Accession._ModifiedBy_key*).

9.6 SEQ_Probe_Cache

A record in this table represents the annotation between a mouse Molecular Segment and a Sequence.

```
_Sequence_key         int not null
_Probe_key            int not null
_Refs_key             int not null
annotation_date       datetime not null
```

1. Primary key is *_Sequence_key*, *_Probe_key*, *_Refs_key.*
2. Foreign key on *_Sequence_key* to *SEQ_Sequence*.
3. Foreign key on *_Probe_key* to *PRB_Probe*.
4. Foreign key on *_Refs_key* to *BIB_Refs*.
5. The *annotation_date* represents the date of the Sequence/Molecular Segment annotation to the Reference.
6. *_CreatedBy_key* is the user who created the annotation (*ACC_Accession._CreatedBy_key*).

7. *_ModifiedBy_key* is the user who created the annotation (*ACC_Accession._ModifiedBy_key*).

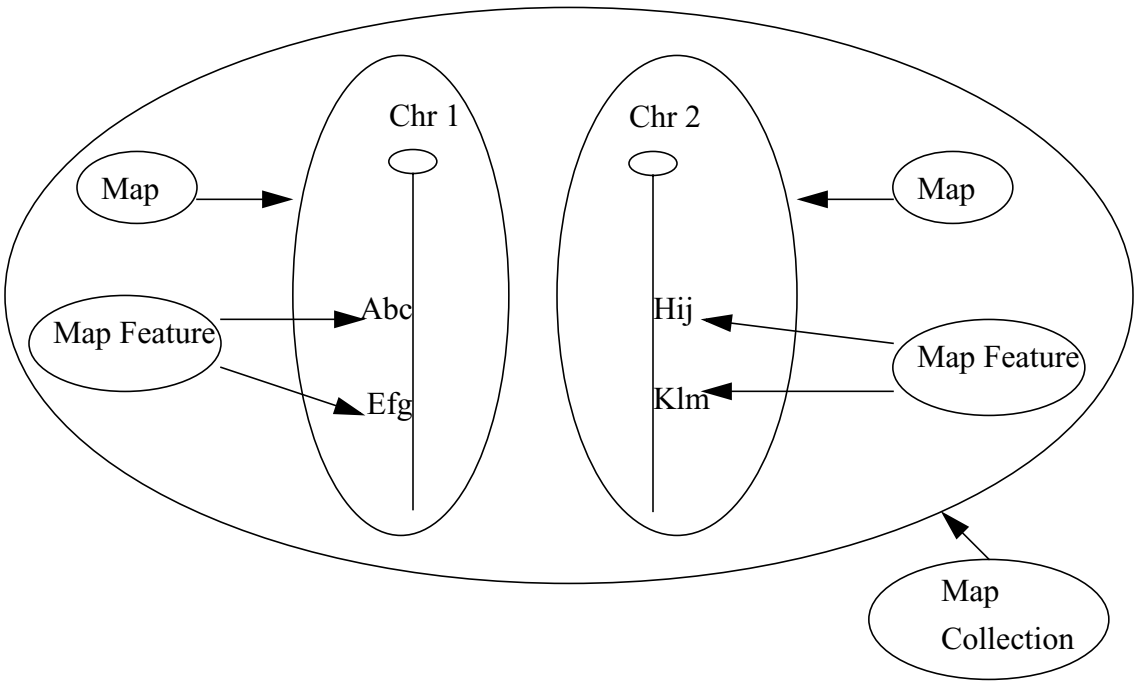
10 Sequence Maps

The section describes the design for coordinate maps, and caching Sequence/Marker associations and representative sequences.

10.1 Map Definitions

- **Coordinate Map Feature** - An object which has a starting base pair and an ending base pair and can be identified by an MGI object (for example, a *Marker*, a *Feature*).
- **Coordinate Map** - A set of Coordinate Map Features of the same unit on a given chromosome. Examples: *Genetic*, *Assembly, RH*.
- **Coordinate Map Collection** - An ordered set of Coordinate Maps.

A Feature, a Map or a Collection can have 0 or more Reference associations (via the generic MGI Reference table) and/or 0 or more Notes (via the generic Notes table).



10.2 MAP_Coord_Feature

A record in this table represents a feature on a coordinate map.

We define a new MGI Object Type of *Map Feature* to enable us to define Map Features which are not represented as Marker or Sequence objects in MGI (for example, *exons*).

```
_Feature_key        int not null
_Map_key            int not null
_MGIType_key        int not null
_Object_key         int not null
startCoordinate     float not null
endCoordinate       float not null
strand              char(1) not null
```

1. Primary key is *_Feature_key*.
2. Non-unique indexes on *_Map_key*, *_MGIType_key*, *_Object_key*.
3. Foreign key on *_Map_key* to *MAP_Coordinate*. This defines the Map on which this feature resides.
4. Foreign key on *_MGIType_key* to *ACC_MGIType*.
5. *startCoordinate* is the coordinate of the first base pair of the object.
6. *endCoordinate* is the coordinate of the last base pair of the object.
7. If *strand* is '+', then the object is on the positive strand. If *strand* is '-', then the object is on the negative strand. If on the negative strand then retrieving the sequence for the object will get the reverse compliment of the base pairs stored in the assembly FASTA file.

