DiscourseDB v0.2
Oliver Ferschke <ferschke@cs.cmu.edu>
Carolyn P. Rosé <cprose@cs.cmu.edu>

System

discoursedb
- id_discoursedb BIGINT
- schema_version VARCHAR(200)
- start_time DATETIME
- end_time DATETIME

Indexes

This table holds basic bookkeeping information - e.g. schema version.
Annotation_Instances refer to this so we know with which
version of the schema the annotations were produced.

Annotations

Annotation_Type
defines the invariant aspects of the type system (everything that's not related to a particular annotation
isEntityAnnotation defines whether an annotation of this type refers to a span in the text field of a Content
the whole entity it is associated with.

Annotation_Instance
Alternatively, we could introduce different types of annotations (Span_Annotation, Entity_Annotation), but
CoveredText explicitly holds the span of text covered by the annotation. This can be inferred by the index
Annotations have a lifetime like all other major entities and relations to indicate the timespan in which the
These field are optional and can be used if this information is available and required.

Annotation
aggregates all annotations related to an entity or relation table.

annotation_type
- id_annotation_type BIGI…
- name VARCHAR(45)
- description VARCHAR(…
- isEntityAnnotation BOO…

Indexes

annotation_instance
- id_annotation_instance BIGINT
- begin_offset INT
- end_offset INT
- covered_text TEXT
- fk_annotation BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation_type BIGINT
- fk_discoursedb BIGINT

Indexes

annotation
- id_annotation BIGINT

Indexes

feature_type
- id_feat_type BIGINT
- name VARCHAR(200)
- datatype VARCHAR(200)
- description VARCHAR(…

Indexes

feature_instance
- id_feat_inst BIGINT
- fk_feat_type BIGINT
- value TEXT
- fk_annotation_instance B…

Indexes

Feature_Type defines name and type of the feature. The name is not part of the feature instance but rather
part of the type so we can develop something like a type system.
However, we reduce the complexity to flat non-hierarchical type system.
The datatype defines how the value in the associated feature entity should be interpreted on the system side
(e.g. String, StringArray, Boolean, etc).
The value field in the Feature entity stores String-representations of feature values.
This could be made more flexible by changing the value data type to BLOB and
fully determine the representation by the feature type.

User Layer

user_memberof_group
- fk_user BIGINT
- fk_group BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

group
- id_group BIGINT
- name VARCHAR(…
- start_time DATETI…
- end_time DATETIME
- fk_group_type BIG…
- fk_annotation BIGI…

Indexes

group_type
- id_group_type BIGINT
- type VARCHAR(200)

Indexes

content_interaction_type
- id_content_interaction_type BIGINT
- type VARCHAR(45)

Indexes

Interactions with content define the relationships
between authors and content (and by extension
also Contribution and Context)
Types could e.g. be WRITE, READ, DELETE etc.

user
- id_user BIGINT
- realname VARCHAR(…
- username VARCHAR(…
- email VARCHAR(200)
- location VARCHAR(200)
- language VARCHAR(2…
- ip VARCHAR(200)
- fk_annotation BIGINT

Indexes

audience_has_user
- fk_audience BIGINT
- fk_user BIGINT
- start_time VARCHAR(45)
- end_time VARCHAR(45)
- fk_annotation BIGINT

Indexes

audience
- id_audience BIGINT
- fk_audience_type BIGINT
- fk_annotation BIGINT

Indexes

audience_has_group
- fk_audience BIGINT
- fk_group BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

content_interaction
- fk_user BIGINT
- fk_content BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_content_interaction_type BIGINT
- fk_annotation BIGINT

Indexes

user_relation
- id_user_relation BIGINT
- fk_user_source BIGINT
- fk_user_target BIGINT
- fk_user_relation_type BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

user_relation_type
- id_user_relation_type BIGINT
- type VARCHAR(200)

Indexes

contribution_has_audience
- fk_contribution BIGINT
- fk_audience BIGINT
- fk_annotation BIGINT
- start_time DATETIME
- end_time DATETIME

Indexes

audience_type
- id_audience_type BIGINT
- type VARCHAR(200)

Indexes

Macro Structure

content
- id_content BIGINT
- creation_time DATETIME
- fk_content_previous_revision…
- fk_content_next_revision BIG…
- text TEXT
- data BLOB
- fk_annotation BIGINT

Indexes

contribution
- id_contribution BIGINT
- fk_content_first_revision BIGINT
- fk_content_current_revision BIGINT
- fk_contribution_type BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

discourse_relation
- id_discourse_relation BIGINT
- fk_contribution_source BIGINT
- fk_contribution_target BIGINT
- fk_discourse_relation_type BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

context
- id_context BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_content_first_revision BIGI…
- fk_content_current_revision …
- fk_context_type BIGINT
- fk_annotation BIGINT

Indexes

contribution_has_context
- id_contribution_has_context BIGINT
- fk_context BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

contribution_type
- id_contribution_type BIGI…
- type VARCHAR(200)

Indexes

contribution_partOf_discourse_part
- fk_contribution BIGINT
- fk_discourse_part BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

discourse_relation_type
- id_discourse_relation_type BIGINT
- type VARCHAR(200)

Indexes

Context is only seen in the light of a contribution.
The content of the context is associated with a user,
but we are not explicitly keeping track of the
audience of a context. That's inferred by the
audience of the contribution.
However, a context could be associated
with multiple contributions from different sources.

context_type
- id_context_type BIGINT
- type VARCHAR(200)

Indexes

discourse_part_relation
- id_discourse_part_relation BIGINT
- fk_source_discourse_part BIGINT
- fk_target_discourse_part BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT
- fk_discourse_part_relation_type …

Indexes

discourse
- id_discourse BIGINT
- fk_annotation BIGINT
- name VARCHAR(200)
- source VARCHAR(200)

Indexes

discourse_has_discourse_part
- fk_discourse BIGINT
- fk_discourse_part BIGINT
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

discourse_part
- id_discourse_part BIGINT
- fk_discourse_part_type BIGINT
- name VARCHAR(200)
- start_time DATETIME
- end_time DATETIME
- fk_annotation BIGINT

Indexes

discourse_part_type
- id_discourse_part_type BIGINT
- type VARCHAR(45)

Indexes

discourse_part_relation_type
- id_discourse_part_relation_type BIGINT
- type VARCHAR(200)

Indexes

DiscoursePart entities are containers for contributions. Context entities are indirectly part of DiscourseParts, since they are connected to contributions.
Examples for DiscourseParts are Forums, Discussion Groups, Discussion sections - i.e. all places where discussion happens within a "Discourse".
Threads are not DiscourseParts, since they represent relationships between contributions rather than containers of contributions.
DiscourseParts can either form a flat hierarchy or can be nested or in other ways connected (previously called embeddings).
If a particular Discourse does not further organize contributions in organizational units (e.g. if you import a Forum without subforms as a Discourse), then
a single Dummy DiscoursePart can be created as a container. This should be reflected in the DiscoursePart_Type.
Beyond hierarchies, DiscoursePart_Relations can also define the relationships between interactions on multiple channels from different sources.