XBRL US Database Changes - June 2017

Overview

The XBRL US database contains public company reports filed with the Securities and Exchange Commission (SEC) filed in XBRL. The database is being expanded to support XBRL reports from other sources. Several changes are being made to to the database to support multiple sources. In addition, improvements are being made to the database. Although there are significant changes to the database model, backward compatibility has been maintained. This document describes these changes.

Table of Contents

Genericized for multiple sources

The previous version of the database was designed to only expect filings from a the SEC and contained no provision in the model to support filings from other reporting sources. This affects the model of the database and how the data is handled during the exchange, transform and load (ETL) process for loading data.

Source table

To identify a reporting source, the “source” table is added to the database:

source table

Column	Type	Nulls	Notes
source_id	integer	No	Internal id for the source.
source_name	text	Yes	Name of the source

For SEC data, an initial row is added with source_id = 1 and source_name = ‘SEC’.

A source_id column is added to the following tables:

- base_namespace
- entity_source (new table)
- namespace_source (new table)
- report (new table)

When a new reporting source is added to the database, a row will be added to the source table.

Entity and source

Entities are shared across sources where the entity scheme and identifiers are the same. The entity_source table links entities (in the entity table) to the reporting source (in the source table).

entity_source table

Column	Type	Nulls	Notes
entity_source_id	integer	No	Internal id for the entity_source.
entity_id	integer	No	Foreign key to the entity table.
source_id	integer	No	Foreign Key to the source table.

Namespace and source

The identification of namespaces that are used are in the namespace table. This table includes the is_base column because, with multiple sources, it is possible that a namespace is a base namespace for one source but not another. The namespace_source table is added to identify which namespaces for a source and whether the namespace is a base namespace.

namespace_source table

Column	Type	Nulls	Notes
namespace_source_id	integer	No	Internal id for the namespace_source.
namespace_id	integer	No	Foreign key to the namespace table.
source_id	integer	No	Foreign key to the source table.
is_base	boolean	No	Determines if the namespace is a base namespace for the source.

The is_base is left on the namespace table for backward compatibility. It is only populated when loading SEC reports.

Report table (replaces accession)

Many columns on the accession table were specific to SEC filings such as filing_accession_number, irs_number and sec_html_url. To support filings from other sources a new table "report" is created. This table replaces the accession table and contains fields that apply to all sources. Source specific columns are included in a flexible JSON structure in the "properties" column.

For backward compatibility, a view for "accession" is created which has the same columns as the former table. This view will only include filings for SEC source reports.

report table

Column	Type	Nulls	Notes
report_id	integer	No	Internal id for the report. For all existing accession rows, this has the same value as the accession_id.
source_id	integer	No	source_id for the source for the source of the report.
entity_id	integer	No	entity_id of the report.

source_report_identifier	text	Yes	The identifier that uniquely identifies the report for the source reporting system. For SEC, this is the filing accession number.
dts_id	integer	No	dts_id of the taxonomy dts for the report. See DTS and Taxonomy.
entry_dts_id	integer	Yes	dts_id of the non taxonomy portion of the report. This is used for footnotes. See DTS and Taxonomy.
creation_timestamp	timestamp	No	Timestamp when the row is added to the table.
accepted_timestamp	timestamp	No	Timestamp when the report is filed as determined by the source reporting system.
is_most_current	boolean	No	Identifies the most current report for an entity.
entity_name	text	Yes	The name of the reporting entity.
creation_software	text	Yes	The name of the software used to create the report.
entry_type	text	No	Identifies if the report is an inline XBRL or standard XBRL instance. The values are ‘inline’ and ‘instance’.
entry_url	text	No	The url of the report.
entry_document_id	integer	NO	The document_id of the entry document.
alternative_document_id	integer	Yes	The document_id of an alternative representation of the the report. This is usually used to include the document id of a non-XBRL (html or text) version of a report.
reporting_period_end_date	timestamp	Yes	The date of the reporting period end.
restatement_index	integer	Yes	Identifies if the filing is a restatement. The most recent statement of a filing will be “1”. The next most recent statement of a filing will be “2”, and so on.
period_index	integer	Yes	Identifies the most recent filing for the entity. This is similar to the restatement

			index, whereas each filing is processed, the period indexes of the previous filings are bumped up by 1.
properties	JSONB	Yes	A flat JSON structure that contains any additional properties of the report. For SEC filings see SEC properties.

