Contact

Invantive UniversalSQL Grammar

sqlBatch:

Compatibility

The Invantive implementation of SQL is based upon ANSI SQL, extended by aspects from popular SQL implementations such as PostgreSQL, MySQL, Oracle, Teradata and Microsoft SQL Server. It is topped of with Invantive-specific extensions, especially for procedural SQL, distributed SQL and distributed transactions. The basis is to implement functions such that as little as possible changes are necessary to run a SQL statement originating from another SQL implementation on Invantive UniversalSQL. For instance, to retrieve the current time you can use 'sysdate', 'now', 'getdate()' and 'sysdatetime' to name a few. The same holds for the procedural extension Invantive Procedural SQL, which reflects SQL/PSM and makes it easy to port Oracle PL/SQL or PostgreSQL PL/pgSQL statements.

Important Safety and Usage Information

Intended Use and Limitations: This software, developed by Invantive, is designed to support a variety of business and information technology data processing functions, such as accounting, financial reporting an sales reporting. It is important to note that this software is not designed, tested, or approved for use in environments where malfunction or failure could lead to life-threatening situations or severe physical or environmental damage. This includes, but is not limited to:

- Nuclear facilities: The software should not be used for operations or functions related to the control, maintenance, or operation of nuclear facilities.
- Defense and Military Applications: This software is not suitable for use in defense-related applications, including but not limited to weaponry control, military strategy planning, or any other aspects of national defense.
- Aviation: The software is not intended for use in the operation, navigation, or communication systems of any aircraft or air traffic control environments.
- Healthcare and Medicine Production: This software should not be utilized for medical device operation, patient data analysis for critical health decisions, pharmaceutical production, or medical research where its failure or malfunction could impact patient health.
- Chemical and Hazardous Material Handling: This software is not intended for the management, control, or operational aspects of chemical plants or hazardous material handling facilities. Any malfunction in software used in these settings could result in dangerous chemical spills, explosions, or environmental disasters.
- Transportation and Traffic Control Systems: The software should not be used for the control, operation, or management of transportation systems, including railway signal controls, subway systems, or traffic light management. Malfunctions in such critical systems could lead to severe accidents and endanger public safety.
- Energy Grid and Utility Control Systems: This software is not designed for the control or operation of energy grid systems, including electrical substations, renewable energy control systems, or water utility control systems. The failure of software in these areas could lead to significant power outages, water supply disruptions, or other public utility failures, potentially endangering communities and causing extensive damage.
- Other High-Risk Environments: Any other critical infrastructure and environments where a failure of the software could result in significant harm to individuals or the environment.

User Responsibility: Users must ensure that they understand the intended use of the software and refrain from deploying it in any setting that falls outside of its designed purpose. It is the responsibility of the user to assess the suitability of the software for their intended application, especially in any scenarios that might pose a risk to life, health, or the environment.

Disclaimer of Liability: Invantive disclaims any responsibility for damage, injury, or legal consequences resulting from the use or misuse of this software in prohibited or unintended applications.

Distributed SQL, Databases and Data Containers

It is easy to exchange and/or combine data across the supported platforms with data. To each platform (such as Salesforce or Exact Online Belgium) multiple connections can be active with the same or different platform-specific connection settings. Each open connection to a platform is named a 'data container'.

All opened connections together are named a 'database'.

When multiple data containers have been opened, each one has an alias to refer it by in Invantive UniversalSQL statements. For instance, a connection can be open for two different customer accounts on Exact Online Netherlands aliased as 'eolnl_comp1' and 'eolnl_comp55') and one for an Exact Online Belgium custom, aliased as 'eolbe_my_new_company'. The aliases can be freely chosen as long as they are valid identifiers and defined in the databases configuration file 'settings.xml'.

Service Providers

A number of special connections are always made, each of which can occur at most once. These are the 'service providers' such as:

- 'datadictionary': metadata of the current database, such as list of tables and executed SQL statements performance.
- 'os': information on the operating system running the SQL engine, such as reading file contents.
- 'smtp': synchronously send mails through SMTP.

Partitioning

Especially online platforms have a multi-tenant structure, in which the data is partitioned per customer, company or person. When the data model is identical across tenants, Invantive UniversalSQL considers them 'partitions'. SQL statements can run across multiple or one partitions, often in parallel. This enables consolidation scenarios across partitions (such as Exact Online or Nmbrs companies) as well as high-performance in MPP environments.

The partitions to be used can be specified with the 'use' statement, either through an explicit list of partitions to be selected across data containers, or through a SQL select statement returning the list of partitions to use. Please note that although the 'use' statement resembles the 'use DATABASE' statement on Microsoft SQL Server or PostgreSQL you can on Invantive UniversalSQL have multiple partitions active at the same time in one user session.

Identifiers

For identifiers, the regular conventions hold for the set of allowed characters. Depending on the platform, the identifiers are case sensitive or not. In general, it is best to assume that the identifier are case insensitive. There is no length limit on an identifier imposed by Invantive UniversalSQL.

Procedural SQL

Invantive Procedural SQL (or "PSQL" for short) is a procedural extension on top of Invantive UniversalSQL. It is based on the ISO-standard 9075-4:2016 (SQL/PSM) and extends Invantive UniversalSQL with procedural options like blocks, variables, conditional execution and loops. The procedural code is - together with the Invantive UniversalSQL contained - as a whole into pseudo-code and then executed.

The procedural code does not lean on the procedural options of the platforms being used, so it is easy to retrieve and change data in all supported cloud, file and database platforms. The pre-compiled procedural code does not perform context switches between procedural and SQL logic.

Licensing

Features

The available functionality of Invantive UniversalSQL features is based upon the licensed features. For instance the free implementation of Invantive UniversalSQL is limited to 15.000 rows and no access to group functions. Please consult the data dictionary contents for your license features.

Usage Fees

For paid products, the fee depends on a number of factors: users, devices and billable partitions. All fees depend on actual use. Additionally, the number and volume in KB of reads and writes is registered to enable communication with the platform partners on the performance of their platform.

A rough estimate is that the billable partitions reflect the product of number of legal entities times number of source systems. The number of billable partitions is determined as follows, where using twice or more often the same billable partition doesn't influence the number:

- Accessing a provider which does not have a concept of "partition" typically counts as one partition per different connection. A so-called data container ID is determined which reflects the backend connected to. For instance, connecting to my-ftp-host.com and another-ftp-host.com counts as two partitions. Examples of such providers are FTP, SQL Server and cbs.nl.
- Accessing a provider which has a concept of "partition" counts as the number of partitions used during the connection. For instance, using the Exact Online company 234 and 456 counts as two partitions. Examples of such providers are Exact Online, NMBRS, Loket and XML Audit File Financieel.
- Accessing data through xmltable, csvtable, internettable, jsontable and exceltable counts as the number of different parameter combination used. For instance, accessing an Excel range 'Sheet1' and a named range 'mydata' counts as two partitions.

settings.xml

The file settings.xml defines for a user or program the list of defined databases. Databases are grouped in 'database groups' for visual display. Database groups have no further functionality. Each database consists of one or multiple data containers.

The file 'settings.xml' is most often found on Microsoft Windows in your '%USERPROFILE%\invantive' folder, such as 'c:\users\john.doe\invantive\settings.xml'. It is shared across all Invantive UniversalSQL product installations for the user.

There are many scenarios to share database specifications across a user community, such as WAN-scenarios with Invantive Web Service, large corporate scenarios using DNS-entries as well as file shares, included files as well as single user solutions. Please involve a consultant when you want to deploy across thousands of users or more.

For user communities of up to 10 users, we recommend that company-specific settings are grouped per role in a separate file named 'settings-ROLE.xml' and placed in the default folder. Invantive UniversalSQL will automatically merge these files in the main settings.xml file.

Group Functions

The Invantive implementation of SQL is based upon ANSI SQL, extended by aspects from popular SQL implementations such as PostgreSQL, MySQL, Oracle, Teradata and Microsoft SQL Server. It is topped of with Invantive-specific extensions, especially for distributed SQL and distributed transactions. The basis is to implement functions such that as little as possible changes are necessary to run a SQL statement designed for use with another SQL implementation on Invantive UniversalSQL. For instance, to retrieve the current time you can use 'sysdate', 'now', 'getdate()' and 'sysdatetime' to name a few.

Popular group functions such as 'stddev' are available. However, currently you can not combine in one unnested SQL statement both group functions as well as expressions on the variables. In that case use an inner (nested) SQL statement to apply the expressions on the data, and execute the group functions in the outer SQL statement with the syntax 'select group() from (select ... from ...)'.

Locking

An Invantive UniversalSQL statement can work with many traditional and online platforms. There are no locking features on data and objects, since few online and traditional platforms connected provide these and the typical use of distributed transactions leave even less opportunity for data and object locking.

Transactions

Invantive UniversalSQL has limited support for transactions. DML is forwarded to a platform and depending on the platform an error can cause part of the work to be registered or everything to be rolled back. Within the SQL engine, multiple changes can be collected and forwarded to the platform at once. For instance, when creating an EDIFACT message you need to combine an invoice header with invoice lines into one EDIFACT message. Collection of multiple changes is done using the 'identified by' and 'attach to' syntax, optionally preceded by 'begin transaction'.

Format Masks

Operations such as to_char and to_date support a limited number of format masks:

- YYYY: 4-digit year.
- YYYYMM: 4-digit year, plus 2-digit number month within year.
- YYYYMMDD: 4-digit year, 2-digit month number within year, plus 2-digit day number within month.
- YYYYMMDDHH24MI: 4-digit year, 2-digit month number within year, 2-digit day number within month, 2-digit 24-hour clock hour number, plus 2-digit minutes number within hour.
- YYYYMMDDHH24MISS: 4-digit year, 2-digit month number within year, 2-digit day number within month, 2-digit 24-hour clock hour number, 2-digit minutes number within hour, plus 2-digit seconds number within minute.
- YYYY-MM-DD: 4-digit year, dash ('-'), 2-digit month number within year, dash ('-'), plus 2-digit day number within month.
- YYYY-MM-DD HH24:MI:SS: 4-digit year, dash ('-'), 2-digit month number within year, dash ('-'), plus 2-digit day number within month, 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.
- YYYY/MM/DD: 4-digit year, slash ('/'), 2-digit month number within year, slash ('/'), plus 2-digit day number within month.
- YYYY/MM/DD HH24:MI:SS: 4-digit year, slash ('/'), 2-digit month number within year, slash ('/'), plus 2-digit day number within month, 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.
- MMDDYY: 2-digit month number within year, 2-digit day number within month, plus 2-digit year.
- MMDDYYYY: 2-digit month number within year, 2-digit day number within month, plus 4-digit year.
- DDMMYYYY: 2-digit day number within month, 2-digit month number within year, 4-digit year.
- DD-MM-YY: 2-digit day number within month, dash ('-'), 2-digit month number within year, dash ('-'), 2-digit year.
- DD-MMM-YY: 2-digit day number within month, dash ('-'), short month name, dash ('-'), 2-digit year.
- DD/MM/YY: 2-digit day number within month, slash ('/'), 2-digit month number within year, slash ('/'), 2-digit year.
- DD/MMM/YY: 2-digit day number within month, slash ('/'), short month name, slash ('/'), 2-digit year.
- DD-MM-YYYY: 2-digit day number within month, dash ('-'), 2-digit month number within year, dash ('-'), 4-digit year.
- DD-MMM-YYYY: 2-digit day number within month, dash ('-'), short month name within year, dash ('-'), 4-digit year.
- DD/MM/YYYY: 2-digit day number within month, slash ('/'), 2-digit month number within year, slash ('/'), 4-digit year.
- DD/MMM/YYYY: 2-digit day number within month, slash ('/'), short month name within year, slash ('/'), 4-digit year.
- DD-MM-YYYY HH24:MI:SS: 2-digit day number within month, dash ('-'), 2-digit month number within year, dash ('-'), 4-digit year, space (' '), 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.
- DD/MM/YYYY HH24:MI:SS: 2-digit day number within month, slash ('/'), 2-digit month number within year, slash ('/'), 4-digit year, space (' '), 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.
- MM/DD/YYYY HH24:MI:SS: 2-digit month number within year, slash ('/'), 2-digit day number within month, slash ('/'), 4-digit year, space (' '), 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.

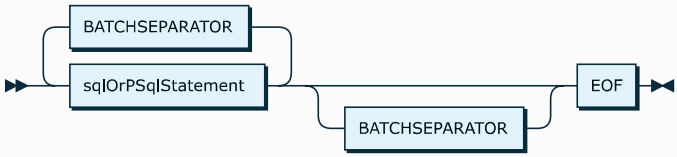
- DD/MMM/YYYY HH24:MI:SS: 2-digit day number within month, slash ('/'), short month name within year, slash ('/'), 4-digit year, space (' '), 2-digit hour on 24 hour clock, colon (':'), 2-digit minutes within hour, colon (':'), 2-digit seconds within minutes.
- YYYYIW: 4-digit year, plus ISO week number.
- YYYY-IW: 4-digit year, dash ('-'), plus ISO week number.
- IW: ISO week number.
- D: day of week according to ISO 8601.
- MMMM DD, YYYY HH24:MI TT: long month, 2-digit day number within month, comma (','), space (' '), 4-digit year number, space (' '), 2-digit hour on 24 hour clock, colon (':'), 2-digit minute within hour, space (' '), time zone.

Release History

The following significant changes were made to the Invantive UniversalSQL grammar:

- Release 17.29 (BETA) adds:
- 20181031: allow named expressions as table function parameters, using => as association operator.
- 20181123: add WHEN EXCEPTION to PSQL BEGIN..END block, with exception being WHEN OTHER or WHEN exception name.
- 20181123: extend SYNCHRONIZE statement.
- 20181224: extend SYNCHRONZIE statement with IGNORE NULLS clause.
- Release 17.30 (Production): All changes sofar.
- Release 17.31 (BETA) adds:
- 20190116: addition of language code parameter to TRANSLATE_RESOURCE.
- 20190117: addition of IS_NUMBER function.
- 20190124: addition of TO_BOOLEAN, IS_BOOLEAN, IS_DATE, IS_GUID functions.
- Release 17.32 (Production): All changes sofar.
- Release 17.33 (BETA) adds:
- 20190131: texts as partition names .
- 20190310: expression as billing id and reference.
- 20190322: select all columns 'except' syntax.
- 20190401: months_between SQL function.
- 20190401: excel_day SQL function.
- 20190403: sqlrowcount, sqlerrm and sqlcode functions.
- 20190410: 'exists' expression.
- 20190601: 'when obsolete within' variant of 'alter persistent cache refresh'.
- 20190605: gzip and ungzip functions.
- 20190624: 'alter session set roles' syntax for connectors with row-level security.
- 20190701: ascii_to_blob and unicode_to_blob functions.
- 20190706: 'alter session set iuid source' syntax.
- 20190821: add htmltable.
- 20191007: add 'except' to synchronize statement for insert and update.
- 20191008: add 'ignore changes to' to synchronize statement for update.
- 20191209: multiple PSQL batches in one statement.
- 20200102: add internettable.
- Release 17.34 (Production): All changes sofar.

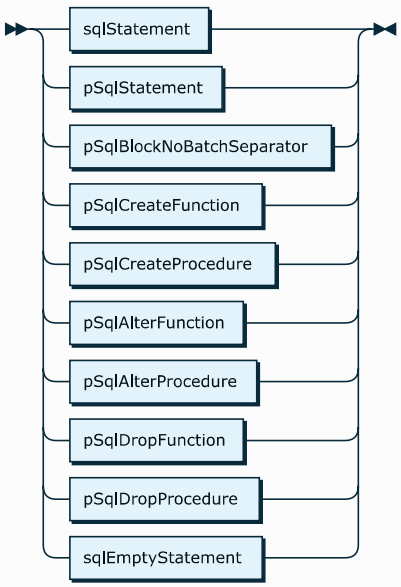
A SQL batch consists of one or more Invantive UniversalSQL and/or PSQL statements separated by the semi-colon batch separator (';').



no references

sqlOrPSqlStatement:

A number of SQL and PSQL statements can be used to compose a batch.



referenced by:

- sqlBatch

sqlEmptyStatement:

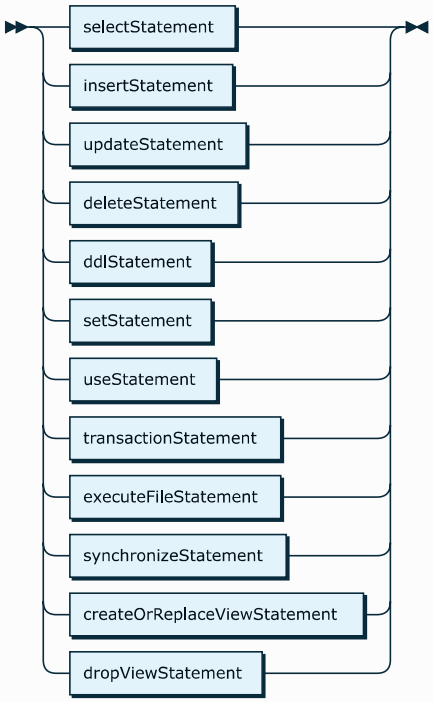
referenced by:

- sqlOrPSqlStatement

sqlStatement:

An Invantive UniversalSQL statement can retrieve or exchange data for many traditional and online platforms. Many platforms also support the use of DML (Data Manipulation Language) statements to change the data contained. On a few platforms you can execute DDL (Data Definition Language) statements to create new data structure or objects such as tables, procedures or sequences.

Popular Invantive UniversalSQL extensions are the synchronize statement to make sure multiple data sources contain the same data and the use statement to run SQL across multiple partitions.



referenced by:

- pSqlStatement
- sqlOrPSqlStatement

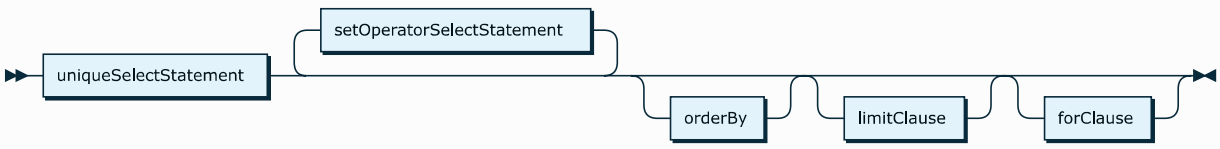
selectStatement:

A SQL select statement retrieves data from one or multiple data containers. A select statement can be composed of multiple data sets retrieved from many platforms, combined by set operators such as 'union'.

Often the performance of cloud platforms is less than traditional database platforms. With the 'limit' clause a limited number of rows can be retrieved quickly from a table or view after applying sorting as specified by the possibly present 'order by'. An alternative for a 'limit' clause is to use the 'top' clause.

A sequence of Invantive UniversalSQL statements, separated by the semi-colon separator character.

Each statement in the SQL batch will be executed consecutively. Execution will be stopped when an error occurs during execution of a statement.



referenced by:

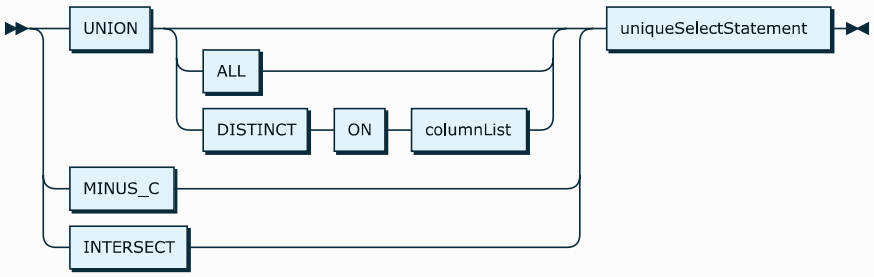
- createOrReplaceViewStatement
- createTableStatement
- embeddedSelect
- expression
- insertStatement
- pSqlForRecordLoopStatement
- sqlStatement
- useStatement

setOperatorSelectStatement:

SQL is based upon a solid mathematical foundation named 'set theory' with some exceptions. The set operators of Invantive UniversalSQL enable you to combine sets of data sets such as merging two sets of data. Please note that SQL actually uses 'bags', which opposed to 'sets', allow duplicates. To change bags of data into sets, either use 'distinct' or the 'union' set operator without 'all'. In general, the extensive use of 'distinct' signals bad database design.

The 'union' set operator returns the union of the data on the left and right side of the union while removing duplicate rows. The 'union all' set operator returns the union of the data on the left and right side of the union without removing duplicate rows. The 'minus' set operator returns all rows from the left side which do not occur in the right side. The 'intersect' set operator returns all rows that occur both in the left and right side.

The 'union' set operator has an extension to facilitate sieving overlapping data sets of decreasing quality using 'distinct on'. The 'union distinct on' set operator returns the union of the data sets listed, but only the first match on the column list is returned. The first match is defined as a fallthrough from left to right. In general, the preferred source of a set of data is listed first, followed by one or multiple 'union distinct on' set operators, each with a data set of less preference.

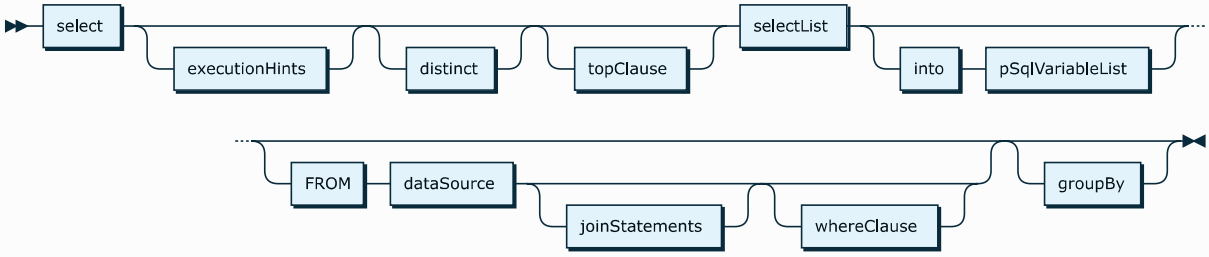


referenced by:

- selectStatement

uniqueSelectStatement:

Retrieves a data set from one or more data containers.