10.3 MAP_Coordinate

A record in this table represents one Coordinate Map; a set of Coordinate Map Features of the specified Units for the specified Object. The Object can be an MGI object (Chromosome or Sequence), or a non-MGI object (in which case the _Object_key and _MGIType_key are null). A Coordinate Map can belong to at most one Collection.

```
_Map_key            int not null
_Collection_key     int not null
_Object_key         int null
_MGIType_key        int null
_MapType_key        int not null
_Units_key          int not null
length              int not null
sequenceNum         int not null
name                varchar(255) null
abbreviation        varchar(255) null
version             varchar(255) null
```

1. Primary key is *_Map_key*.
2. Non-unique indexes on *_Object_key*, *_MGIType_key, _Collection_key*, *_MapType_key*, *_MapUnits_key*.
3. Foreign key on *_Collection_key* to *MAP_Coord_Collection*. This defines the Collection to which this map belongs.

4. Foreign key on *_MGIType_key* to *ACC_MGIType*. If *_MGIType_key* is null, then so is *_Object_key*, and this specifies that the feature is a non-MGI object (like a contig that does not correspond to a chromosome).
5. Foreign key on *_MapType_key* to *VOC_Term*. Examples: *Genetic*, *Assembly*, *RH*.
6. Foreign key on *_Units_key* to *VOC_Term*. Examples: *centiMorgan*, *centiRay*, *base pair*.
7. *length* is the map length in the specified Units (*_Units_key*).
8. *sequenceNum* orders the Map within the specified Map Collection (*_Collection_key*).
9. *name* is the public Map label which is printed in the WI or on reports. Most of the time the Map name will be the Chromosome. (e.g. *Chromosome 1*)
10. *abbreviation* is a short version of the *name* used for pick lists, etc.
11. *version* is the version of the specified map. (e.g. *NCBI Build 33* for an Assembly map)

10.4 MAP_Coord_Collection

A record in this table represents one Coordinate Map Collection; an ordered set of Coordinate Maps. A Coordinate Map Collection can contain Coordinate Maps of different Map Types.

```
_Collection_key     int not null
name                varchar(255) not null
abbreviation        varchar(255) not null
```

1. Primary key is *_Collection_key*.
2. *name* is the data provider of the collection. Example: *NCBI Gene Model*.
3. *abbreviation* is a short version of the *name* used for pick lists, etc.

10.5 MRK_CuratedRepSequence

A record in this table represents a *curated* representative sequence association between a Sequence and a Marker as provided by the MGS group. The table is loaded from the genomic sequence/marker association file.

```
_Sequence_key       int not null
_Marker_key         int not null
_Refs_key           int not null
_Qualifier_key      int not null
```

1. Primary key is *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key*.
2. Unique index on *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key*.
3. Non-unique indexes on *_Sequence_key*, *_Marker_key*, *_Refs_key*, *_Qualifier_key*.
4. Foreign key on *_Sequence_key* to *SEQ_Sequence*.
5. Foreign key on *_Marker_key* to *MRK_Marker*.
6. Foreign key on *_Refs_key* to *BIB_Refs*.
7. Foreign key on *_Qualifier_key* to *VOC_Term*.

11 Strains

11.1 PRB_Strain

A record in this table represents a unique strain or genetic background. Accession IDs for Strains (MGI, JRS, EMMA, MMRC, etc.) are stored in the *ACC_Accession* table.

```
_Strain_key       int not null
_Species_key      int not null
strain            varchar(255) not null
standard          bit not null
needsReview       bit not null
private           bit not null
```

1. Primary key is *_Strain_key.*
2. Non-unique index on *strain*.
3. Foreign key on *_Species_key* to *VOC_Term*.
4. If *standard* = 1 then the Strain is standard.
5. If *needsReview* = 1, then the Strain needs nomenclature review. This is set to 1 if any of the Strain's Markers (see *PRB_Strain_Marker*) undergoes a nomenclature event.
6. If *private* = 1, then the Strain is private and is removed from any public version of the database (PUB_MGI, ADHOC_MGI).

11.2 PRB_Strain_Genotype

A record in this table represents a Strain/Genotype association. An association is qualified using the *_Qualifier_key.* A Strain can have 0 or more Genotypes associated with it.

```
_StrainGenotype_key int not null
_Strain_key         int not null
_Genotype_key       int not null
_Qualifier_key      int not null
```

1. Primary key is *_StrainGenotype_key.*
2. Foreign on *_Strain_key* to *PRB_Strain*.
3. Foreign key on *_Genotype_key* to *GXD_Genotype*.
4. Foreign key on *_Qualifier_key* to *VOC_Term* (Strain/Genotype Qualifier).

11.3 PRB_Strain_Marker

A record in this table represents a Strain/Marker/Allele association. An associations is qualified using the *_Qualifier_key.* A Strain can have 0 or more Marker/Allele pairs associated with it.

```
_StrainMarker_key int not null
_Strain_key       int not null
_Marker_key       int not null
_Allele_key       int null
_Qualifier_key    int not null
```

1. Primary key is *_StrainMarker_key*.
2. Foreign on *_Strain_key* to *PRB_Strain*.
3. Foreign key on *_Marker_key* to *MRK_Marker*.
4. Foreign key on *_Allele_key* to *ALL_Allele*.
5. Foreign key on *_Qualifier_key* to *VOC_Term*.

11.4 PRB_Strain_Type

A record in this table represents the association between a Strain and a Strain Type. A Strain can have 0 or more Strain Types associated with it.

```
_Type_key         int not null
_Strain_key       int not null
_StrainType_key   int not null
```

1. Primary key is *_Type_key*.
2. Foreign on *_Strain_key* to *PRB_Strain*.
3. Foreign key on *_StrainType_key* to *VOC_Term*.

12 DAGs

The DAG structures are designed to support the mathematical model of a DAG along with:

- node-child ordering; ordering of children within a node
- node and edge “labels” (for example, “is-a”, “part-of”, “printStop”, anything else we can dream up)

DAG structures are independent of any Vocabulary.