Note about accession_id on other tables. The report_id is a replacement for accession_id. In order to maintain backward compatibility, any existing table with an accession_id was not changed to report_id. When joining the "report" table to a pre-existing table, the join will be made on report.report_id = *other_table*.accession_id.

SEC properties

For SEC filings, the "properties" column contains the following JSON structure:

```
{
        "zip_url": "",
        "filing_date": "",
        "sec_html_url": "",
        "document_type": "",
        "business_phone": "",
        "business_address": "",
        "percent_extended": null,
        "state_of_incorporation": "",
        "filing_accession_number": "",
        "internal_revenue_service_number":"",
        "standard_industrial_classification":""
}
```

These are the columns from the accession table that are not on the report table. To access these properties, use the "->>" operator.

```
SELECT properties->>'document_type'
FROM report
LIMIT 10
```

The "->>" operator returns the value as a string. For non string properties you may cast the returned value.

```
SELECT (properties->>'standard_industrial_classification')::integer
FROM report
LIMIT 10
```

Report element table

The report_element table partially replaces the accession_element table. The report_element table lists the elements that are used in a report. An element is considered used if it is used on a fact as a primary item, dimension or member.

report_element table

Column	Type	Nulls	Notes
report_element_id	integer	No	Internal id for the report_element.
report_id	integer	No	Foreign key to the report table.
element_id	integer	No	Foreign key to the element table.
is_base	booleans	No	Identifies if the element is a base or extension element for the report.
primary_count	integer	Yes	The number of times the element is used on a. fact as a primary item in the report
dimension_count	integer	Yes	The number of times the element is used on a fact as a dimension in the report.
member_count	integer	Yes	The number of times the element is used on a fact as a dimension member in the report.

Unlike the accession_element table, elements that are not directly used on a fact (i.e. abstracts used as headings) are not in the report_element table. These elements are identified on the the dts_element table. See the accession_element backward compatibility section for more information about how the report_element table replaces the accession_element table.

Report document table

The report_document table partially replaces the accession_document_association table. The report_document table identifies the non DTS documents that make up the report. This will always include the instance or inline document for the report. the report_document table may include additional non XBRL documents (such as a text or html version of the report).

report_document table

Column	Type	Nulls	Notes
report_document_id	integer	No	Internal id for the report_document.
report_id	integer	No	Foreign key to the report table.

document_id	integer	No	Foreign key to the document table.

The documents that make up the DTS are in the dts_document table. See the accession_document_association backward compatibility section for more information.

DTS (Discoverable Taxonomy Set)

The database now supports identification of a DTS (discoverable taxonomy set). A DTS is the combination of the documents that support an instance. Prior to this change, there was no easy way to determine if two reports used the same DTS. For SEC reporting, this isn't very important because filers create an extension taxonomy for each report so no two filings ever refer to the same DTS.

In the previous version of the database, networks of relationships were associated with the accession. If multiple accessions used the same DTS, then there would be a separate set of networks and relationships for each report in the database (in the network table and the relationship table). This version of the database associates the networks to the DTS. In this model if two reports use the same DTS there would only be one copy of the networks and relationships and each report would refer to the same DTS.

DTS table

The dts table identifies each DTS in the database.

dts table

Column	Type	Nulls	Notes
dts_id	integer	No	Internal id for the dts.
dts_hash	byeta	No	Unique hash for the dts.
dts_name	text	Yes	Name of the DTS.

The dts_hash is a sha-224 digest of the sorted urls of the top level documents that make up the DTS. It is unique for each DTS. When a report is loaded, the hash is calculated for the report and checked against the dts table to determine if the DTS exists in the database.

A DTS can be loaded in two ways. It can be loaded as part of loading a report or can be directly loaded to the database. If the DTS is directly loaded, a name can be applied to the dts (contained in the dts_name column). This is done for commonly used DTSs such as the US GAAP taxonomy.