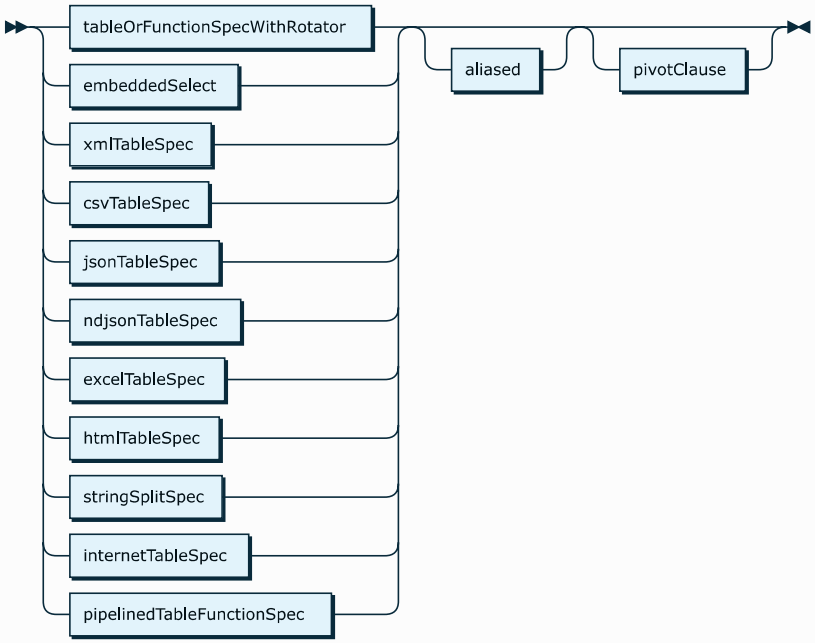


referenced by:

- selectStatement
- setOperatorSelectStatement

dataSource:

A data source can be a table, a table with parameters or a nested select (an 'inline view'). Also, a data source can be a binary or text string being interpreted as XML, CSV, JSON or binary Excel Open XML format.



referenced by:

- joinStatement
- uniqueSelectStatement

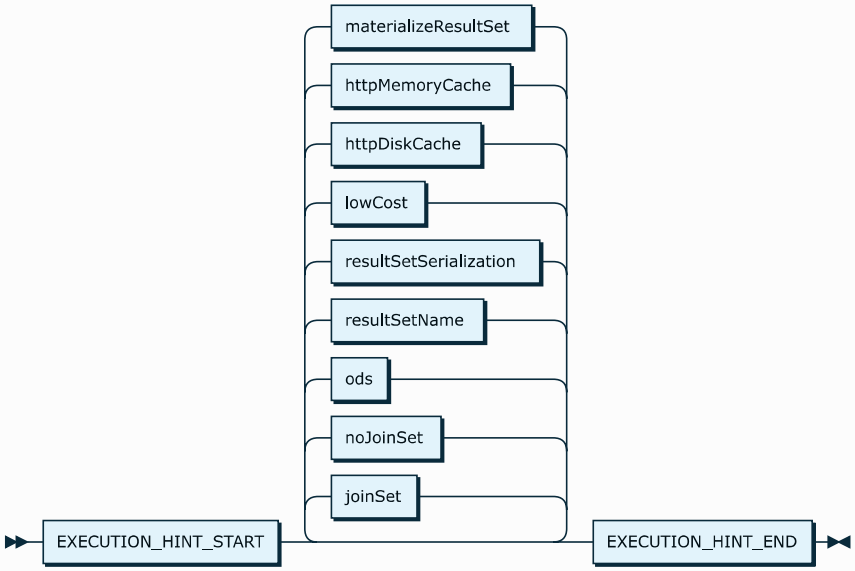
select:

referenced by:

- uniqueSelectStatement

executionHints:

Execution hints allow you to control individually the execution of SQL statements. Whenever possible, the hints will be used. In contrary to other platforms, Invantive UniversalSQL requires a hint to be valid according to the grammar when specified. This reduces the engineering risk that hints become invalid by accident.



referenced by:

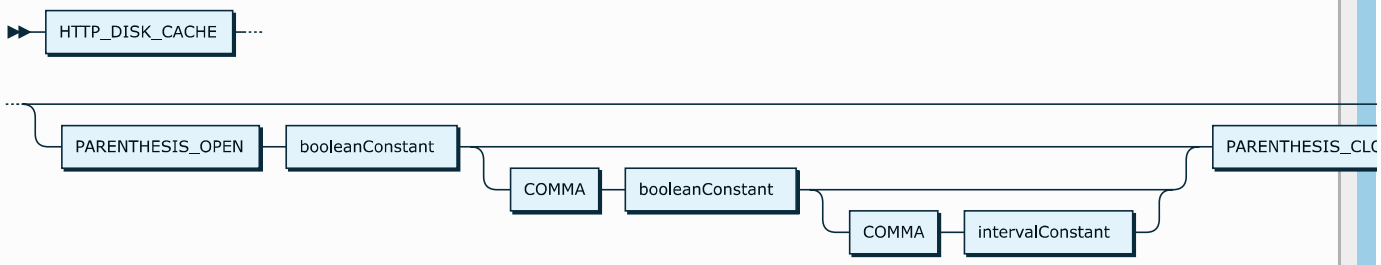
- deleteStatement
- uniqueSelectStatement
- updateStatement

httpDiskCache:

The http_disk_cache-hint specifies whether messages may be cached on disk when the provider uses HTTP to exchange data with the backing platform. This typically holds only for cloud-based platforms such as Exact Online, Teamleader or Salesforce. The default setting is false. The first parameter is a boolean whether data may be taken from the disk cache, the second parameter is a boolean whether data retrieved must be stored also in the disk cache and the third parameter is an interval that specifies the timespan before a disk cache hit found is to considered stale, such as "interval '4 hours'".

The use of the http_disk-cache-hint is recommended for data which is known to change seldom such as seeded or reference data. The contents of the disk cache are persistent across Invantive UniversalSQL sessions.

The disk cache is located in the Cache folder of the Invantive configuration folder.



referenced by:

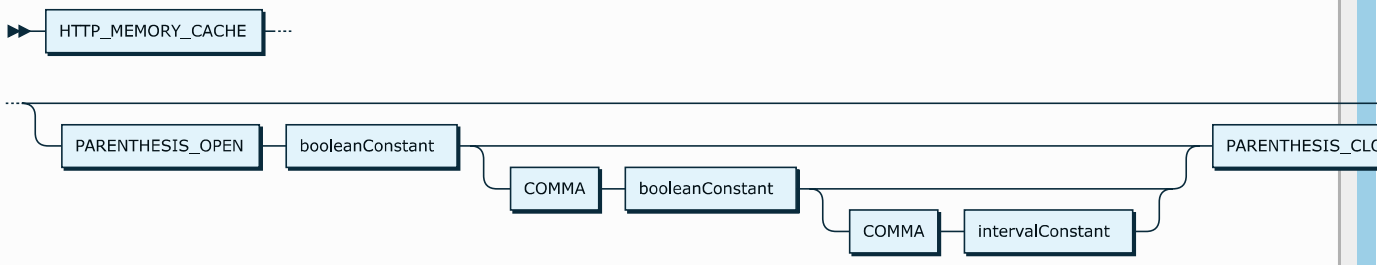
- executionHints

httpMemoryCache:

The http_memory_cache-hint specifies whether messages may be cached in memory when the provider uses HTTP to exchange data with the backing platform. This typically holds only for cloud-based platforms such as Exact Online, Teamleader or Salesforce. The default setting is false. The first parameter is a boolean whether data may be taken from the memory cache, the second parameter is a boolean whether data retrieved must be stored also in the memory cache and the third parameter is an interval that specifies the timespan before a memory cache hit found is to considered stale, such as "interval '4 hours'".

The use of the http_memory-cache-hint is recommended for data which is known to change seldom such as seeded or reference data. The contents in the memory cache are forgotten across Invantive UniversalSQL sessions.

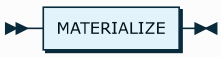
The memory cache is located in the Cache folder of the Invantive configuration folder.



referenced by:

- executionHints

materializeResultSet:



referenced by:

- executionHints

ods:

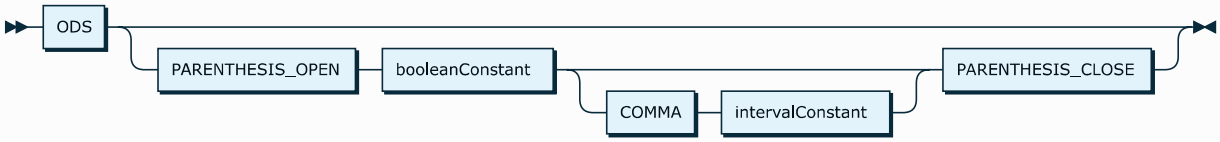
The ods-hint controls the use of the Invantive Data Cache stored in a relational database. The Invantive Data Cache is also the basis of the Operational Data Store managed by Invantive Data Replicator and the data warehouses managed by Invantive Data Vault. The ods-hint specifies the maximum age data from the data cache eligible for use.

The boolean specifies whether the Data Cache may be used to answer a query. Set it to false to disable use of Data Cache for the duration of the query. Keep it on the default true to use Data Cache.

The interval specifies the period of time during which cached results are considered sufficiently fresh for use, such as '30 minutes'.

When no interval is present, the actual platform is consulted. The default with Invantive Data Cache enabled is to always use the data cache contents when not stale according to the metadata of the data cache. In general, that defaults to a maximum age of 7 days.

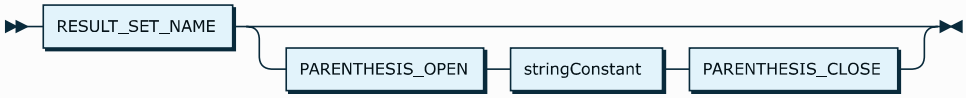
When use of data cache is requested and a filter is applied, all data will be retrieved from the actual platform and filtered on the Invantive UniversalSQL engine. This also applies when a filter is applied that can be forwarded to the actual platform. When the data is requested a limited number of times and the filter severely reduces the amount of data to be transported, it can be faster to not use the data cache.



referenced by:

- executionHints

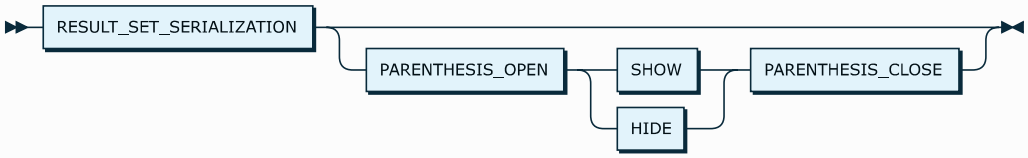
resultSetName:



referenced by:

- executionHints

resultSetSerialization:



referenced by:

- executionHints

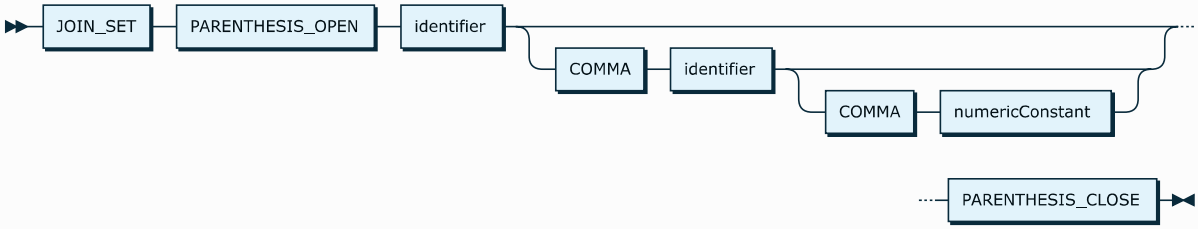
joinSet:

Control join approach between two data sources. A column-indexed lookup will be used instead of a full table scan when the number of rows on the left-hand side does not exceed the maximum number of rows specified in the hint. When not specified, a hash lookup will only be used when the number of rows on the left-side does not exceed 5.000.

The actual implementation of a hash lookup depends on the platform on which the data container runs. For instance with OData, a number of requests will be made using an in-construct with a limited number of in-values. With a relation database platform, a native SQL 'in' will be used.

The first identifier is the alias of the table on the right-hand side of the join. The second identifier is the name of the column used to join upon in the right-hand side. The numeric constant specifies upto what number of rows on the left-hand side of the join will allow the join set hint to be used. When the number of rows exceeds the numeric constant, a full table join is made.

The following example takes for instances 5.000 sales invoices from an Exact Online environment with 100.000 sales invoices. Each sales invoice has 4..10 lines. The join does not retrieve all sales invoices nor all invoice lines, but instead fetches the 5.000 sales invoices using the where-clause, and then retrieves the related invoice lines using a column-indexed lookup by invoiceid. Since Exact Online is an OData source, the approximately 30.000 invoice lines will be retrieves in 300 session I/Os each having an in-construct for 100 lines on invoiceid.

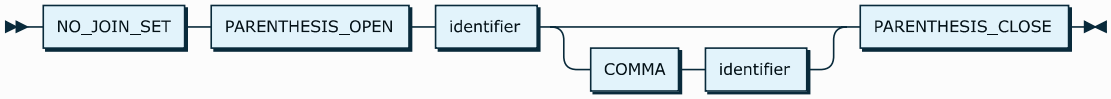


referenced by:

- executionHints

noJoinSet:

The no_join_set hint disables the use of hash-joins. It can be enabled using the join_set hint.



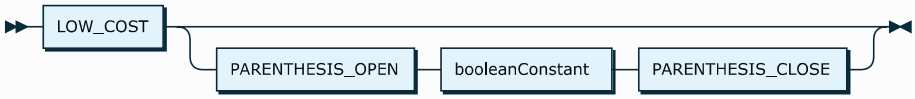
referenced by:

- executionHints

lowCost:

The low_cost-hint specifies that the select with the hint must be considered a select with low execution costs. Low execution costs trigger early evaluation during parsing. By default, select statements using solely in memory storage, dummy and data dictionary are considered low cost and evaluated early. The evaluation of all others is delayed as long as possible.

The use of the low_cost-hint is recommended when the select is used with a 'in (select ...)' syntax and the developer knows beforehand that it will evaluate fast to values and that the use of these values will allow the use of server-side filtering for the outer select.



referenced by:

- executionHints

distinct:

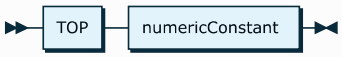
Addition of the 'distinct' keyword to a SQL select statement de-duplicates the rows returned. Rows are considered duplicates when the values in all selected columns are identical, with two null-values considered equal.

referenced by:

- avgAggregateFunction
- countAggregateFunction
- listAggAggregateFunction
- productAggregateFunction
- stdDevAggregateFunction
- sumAggregateFunction
- uniqueSelectStatement

topClause:

With the 'top' clause a limited number of rows can be retrieved quickly from a table or view after applying sorting as specified by the possibly present 'order by'.

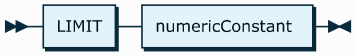


referenced by:

- uniqueSelectStatement

limitClause:

With the 'limit' clause a limited number of rows can be retrieved quickly from a table or view after applying sorting as specified by the possibly present 'order by'.

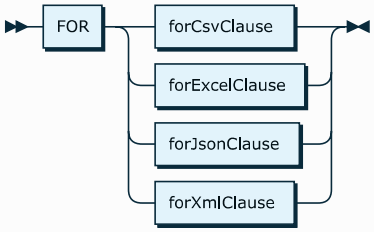


referenced by:

- deleteStatement
- selectStatement
- updateStatement

forClause:

The 'for' clause replaces the results from a query by a single row or a few rows, single column embedding the full results in the respective format.



referenced by:

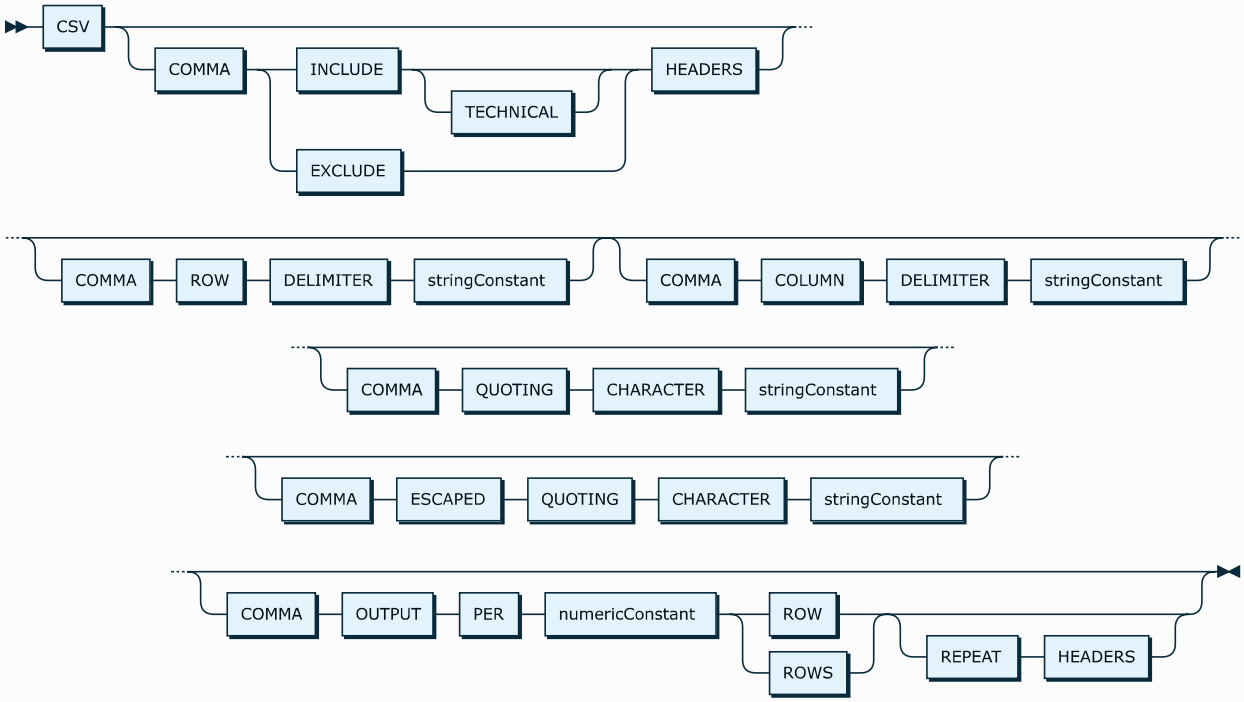
- selectStatement

forCsvClause:

The 'for csv' clause replaces the results from a query by few rows and asingle column named 'CSV' embedding the full results in an CSV format. Depending on the data volume, multiple rows can be returned. By default, each output row contains at most 1.000 rows from the query. The actual number of rows from the query can be set using 'OUTPUT PER ... ROWS'.

By default, the first output row starts with a technical header of the respective column names. The headers can be replaced by labels using 'INCLUDE HEADERS'. The headers can be fully disabled by adding 'EXCLUDE HEADERS'. The output of headers in each output row can be enabled using 'REPEAT HEADERS' with 'OUTPUT PER ... ROWS'. In that case, 'OUTPUT PER ... ROWS' must always be specified even with the default of 1.000 query rows per output row.

The CSV-output can use other texts than CR/LF for row delimiter, ',' for column delimiter, '"' for quoting character and '\"' for escaped quoting character. These can be set using, respectively, 'ROW DELIMITER', 'COLUMN DELIMITER', 'QUOTING CHARACTER' and 'ESCAPED QUOTING CHARACTER'.



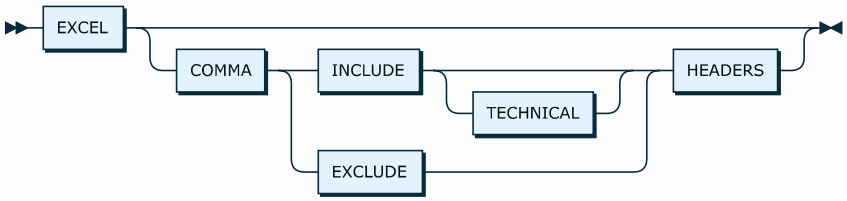
referenced by:

- forClause

forExcelClause:

The 'for excel' clause replaces the results from a query by a single row, single column named 'Excel' embedding the full results in an Excel xlsx-file format. The Excel xlsx-file has an Excel worksheet with a name (by default 'resultset1') with the rows.

By default, the first Excel worksheet starts with a header with all column names. No header is added when 'EXCLUDE HEADERS' is specified. Labels can be used as headers by specifying 'INCLUDE HEADERS'.



referenced by:

- forClause

forJsonClause:

The 'for json' clause replaces the results from a query by a single column named 'JSON' embedding the full results in a JSON format. Depending on the data volume, multiple rows can be returned. By default, each output row contains at most 1.000 rows from the query. The actual number of rows from the query can be set using 'OUTPUT PER ... ROWS'.

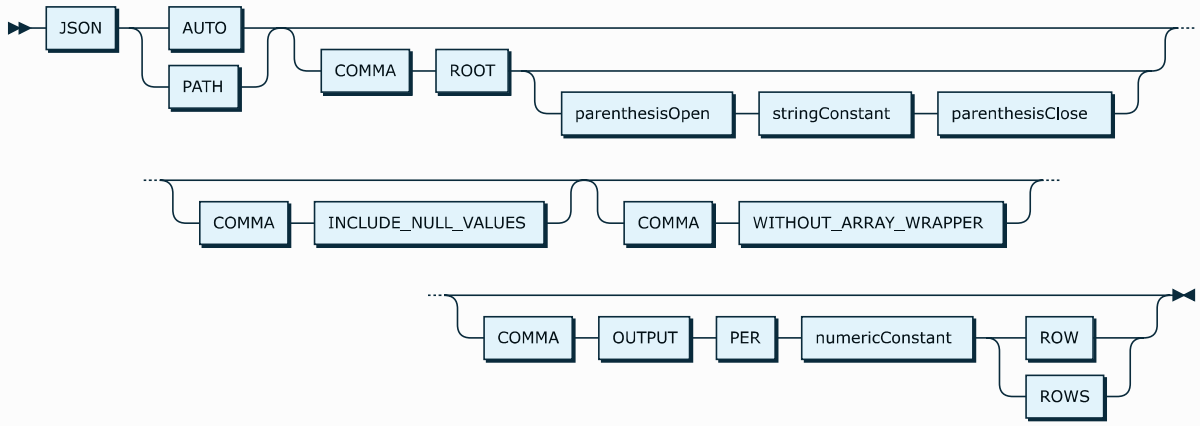
One of the naming configurations is 'AUTO'; the rows return one JSON object, with column names and their values as, respectively, JSON property names and values. A period ('.') in a column name will not influence the generated JSON differently from any other character. Casing of the column names will be reflected in the JSON property names. Casing is upper-case by default in Invantive UniversalSQL. Escape column names using square brackets to accurately specify casing of JSON property names.

As an alternative, the naming configuration can be switched to 'PATH'. With 'PATH', the JSON property names are identical to 'AUTO' except for a period ('.') in a column name. Each period in a column name introduces a new nesting level in the generated JSON.

A wrapper can be specified using 'ROOT'. The default root JSON property name is 'root'. A deviating name can be specified as a text constant between parentheses.

Properties are excluded for column null values. JSON properties are generated even for null values when 'INCLUDE_NULL_VALUES' is present.

The JSON objects are combined into a JSON array unless 'WITHOUT_ARRAY_WRAPPER' is present. With 'WITHOUT_ARRAY_WRAPPER' present, the output becomes NDJSON, where each individual output row represent a single source row.

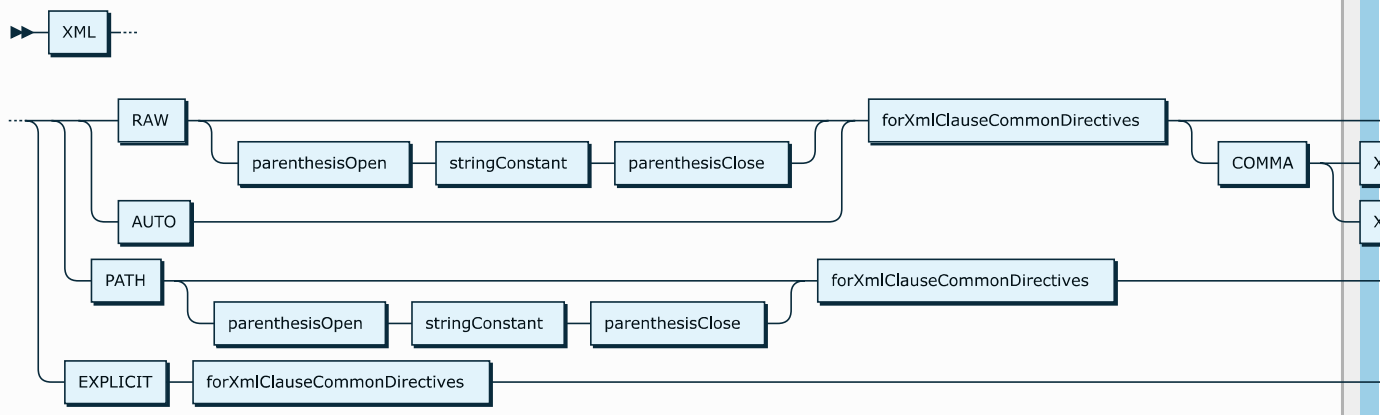


referenced by:

- forClause

forXmlClause:

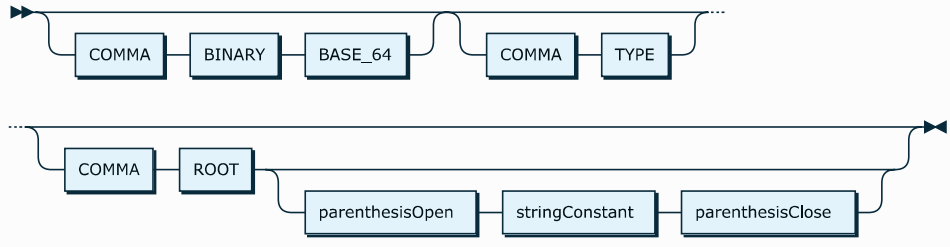
The 'for xml' clause replaces the results from a query by a single row, single column named 'XML' embedding the full results in an XML format.



referenced by:

- forClause

forXmlClauseCommonDirectives:



referenced by:

- forXmlClause

embeddedSelect:

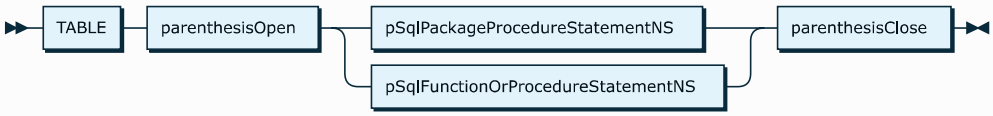
An embedded select, also known as an 'inline view', retrieves rows using the specified select statement. These rows are consumed by the outer select as were it the results of retrieving the rows from a table.