12.1 DAG_DAG

A record in this table represents a specific DAG.

```
_DAG_key       int not null
_Refs_key      int not null
_MGIType_key   int not null
name           varchar(255) not null
abbreviation   char(5) null
```

1. Primary key is *_DAG_key*.
2. Foreign key on *_Refs_key* to *BIB_Refs*.
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.

The *_MGIType_key* defines the object type of the DAG’s Nodes or its Node “type”. In the case of a GO DAG, this is a Vocabulary Term.

For a DAG that has a Vocabulary, the *_Refs_key* is the same as the *_Refs_key* for the Vocabulary. But this model supports DAGs which may not have Vocabularies, so we want to be able to associated References to these DAGs as well.

12.2 DAG_Node

A record in this table represents a specific Node within a DAG.

```
_Node_key    int not null
_DAG_key     int not null
_Object_key  int not null
_Label_key   int not null
```

1. Primary key is *_Node_key*.
2. Non-unique index on *_Object_key*.
3. Foreign key on *_DAG_key* to *DAG_DAG*.
4. Foreign key on *_Label_key* to *DAG_Label*.

The *_Object_key-DAG_DAG._MGIType_key* combination identifies the specific Node object. In the case of a GO DAG this is a specific Vocabulary Term. The *DAG_DAG._MGIType_key* identifies the type of the object (a Vocabulary Term) and the *_Object_key* identifies the object itself.

12.3 DAG_Edge

A record in this table represents an edge of a DAG, that is the relationship between 2 nodes.

```
_Edge_key         int not null
_DAG_key          int not null
_Parent_key       int not null
_Child_key        int not null
_Label_key        int not null
sequenceNum       int not null
```

1. Primary key is *_Edge_key*.
2. Unique index on *_Parent_key* + *_Child_key*.
3. Unique index on *_Parent_key* + *sequenceNum*.
4. Foreign key on *_DAG_key* to *DAG_DAG*.
5. Foreign key on *_Parent_key* to *DAG_Node._Node_key*.
6. Foreign key on *_Child_key* to *DAG_Node._Node_key*.
7. Foreign key on *_Label_key* to *DAG_Label._Label_key*.

The *sequenceNum* orders the child within the parent and is unique for a given parent. All parent-child combinations are unique; a parent can have at most one edge to a given child.

Note that the *_DAG_key* is not technically necessary (since it can be obtained by joining *_Parent_key* to *DAG_Node._Node_key*), but it is included for performance reasons.

12.4 DAG_Label

A record in this table represents a "label" for a Node or an Edge. A "label" is an attribute of the Node or Edge which may be specific to a given DAG. A "label" is not necessarily something which is printed.

```
_Label_key   int not null
label        varchar(255) not null
```

1. Primary key is *_Label_key*.
2. Unique index on *label*.

Examples of DAG labels are:

- is-a (used to specify an Edge type in the GO Vocabulary)
- part-of (used to specify an Edge type in the GO Vocabulary)
- printStop (used to construct a "print name" for a Node in the Anatomical Dictionary)
- header (used to specify that a Mammalian Phenotype term is a Header term)

12.5 DAG_Closure

A record in this table represents an ancestor/descendent pair within a DAG. It caches data for performance reasons. Loaded via the *vocload* product. Two common queries of a DAG are:

- retrieve all the descendents of a node
- retrieve all the ancestors of a node

This table provides a mechanism for performing such queries within one SQL statement.

```
_DAG_key               int not null
_MGIType_key           int not null
_Ancestor_key          int not null
_Descendent_key        int not null
_AncestorObject_key    int not null
_DescendentObject_key  int not null
_AncestorLabel_key     int not null
_DescendentLabel_key   int not null
```

1. Primary key is *_DAG_key* + *_Ancestor_key* + *_Descendent_key*.
2. Foreign key on *_DAG_key* to *DAG_DAG*.
3. Foreign key on *_MGIType_key* to *ACC_MGIType*. This represents the type of object of the *_AncestorObject_key* and the *_DescendentObject_key*.
4. Foreign key on *_Ancestor_key* to *DAG_Node._Node_key*. This represents an Ancestor (parent) Node.
5. Foreign key on *_Descendent_key* to *DAG_Node._Node_key*. This represents a Descendent (child) Node.
6. Foreign key on *_AncestorLabel_key* to *DAG_Label._Label_key*. This represents the Label Type of the Ancestor Node; used to determine if a Node is a Header Node.

7. Foreign key on *_DescendentLabel_key* to *DAG_Label._Label_key*. This represents the Label Type of the Descendent Node; used to determine if the Nodeis a Header Node.
8. *_AncestorObject_key* represents the Object that corresponds to the Ancestor Node (*_Ancestor_key*).
9. *_DescendentObject_key* represents the Object that corresponds to the Descendent Node (*_Descendent_key*).