The report table includes a dts_id column to identify the DTS of the report. Footnotes present a issue in the model for the DTS. Footnote networks are contained in the instance/inline document and are therefore not technically part of a DTS. However, the DTS model associates the networks and relationships to the DTS. To handle this, when a report contains footnotes a second DTS is created in the database for the report. This DTS is identified in the entry_dts_id column for the report. The hash for the footnote DTS is the url of the instance/inline document. If two reports use the same DTS and each has footnotes, the dts_id for each report will be the same, however each report will have a separate entry_dts_id for the DTS row that contains the footnote networks.

DTS document table

The dts_document table identifies the files that make up a DTS.

dts_document table

Column	Type	Nulls	Notes
dts_document_id	integer	No	Internal id for the dts_document.
dts_id	integer	No	Foreign key to the dts table.
top_level	boolean	No	Identifies if the document is at the top of the DTS document tree.
document_id	integer	No	Foreign key to the document table.

The top_level column identifies if the document is part of the initial entry point of the DTS. The top level documents are the document that are directly referenced from the instance/inline document. Only the uris of top level documents are used to create the dts_hash (on the dts table) for the DTS.

The dts_document table partially replaces the accession_document_association table. Documents that are part of a report but not the DTS are in the report_element table. See the accession_document_association backward compatibility section for more information.

DTS element table

The dts_element table identifies the elements that are "used" by the DTS. An element is considered "used" by a DTS if it is included in at least one extended link in the DTS.

dts_element table

Column	Type	Nulls	Notes
dts_element_id	integer	No	Internal id for the dts_element.
dts_id	integer	No	Foreign key to the dts table.

element_id	integer	No	Foreign key to the element table.
is_base	boolean	No	Identifies if the element is a base or extension element for the DTS.
in_relationship	boolean	No	Identifies if the element is in a relationship in the DTS. Currently only elements that are included in at least one extended link in the DTS are included in the dts_element table. Therefore, this value is always true.

The dts_element table partially replaces the accession_element table. See the accession_element backward compatibility section for more information.

DTS and networks

Networks of relationship are now linked to the DTS instead of the accession. The dts_network and dts_relationship tables replace the former network and relationship tables, respectively. The dts_network table contains the same columns as the former network table except the accession_id is replaced with dts_id. The dts_relationship table has the same columns as the former relationship table except the network_id is replaced with the dts_network_id.

Joins between the dts_network table and the report table are done on the dts_id columns of the tables. This replaces using the accession_id column on the former accession table and network table.

In the previous version of the database, a query to get the list of networks for an accession:

```
SELECT n.*
FROM accession a
JOIN network n
  ON a.accession_id = n.accession_id
WHERE a.filing_accession_number = '0001144204-17-034457'
```

In the new version:

```
SELECT dn.*
FROM report r
JOIN dts_network dn
  ON r.dts_id = dn.dts_id
  OR r.entry_dts_id = dn.dts_id       --to include the footnote networks
WHERE r.source_report_identifier = '0001144204-17-034457'
```

In the new version of the query, the join between report and dts_network is on the dts_id.

DTS and taxonomy identification

The existing taxonomy and taxonomy_version tables are used to identify the taxonomy "families". These can be linked to the DTS via the taxonomy_version_dts table.

The terminology for DTS and taxonomy are often used interchangeable and can create some confusion. Here are some definitions that apply to the usage of these terms In the database.

DTS	The collection of taxonomy schema and linkbase files that support an instance/inline document. A small exception, is that there is a DTS that is the instance/inline document when there are footnotes. This is to support the network and resources for the footnotes in the database.
Taxonomy	In the database, this is a notion of taxonomy "family". For example, the US GAAP taxonomy is really a collection of taxonomies (US GAAP, dei, currency, country, …). This constitutes a family of taxonomies that are often used together.

The taxonomy_version_dts table, links the version of a taxonomy family to the DTSs that are part of that family.

taxonomy_version_dts table

Column	Type	Nulls	Notes
taxonomy_version_dts_id	integer	No	Internal id for the taxonomy_version_dts.
taxonomy_version_di	integer	No	Foreign key to the taxonomy_version table.
dts_id	integer	No	Foreign key to the dts table.

Documents

The document_type and target_namespace columns have been added to the document table. The document types are:

- schema
- linkbase
- instance
- inline
- report - text or html version of the report

If a document type cannot be determined, the document_type column will be null.

For schema documents, the target_namespace column contains the target namespace of the schema.