Invantive UniversalSQL does not allow grouping rows with expressions as columns. An embedded select is typically used to evaluate expressions to rows with solely constants. After applying the embedded select the group operators can be applied.

referenced by:

- dataSource

pipelinedTableFunctionSpec:



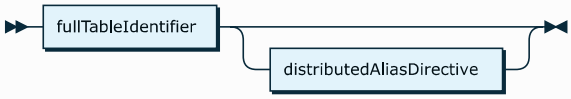
referenced by:

- dataSource

tableSpec:

A table specification without parameters. The optional alias after the at-sign specifies a specific data source to be used, such as 'exactonlinerest..journals@eolbe' specifying the use of Exact Online Belgium when 'eolbe' is associated by the database definitions in settings.xml with Exact Online Belgium.

A number of special so-called 'service providers' are always present, such as 'datadictionary' for use by an alias.



referenced by:

- alterPersistentCacheConfigureWebhooksStatement
- alterPersistentCacheDropStatement
- alterPersistentCacheSetTableOptions
- alterPersistentCacheTableRefreshStatement
- applyToClause
- createTableStatement
- dropTableStatement
- synchronizeStatement

tableSpecWithRotator:

referenced by:

- deleteStatement
- insertStatement
- updateStatement

tableOrFunctionSpec:

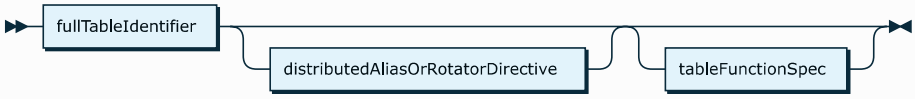
A table specification requiring a comma-separated list of parameters to determine the rows to be retrieved.

Traditional SQL syntax did not provide for parameterized queries, matching set theory. Modern variants such as pipelined table functions allow a stored procedure or other imperative language-based approaches to generate rows based upon parameter values. Many data containers support queries that returns rows based upon parameter values. This holds especially for SOAP web services. Table specifications with parameters ease queries on such data containers.

The optional alias after the at-sign specifies a specific data source to be used, such as 'exactonlinerest..journals@eolbe' specifying the use of Exact Online Belgium when 'eolbe' is associated by the database definitions in settings.xml with Exact Online Belgium.

no references

tableOrFunctionSpecWithRotator:



referenced by:

- dataSource

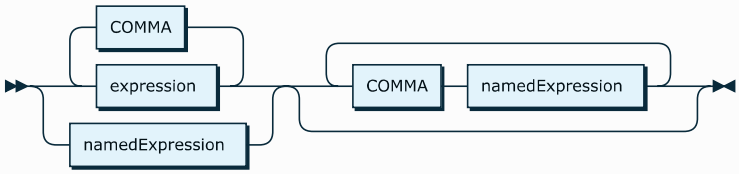
tableFunctionSpec:

A list of parameter value expressions identified by position or name to determine the rows to be retrieved by a tableOrFunctionSpec.

referenced by:

- tableOrFunctionSpec
- tableOrFunctionSpecWithRotator

numberedOrNamedExpressionList:

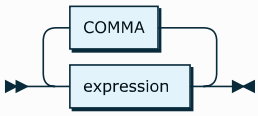


referenced by:

- pSqlFunctionOrProcedureStatementNS
- pSqlPackageProcedureStatementNS
- pSqlRaiseStatement
- tableFunctionSpec

expressionList:

An ordered comma-separated list of value expressions.

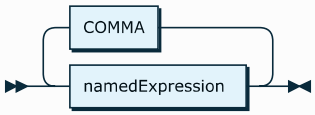


referenced by:

- functionExpression
- listAggAggregateFunction
- zipAggregateFunction

namedExpressionList:

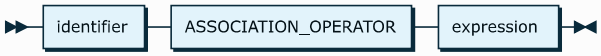
An unordered list of value expressions, identified by the parameter name.



no references

namedExpression:

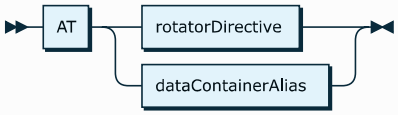
A value expression, identified by the parameter name, the association operator '=>' and the value expression.



referenced by:

- namedExpressionList
- numberedOrNamedExpressionList

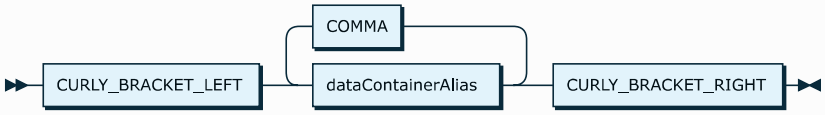
distributedAliasOrRotatorDirective:



referenced by:

- tableOrFunctionSpecWithRotator
- tableSpecWithRotator

rotatorDirective:



referenced by:

- distributedAliasOrRotatorDirective

distributedAliasDirective:

The distributed alias after the at-sign specifies a specific data source to be used, such as 'exactonlinerest..journals@eolbe' specifying the use of Exact Online Belgium when 'eolbe' is associated by the database definitions in settings.xml with Exact Online Belgium.

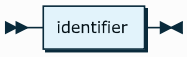
A number of special so-called 'service providers' are always present, such as 'datadictionary' for use by an alias.

referenced by:

- pSqlPackageProcedureStatementNS
- partitionIdentifierWithAlias
- setIdentifier
- tableOrFunctionSpec
- tableSpec

dataContainerAlias:

When multiple data containers have been defined in settings.xml for a database, each one is assigned an alias. An alias typically takes the form of a limited number of characters. The presence of an alias allows Invantive UniversalSQL to precisely determine to what data container forward a request for data.

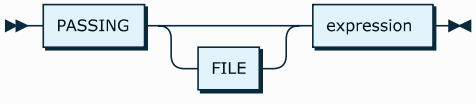


referenced by:

- alterPersistentCacheRefreshStatement
- distributedAliasDirective
- distributedAliasOrRotatorDirective
- rotatorDirective

passingSourceOrPathExpression:

The passing option specifies the source of the contents to be interpreted. The contents can be specified as the outcome of an expression such as from a previous read_file() table function, a URL downloaded using httpget() or a string concatenation. The contents can also be specified as to be taken from a specific file identified by it's file name and path as an expression.



referenced by:

- csvTablePassing
- excelTablePassing
- htmlTablePassing
- jsonTablePassing
- xmlTablePassing

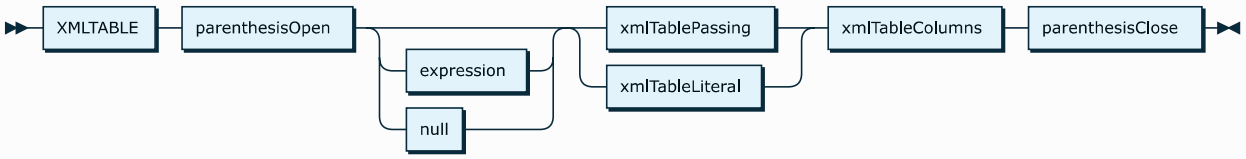
xmlTableSpec:

Data stored in XML format can be interpreted as a data source using the xmltable keyword.

The expression specifies a master XPath expression within the context of which the rows are evaluated using the column specifications.

The passing option specifies the source of the data in XML format. The source is often the outcome of a read_file() table function, a URL download using httpget() or a previous xmlformat() SQL function.

The columns are specified using their XPath relative to the master path.



referenced by:

- dataSource

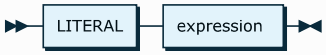
xmlTablePassing:

referenced by:

- xmlTableSpec

xmlTableLiteral:

A literal value containing a valid XML document.

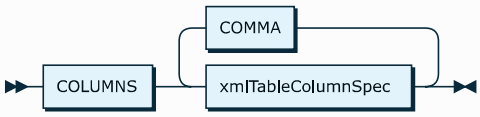


referenced by:

- xmlTableSpec

xmlTableColumns:

A list of XML table column specifications.

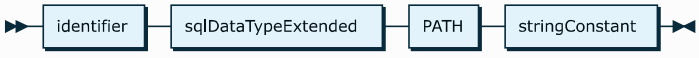


referenced by:

- xmlTableSpec

xmlTableColumnSpec:

The columns are specified using their XPath relative to the master path.



referenced by:

- xmlTableColumns

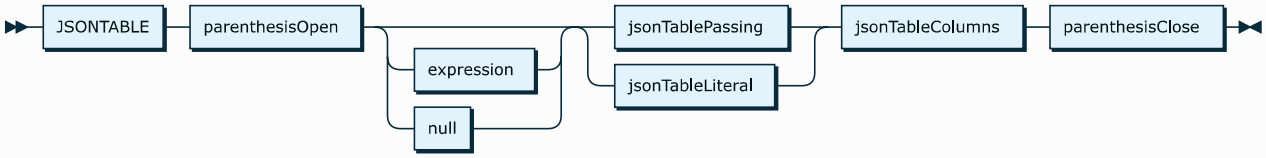
jsonTableSpec:

Data stored in JSON format can be interpreted as a data source using the jsontable keyword.

The expression specifies a master JSON expression within the context of which the rows are evaluated using the column specifications.

The passing option specifies the source of the data in JSON format. The source is often the outcome of a read_file() table function or a URL download using httpget().

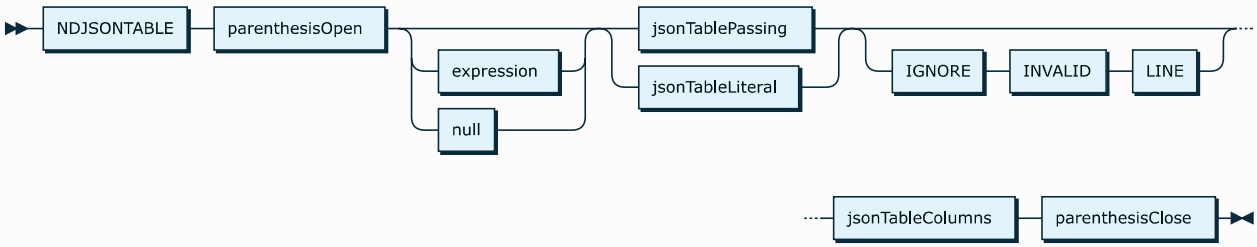
The columns are specified using their JSON path relative to the master path.



referenced by:

- dataSource

ndjsonTableSpec:



referenced by:

- dataSource

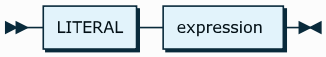
jsonTablePassing:

referenced by:

- jsonTableSpec
- ndjsonTableSpec

jsonTableLiteral:

A literal value containing a valid JSON document.

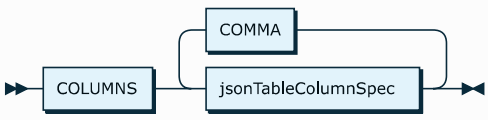


referenced by:

- jsonTableSpec
- ndjsonTableSpec

jsonTableColumns:

A list of JSON table column specifications.

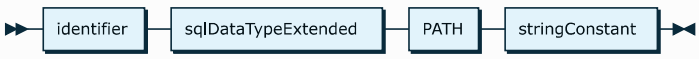


referenced by:

- jsonTableSpec
- ndjsonTableSpec

jsonTableColumnSpec:

The columns are specified using their JSON path relative to the master path.



referenced by:

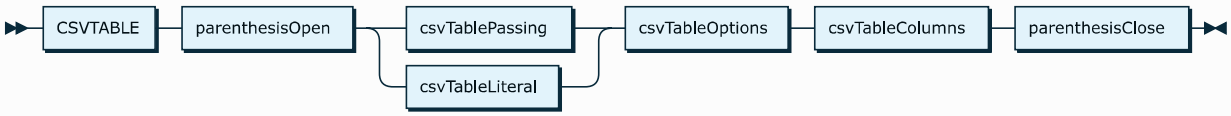
- jsonTableColumns

csvTableSpec:

Data stored in CSV format can be interpreted as a data source using the csvtable keyword.

The passing option specifies the source of the data in CSV format. The source is often the outcome of a read_file() table function or a URL download using httpget().

The interpretation process can be controlled on table level using the table options and the specification ends with the CSV columns being mapped to data source columns using their relative position within a row.



referenced by:

- dataSource

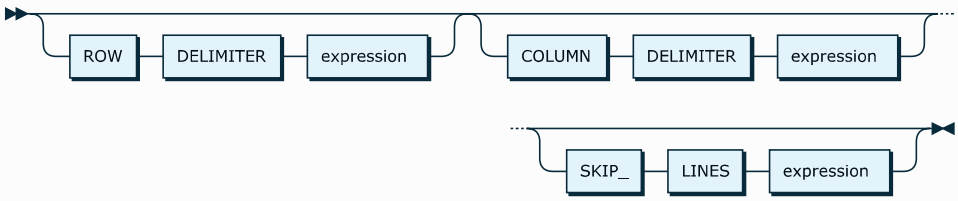
csvTableOptions:

The interpretation process can be controlled on table level using the table options.

The row delimiter is a text that separates two CSV rows, such as "chr(13) || chr(10)" or simply a pipe character "'|'". The default is the operating system-specific variant of new line.

The column delimiter is a text that separates two CSV columns such as "';'". The default is comma.

Data in CSV will typically have one or more header CSV rows labeling the individual columns. The 'skip lines' statement excludes the first number of CSV rows from processing.

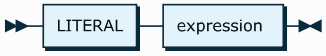


referenced by:

- csvTableSpec

csvTableLiteral:

A literal value containing a valid CSV document.



referenced by:

- csvTableSpec

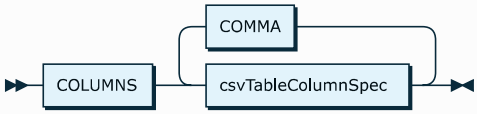
csvTablePassing:

referenced by:

- csvTableSpec

csvTableColumns:

A list of CSV table column specifications.

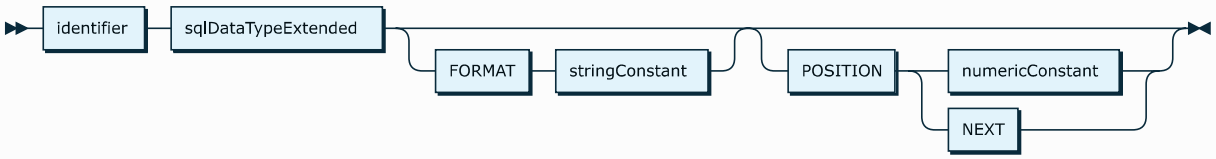


referenced by:

- csvTableSpec

csvTableColumnSpec:

Each CSV column is mapped to a data source column using it's relative position within the CSV row. Position 1 represents the first CSV column. A position can be represented using an integer, or by specifying the keyword 'NEXT'. The keyword 'NEXT' specifies that the column is located one position higher than the previous column. When there is no previous column, the column is located at position 1.



referenced by:

- csvTableColumns

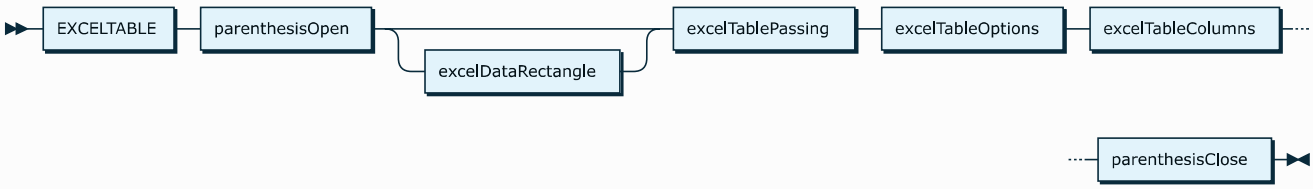
excelTableSpec:

Excel file contents in Open XML, such as used with the file extensions 'xlsx' and 'xlsm', can be interpreted as a data source using the exceltable keyword.

The rectangle specifies the rectangular area from which should be taken. The first worksheet is used when no rectangle is specified or a null value as worksheet name. Rows in Excel are interpreted as rows from the data source, whereas columns in Excel are interpreted as columns.

The passing option specifies the source of the (binary) contents in zipped Open XML format such as used with the file extensions 'xlsx' and 'xlsm'. The source is often the outcome of a read_file() table function or URL download using httpget().

The interpretation process can be controlled on table level using the table options and the specification ends with the Excel columns being mapped to data source columns using their relative position within the rectangular area.



referenced by:

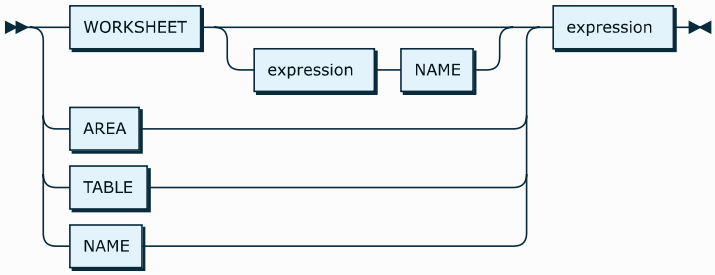
- dataSource

excelDataRectangle:

The rectangle specifies the rectangular area from which data should be taken. Rows in Excel are interpreted as rows from the data source, whereas columns in Excel are interpreted as columns. All cells within the rectangle are considered data; headings should be excluded from the rectangle specification.

A rectangle can be either:

- The contents of a complete worksheet, specified by an expression returning the worksheet name. Specify null to select the first worksheet independent of name. Alternatively, a worksheet can be selected by specifying an integer indicating the relative position, including hidden worksheet.
- A named range within a specific worksheet, specified by expressions for optional worksheet name and named range name.
- A cell range identified by an expression with it's dimensions including optional worksheet name and a range definition. The range definition can have various forms: 'A1:Z999' or '$A$1:$Z$999' for an area, 'A:Z' or '1:26' for all cells in column(s) or 'A1' to refer to a specific cell.
- An Excel table identified by an expression with the table name.
- A named range identified by an expression with the named range name.
- Unspecified, selecting the first worksheet independent of name.



referenced by:

- excelTableSpec

excelTablePassing:

referenced by:

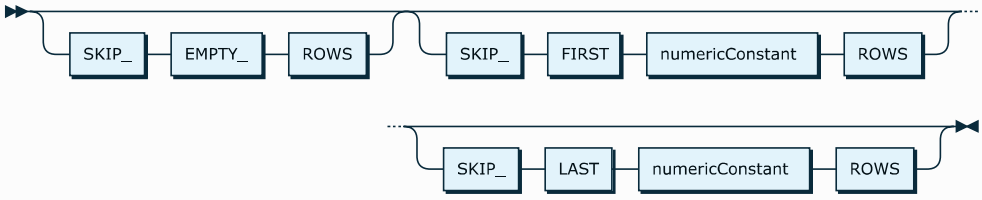
- excelTableSpec

excelTableOptions:

The interpretation process can be controlled on table level using the table options.

A position can be represented using an integer, or by specifying the keyword 'NEXT'. The keyword 'NEXT' specifies that the column is located one position higher than the previous column. When there is no previous column, the column is located at position 1.

Empty rows will typically be found when consuming a rectangular area larger than the actual data, such a complete worksheet. The 'skip empty rows' statement eliminates all completely empty rows from the output.

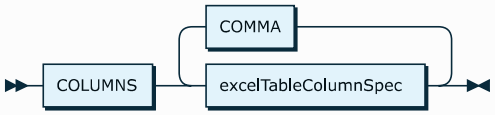


referenced by:

- excelTableSpec

excelTableColumns:

A list of Excel table column specifications.

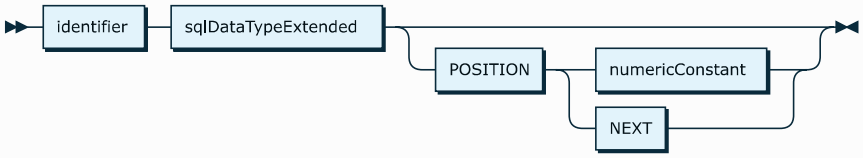


referenced by:

- excelTableSpec

excelTableColumnSpec:

Each Excel column is mapped to a data source column using it's relative position within the rectangular area. Position 1 represents the first Excel column falling within the Excel rectangular area.



referenced by:

- excelTableColumns

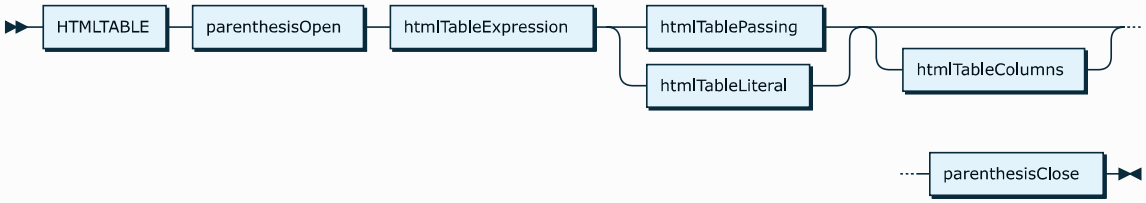
htmlTableSpec:

Data lossy stored in HTML format can be interpreted as a data source using the htmltable keyword.

The expression specifies a master path within the context of which the rows are evaluated using the column specifications.

The passing option specifies the source of the data in HTML format. The source is often the outcome of a read_file() table function or a URL download using httpdownload(). Full adherence to the HTML format is not required; the parser often automatically corrects many HTML format errors.

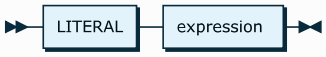
The columns are specified using their path relative to the master path.



referenced by:

- dataSource

htmlTableLiteral:



referenced by:

- htmlTableSpec

htmlTablePassing:

referenced by:

- htmlTableSpec

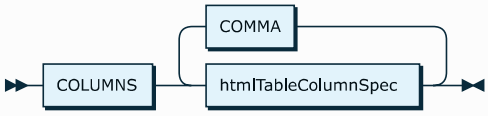
htmlTableExpression:

referenced by:

- htmlTableSpec

htmlTableColumns:

A list of HTML table column specifications.