12.6 Examples

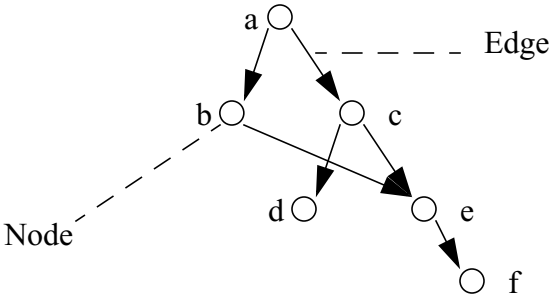


FIGURE 1. Example DAG

1. The *DAG_Edge* records (parent, child, sequenceNum) for this DAG are (note that this example does not show *edge type* or *node type*):
```
a b 1
a c 2
b e 1
c d 1
c e 2
e f 1
```
2. The *DAG_Closure* records are:
```
a,b
a,c
a,e
a,f
a,d
b,e
b,f
c,d
c,e
c,f
e,f
```
3. To retrieve all ancestors of Node e:
```
select _Ancestor_key from DAG_Closure where _Descendent_key
= e
```

4. To retrieve all descendents of Node `c`:
   ```
   select _Descendent_key from DAG_Closure where _Ancestor_key
   = c
   ```

13 Vocabularies

The VOC structures are designed to support the general attributes of a Vocabulary. A Vocabulary is a set of terms which is organized either as a simple list or as one or more DAGs. For example, the GO Vocabulary contains 3 DAGS, the Anatomical Dictionary contains 28 DAGs (each Stage is a DAG) and GO Evidence Codes contain 0 DAGs.

13.1 VOC_Vocab

A record in this table represents a Vocabulary. Different versions of the same Vocabulary are represented by records with different *names* (as in GO 1.0, GO 2.0, etc.).

```
_Vocab_key        int not null
_Refs_key         int not null
_LogicalDB_key    int null
isPrivate          bit
isSimple          bit
name              varchar(255) not null
```

1. Primary key is *_Vocab_key.*
2. Unique index on *name*.
3. Foreign key on *_Refs_key* to *BIB_Refs* table.
4. Foreign key on *_LogicalDB_key* to *ACC_LogicalDB* table.
5. The *isSimple* attribute describes the Vocabulary as either *simple* or *structured*. If *isSimple* = 1, then *simple*, else *structured*.
6. The *_LogicalDB_key* attribute designates the logical database of the Accession IDs of the Vocabulary Terms. If *_LogicalDB_key* is null, then the Vocabulary Terms do not have Accession IDs (for example, GO Evidence Codes). If _LogicalDB_key = 1, then the Vocabulary Terms have MGI Accession IDs (for example,Mammalian Phenotype). If *_LogicalDB_key* > 1, then the Vocabulary Terms have non-MGI Accession IDs (for example, GO Terms have GO IDs).
7. The *isPrivate* attribute designates whether the Accession IDs of the Vocabulary Terms are private (*isPrivate* = 1) or public (*isPrivate* = 0). If *_LogicalDB_key* is null, then *isPrivate* is not relevant.

13.2 VOC_Term

A record in this table represents a Vocabulary Term for a specific Vocabulary. There is one big bucket of Terms for each Vocabulary. A Term can appear in 0 or more DAGS within the same Vocabulary. Remember that a Term is an accessionable object, so there is a corresponding entry for this table in *ACC_MGIType*.

```
 _Term_key   int not null
_Vocab_key   int not null
term         varchar(255) not null
abbreviation char(5) null
sequenceNum  int null
isObsolete   bit
```

1. Primary key is *_Term_key.*
2. Non-unique index on *term*.
3. Foreign key on *_Vocab_key* to *VOC_Vocab.*
4. The *abbreviation* attribute is shorthand for the *term*. For example, in the GO Evidence Vocabulary the term *traceable author statement* has an abbreviation of *TAS*.
5. The *sequenceNum* attribute enables us to define an ordered list of terms for a Vocabulary which contains 0 DAGS (i.e. a simple vocabulary such as the GO Evidence Vocabulary). For structured vocabularies this attribute stores the topological order of the Vocabulary term (determined via a traversal of the DAG in *vocload*).
6. The *isObsolete* attribute is cached information for easily determinig if a Term has been obsoleted or not. This attribute is only set during the GO DAG load.

13.3 VOC_VocabDAG

A record in this table represents an ownership relationship of a Vocabulary over a DAG. A Vocabulary can have many DAGS, but a DAG can belong to at most one Vocabulary.

This table establishes the relationship between a Vocabulary and a DAG.

```
_Vocab_key  int not null
_DAG_key    int not null
```

1. Primary key is *_Vocab_key* + *_DAG_key.*
2. Foreign key on *_Vocab_key* to *VOC_Vocab.*
3. Foreign key on *_DAG_key* to *DAG_DAG*.

13.4 VOC_Text

A record in this table represents textual information of a Vocabulary Term. The Term's *definition* and *example* are part of the textual information. This implements one unlimited-length note per Term and each record represents one 255-char chunk of the note.

```
_Term_key int not null
sequenceNum int not null
note char(255) not null
```

1. Primary key is *_Term_key*, *sequenceNum*.
2. Non-unique index on *note*.
3. Foreign key on *_Term_key* to *VOC_Term*.

14 Vocabulary-to-Vocabulary Associations

A generalized design that allows one to define an association between any two MGI vocabularies.

First, define an Association Type (*MGI_VocAssociationType*):

- what 2 Vocabularies am I associating? for example, *Allele QF Categories* to *Allele Types*

Then, define the association between the terms from the 2 vocabularies. Note that this design supports a many-to-many association between vocabularies.

14.1 MGI_VocAssociation

A record in this table represents the association between a vocabulary term in one Vocabulary (*_Term_key_1*) to a vocabulary term in another Vocabulary (*_Term_key_2*) for a specific Association Type (*_AssociationType_key*).

```
_Association_key      int not null
_AssociationType_key  int not null
_Term_key_1           int not null
_Term_key_2           int not null
sequenceNum           int not null
```

1. Primary key is *_Association_key.*
2. Foreign key on *_Term_key_1* to *VOC_Term*.
3. Foreign key on *_Term_key_*2 to *VOC_Term*.
4. Foreign key on *_AssociationType_key* to *MGI_VocAssociationType*.
5. *_AssociationType_key* specifies the Vocabulary Association Type for the pair of terms.
6. *_Term_key_*1 represents the Term from one Vocabulary. All *_Term_key_1* terms for a given Association Type are from the same Vocabulary.
7. *_Term_key_*2 represents the Term from the other Vocabulary. All *_Term_key_2* terms for a given Association Type are from the same Vocabulary.
8. *sequenceNum* is used to order the pairs of associated Terms.