Document structure

The database now tracks the relationships between documents. A document can reference other documents via imports, includes, schemaRefs, roleRefs, arcroleRefs and locators. The document structure is the network by following these links.

NOTE: For filings loaded previous to this version of the database, the document structure is not loaded.

document_structure table

Column	**Type**	**Nulls**	**Notes**
document_structure_id	integer	No	Internal id for the document_structure..
parent_document_id	integer	No	Foreign key for the parent document to the document table.
child_document_id	integer	No	Foreign key for the child document to the document table.

For extracting the document structure see the dts_tree function.

Units

The former unit table contained units as they were defined in an instance or inline document. This meant that common units (i.e. USD) were added to the table for each accession. This created unnecessary duplication in the table and made it more difficult to determine if the same unit is used across multiple accessions, as facts from different accessions would have different unit_id values for the same unit.

To improve the handling of units, the unit and unit_measure tables were replaced by the unit_base, unit_measure_base and unit_report tables.

Unit base table

The unit_base table contains a single row for each unit of measure defined in any instance or inline document.

unit_base table

Column	**Type**	**Nulls**	**Notes**
unit_base_id	integer	No	Internal id for the unit_base.
unit_hash	bytea	No	A hash of the unit based on a canonical form of the unit (the hash_string).

unit_hash_string	character varying	No	The string before hashing.
unit_string	character varying	Yes	A presentational form of the unit.

The unit table has a unique index on the unit_hash column.

Unit measure base table

The unit_measure_base table identifies the numerator and denominator parts of the unit. It serves the same function as the former unit_measure table.

unit_measure_base table

Column	Type	Nulls	Notes
unit_measure_base_id	integer	No	Internal id for the unit_measure_base.
unit_base_id	integer	No	Foreign key to the unit table.
qname_id	integer	No	Foreign key to the qname table.
location_id	integer	Yes	Identifies the use of the unit part. 1 - measure - for units that don't have a numerator and denominator. 2 - numerator 3 - denominator

Unit report table

The unit_report_table identifies which units are used in a report. The combination of the unit_base and unit_report tables replace the former unit table.

unit_report table

Column	Type	Nulls	Notes
unit_report_id	integer	No	Internal id for the unit_report.
report_id	integer	No	Foreign key to the report table.
unit_base_id	integer	No	Foreign key to the unit_base table.
unit_xml_id	character varying	No	The ID used on the xml unit element in the instance.

Tuples

Tuple support has been added to the database. A tuple fact is stored on the fact table like any other fact. The tuple_fact_id column is used to identify the tuple parent of a fact. The null constraint for context_id on the fact table is dropped as tuple facts do not have a context.

The null constraint for period_type_id on the element table is dropped. This allows tuple facts, which do not have a period, to be stored on the fact table. The column is_tuple is added to the element table. This is a boolean column indicating if the element is a tuple element.

Fiscal based ultimus index

A fiscal based ultimus (and hash) has been added to the database via the columns fiscal_ultimus_index and fiscal_hash columns on the fact table. The ultimus indexes are used to rank repeatedly reported facts. The hashes are used to identify “equivalent” facts. There are 3 hashes: fact_hash, calendar_hash and now, the fiscal_hash. The hashes are the same except for how the period of a fact is included in the hash. The fact_hash uses the actual reported period of the fact. The calendar_hash uses the period as it fits into the calculated calendar period. The new fiscal_hash uses the the period as it fits into the calculated fiscal period.

Not all reported periods can be calculated into a calendar period and/or a fiscal period. These facts will not have a calendar and/or fiscal hash and will not be part of the calendar and/or fiscal ultimus rankings.

Resource table changes

The null constraint for the role_id of the resource table has been dropped. This is to support label, footnote or reference resources that do not have an explicit role.

Standard role definitions

The standard_role_definition table is added. This is a static table of role uris defined in the XBRL core spec with a label (short name) and definition. This is useful when label or reference resources to provide a human readable label are needed.

standard_role_definition table

Column	**Type**	**Nulls**	**Notes**
standard_role_definition_id	integer	No	Internal id for the standard_role_definition.
uri	text	No	The role uri.
label	text	Yes	A short human readable name for the label.

definition	text	Yes	The definition provided in the XBRL core specification.