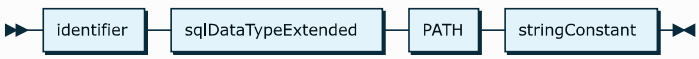


referenced by:

- htmlTableSpec

htmlTableColumnSpec:

Each HTML table column is mapped to a data source column using the path relative to the master path.



referenced by:

- htmlTableColumns

stringSplitSpec:

Splits a text upon another text, return one row per element.

Parameters:

- Input: The text to split.
- Splitter: Text being split upon.

Returns: a number of rows with one column named 'value'; one per element in the input.

referenced by:

- dataSource

internetTableSpec:

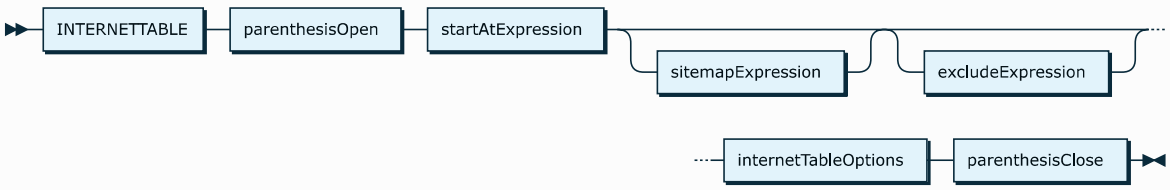
Data stored on the Internet accessible through the HTTP protocol can be interpreted as a data source using the internettable keyword. A depth-first scan is done in which solely unique URLs are returned. Starting of at one starting URL, URLs are downloaded from the Internet consisting of webpages and content. Contents is made available with no further deeper inspection. Webpages in HTML format are scanned for more URLs by default for the following paths:

- //a[@href]: all hrefs in anchors;
- //script[@src]: all sources of scripts;
- //link[@href]: all hrefs of links.
- //img[@src]: all hrefs of images.

The startAtExpression specifies the initial webpage to retrieve data for.

A pre-defined list of columns is available per retrieved URL:

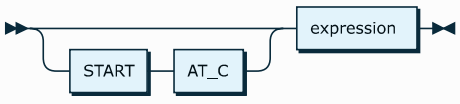
- URL: URL of page;
- Contents_char: the character contents, converted from the original character set into UTF-8;
- Contents_blob: the binary contents;
- Mime_type: MIME-type returned by the web server;
- Http_status_code: numeric HTTP response status code;
- Date_retrieval_utc: date/time when the response was received (UTC);
- Retrieval_duration_ms: time between the request and complete response in milliseconds;
- Bytes_retrieved: number of bytes retrieved;
- Depth: recursion depth, starting at 1 for the initial URL;
- Retrieval_successful: indicator whether the response was completely successful retrieved;
- Last_modified: date/time when the response's content was last modified;
- Etag: ETAG on the content as returned by the web server;
- Content_disposition: preferred file name and encoding to be used;
- Cache_Control: contents of cache-control HTTP response header;
- Expires: contents of the Expires HTTP response header;
- Error_message_code: Invantive UniversalSQL engine error message code if any occurred;
- Error_message_text: Invantive UniversalSQL engine error message code if any occurred.



referenced by:

- dataSource

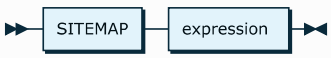
startAtExpression:



referenced by:

- internetTableSpec

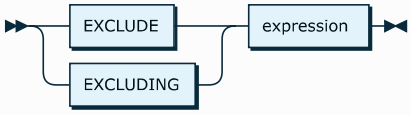
sitemapExpression:



referenced by:

- internetTableSpec

excludeExpression:



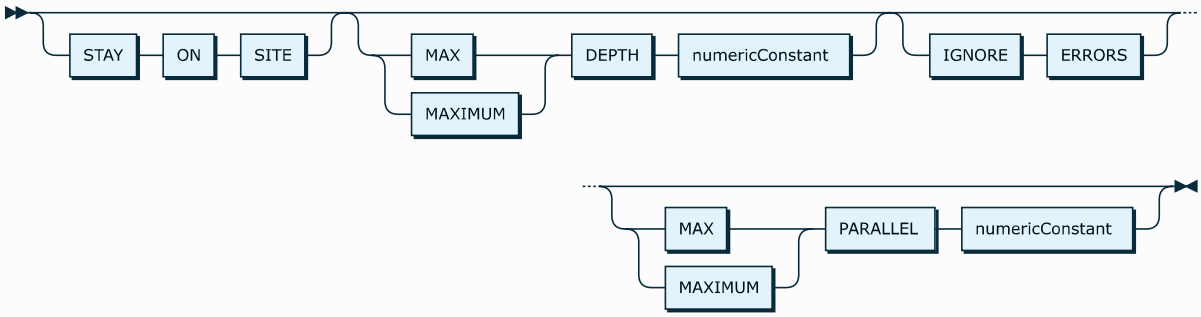
referenced by:

- internetTableSpec

internetTableOptions:

The process can be controlled using options:

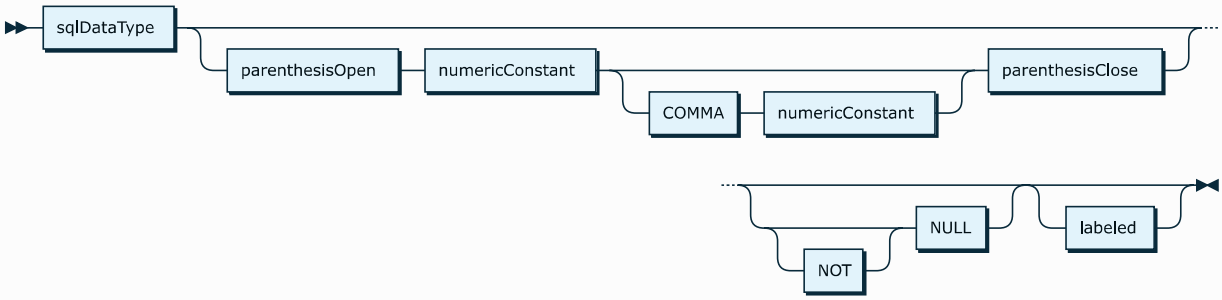
- Stay on site: when present, recursion restricts to URLs on the same host name as the starting URL. Default behaviour is to branch out to other sites too.
- Maximum depth: limit the depth of the recursion to a specific number.
- Ignore errors: do not stop on the first error but continue. Default behaviour is to stop on the first error or result completion, whatever comes first.



referenced by:

- internetTableSpec

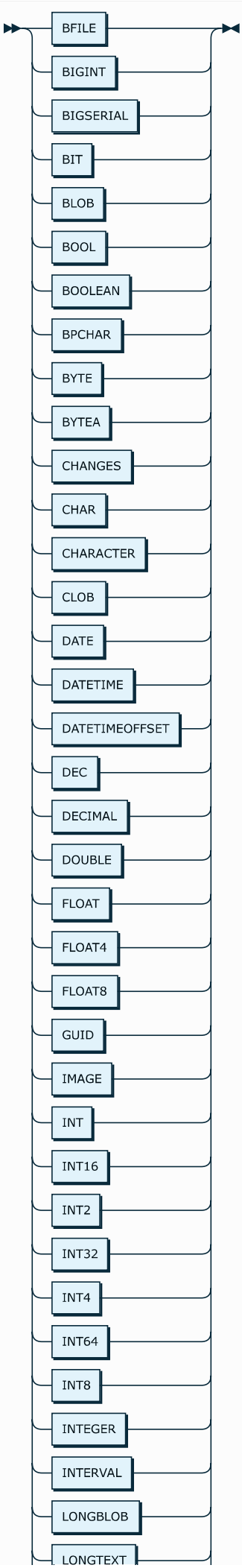
sqlDataTypeExtended:

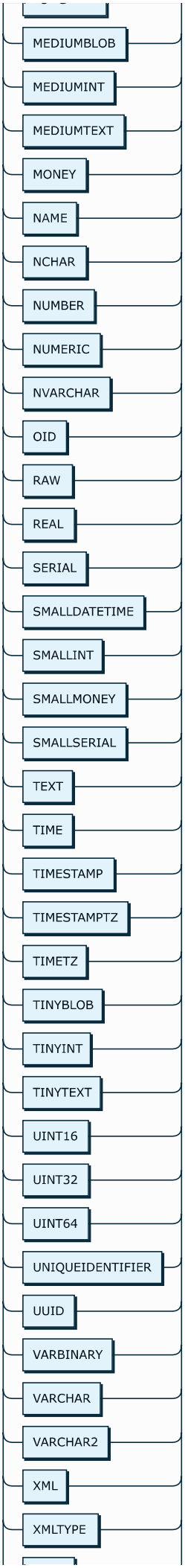


referenced by:

- castExpression
- createTableArgument
- csvTableColumnSpec
- excelTableColumnSpec
- htmlTableColumnSpec
- jsonTableColumnSpec
- xmlTableColumnSpec

sqlDataType:


BFILE
BIGINT
BIGSERIAL
BIT
BLOB
BOOL
BOOLEAN
BPCHAR
BYTE
BYTEA
CHANGES
CHAR
CHARACTER
CLOB
DATE
DATETIME
DATETIMEOFFSET
DEC
DECIMAL
DOUBLE
FLOAT
FLOAT4
FLOAT8
GUID
IMAGE
INT
INT16
INT2
INT32
INT4
INT64
INT8
INTEGER
INTERVAL
LONGBLOB
LONGTEXT


MEDIUMBLOB
MEDIUMINT
MEDIUMTEXT
MONEY
NAME
NCHAR
NUMBER
NUMERIC
NVARCHAR
OID
RAW
REAL
SERIAL
SMALLDATETIME
SMALLINT
SMALLMONEY
SMALLSERIAL
TEXT
TIME
TIMESTAMP
TIMESTAMPTZ
TIMETZ
TINYBLOB
TINYINT
TINYTEXT
UINT16
UINT32
UINT64
UNIQUEIDENTIFIER
UUID
VARBINARY
VARCHAR
VARCHAR2
XML
XMLTYPE

referenced by:

- pSqlDataType
- sqlDataTypeExtended

groupBy:

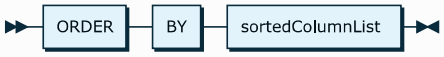
Grouping of multiple rows into groups is specified by the groupBy. A group will be introduced for each distinct combination of column values for the columns listed. The values of grouped columns can be used in the select clause. Columns not being grouped upon can only be used within the context of a group function listed as 'aggregateFunction'.

referenced by:

- uniqueSelectStatement

orderBy:

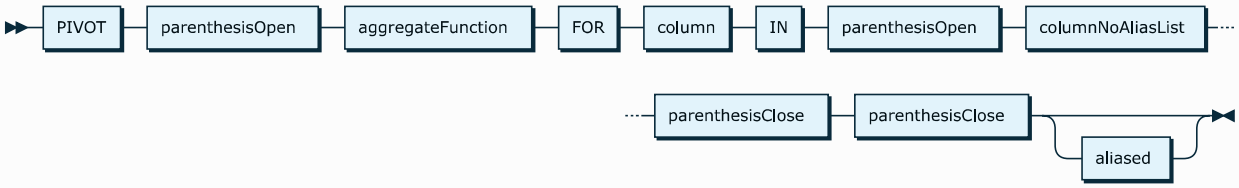
Sort the rows returned as specified by the list of columns. Values are either sorted ascending (the default) or descending.



referenced by:

- listAggAggregateFunction
- selectStatement

pivotClause:

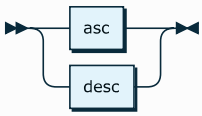


referenced by:

- dataSource

sortDirection:

A sort direction can be either 'asc' for 'ascending' (the default) or 'desc' for 'descending'.

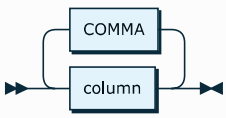


referenced by:

- sortedColumn

columnList:

A comma-separated list of columns.



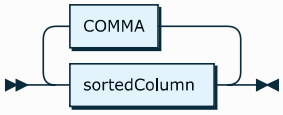
referenced by:

- allColumnsSpecId
- groupBy
- identifiedByMultipleClause
- insertFieldList
- setOperatorSelectStatement
- synchronizeInsertStatement

- synchronizeUpdateStatement

sortedColumnList:

An ordered comma-separated list of column values to sort upon.

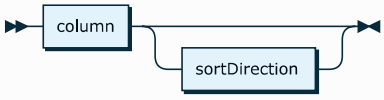


referenced by:

- orderBy
- resolveByClause

sortedColumn:

A column values to sort upon.

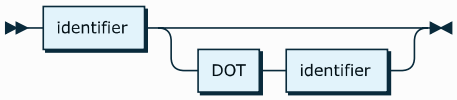


referenced by:

- sortedColumnList

column:

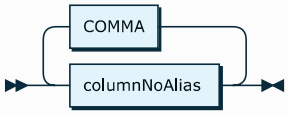
A column is identified by an identifier, possibly prefixed by the name of the table or the alias of the table from which the column is to be taken.



referenced by:

- columnList
- pivotClause
- sortedColumn
- updateValue

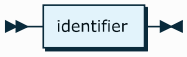
columnNoAliasList:



referenced by:

- pivotClause

columnNoAlias:

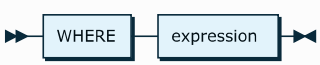


referenced by:

- columnNoAliasList

whereClause:

The where-clause restricts the number of rows in a result set by applying one or more boolean condiditions which rows must satisfy.

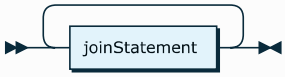


referenced by:

- deleteStatement
- uniqueSelectStatement
- updateStatement

joinStatements:

A list of join statements.



referenced by:

- uniqueSelectStatement

joinStatement:

A join statement combines two result sets. Only combinations of rows taken from both result sets are returned when they meet the join conditions.

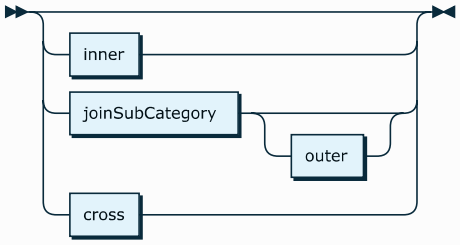
referenced by:

- joinStatements

joinCategory:

The join category specifies what combinations of rows are considered. The following variants can be used:

- inner join, as indicated by 'join' or 'inner join': an inner join returns all combinations of rows from both result sets that meet the join conditions.
- left outer, as indicated by 'left outer join': a left outer join returns the same rows as an inner join, extended by one row for each row in the left result set having no matching rows in the right result set. Each column that originates from the right result set is assigned a null value.
- right outer, as indicated by 'right outer join': a right outer join returns the same rows as an inner join, extended by one row for each row in the right result set having no matching rows in the left result set. Each column that originates from the left result set is assigned a null value.
- full outer, as indicated by 'full outer join': a full outer join returns the same rows as an inner join, extended by one row for each row in the right result set having no matching rows in the left result set. Each column that originates from the left result set is assigned a null value. The results are also extended by one row for each row in the left result set having no matching rows in the right result set. Each column that originates from the right result set is assigned a null value.
- cross join, as indicated by 'cross join': a cross join returns a Cartesian product of the rows from both result sets. A 'Cartesian product' is a term from set theory, which indicates that all combinations are returned.

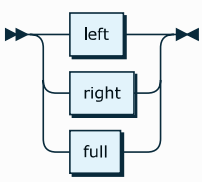


referenced by:

- joinStatement

joinSubCategory:

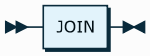
The join sub-category refines the join category. Please see 'joinCategory' for an explanation.



referenced by:

- joinCategory

join:



referenced by:

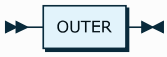
- joinStatement

inner:

referenced by:

- joinCategory

outer:



referenced by:

- joinCategory

left:

Extracts a substring from a value with the given length from the left side. Keyword is also used with joins.
Parameters:

- Input: Text to extract substring from.
- Length: Maximum length of the substring.

Returns: Substring from the left side of the input.

referenced by:

- functionExpression
- joinSubCategory

right:

Extracts a substring from a value with the given length from the right side. Keyword is also used with joins.
Parameters:

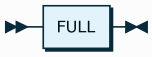
- Input: Text to extract substring from.
- Length: Maximum length of the substring.

Returns: Substring from the right side of the input.

referenced by:

- functionExpression
- joinSubCategory

full:



referenced by:

- joinSubCategory

cross:

referenced by:

- joinCategory

sum:

Group function to sum together individual numerical values. Occurrences of null are considered 0, unless there are only null values. In that case the outcome is null.

referenced by:

- sumAggregateFunction

product:

Group function to multiply together individual numerical values. Multiplying large values can quickly exceed the range of the resulting Decimal data type. The product group function is typically used in financial and probability calculations with values near 1.

referenced by:

- productAggregateFunction

min:

Group function to find the minimum value from a group of numerical values or text values.

referenced by:

- minAggregateFunction

max:

Group function to find the maximum value from a group of numerical values or text values.

referenced by:

- maxAggregateFunction

avg:

Group function to find the average value from a group of numerical values.

referenced by:

- avgAggregateFunction

first:

Group function to the first non-null value in an ordered result set.

referenced by:

- firstAggregateFunction

last:

Group function to the last non-null value in an ordered result set.

referenced by:

- lastAggregateFunction

stddev:

Group function to find the standard deviation from a group of numerical values.

referenced by:

- stdDevAggregateFunction

count:

Group function to find the number of values from a group of values.

referenced by:

- countAggregateFunction

listagg:

Group function which concatenates all individual values, separated by the separator when provided and comma plus space otherwise.

referenced by:

- listAggAggregateFunction

asc:

referenced by:

- sortDirection

desc:

referenced by:

- sortDirection

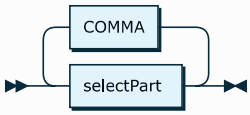
joinConditions:

A boolean expression which defines valid combinations of the join.

referenced by:

- joinStatement

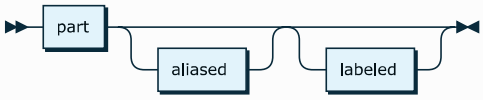
selectList:



referenced by:

- uniqueSelectStatement

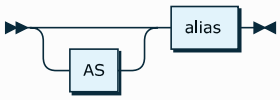
selectPart:



referenced by:

- selectList

aliased:



referenced by:

- dataSource
- pivotClause
- selectPart

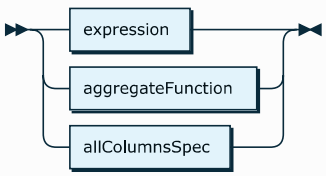
labeled:

Defines the textual label of an expression. The label may contain resources in the format '{res:resource code}' such as 'My {res:itgen_description}'. Similar to the data type and alias of an expression, the label is maintained across selections. Application of a calculation or a SQL function resets the label to the empty value. User interfaces can choose to display the label when available instead of the column name to provide a more natural interface.

referenced by:

- selectPart
- sqlDataTypeExtended

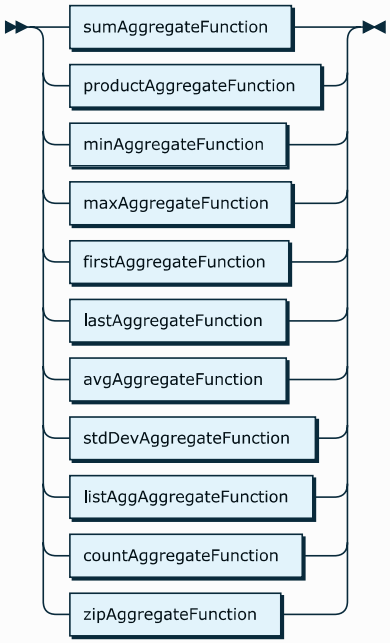
part:



referenced by:

- countAggregateFunction
- selectPart

aggregateFunction:



referenced by:

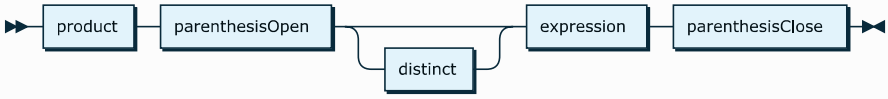
- part
- pivotClause

sumAggregateFunction:

referenced by:

- aggregateFunction

productAggregateFunction:



referenced by:

- aggregateFunction

minAggregateFunction:

referenced by:

- aggregateFunction

maxAggregateFunction:

referenced by:

- aggregateFunction

firstAggregateFunction:

referenced by:

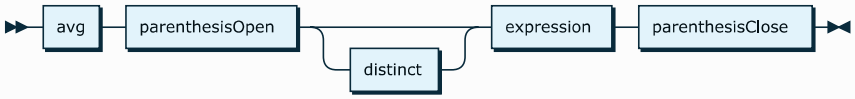
- aggregateFunction

lastAggregateFunction:

referenced by:

- aggregateFunction

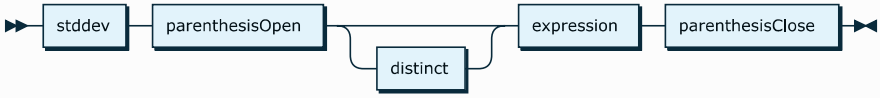
avgAggregateFunction:



referenced by:

- aggregateFunction

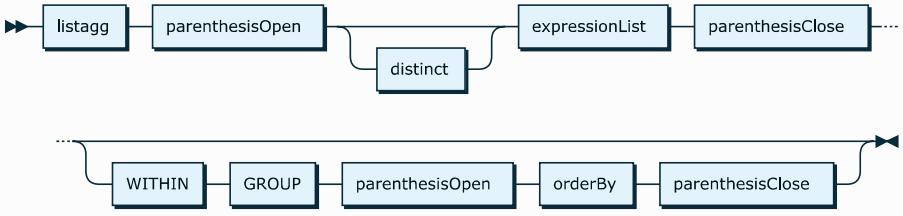
stdDevAggregateFunction:



referenced by:

- aggregateFunction

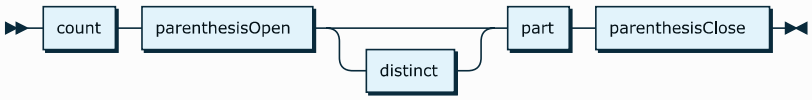
listAggAggregateFunction:



referenced by:

- aggregateFunction

countAggregateFunction:



referenced by:

- aggregateFunction

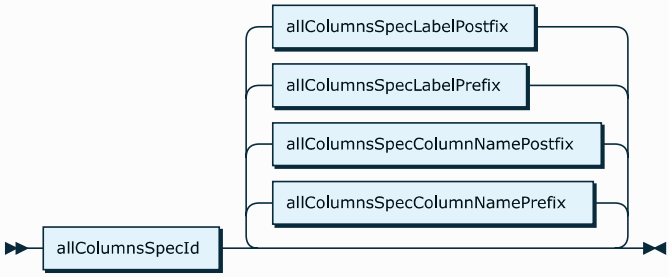
zipAggregateFunction:

referenced by:

- aggregateFunction

allColumnsSpec:

Selects all columns from an SQL statement or from one or more data sources used. The except clause allows retrieval of all columns except the listed names. The except clause is typically used with wide result sets with possibly varying column lists from which a few pre-defined columns need to excluded.



referenced by:

- part

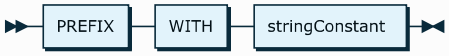
allColumnsSpecId:

referenced by:

- allColumnsSpec

allColumnsSpecColumnNamePrefix:

Defines the aliases of columns selected by the select all ('*') operator as their original alias, prefixed with the specified text. For instance, the select operator changes the alias of the column 'code' with the prefix 'pjt_' into 'pjt_code'.