14.2 MGI_VocAssociationType

A record in this table represents a Vocabulary Association Type (e.g. “Allele QF Category-to-Allele Type”).

```
_AssociationType_key   int not null
_Vocab_key_1           int not null
_Vocab_key_2           int not null
associationType        varchar(255) not null
definition             varchar(255) null
```

1. Primary key is *_AssociationType_key.*
2. Foreign key on *_Vocab_key_1* to *VOC_Vocab.*

3. Foreign key on *_Vocab_key_2* to *VOC_Vocab.*
4. *_Vocab_key_1* is the Vocabulary of *MGI_VocAssociation._Term_key_1*.
5. *_Vocab_key_2* is the Vocabulary of *MGI_VocAssociation._Term_key_2*.
6. *associationType* is the user-defined name of the Association (*Allele QF Category-to-Allele Type*).
7. *definition* is the definition of the Association.

15 Annotations

The VOC_Annot structures are designed to support all Vocabulary Term-to-Object Annotations.

A Vocabulary Term-Object Annotation contains the following attributes:

- Reference (J:)
- Qualifying Evidence (how the authors support their conclusions). The GO provides a vocabulary list of Evidence Codes which will be implemented as an ordered Vocabulary list with 0 DAGs.
- Editor (as long as we're recording this, we may as well record the editor who created the record and the editor who last modified the record).

15.1 VOC_AnnotType

A record in this table defines an Annotation Type. Each Annotation instance specifies its type. An Annotation Type specifies an Object type (*_MGIType_key*), the Vocabulary to which an Object of that type is being annotated (*_Vocab_key*), and the Evidence Vocabulary (*_EvidenceVocab_key*) used to annotate the Object of the specified type. An Object which is to be annotated (a Marker or a Genotype) is an accessionable object which requires an entry in *ACC_MGIType*.

```
_AnnotType_key int not null
_MGIType_key   int not null
_Vocab_key     int not null
_EvidenceVocab_key int not null
name               varchar(255) not null
```

1. Primary key is *_AnnotType_key.*
2. Unique index on *_MGIType_key* + *_Vocab_key* + *_EvidenceVocab_key.*
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.
4. Foreign key on *_Vocab_key* to *VOC_Vocab.*
5. Foreign key on *_EvidenceVocab_key* to *VOC_Vocab.*

For example, *name* = GO-Marker, *_MGIType_key* = Marker, *_Vocab_key* = GO Vocabulary, *_EvidenceVocab_key* = GO Evidence Code Vocabulary defines an Annotation Type for annotating Marker objects to the GO Vocabulary using the list of GO Evidence Codes.

15.2 VOC_Annot

A record in this table represents the Annotation between a Vocabulary Term and an Object of type specified by the Annotation Type.

There is one Annotation per Object/Term/Annotation Type/isNot.

```
_Annot_key      int not null
_AnnotType_key  int not null
_Object_key     int not null
_Term_key       int not null
isNot           bit (default is 0)
```

1. Primary key is *_Annot_key*.
2. Unique index on *_AnnotType_key* + *_Object_key* + *_Term_key* + *isNot*. We cannot create an index with a bit field, so we will have to enforce this constraint in a trigger.
3. Non-unique index on *Object_key*.
4. Foreign key on *_AnnotType_key* to *VOC_AnnotType*.
5. Foreign key on *_Term_key* to *VOC_Term*.
6. Bind *bit_default* to *isNot*.
7. No foreign key on *_Object_key* because this can be a key to any Object.
8. The *isNot* bit is used to enable annotations which specifically state that the Term does *not* describe the Object. The default is 0 (the Term *does* describe the Object).

In order to be annotated, an Object must be an accessionable object (which means there must be an entry for its type in the *ACC_MGIType* table).

15.3 VOC_Evidence

A record in this table represents one evidence statement which supports one Annotation. There can be one or more Evidence records per Annotation. Although the schema supports zero or more Evidence records per Annotation, editorially at least one Evidence record is required per Annotation.

The *_EvidenceTerm_key* in this table identifies the Evidence Code (which is itself a member of a simple Vocabulary).

```
_Annot_key        int not null
_EvidenceTerm_key int not null
_Refs_key         int not null
inferredFrom      varchar(255) null
createdBy         varchar(30) not null
modifiedBy        varchar(30) not null
notes             varchar(255) null
```

1. Primary key is *_Annot_key* + *_Evidence_key* + *_Refs_key*.
2. Foreign key on *_EvidenceTerm_key* to *VOC_Term*.
3. Foreign key on *_Refs_key* to *BIB_Refs*.

15.4 VOC_AnnotHeader

A record in this table represents the unique use of a Header term for a specific annotated Object of a specific Annotation Type. A set of records grouped by Annot Type/Object represents the *ordered* list of Header terms for an annotated Object of a specific Annotation Type.