Backward compatibility

Some tables in the former version of the database have been replaced by a new table or a combination of new tables. The replaced tables have been dropped from the database. For each dropped table, a view is created with the same name to allow backward compatibility. The following is a list of tables that have been replaced by views:

- accession
- accession_document_association
- accession_element
- accession_timestamp
- network
- relationship
- unit
- unit_measure

The use of these views should produce the same results as the former tables. However, it is possible that using the views may affect query performance. Rewriting the queries to use the new tables may improve performance.

accession_document_association

The accession_document_association table is replaced by a combination of the report_document and dts_document tables. The report_document table can be joined directly to the report table via the report_id columns on both tables. This will yield the non-DTS documents associated with the report, which includes the instance/inline document and possibly alternative versions of the report (i.e. a text or html version of the report).

The DTS documents for the report are identified by joining the report and dts_document tables using the dts_id columns on both tables. This will yield the DTS documents associated with the report.

The accession_document_association view is a union of querying the report_document and dts_document tables.

accession_element

The accession_element table is replaced by a combination of the report_element and the dts_element tables. The report_element table identifies the elements that are used on facts in the report with the associated counts (primary, dimension and member). The dts_element table identifies elements used by a DTS (used in a network in the DTS).

For a report, an element will usually appear on both the report_element table and the dts_element table (based on the dts_id of the report). The accession_element view unions the queries for the report_element and dts_element table and eliminates the duplicates.

Ordering of columns

Some existing queries may be affected by columns that were added to existing table. Queries that use SELECT * are particularly susceptible when the results are being handled by an application (i.e. Excel) or a program. The number and order of returned columns may be different from what is expected based on the previous version of the database.

Function Changes

dts_tree (new function)

The dts_tree returns a set of rows (table) showing the document hierarchy for a dts. It takes a dts id as a single parameter. The columns of the returned table are:

Column	Type	Notes
tree_order	integer	Order of the recursive traversal of the document structure. For sibling documents, the order is not guaranteed.
level	integer	The depth of the document in the document structure.
document_id	integer	Foreign key to the document table.
starts_loop	boolean	If true, this document is the beginning of a loop in the document structure. The children for this document are included in the result the first time the document is encountered in the document structure. Once the document is encountered further down in the structure the loop is detected and the children are not repeated.

Generally, this function is used in the FROM clause of a query. For example, to get the document structure for the DTS of the latest report loaded in the database:

```
SELECT dt.*
          ,repeat('   ', dt.level) || d.document_uri --indent the uris according to the
structure
FROM dts_tree((
                SELECT dts_id
                FROM report
```

```
                        WHERE report_id = (SELECT max(report_id) FROM report)
                )) dt
        JOIN document d
          ON dt.document_id = d.document_id
```

document_tree (new function)

The document_tree function works exactly like the dts_tree function except that it takes a document id as the starting point (instead of a DTS id).

document_navigate (new function)

The document_navigate function is the underlying function that is used by the dts_tree and document_tree functions.

ticker (updated)

The ticker function has been modified to better handle inline documents.

Appendix - List of affected tables and views

Name	Notes
accession	Dropped table. Replaced by view.
accession_document_association	Dropped table. Replaced by view.
accession_element	Dropped table. Replaced by view.
accession_timesteamp	Dropped table. Replaced by view.
base_namespace	Added column source_id
document	Added columns document_type and target_namespace
document_structure	New Table
dts	New Table
dts_document	New Table
dts_element	New Table
dts_network	New Table
dts_relationship	New Table
element	Added column is_tuple. Dropped not null constraint on the period_type_id column.
element_attribute	Dropped table. This table was never used.
entity_source	New table
fact	Added columns fiscal_hash, fiscal_ulitmus_index, unit_base_id. Dropped null constraint on column context_id.
namespace_source	New table
network	Dropped table. Replaced by view.
reference_part_type	Dropped table. This table was never used.
reference_resource	Dropped table. This table was never used.
relationship	Dropped table. Replaced by view.

report	New table
report_document	New Table
report_element	New Table
resource	Dropped “not null” constraint on the resource_role_id column.
source	New table
standard_role_definition	New Table
taxonomy_version_dts	New Table
unit	Dropped table. Replaced by view.
unit_base	New Table
unit_measure	Dropped table. Replaced by view.
unit_measure_base	New Table
unit_report	New Table