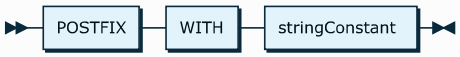


referenced by:

- allColumnsSpec

allColumnsSpecColumnNamePostfix:

Defines the aliases of columns selected by the select all ('*') operator as their original alias, postfixed with the specified text. For instance, the select operator changes the alias of the column 'code' with the postfix '_from_table2' into 'code_from_table2'.



referenced by:

- allColumnsSpec

allColumnsSpecLabelPrefix:

Defines the label of columns selected by the select all ('*') operator as their original label, prefixed with the specified text. For instance, the select operator changes the label 'Name' of the column 'name' with the prefix 'Project ' into 'Project Name'.

referenced by:

- allColumnsSpec

allColumnsSpecLabelPostfix:

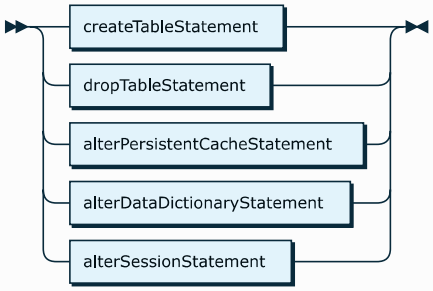
Defines the label of columns selected by the select all ('*') operator as their original label, postfixed with the specified text. For instance, the select operator changes the label 'Name' of the column 'name' with the postfix ' from Table2' into 'Name from Table2'.

referenced by:

- allColumnsSpec

ddlStatement:

All available Data Definition Language statements.



referenced by:

- sqlStatement

alterPersistentCacheStatement:

Besides an in-memory cache valid during the duration of a session, Invantive UniversalSQL offers an integrated cache storing data persistently using an on-premise or cloud relation database such as SQL Server or PostgreSQL. When configured, Invantive UniversalSQL first tries to find sufficiently fresh data in the cache. This reduces the number of data loads from slow data containers such as some cloud platforms. In general, the performance increase when the rows can be fully retrieved from a cache is between a factor 25 and 2.500.

Invantive UniversalSQL itself manages the table structure and table contents in the relation database used as a data cache. On initial use just provide an empty database. Platforms supported include SQL Server, Oracle, PostgreSQL and MySQL. Invantive UniversalSQL installs a repository consisting of a dozen tables. The repository tables have names starting with 'dc_'. Most repository tables are accompanied by a view named identically except for a trailing '_r'. The repository views join in all master tables for easier analysis from within the database storing the facts

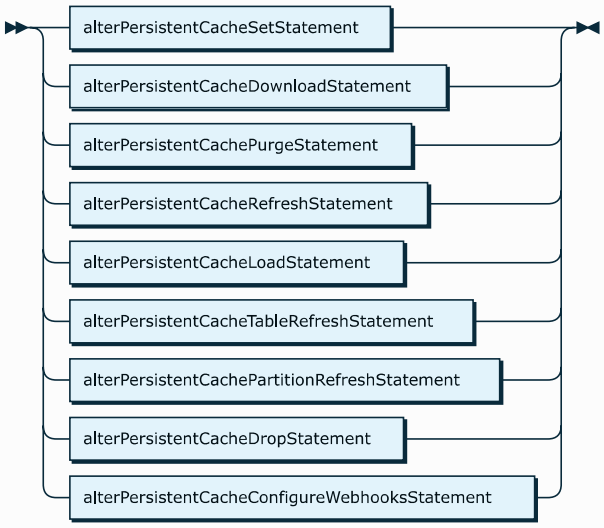
For each table partition version, a so-called 'facts table' is created. A facts table contains a full copy of the rows retrieved from the data container. Facts tables have names starting with 'dcd_', followed by a unique hash signaling the table partition version. When necessary, additional database objects are maintained such as indexes to improve performance. As with facts table names, all column names are also hashed based upon an algorithm including the original column name. These facts tables are not intended for direct use using native SQL.

Each facts table has a unique state from the following state, with 'Ready' state signaling the now current version:

- Initializing ('I'): the facts table will be created.
- View creation ('V'): logical views will be created.
- Prepared ('P'): the facts table has been created, but contains yet no rows.
- Seeding ('S'): the facts table is being seeded with the contents of the previously current version.
- Loading ('L'): loading new facts from data container using water shed or another algorithm.
- Ready ('R'): the facts table is available and the current one to be used.
- Obsoleted ('O'): the facts table still exists, but the data has passed it's conservation period. Often a newer version is now current.
- Dropped ('D'): the facts table now longer exist, but the metadata is still present in the repository tables.

The persistent cache in the database can be used with native SQL when extended by Invantive Data Replicator. Invantive Data Replicator creates and maintains database view (a so-called 'partition view') for the now current version of table partition. Similarly, it can create an 'overall view', showing the rows across all partitions of the now current versions per partition. Invantive Data Replicator provides a high-performance high-volume operational data store with little effort.

The overall views are typically used for consolidation purposes, bringing together data across multiple companies or persons.



referenced by:

- ddlStatement

alterPersistentCachePurgeStatement:

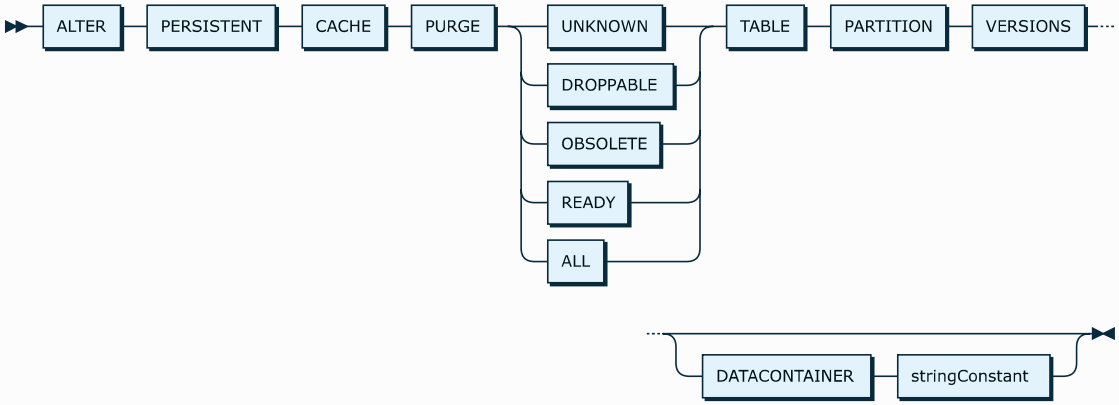
The purge statement maintains existing table partition versions. It can purge table partition versions in several ways:

- ready: selects table partition versions in state 'Ready'. Table partition versions go to the state 'Obsolete'. Facts are not deleted.
- obsolete: selects table partition versions in state 'Obsolete'. Facts of these are deleted. Table partition versions go to the state 'Dropped'. The facts deleted might be used in queries that started running when they were in state 'Ready'. The facts can be deleted safely when the database platform supports reading data from rows already deleted, such as Oracle. Purging obsolete facts is often used as a background task separate from refresh to reduce refresh runtime, since the deletion of data has already been executed before refresh.
- droppable: selects table partition versions in state 'Dropped'. Facts are deleted when any facts are still present despite the dropped state. This will be useful only under exceptional conditions, since the dropped status indicates that the facts have already been deleted. Table partition versions remain in the state 'Dropped'.
- unknown: queries the database for tables that match the naming convention 'dcd...' or 'dcs...'. Any tables that is not registered in the repository is dropped from the database. This will be useful only under exceptional conditions, such as when the repository was unintentionally dropped or incorrectly changed.
- all: first performs the "unknown" variant and then the "droppable" variant. This will be useful only under exceptional conditions.

Specification of a data container limits the scope of table partition versions to the listed data container.

PSQL Packages

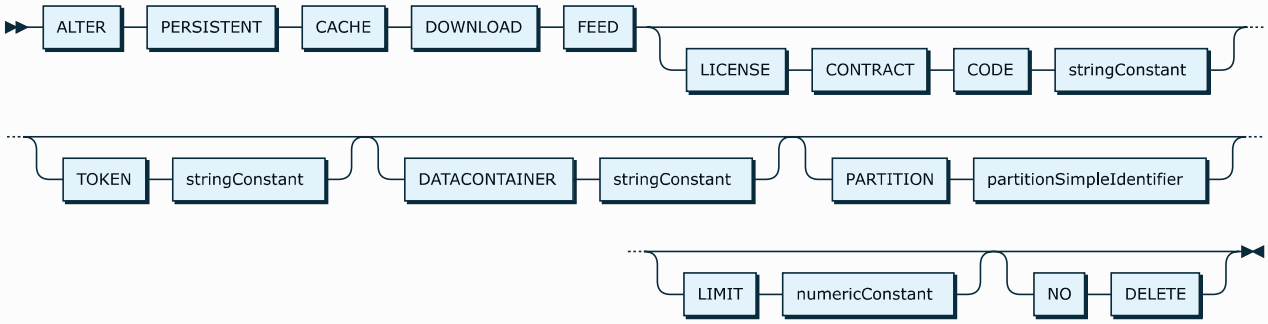
Invantive PSQL comes with a number of pre-defined packages. These can be queried using SystemPackages@DataDictionary and SystemPackageFunctions@DataDictionary.



referenced by:

- alterPersistentCacheStatement

alterPersistentCacheDownloadStatement:



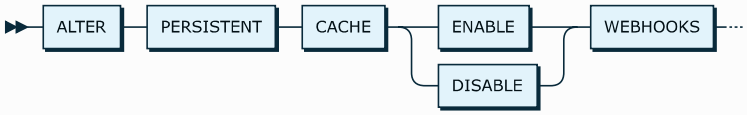
referenced by:

- alterPersistentCacheStatement

alterPersistentCacheConfigureWebhooksStatement:

Trickle loading is an advanced strategy to efficiently and fast update the replicate data. It requires all changes (insert, update and delete) on the replicated data to be registered as event messages, possibly with a delay.

The registration of changes can be activated on selected platforms using webhooks. Configuration of the webhooks can be done using this statement. When not specified, the license contract code of the current subscription will be used.



referenced by:

- alterPersistentCacheStatement

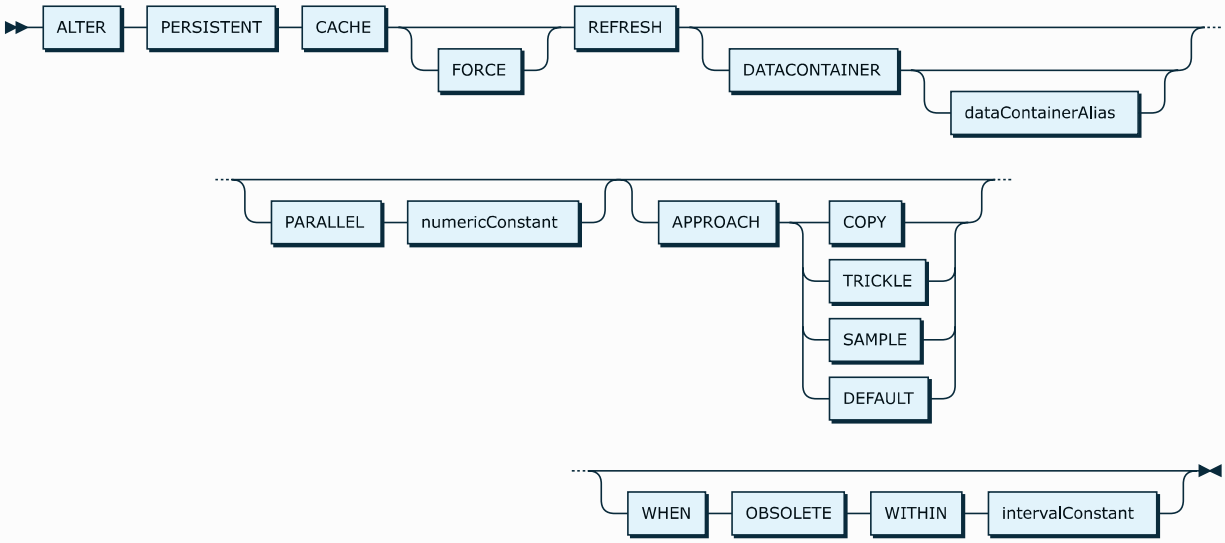
alterPersistentCacheRefreshStatement:

This statement triggers the refresh of replicated data of all connected or one specific data container alias. A refresh of the replicated data can also be triggered by executing a SQL statement specifying the use of replicated data of a specific age.

In default mode (without 'force'), a refresh is only executed on the specified data when the age of the replicated data exceeds the configured maximum age. An offset to the configured maximum age can be specified using the 'when obsolete within' clause. With a forced refresh, the replicated data is always replaced by a recent version of the source data.

The maximum number of parallel processes to replicate the data can be configured using the 'parallel' clause to increase throughput and decrease runtime of the replication process.

By default the approach as configured on each table partition is used to update the replica. However, a deviating approach can be specified. This is typically done after switching a table partition to trickle loading to run a one-time replication process with the copy approach, effectively ensuring that no changes have gone unnoticed.



referenced by:

- alterPersistentCacheStatement

alterPersistentCacheLoadStatement:

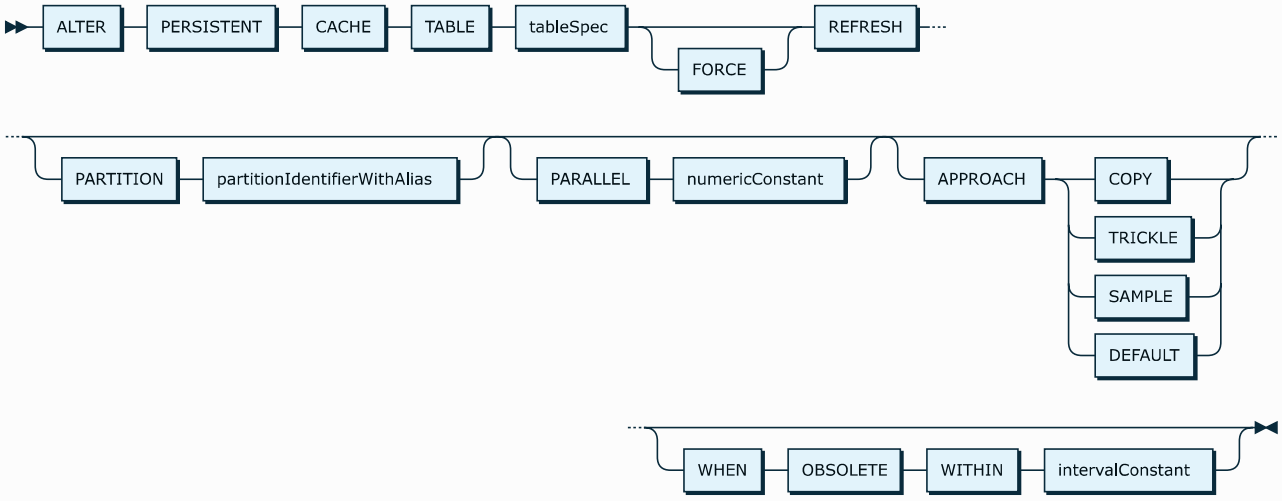
Loads all available tables across all connected data containers in the cache. Typically used for demonstrations.

referenced by:

- alterPersistentCacheStatement

alterPersistentCacheTableRefreshStatement:

Refresh all data of a specificied table. The options are explained at alterPersistentCacheRefreshStatement.

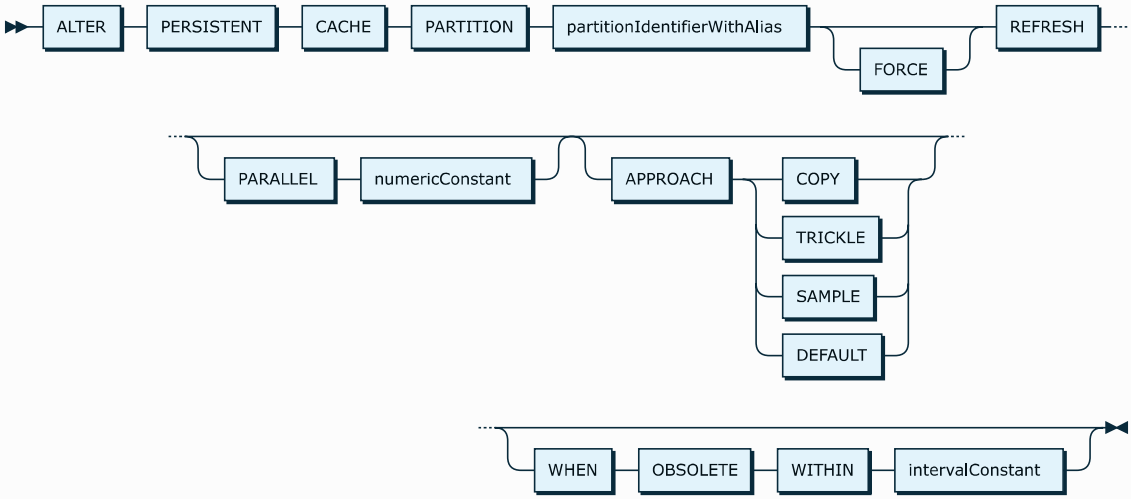


referenced by:

- alterPersistentCacheStatement

alterPersistentCachePartitionRefreshStatement:

Refresh all data of a specificied partition. The options are explained at alterPersistentCacheRefreshStatement.

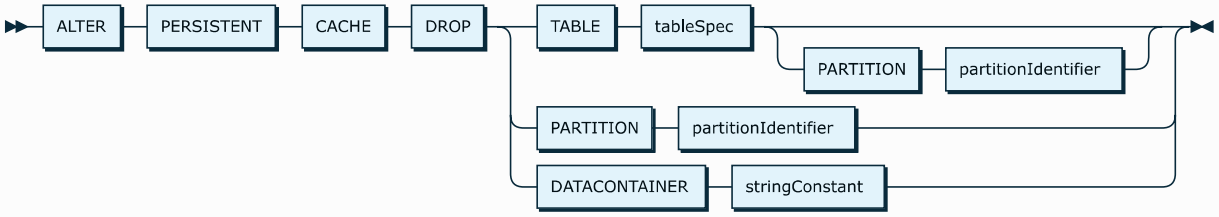


referenced by:

- alterPersistentCacheStatement

alterPersistentCacheDropStatement:

Drops all facts stored in table partition versions from the database and the associated metadata from the cache repository for a range of table partitions. The tables and partitions can be specified using the table, partition and/or data container clause.



referenced by:

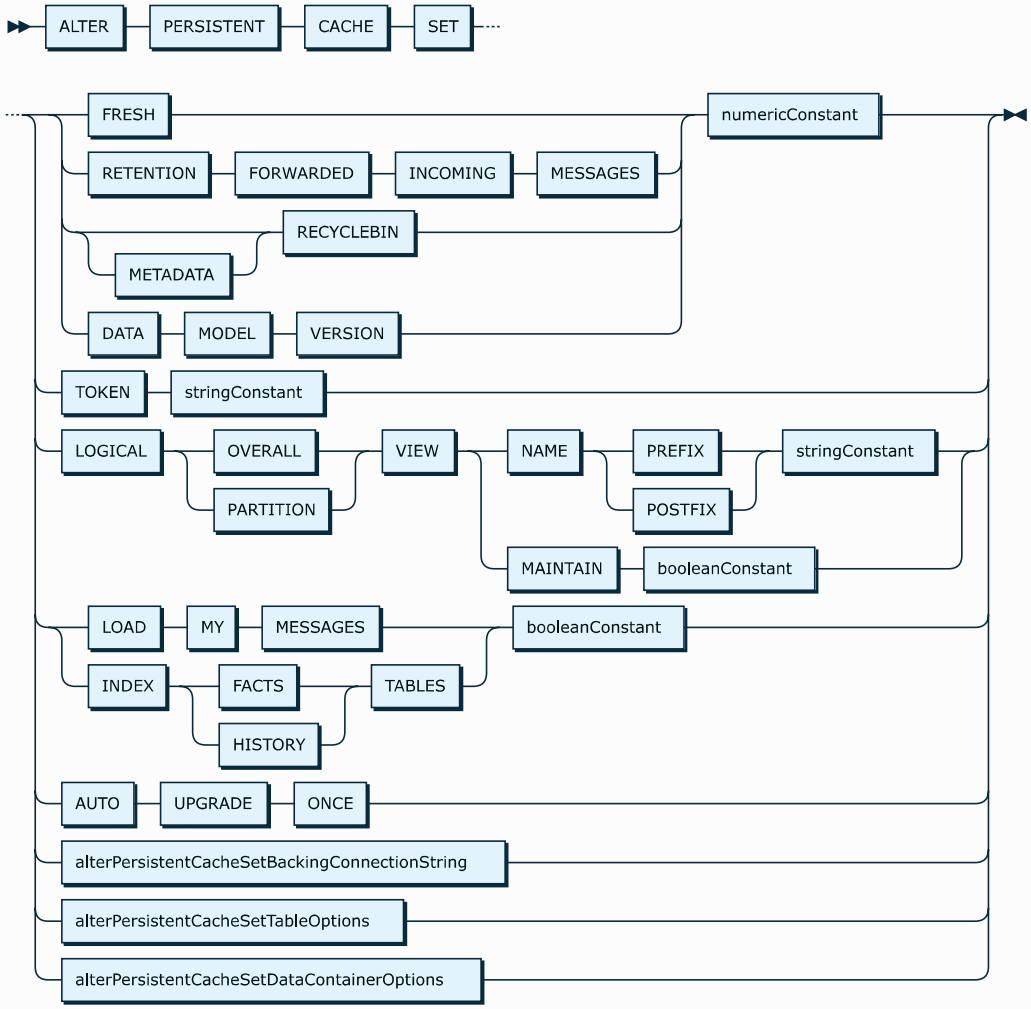
- alterPersistentCacheStatement

alterPersistentCacheSetStatement:

Change central maintained settings for the cache, including defaults for new tables or data containers.

- Fresh: number of seconds during which facts are considered fresh after completiong of loading.
- Retention: number of seconds trickle loading messages are kept available after processing.
- Metadata: number of seconds metadata are kept available after dropping the associated facts in the version.

- Metadata recyclebin: number of seconds dropped metadata are kept available.
- Version: reset data model version to another version to re-run upgrade scripts.
- Token: secret text used as token to retrieve trickle loading messages.
- Prefix/postfix: prefix and/or postfix for Invantive Data Replicator maintained database views.
- Maintain: indicator whether to maintain database views per table partition and/or overall table.
- My Messages: indicator whether to place a copy of incoming trickle loading messages in dc_my_incoming_messages for custom code purposes.
- Index Tables: whether to create non-unique indexes on the facts tables for speedier retrieval on database views.
- Auto Upgrade: whether to run a data model upgrade once.
- Data Container Prefix/postfix: prefix and/or postfix for Invantive Data Replicator maintained overall views.



referenced by:

- alterPersistentCacheStatement

alterPersistentCacheSetBackingConnectionString:

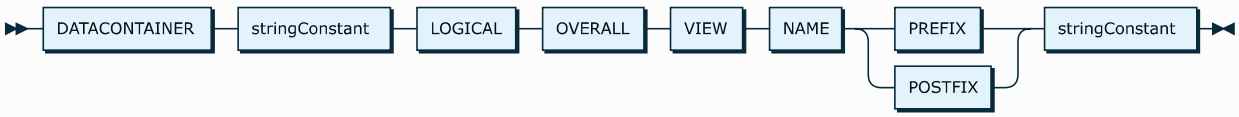
referenced by:

- alterPersistentCacheSetStatement

alterPersistentCacheSetDataContainerOptions:

Changes settings of a data container.

The prefix and postfix for the logical overall view of every new replicated table registered in the data container can be specified. This is typically done when there are multiple data containers pointing to the same platform, such as multiple subscriptions on Teamleader or multiple Exact Online countries. Specifying a prefix ensures that the automatically generated logical overall views have unique and recognizable names.