Implemented for these Annotation Types:

- Genotypes annotated to Mammlian Phenotype

If a record for a given *_AnnotType_key*/ *_Object_key* is approved, then **all** records for the same *_AnnotType_key*/*_Object_key* are considered approved and all *_ApprovedBy_key* and *approval_date* fields for the *_AnnotType_key*/*_Object_key* are updated.

```
_AnnotHeader_key  int not null
_AnnotType_key    int not null
_Object_key       int not null
_Term_key         int not null
sequenceNum       int not null
_ApprovedBy_key   int null
approval_date     datetime null
```

1. Primary key is *_AnnotHeader_key*.
2. Foreign key on *_AnnotType_key* to *VOC_AnnotType*.
3. Foreign key on *_Term_key* to *VOC_Term*.
4. Foreign key on *_ApprovedBy_key* to *MGI_User*.
5. *_AnnotType_key* specifies the type of Annotation (the type of object being annotated and the Vocabulary to which the object is annotated). This is necessary because Objects can be annotated to different Vocabularies.
6. *_Object_key* is the primary key of the Object that is annotated.
7. *_Term_key* is the Header Term for the annotation (via *VOC_Term*, *DAG_Node*).
8. *sequenceNum* is the order of the Header Term for the all records grouped by *_Object_key*. The default order (when the record is initially created) is by topological order (stored in *VOC_Term.sequenceNum*).
9. *_ApprovedBy_key* and *approval_date* are the user and date on which the Header order was approved by a curator. If these fields are null, then the Headers are considered "Not Approved".

16 Editorial Tracking

Used to track the modification history of specific table attributes in order to determine if a process is allowed to modify an attribute.

For example, we track the modification history for the following:

- all *PRB_Source* attributes (resolved Molecular Source)

- *SEQ_Sequence._SequenceType_key*

16.1 MGI_AttributeHistory

A record in this table represents the *most recent* modification history of a specific attribute (*columnName*) of an MGI Object (*_Object_key*) of a specific type (*_MGIType_key*).

The *_MGIType_key* specifies the *tableName* (via a join to *ACC_MGIType*) to which the *columnName* belongs.

```
_Object_key             int not null
_MGIType_key            int not null
columnName              varchar(30)
```

1. Primary key is *_Object_key* + *_MGIType_key* + *columnName*.
2. Non-unique index on *_Object_key.*
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.
4. Note that the *modification_date* field stores the most recent modification date of the columnName and that *modifiedBy* field stores the user ID/load ID which made the most recent modification.

17 Images

An MGI Image is an object that represents an actual picture that we wish to associate to some other MGI object (like an expression result, an allele or a genotype). It is not the actual picture itself (the jpeg file).

An MGI Image consists of one or more Image Pane objects and every MGI Image has at least one Image Pane object. It is the Image Pane object that is associated to the MGI expression result, allele or genotype.

When associating an MGI Image Pane with an MGI expression result, allele or genotype, it is not necessary that the MGI Image Pane have a corresponding image file (jpeg).

Image files (jpegs) are stored in the file system in a "database" called PixelDB. Each image that is scanned into PixelDB is identified by a PixelDB accession id (PIX:#####). Image files are associated with MGI Image objects (not the individual Panes) by associating the PixelDB accession id of the image file to the MGI Image object.

So, you can associate an MGI object (expression result, allele, genotype) with an MGI Image Pane object without having an actual picture to display. GXD curates expression results to MGI Image Panes in this way much of the time.

A generalized design that allows one to define any number of Images and to associate an Image Pane to any object in the database that has a non-composite primary key.

Implemented for:

- Alleles

- Genotypes

17.1 IMG_Image

A record in this table represents a full size or a thumbnail Image.

If the X dimension (width) and Y dimension (height) are non-null, then a corresponding PixelDB Image file is associated with the Image object (via the accession id PIX:####, *ACC_Accession*).

A full size Image:

- has at most one Thumbnail Image or may have no Thumbnail Image.
- has a Caption of unlimited size.
- may have Private Curatorial Notes.

A Thumbnail Image:

- belongs to at most one full size Image.
- has a Caption of unlimited size.
- may have Private Curatorial Notes.
- has the same Reference as its full size Image.

Captions, Copyright and Private Curatorial Notes are implemented using the MGI generic Notes data structures.

```
_Image_key           int not null
_Refs_key            int not null
_ImageType_key       int not null
_ThumbnailImage_key  int null
xDim                 int null
yDim                 int null
figureLabel          varchar(255) not null
```

1. Primary key is *_Image_key*.
2. Foreign key on *_Refs_key* to *BIB_Refs*.
3. Foreign key on *_ImageType_key* to *VOC_Term* (Image Type).
4. Foreign key on *_ThumbnailImage_key* to *IMG_Image*.
    - If *_ImageType_key* is Full Size, then *_ThumbnailImage_key* refers to an *IMG_Image* object where *_ImageType_key* is Thumbnail.
    - If *_ImageType_key* is Thumbnail, then *_ThumbnailImage_key* is null.

17.2 IMG_ImagePane

A record in this table represents an Image Pane. An Image has at least one Image Pane.

```
_ImagePane_key       int not null
_Image_key           int not null
paneLabel            varchar(255) null
```

1. Primary key is *_ImagePane_key*.
2. Foreign key on *_Image_key* to *IMG_Image*.

17.3 IMG_ImagePane_Assoc

A record in this table represents an association between an Image Pane and an MGI object (allele, genotype). At most one association for a given MGI object can be primary.

References of Image Pane Associations are implemented using the MGI generic Reference Association data strucutres.

```
_Assoc_key            int not null
_ImagePane_key        int not null
_MGIType_key          int not null
_Object_key           int not null
isPrimary             bit not null
```

1. Primary key is *_Assoc_key*.
2. Foreign key on *_ImagePane_key* to *IMG_ImagePane*.
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.
4. An ImagePane association may have Private Curatorial Notes.
5. An ImagePane association may have References.

17.4 GXD_InSituResultImage

A record in this table represents an association between an Image Pane and a GXD InSitu Result. An InSitu Result may have 1 or more Images Panes.

For associations of GXD Gel Assays to Image Panes, see *GXD_Assay*.

```
_Result_key           int not null
_ImagePane_key        int not null
```

1. Primary key is *_Result_key*, *_ImagePane_key*.
2. Foreign key on *_Result_key* to *GXD_InSituResult*.
3. Foreign key on *_ImagePane_key* to *IMG_ImagePane*.

18 Notes

A generalized design that allows one to define any number of any kind of Note to any object in the database that has a non-composite primary key.

Implemented for:

- Alleles
- Annotation Evidence
- Genotypes
- Images

- Markers (GO Notes)
- Nomenclature
- Molecular Sources
- Sequences
- Strains
- GO Text
- Vocabulary Terms

18.1 MGI_Note

A record in this table represents a Note of a specific type (*_NoteType_key*, e.g. *General Purpose*) for a specific object (*_Object_key*, e.g. *Sequence "A12345"*) of a specific object type (*_MGIType_key*, e.g. *Sequence*, *Sequence Set*, *Marker/Sequence Set Association*) with a specific privacy setting (*private* or *public*).

```
_Note_key           int not null
_Object_key         int not null
_MGIType_key        int not null
_NoteType_key       int not null
```

1. Primary key is *_Note_key.*
2. Unique index on *_Object_key* + *_MGIType_key* + *_NoteType_key.*
3. Non-unique index on *_Object_key.*
4. Foreign key on *_MGIType_key* to *ACC_MGIType*.
5. Foreign key on *_NoteType_key* to *MGI_NoteType*.

For example, a general purpose, public note for a Sequence with primary key = 1000 would have a *MGI_Note* record that looks like this:

- _Note_key = next available primary key
- _Object_key = 1000
- _MGIType_key = _MGIType_key of *Sequence* (from ACC_MGIType)
- _NoteType_key = _NoteType_key of *General Purpose Note* (from MGI_NoteType)

18.2 MGI_NoteType

A record in this table represents a Note Type (e.g. *General*, *Molecular*).

```
__NoteType_key     int not null
_MGIType_key       int null
noteType           varchar(255) not null
private            bit not null
```

1. Primary key is *_NoteType_key.*
2. Foreign key on *_MGIType_key* to *ACC_MGIType*.

3. If *_MGIType_key = null*, then the Note Type can be used for a Note to any type of database object. If *_MGIType_key* is not *null*, then the Note Type can only be used for a Note to an object of that type.

   We will be able to use this mechanism to control the placement of Note buttons in EI modules. All Note Types where *_MGIType_key = null* will appear in all EI modules for which Note support is requested. Note Types where *_MGIType_key* is not *null* will only appear in EI modules of that type. For example, the *General* note button will appear in the Marker module and the Allele module. But the *Molecular* note button will only appear in the Allele module.
4. If *private* = true (1), then the Note is removed from public versions of the database via the *MGI_deletePrivateData* stored procedure.

18.3 MGI_NoteChunk

A record in this table represents a 255-character segment of a specific Note object (defined in *MGI_Note*).

```
_Note_key           int not null
sequenceNum         int not null
note                char(255) not null
```

1. Primary key is *_Note_key* + *sequenceNum*.
2. Non-unique index on *sequenceNum*.
3. Foreign key on *_Note_key* to *MGI_Note*.

19 Organisms

Provides the following functionality:

- makes Organisms accessionable objects (assign private MGI Accession IDs to all Organisms; assign NCBI's taxon IDs to mouse, human, rat)
- allows us to produce organism lists for a specific MGI Type (Homology, Molecular Segment, Antigen, Antibody, Markers, Sequences)

19.1 MGI_Organism

A record in this table represents an Organism.

```
_Organism_key         int not null
latinName             varchar(50) not null,
commonName            varchar(50) not null
```

1. Primary key is *_Organism_key.*
2. Unique index on *_Organism_key*, *commonName*.

19.2 MGI_Organism_MGIType

A record in this table represents the relationship between an Organism and a MGI Type; what Organisms are valid for a given MGI Type. An Organism can be associated with one or more MGI Types. This enables us to generate a list of Organisms by MGI Type. Examples are *Orthology*, *Molecular Segment* , *GXD Antigen*, *GXD Antibody*, *Sequence, Marker.*

```
_Organism_key          int not null
_MGIType_key           int not null
```

1. Primary key is *_Organism_key* + *_MGIType_key.*
2. Foreign key on *_Organism_key* to *MGI_Organism*.
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.

20 Sets

20.1 MGI_Set

A record in this table represents a Set (e.g. “Clone Set”).

```
_Set_key               int not null
_MGIType_key           int not null
name                   varchar(255) not null
```

1. Primary key is *_Set_key.*
2. Foreign key on *_MGIType_key* to *ACC_MGIType*.
3. *_MGIType_key* is the type of MGI Object (Logical DB, Marker, etc.) of a Set member. Used to verify the *MGI_Set._Object_key.*
4. *name* is the user-defined name of the Set (example: *Clone Set*).

20.2 MGI_SetMember

A record in this table represents the ordered member (*_Object_key*) of a specific Set (*_Set_key*).

```
_SetMember_key        int not null
_Set_key              int not null
_Object_key           int not null
sequenceNum           int not null
```

1. Primary key is *_SetMember_key.*
2. Non-unique index on *_Object_key.*
3. Foreign key on *_Set_key* to *MGI_Set*.
4. *_Set_key* specifies the Set of which the object is a member.
5. *_Object_key* is the primary key of a MGI object (for example, *ACC_LogicalDB._LogicalDB_key* and represents a member of the set.
6. *sequenceNum* is used to order the member within the set.