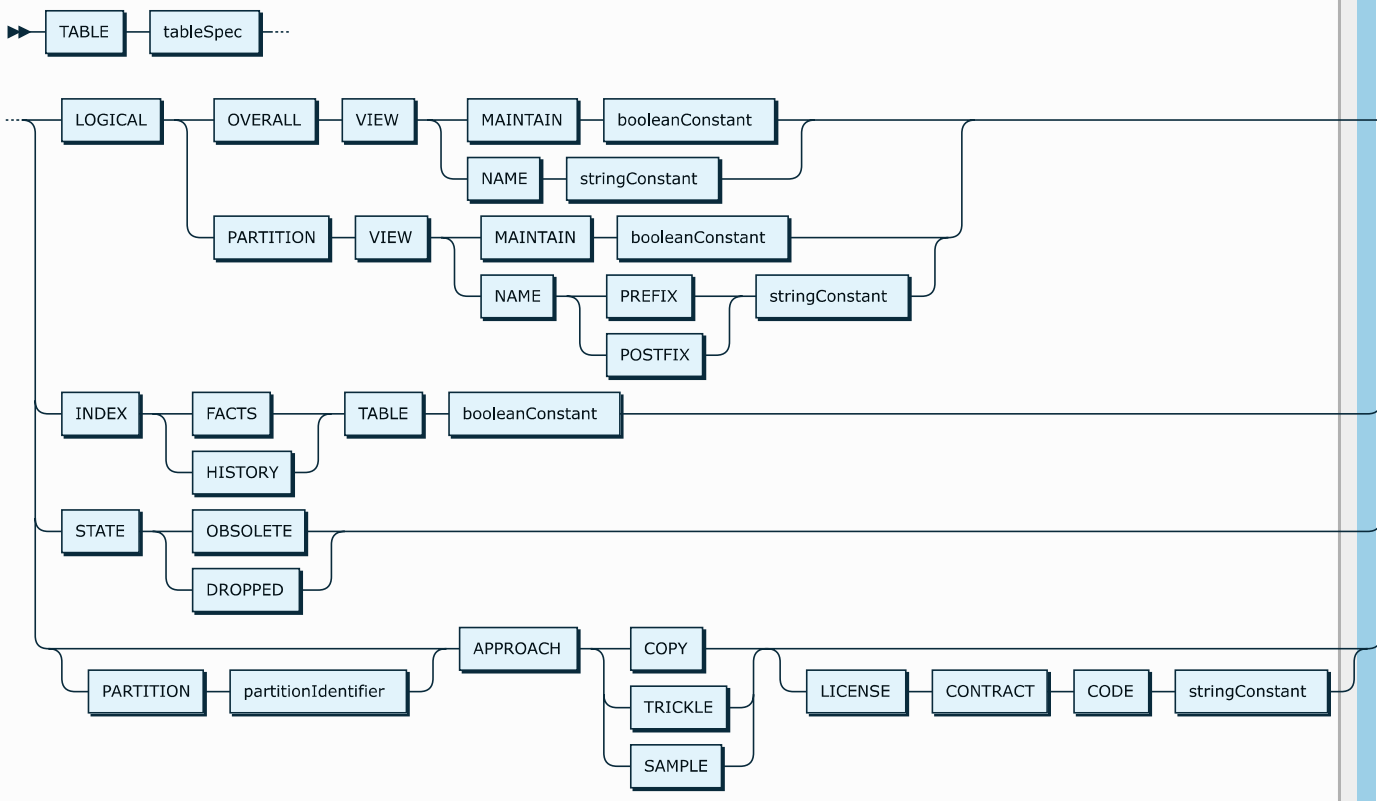
referenced by:

- alterPersistentCacheSetStatement

alterPersistentCacheSetTableOptions:

Change table and table partition-specific settings for the cache. Most options are identical as on alterPersistentCacheSetStatement, but intended for table/table partition-specific deviations. Additional options are:

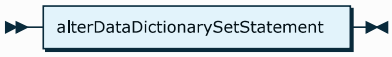
- State: move all table partition versions with an earlier state to a specific end state.
- Approach: replication strategy.



referenced by:

- alterPersistentCacheSetStatement

alterDataDictionaryStatement:



referenced by:

- ddlStatement

alterSessionStatement:

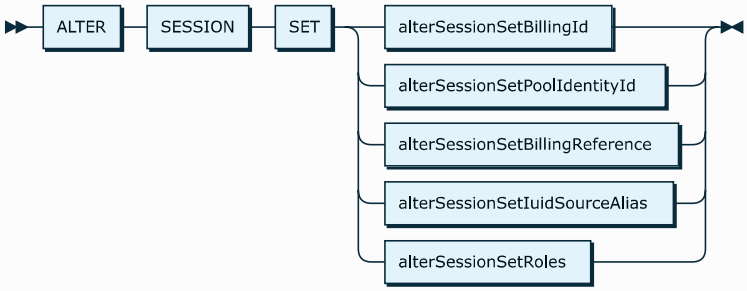
Various statement to alter properties of the current Invantive UniversalSQL session.

referenced by:

- ddlStatement

alterSessionSetStatement:

Change properties of the current Invantive UniversalSQL session.

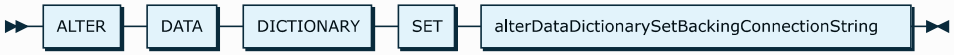


referenced by:

- alterSessionStatement

alterDataDictionarySetStatement:

Associate a persistent store to back the data dictionary associated with the current Invantive UniversalSQL session. The persistent store will be used to store and retrieve object definitions such as PSQL procedures, SQL views and PSQL packages.



referenced by:

- alterDataDictionaryStatement

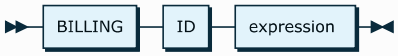
alterDataDictionarySetBackingConnectionString:

referenced by:

- alterDataDictionarySetStatement

alterSessionSetBillingId:

Change the ID used for to charge usage to another agreement as specified in the license. Used solely for hosted use of Invantive products to associate usage statistics on data downloaded, data uploaded, users, devices and partitions use with individual billing IDs. Original agreement code is also registered.



referenced by:

- alterSessionSetStatement

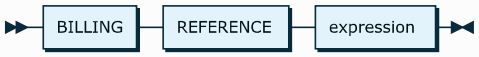
alterSessionSetPoolIdentityId:

referenced by:

- alterSessionSetStatement

alterSessionSetBillingReference:

Add a reference to billing to differentiate use within one agreement depending on application. department or operating company. Typically used for use of Invantive products by larger organizations to associate usage statistics on data downloaded, data uploaded, users, devices and partitions use with individual billing references.



referenced by:

- alterSessionSetStatement

alterSessionSetIuidSourceAlias:

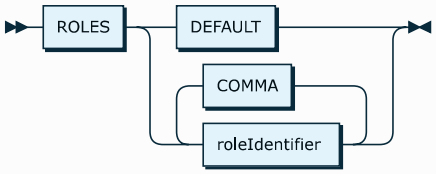
Use a non-default data container to determine the IUID (Invantive User ID) value used for identifying the user. Typically used when the actual user is available in a user directory of a higher-numbered data container.

referenced by:

- alterSessionSetStatement

alterSessionSetRoles:

Invantive UniversalSQL offers advanced security features including row-level security with custom connectors. Sets the active roles of a session.



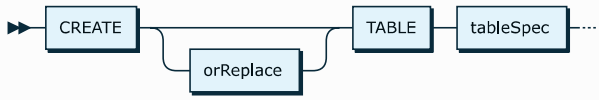
referenced by:

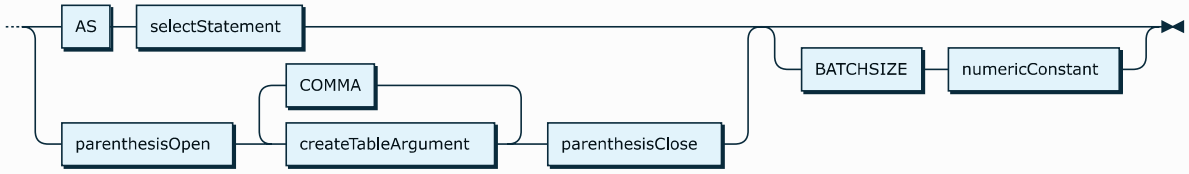
- alterSessionSetStatement

createTableStatement:

Create a table on the specified platform, filled with data from the select statement. An error is raised when a table with the same name already exists. When 'or replace' is specified, a previously existing table with the same name is dropped before creating the new table.

A new table is assigned a technical primary key column. Also indexes are created by default where deemed useful.





referenced by:

- ddlStatement

createTableArgument:

referenced by:

- createTableStatement

dropTableStatement:

Drop the specified table on the specified platform. An error is raised when no table exists by that name.

referenced by:

- ddlStatement

orReplace:

referenced by:

- createTableStatement

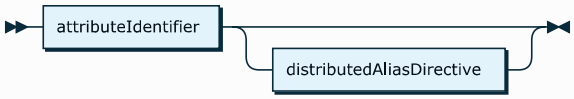
setStatement:

Replaces the value of a provider attribute by a new value.

referenced by:

- sqlStatement

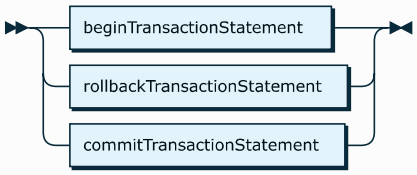
setIdentifier:



referenced by:

- setStatement

transactionStatement:



referenced by:

- sqlStatement

executeFileStatement:

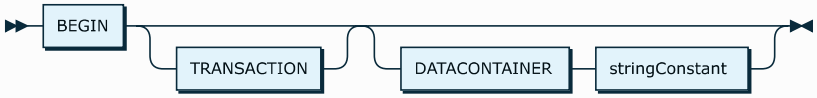
Execute a file loaded from the operating system with Invantive UniversalSQL statements. The file may contain Invantive UniversalSQL and Procedural SQL. The use of Invantive Script is not possible; Invantive Script is a client-side solution such as with Invantive Data Hub and Invantive Query Tool.

referenced by:

- sqlStatement

beginTransactionStatement:

A begin transaction statement initiates a transaction. Invantive UniversalSQL typically provides no transaction logic given the distributed nature and the limitations of the possible platforms. Some platforms enable collection of transaction data, which are to be handed over to the backing platform all together.

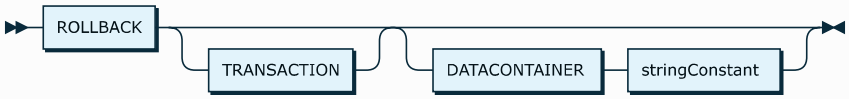


referenced by:

- transactionStatement

rollbackTransactionStatement:

Forgets all collected transaction data not yet handed over to the backing platform.

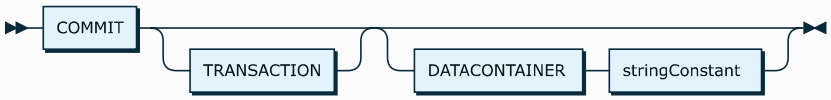


referenced by:

- transactionStatement

commitTransactionStatement:

Hand over all collected transaction to the backing platform for registration.



referenced by:

- transactionStatement

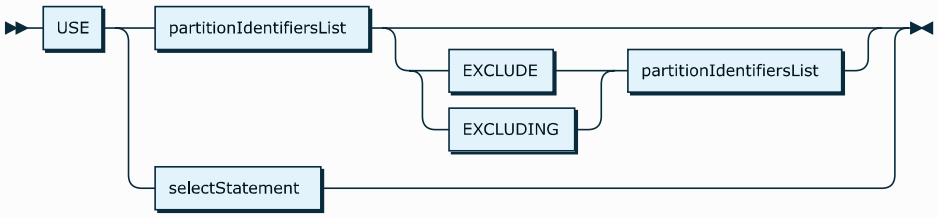
useStatement:

The use statement enables you to specify which partitions should be accessed by subsequent select, insert, update and delete statements. You can specify one or multiple partitions as a comma-separated list, possibly for a specific data container by appending an at-sign plus data container alias to the partition code. The value 'default' has a special meaning; it specifies to use the partition(s) originally selected when you logged on. The value 'all' also has a special meaning: it selects all partitions available.

For instance, to select partition '35' in the data container with alias 'eoInl' and partition '57345' in the data container with alias 'nmbrsnl', you can execute: 'use 35@eoInl, 57345@nmbrsnl'.

For complex scenarios, you can specify any valid Invantive UniversalSQL select statement which returns one or two columns. Each row from the query specifies one partition to select. The first column specifies the partition code, whereas the optional second column specifies a specific data container alias.

For instance, to select partition '35' in the data container with alias 'eoInl' and partition '57345' in the data container with alias 'nmbrsnl', you can execute: 'use select '35', 'eoInl' from dual@datadictionary union all select '57345', 'nmbrsnl' from dual@datadictionary'.



referenced by:

- sqlStatement

partitionIdentifiersList:

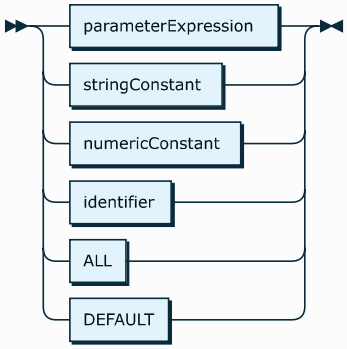
referenced by:

- useStatement

partitionIdentifier:

A partition identifier uniquely identifies one or more partitions from the currently available data containers.

The special partition identifer 'default' stands for all partitions that were chosen as default partitions during setup of the database connection. The special partition identifier 'all' stands for all available partitions across all available database connections.

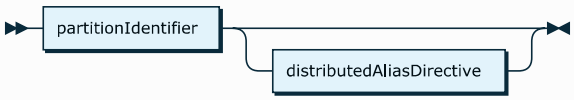


referenced by:

- alterPersistentCacheDropStatement

- alterPersistentCacheSetTableOptions
- partitionIdentifierWithAlias

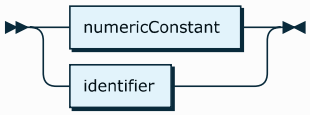
partitionIdentifierWithAlias:



referenced by:

- alterPersistentCachePartitionRefreshStatement
- alterPersistentCacheTableRefreshStatement
- partitionIdentifiersList

partitionSimpleIdentifier:



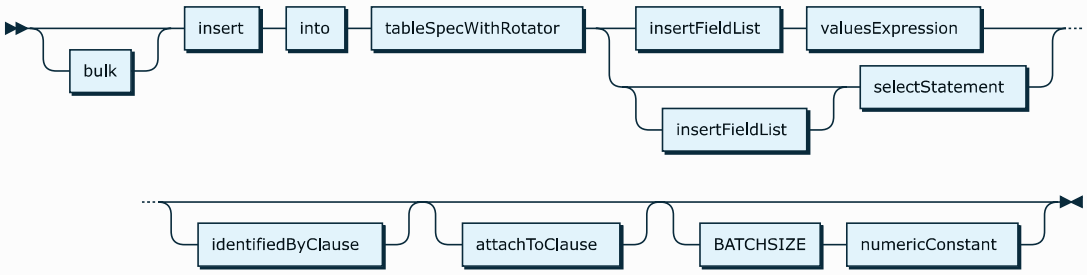
referenced by:

- alterPersistentCacheConfigureWebhooksStatement
- alterPersistentCacheDownloadStatement

insertStatement:

Inserts data into a table of a data container. Low-volume inserts are typically done one-by-one, but on some connectors a bulk insert option is available which inserts data with multiple rows each time for higher throughput. The data can be sourced from a select statement or values. Values can be single row values or multiple row values.

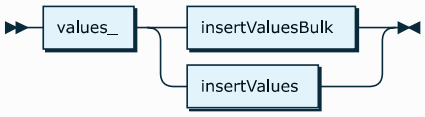
The identified by and attach to clause can be used in combination with transactions. The identified by clause is used to register the master rows, whereas the attach to clause is used to register which master rows detail rows belong. Upon commit, a master row (such as a sales order) plus it's details (such as multiple sales order lines) are sent to the target data container.



referenced by:

- sqlStatement

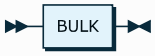
valuesExpression:



referenced by:

- insertStatement

bulk:



referenced by:

- insertStatement

into:

referenced by:

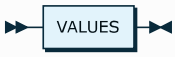
- insertStatement
- pSqlExecuteImmediateStatementNS
- pSqlExecuteNativeStatementNS
- uniqueSelectStatement

insert:

referenced by:

- insertStatement

values_:



referenced by:

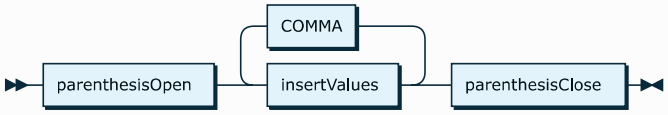
- valuesExpression

insertFieldList:

referenced by:

- insertStatement

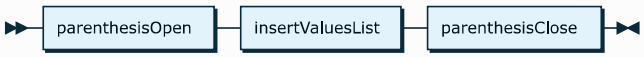
insertValuesBulk:



referenced by:

- valuesExpression

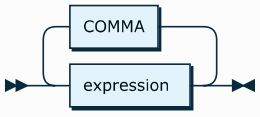
insertValues:



referenced by:

- insertValuesBulk
- valuesExpression

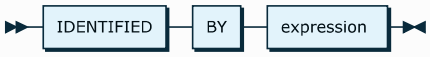
insertValuesList:



referenced by:

- insertValues

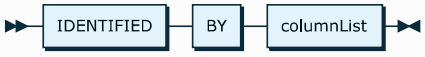
identifiedByClause:



referenced by:

- insertStatement

identifiedByMultipleClause:



referenced by:

- synchronizeStatement

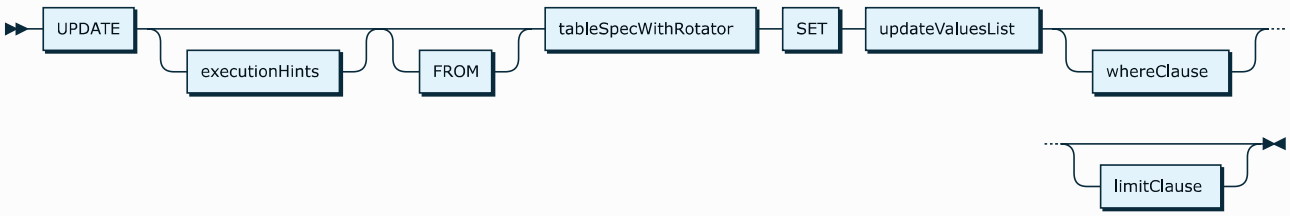
attachToClause:

referenced by:

- insertStatement

updateStatement:

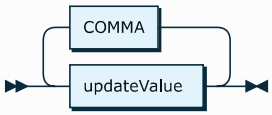
Updates data in a table of a data container.



referenced by:

- sqlStatement

updateValuesList:



referenced by:

- updateStatement

updateValue:

referenced by:

- updateValuesList

deleteStatement:

Deletes data from a table of a data container.

referenced by:

- sqlStatement

delete:

referenced by:

- deleteStatement

synchronizeStatement:

The compare and synchronize statement provides one-directional and bi-directional analysis and alignment of data in two tables. It is an extended version of SQL statements like 'upsert' and 'merge' found in other SQL implementations. The two tables can be located in any data container, enabling cross-data container synchronization of data. In most scenarios, data from both tables is downloaded and compared by Invantive UniversalSQL to establish the necessary actions. Some specific edge cases may have an optimized algorithm to reduce data downloads.

In an one-directional approach as specified by 'FROM' or 'TO', the compare and synchronize statement typically analyzes the differences between two tables with identically named columns, separating all differences in one out of four groups:

- insert: a row exists in the source table and not in the target table;
- delete: a row exists not in the source table, but exists in the target table;
- update: the row exists in both tables, but the column contents are different.
- none: the row exists in both tables and the column contents are equal.

Column values are matched on name. By default, all columns having identical names will be included in the matching process. The IGNORE CHANGES TO can be used to exclude columns by name from matching. IGNORE CHANGES TO is typically used to exclude changes in column values that should not trigger a DML-operation, such as for columns whose value is managed by the application, including technical keys.

Unique rows in both tables are matched using the list of columns as specified by 'IDENTIFIED BY'. Identifying column NULL contents are considered a specific value for matching; when both the source and target table have a null value, the rows will be compared. Rows are completely ignored for synchronisation when all identifying columns have a NULL value and the clause "IGNORE NULLS" is present.

After analysis, the differences are be filtered to only contain a combination of DML-operations specified using 'WITH'. Applying all DML-operation types INSERT, UPDATE and DELETE would result in the two tables having identical contents. The INSERT and UPDATE operations by default apply to all columns, but columns can be excluded using 'ALL EXCEPT'. Excluding columns is typically used to leave out audit columns such as a generated creation date upon update and still have it included upon insert.

For better performance, SYNCHRONIZE uses bulk operations where supported, similar to CREATE TABLE and BULK INSERT. Bulk operations typically offer an order of magnitude better performance than solitary operations. However, error handling can be inconvenient since it is often unclear what specific row triggered an error. Neither, it is not easily established upon an error whether other rows in the same bulk operation have been processed. The BATCHSIZE clause allows specification of the number of rows per bulk operation with a minimum of 1. When BATCHSIZE is not specified, a platform-specific default value is used which can fluctuate due to dynamical management by Invantive UniversalSQL.

In case of bi-directional synchronization, the RESOLVE BY-clause enables specification of the column names whose value determines what table contains the preferred row with most current values. Using left-to-right column preference, both values of all RESOLVE BY columns are inspected. The first column value to have a higher ranking value than the other table selects that table as source. The logic is reversed for a column from higher ranking to lower ranking when DESC is specified. The RESOLVE BY-clause is typically used in combination with columns containing a (synchronized) timestamp value such as UTC time.

The 'APPLY TO' syntax is reserved for future use to allow routing DML-operations to other tables.

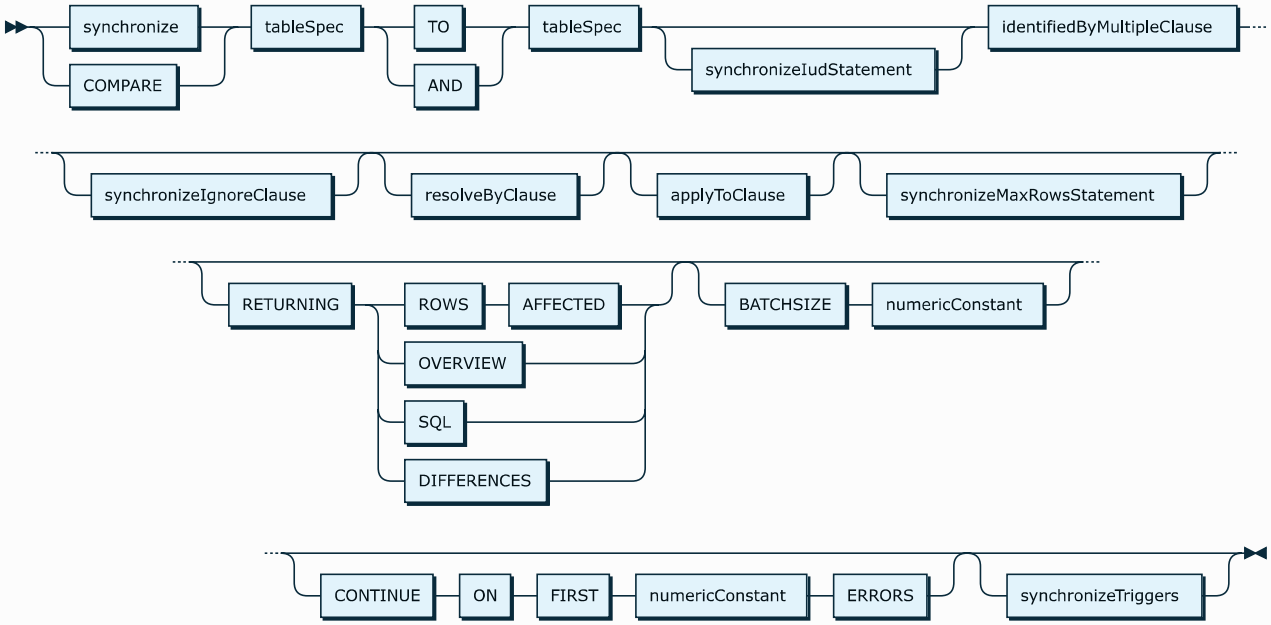
By default, the SYNCHRONIZE statement will fail upon the first error returned. Using 'CONTINUE ON FIRST ... ERRORS' the threshold can be increased. However, the statement will fail whenever any error occurred during execution.