21 Synonyms

A generalized design that allows one to specify any number of Synonyms for a database object that has a non-composite primary key.

Implemented for:

- Alleles
- Markers
- Nomenclature Markers
- Strains
- Vocabulary Terms

21.1 MGI_Synonym

A record in this table repesents a Synonym of a specific type (*_SynonymType_key*, e.g. *Exact*) and a specific database object (*_Object_key*, e.g. *Marker "Kit"*) of a specific object type (*_MGIType_key*, e.g., *Marker*). The Synonym may also have a Reference associated with it.

```
_Synonym_key         int not null
_Object_key          int not null
_MGIType_key         int not null
_SynonymType_key     int not null
_Refs_key            int null
synonym              varchar(255) not null
```

1. Primary key is *_Synonym_key*.
2. Non-unique index on *synonym*.
3. Foreign key on *_MGIType_key* to *ACC_MGIType*.
4. Foreign key on *_SynonymType_key* to *MGI_SynonymType*.
5. Foreign key on *_Refs_key* to *BIB_Refs*.

21.2 MGI_SynonymType

A record in this table represents a Synonym Type (e.g. *Exact*) for a specific Object Type. The Object may be of a specific Organism (for example, Marker, Mouse).

```
__SynonymType_key         int not null
_MGIType_key              int not null
_Organism_key             int nill
allowOnlyOne              bit not null
synonymType               varchar(255) not null
definition                varchar(255) not null
```

1. Primary key is *_SynonymType_key*.
2. Foreign key on *_MGIType_key* to *ACC_MGIType*.
3. Foreign key on *_Organism_key* to *MGI_Organism*.

4. If *allowOnlyOne* = 1, then there can be at most one Synonym of this type for a given object.
5. The Synonym Type can only be used for a Synonym to an object of that type.

22 Translations

A generalized design that allows one to define a bad name to good name translation implementation.

First, define a Translation Type (*MGI_TranslationType*):

- what type of object am i translating? for example, *a Strain object*.
- what do i want to call this translation type? for example, *Bad Strain/Good Strain*.

Then define the bad name/good name list for the Translation Type (*MGI_Translation*). That is, the list of all bad name/good name pairs where each *good name* is represented by an *_Object_key* of the type specified by the *MGI_Translation._TranslationType_key.*

22.1 MGI_Translation

A record in this table represents the translation between a "bad name" (*badName*) and the corresponding MGI object (the "good name") of a specific Translation Type (*_TranslationType_key*).

```
_Translation_key      int not null
_TranslationType_key  int not null
_Object_key           int not null
badName               varchar(255)
sequenceNum           int not null
```

1. Primary key is *_Translation_key.*
2. Non-unique index on *_Object_key*, *badName*.
3. Foreign key on *_TranslationType_key* to *MGI_TranslationType*.
4. *_TranslationType_key* specifies the Translation Type for the bad name/good name pair, which specifies what type of object the good name is (see *_Object_key*).
5. *_Object_key* is typically a Vocabulary Term (*VOC_Term._Term_key*) or a primary key of a Controlled Vocabulary table (for example, *PRB_Strain._Strain_key*).
6. badName is the incorrent term that is being mapped to *_Object_key.*
7. *sequenceNum* is used to order the list of bad names. Order is significant if regular expressions are supported (because in this case all bad names have to be searched).

22.2 MGI_TranslationType

A record in this table represents a Translation Type (e.g. "SwissProt-to-GO" or "Strains").

```
_TranslationType_key   int not null
_MGIType_key           int not null
_Vocab_key             int null
```

```
translationType        varchar(255) not null
compressionChars       varchar(50) not null
regularExpression      bit not null
```

1. Primary key is *_TranslationType_key*.
2. Foreign key on *_MGIType_key* to *ACC_MGIType*.
3. Foreign key on *_Vocab_key* to *VOC_Vocab*.
4. *_MGIType_key* is the type of MGI Object (Strain, Tissue, etc.) that a Translation Type defines. Used to verify the *MGI_Translation._Object_key*.
5. *_Vocab_key* is the Vocabulary to which the MGI Objects refer if they are of MGI Type *Vocabulary Term*.
6. *translationType* is the user-defined name of the Translation (*SWISSProt-to-GO* or *Strains*).
7. *compressionChars* is the list of characters to remove from a translation target before translation is applied. For example, if translation target = "C57.BL6/J" and *compressionChars* includes ".", then translation target will be compressed to "C57BL/6J" before it is searched for in *MGI_Translation*.
8. if *regularExpression* = 1 (true), then apply a regular expression algorithm to the lookup of the translation target in *MGI_Translation*. Note that a regular expression algorithm is a totally separate algorithm than the non-regular expression bad name-to-good name algorithm. For JSAM, we do not anticipate using a regular expression algorithm. Therefore, a regular expression algorithm will only be implemented when it is needed.

23 User Table

23.1 MGI_User

A record in this tables represents a valid MGI User.

```
_User_key         int not null
_UserType_key     int not null
_UserStatus_key   int not null
login             varchar(30) not null
name              varchar(255) not null
```

1. Primary key is *_User_key*.
2. Foreign key on *_UserType_key* to *VOC_Term* (a term in the User Type CV). Examples: Curator, Software Engineer, PI.
3. Foreign key on *_UserStatus_key* to *VOC_Term* (a term in the User Status CV). Examples: Active, Inactive.
4. *login* is equivalent to the Sybase user id and UNIX login id. Examples: jdoe.
5. *name* is the full name of the user. Examples: Jane Doe.