The differences can then either be:

- applied on the target table to make the tables equal, using bulk operations where supported.
- returned as a list of content differences (not yet available);
- returned as Invantive UniversalSQL DML-statements to achieve synchronization (not yet available).

After checking the output of the last two categories of analysis, the differences can be applied on the target table.

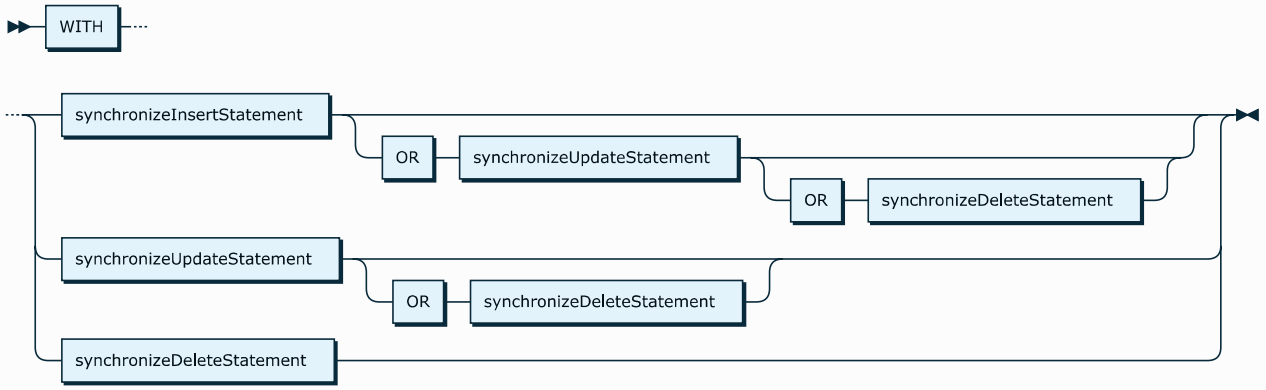
In a bi-directional approach, as specified by using the keyword 'AND' between the two table identifiers, deletes will not be returned from the analysis phase and replaced by inserts on the table which does not have the other table's row. The determination of the target for an update is based upon the resolution preference as defined by the 'RESOLVE' clause. The 'RESOLVE' clause either defines to always apply updates to the left or right table as listed in their order using, respectively, 'PREFER LEFT' and 'PREFER RIGHT'. Otherwise, the combined ordinal value of the columns listed in the 'RESOLVE BY' clause will be evaluated and the highest-ranking value is considered to be leading.



referenced by:

- sqlStatement

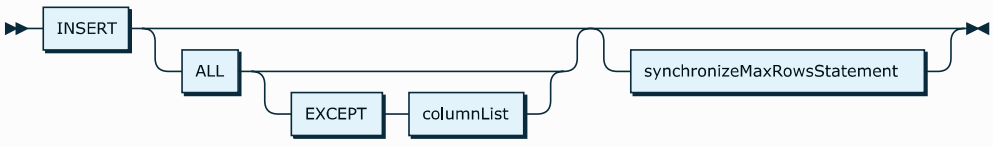
synchronizeIudStatement:



referenced by:

- synchronizeStatement

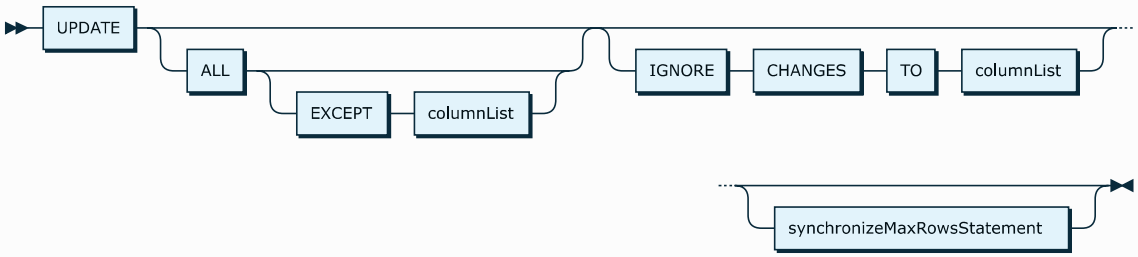
synchronizeInsertStatement:



referenced by:

- synchronizeIudStatement

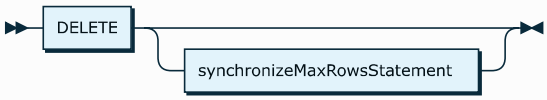
synchronizeUpdateStatement:



referenced by:

- synchronizeIudStatement

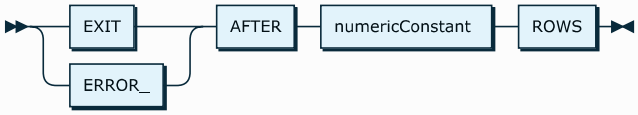
synchronizeDeleteStatement:



referenced by:

- synchronizeIudStatement

synchronizeMaxRowsStatement:



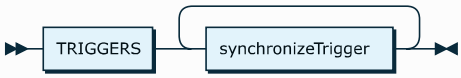
referenced by:

- synchronizeDeleteStatement
- synchronizeInsertStatement
- synchronizeStatement
- synchronizeUpdateStatement

synchronizeUsingPSqlBlock:

no references

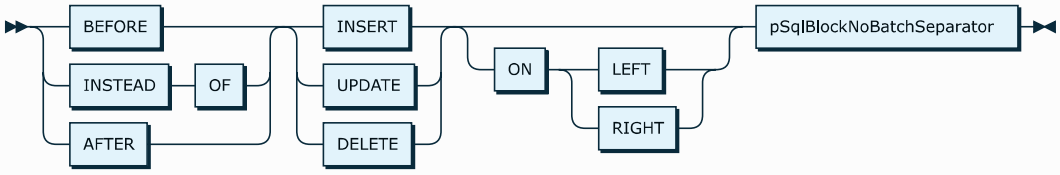
synchronizeTriggers:



referenced by:

- synchronizeStatement

synchronizeTrigger:



referenced by:

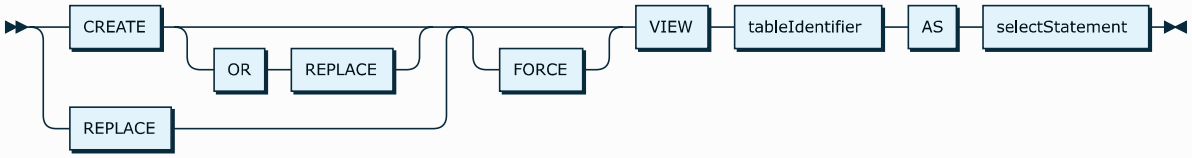
- synchronizeTriggers

createOrReplaceViewStatement:

Creates a database view with associated user-definable SQL select statement. The SQL select statement can access any object available across all data containers of the database. A view is stored for the duration of the session in the data dictionary under the catalog DataDictionary and schema Custom. When creating, modifying and deleting views, it is not necessary to specify that the view is in the data dictionary; after all, there is only one place it can be. All defined views can be queried for using the data dictionary view SystemViews. Upon creation, the view is checked for syntactic correctness. Syntactic errors will lead to an error, unless the force option is also specified in the statement. Upon execution, both syntax and validity are checked against the then active database.

To run the query associated with the view, it is necessary to specify the origin via the DataDictionary alias. A view can be embedded through it's name in another view or query as if it was a table.

Database views complement the unalterable Invantive-provided views which are always driver-specific. Invantive's driver-views only combine data within the driver that contains it. Driver-views can always be used independent of the database they are contained in, as long as the driver is present.

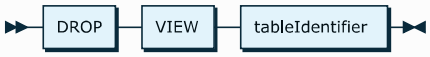


referenced by:

- sqlStatement

dropViewStatement:

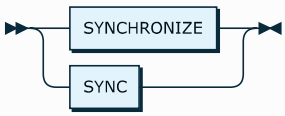
The drop view statement removes a view from the data dictionary. After completion, the view can no longer be used by queries being started.



referenced by:

- sqlStatement

synchronize:



referenced by:

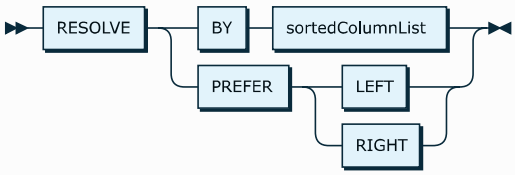
- synchronizeStatement

synchronizeIgnoreClause:

referenced by:

- synchronizeStatement

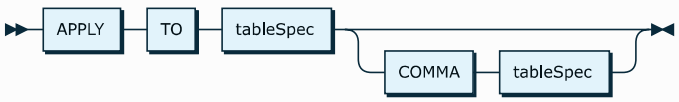
resolveByClause:



referenced by:

- synchronizeStatement

applyToClause:

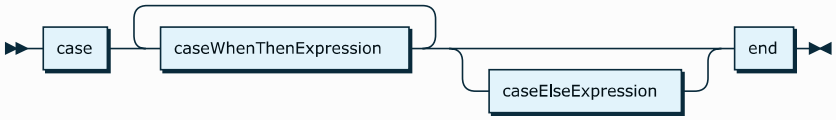


referenced by:

- synchronizeStatement

caseExpression:

Evaluates to an expression, testing a number of boolean expressions in sequence for true. When no boolean expression evaluates to true, the else expression is returned.



referenced by:

- expression

caseWhenThenExpression:

referenced by:

- caseExpression

caseElseExpression:

referenced by:

- caseExpression

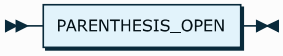
castExpression:

Changes the data type of an expression into the indicated data type.

referenced by:

- expression

parenthesisOpen:

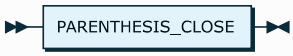


referenced by:

- avgAggregateFunction
- castExpression
- countAggregateFunction
- createTableStatement
- csvTableSpec
- embeddedSelect
- excelTableSpec
- expression
- firstAggregateFunction
- forJsonClause
- forXmlClause
- forXmlClauseCommonDirectives
- functionExpression
- htmlTableSpec
- insertFieldList
- insertValues
- insertValuesBulk
- internetTableSpec
- jsonTableSpec
- lastAggregateFunction
- listAggAggregateFunction
- maxAggregateFunction
- minAggregateFunction
- ndjsonTableSpec
- now
- pSqlFunctionOrProcedureStatementNS
- pSqlPackageProcedureStatementNS
- pipelinedTableFunctionSpec
- pivotClause
- productAggregateFunction
- sqlDataTypeExtended
- stdDevAggregateFunction
- stringOrChar

- stringSplitSpec
- sumAggregateFunction
- tableFunctionSpec
- utc
- xmlTableSpec
- zipAggregateFunction

parenthesisClose:



referenced by:

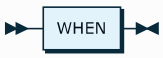
- avgAggregateFunction
- castExpression
- countAggregateFunction
- createTableStatement
- csvTableSpec
- embeddedSelect
- excelTableSpec
- expression
- firstAggregateFunction
- forJsonClause
- forXmlClause
- forXmlClauseCommonDirectives
- functionExpression
- htmlTableSpec
- insertFieldList
- insertValues
- insertValuesBulk
- internetTableSpec
- jsonTableSpec
- lastAggregateFunction
- listAggAggregateFunction
- maxAggregateFunction
- minAggregateFunction
- ndjsonTableSpec
- now
- pSqlFunctionOrProcedureStatementNS
- pSqlPackageProcedureStatementNS
- pipelinedTableFunctionSpec
- pivotClause
- productAggregateFunction
- sqlDataTypeExtended
- stdDevAggregateFunction
- stringOrChar
- stringSplitSpec
- sumAggregateFunction
- tableFunctionSpec
- utc
- xmlTableSpec
- zipAggregateFunction

case:

referenced by:

- caseExpression

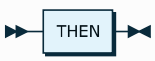
when:



referenced by:

- caseWhenThenExpression

then:



referenced by:

- caseWhenThenExpression

else:

referenced by:

- caseElseExpression

end:

referenced by:

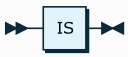
- caseExpression

not:

referenced by:

- expression

is:



referenced by:

- expression

are:

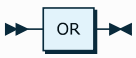
no references

and:

referenced by:

- expression

or:



referenced by:

- expression

true:

referenced by:

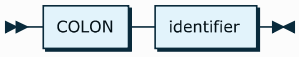
- booleanConstant

false:

referenced by:

- booleanConstant

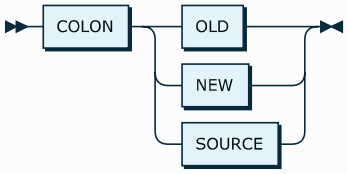
parameterExpression:



referenced by:

- expression
- partitionIdentifier

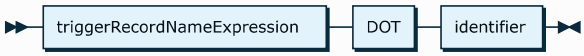
triggerRecordNameExpression:



referenced by:

- triggerRecordVariableExpression

triggerRecordVariableExpression:

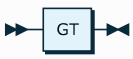


referenced by:

- expression
- pSqlAssignmentStatement

gt:

Greater then is a binary operator which returns true when the left value is greater than the right value. When one of both values is null, the outcome is null. Otherwise it is false.

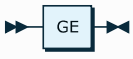


referenced by:

- expression

ge:

Greater or equal is a binary operator which returns true when the left value is greater than or equal to the right value. When one of both values is null, the outcome is null. Otherwise it is false.

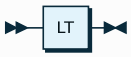


referenced by:

- expression

lt:

Less then is a binary operator which returns true when the left value is less than the right value. When one of both values is null, the outcome is null. Otherwise it is false.

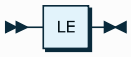


referenced by:

- expression

le:

Less or equal is a binary operator which returns true when the left value is less than or equal to the right value. When one of both values is null, the outcome is null. Otherwise it is false.



referenced by:

- expression

eq:

Equals is a binary operator which returns true when the left value and right value are identical. When one of both values is null, the outcome is null.

referenced by:

- expression

neq:

Not equals is a binary operator which returns true when the left value and right value are not identical. When one of both values is null, the outcome is null.

referenced by:

- expression

like:

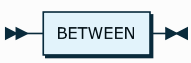
Like is a operator which returns true when the left value matches the right value. Occurrences of '%' in the right value can be matched by a sequence of 0, 1 or more characters in the left value. Occurrences of '_' in the right value can be matched by any character in the left value.

referenced by:

- expression

between:

Between is a tertiary operator which returns true when the left value has a value between the second and third values, including edges. When one of values is null, the outcome is null.

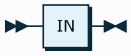


referenced by:

- expression

in_:

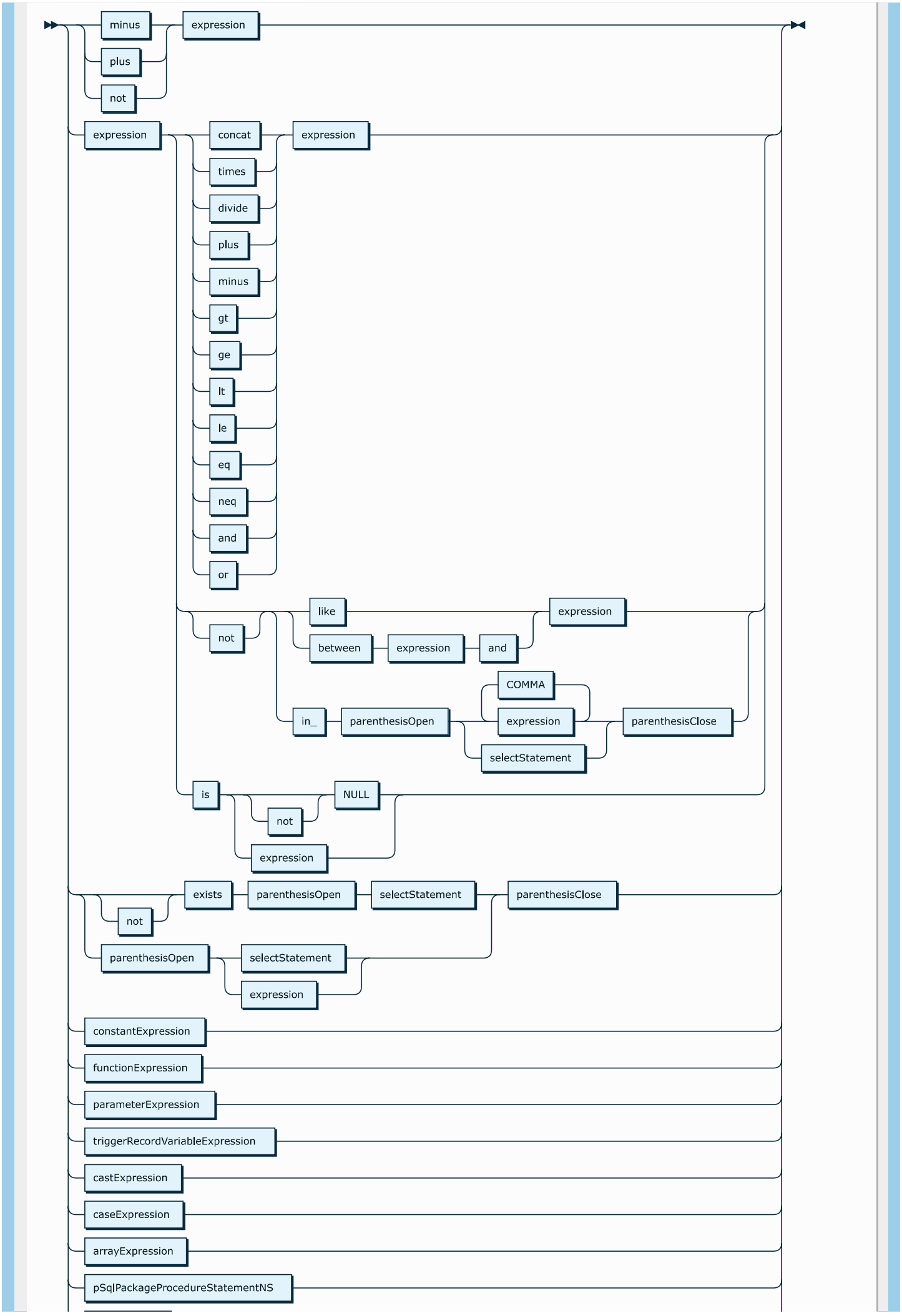
In is a n-ary operator which returns true when the left value is one of the right-hand side values after the 'in'.



referenced by:

- expression

expression:


minus
plus
not
expression
expression
concat
times
divide
plus
minus
gt
ge
lt
le
eq
neq
and
or
expression
not
like
between
expression
and
expression
in_
parenthesisOpen
COMMA
expression
selectStatement
parenthesisClose
is
not
NULL
expression
not
exists
parenthesisOpen
selectStatement
parenthesisClose
parenthesisOpen
selectStatement
expression
constantExpression
functionExpression
parameterExpression
triggerRecordVariableExpression
castExpression
caseExpression
arrayExpression
pSqlPackageProcedureStatementNS

referenced by:

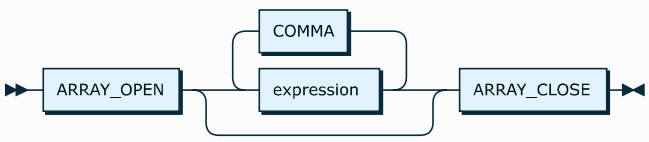
- alterSessionSetBillingId
- alterSessionSetBillingReference
- alterSessionSetIuidSourceAlias
- alterSessionSetPoolIdentityId
- arrayExpression
- attachToClause
- avgAggregateFunction
- caseElseExpression
- caseWhenThenExpression
- castExpression
- csvTableLiteral
- csvTableOptions
- excelDataRectangle
- excludeExpression
- expression
- expressionList
- firstAggregateFunction
- htmlTableLiteral
- identifiedByClause
- insertValuesList
- joinConditions
- jsonTableLiteral
- jsonTableSpec
- lastAggregateFunction
- maxAggregateFunction
- minAggregateFunction
- namedExpression
- ndjsonTableSpec
- numberedOrNamedExpressionList
- pSqlAssignmentStatement
- pSqlContinueStatement
- pSqlElsIfExpression
- pSqlExecuteImmediateStatementNS
- pSqlExecuteNativeStatementNS
- pSqlExitStatement
- pSqlForNumberLoopStatement
- pSqlIfStatement
- pSqlItemDeclaration
- pSqlReturnStatement
- pSqlWhileLoopStatement
- part
- passingSourceOrPathExpression
- productAggregateFunction
- setStatement
- sitemapExpression
- startAtExpression
- stdDevAggregateFunction
- stringSplitSpec
- sumAggregateFunction
- updateValue
- whereClause
- xmlTableLiteral
- xmlTableSpec

bindVariable:

referenced by:

- expression

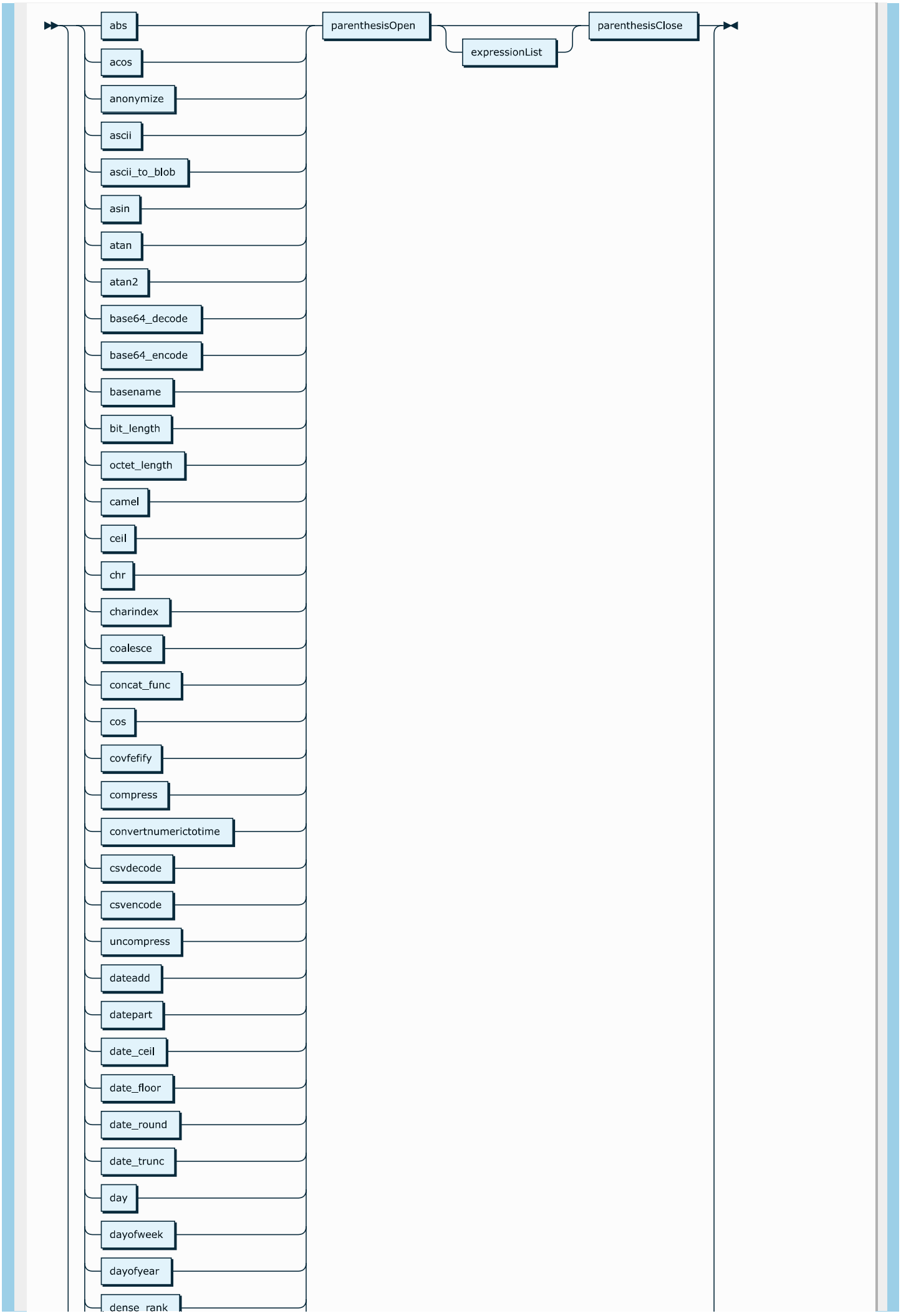
arrayExpression:

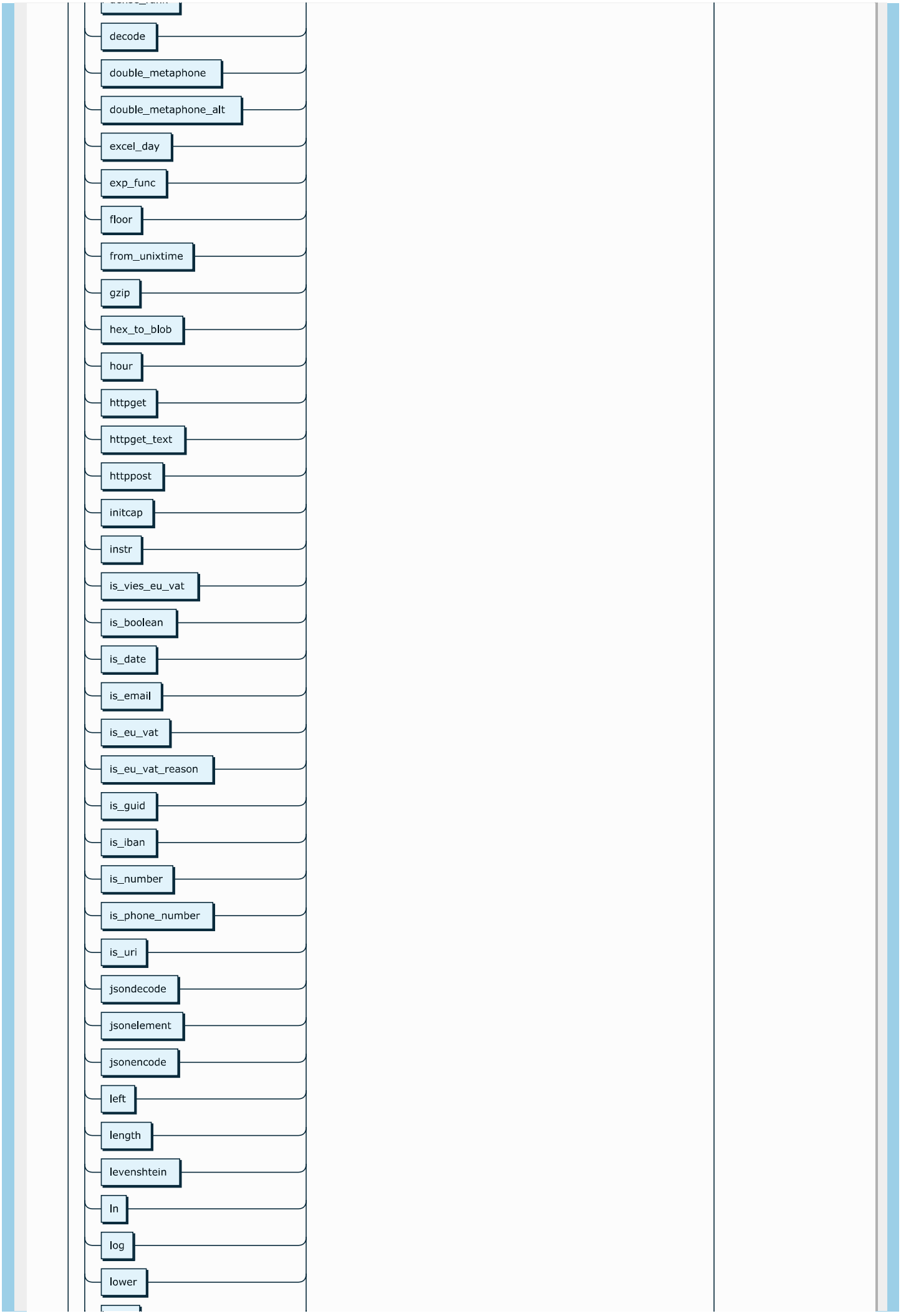


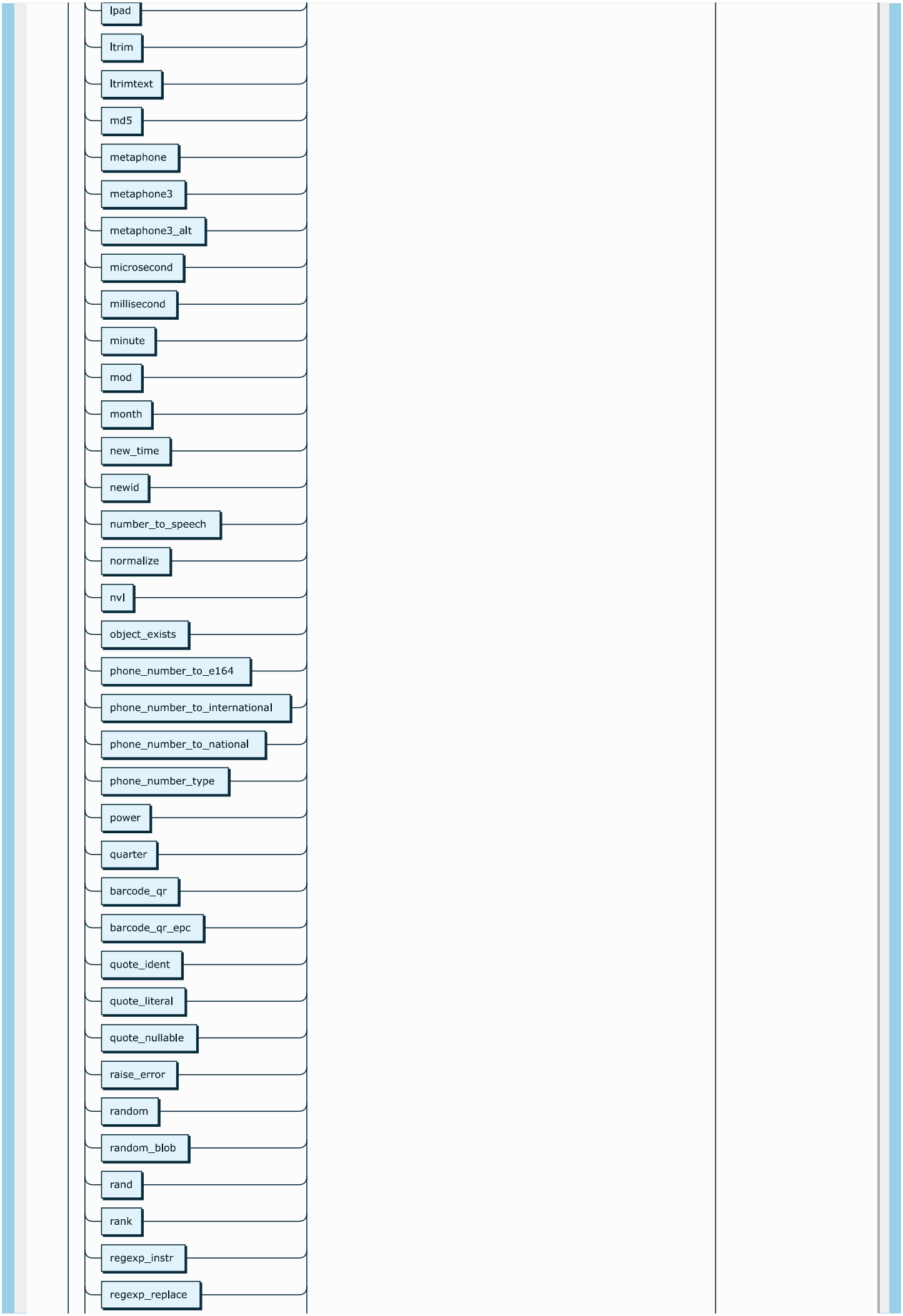
referenced by:

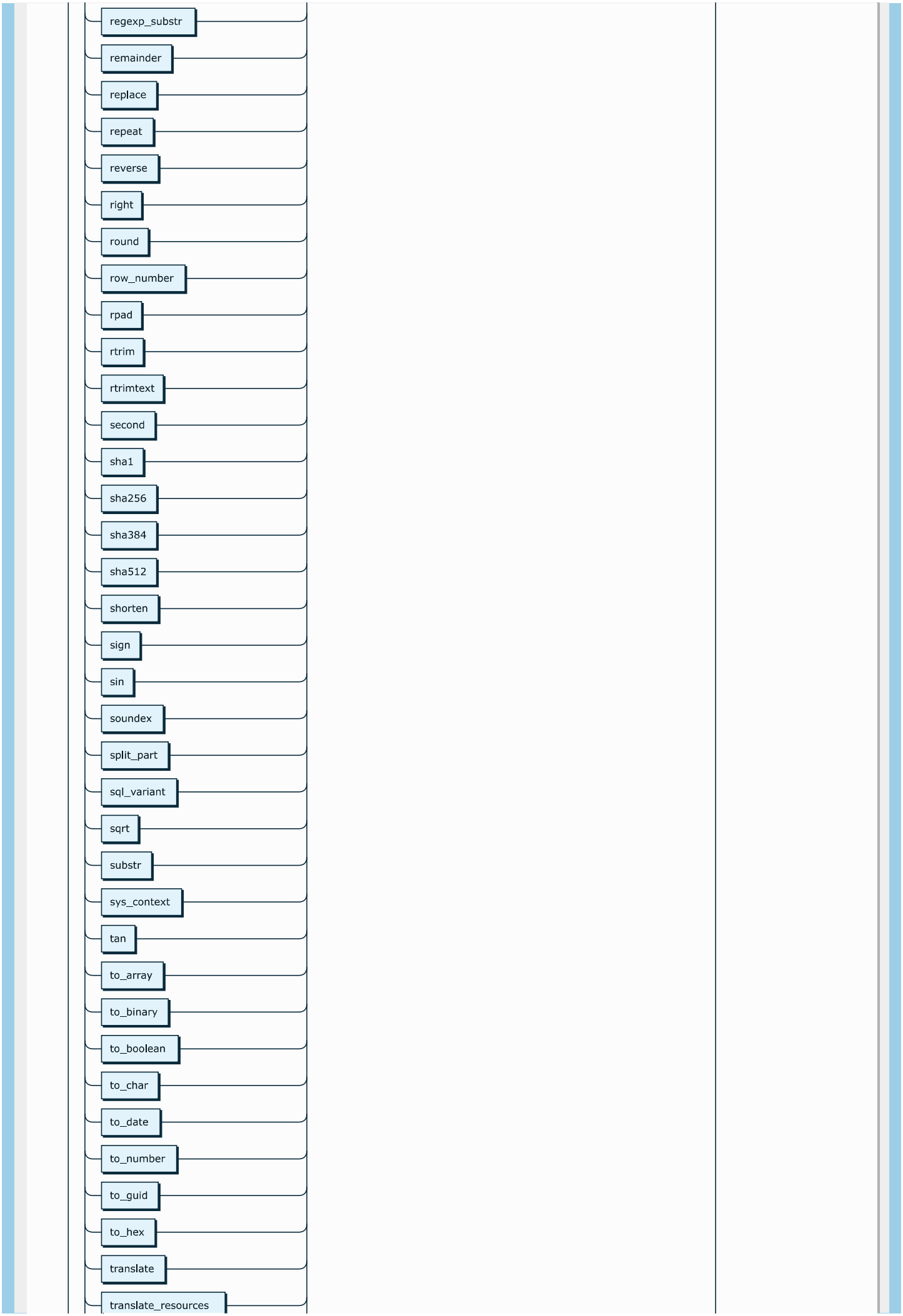
- expression

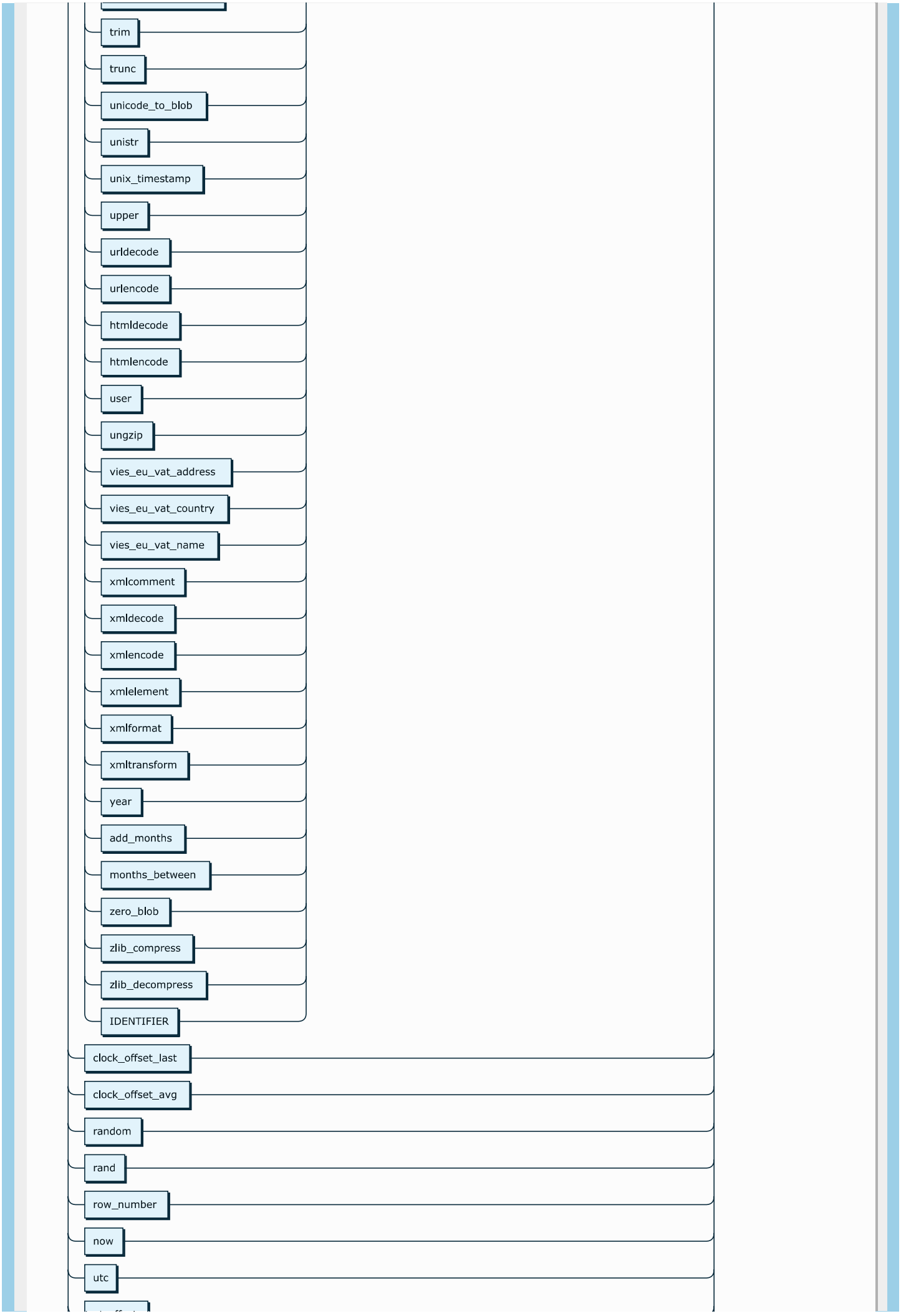
functionExpression:

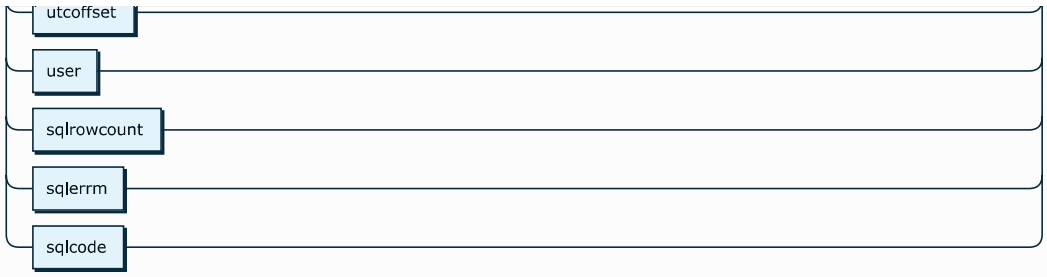

abs
acos
anonymize
ascii
ascii_to_blob
asin
atan
atan2
base64_decode
base64_encode
basename
bit_length
octet_length
camel
ceil
chr
charindex
coalesce
concat_func
cos
covfefify
compress
convertnumerictotime
csvdecode
csvencode
uncompress
dateadd
datepart
date_ceil
date_floor
date_round
date_trunc
day
dayofweek
dayofyear
dense_rank
parenthesisOpen
expressionList
parenthesisClose


decode
double_metaphone
double_metaphone_alt
excel_day
exp_func
floor
from_unixtime
gzip
hex_to_blob
hour
httpget
httpget_text
httppost
initcap
instr
is_vies_eu_vat
is_boolean
is_date
is_email
is_eu_vat
is_eu_vat_reason
is_guid
is_iban
is_number
is_phone_number
is_uri
jsondecode
jsonelement
jsonencode
left
length
levenshtein
ln
log
lower


lpad
ltrim
ltrimtext
md5
metaphone
metaphone3
metaphone3_alt
microsecond
millisecond
minute
mod
month
new_time
newid
number_to_speech
normalize
nvl
object_exists
phone_number_to_e164
phone_number_to_international
phone_number_to_national
phone_number_type
power
quarter
barcode_qr
barcode_qr_epc
quote_ident
quote_literal
quote_nullable
raise_error
random
random_blob
rand
rank
regexp_instr
regexp_replace


regexp_substr
remainder
replace
repeat
reverse
right
round
row_number
rpad
rtrim
rtrimtext
second
sha1
sha256
sha384
sha512
shorten
sign
sin
soundex
split_part
sql_variant
sqrt
substr
sys_context
tan
to_array
to_binary
to_boolean
to_char
to_date
to_number
to_guid
to_hex
translate
translate_resources


trim
trunc
unicode_to_blob
unistr
unix_timestamp
upper
urldecode
urlencode
htmldecode
htmlencode
user
ungzip
vies_eu_vat_address
vies_eu_vat_country
vies_eu_vat_name
xmlcomment
xmldecode
xmlencode
xmlelement
xmlformat
xmltransform
year
add_months
months_between
zero_blob
zlib_compress
zlib_decompress
IDENTIFIER
clock_offset_last
clock_offset_avg
random
rand
row_number
now
utc



referenced by:

- expression

abs:

Returns the absolute value of a number.
Parameters:

- Input: A number that is greater than or equal to System.Double.MinValue, but less than or equal to System.Double.MaxValue.

Returns: decimal.

referenced by:

- functionExpression

acos:

Returns the angle of the provided cosine.
Parameters:

- Input: the cosine to get the angle of.

Returns: A number which represents the angle of the provided cosine.

referenced by:

- functionExpression

anonymize:

Anonymize a text or number. Anonymization is executed such that when the same original value is anonymized within the same session, the anonymized value will be identical. The anonymized value also uniquely matches the original value. With no access to the anonymization map however, the original value can however not be calculated from the anonymized value.
In mathematics, the anonymization function is a bijection: each element of the original set is paired with exactly one element of the anonymized set, and each element of the anonymized set is paired with exactly one element of the original set.
Parameters:

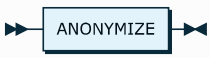
- Value: A text or number to be obfuscated.
- Maximum length (optional): Maximum length in digits for numbers or characters for text of anonymized value. Null means no restriction on maximum length.
- Mapping (optional): algorithm to use. The default algorithm is 'DEFAULT' which maps text values to a range of hexadecimal characters and numbers to a range of numbers. Alternative mappings are described below.

The following anonymization maps are available on installation:

- DEFAULT: the default algorithm.
- IVE-GL-JOURNAL-DESCRIPTION: general ledger journal descriptions: no preferred anonymizations, leave familiar and non-confidential descriptions in original state.
- IVE-GL-ACCOUNT-DESCRIPTION: general ledger account descriptions: no preferred anonymizations, leave familiar and non-confidential descriptions in original state.
- IVE-PSN-FIRST-NAME: person first names: prefer readable alternative first names, anonymize all.
- IVE-PSN-LAST-NAME: person last names: prefer readable alternative last names, anonymize all.
- IVE-ADS-CITY-NAME: address city names: prefer readable alternative city names, anonymize all.
- IVE-ADS-STREET-NAME: address street names: prefer readable alternative street names, anonymize all.

The data dictionary contains the anonymization maps used sofar in the session and their corresponding values:

Returns: Anonymized value.



referenced by:

- functionExpression

ascii:

Get the position of a character on database character set.
Parameters:

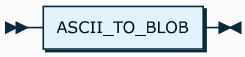
- Input: character to get position from.

Returns: the position of the character on database character set.

referenced by:

- functionExpression

ascii_to_blob:



referenced by:

- functionExpression

asin:

Returns the angle of the provided sine.
Parameters:

- Input: the sine to get the angle of.

Returns: A number which represents the angle of the provided sine.

referenced by:

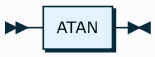
- functionExpression

atan:

Returns the angle of the provided tangent.
Parameters:

- Input: the tangent to get the angle of.

Returns: A number which represents the angle of the provided tangent.



referenced by:

- functionExpression

atan2:

Returns the angle of the provided tangent.
Parameters:

- First number: the first number to get the angle of.
- Second number: the second to get the angle of.

Returns: A number which represents the angle of the provided tangent.

referenced by:

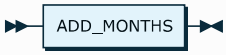
- functionExpression

add_months:

Add an amount of months to a datetime.
Parameters:

- Date: datetime to add the months to.
- Months: the amount of months to add.

Returns: A new datetime with the amount of months added.



referenced by:

- functionExpression

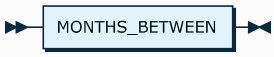
months_between:

Get the number of months between two dates. The fractional part is determined using a 30-day month length.
Parameters:

- Date 1: first date.
- Date 2: second date.

Returns: A decimal with the number of months and fractional part. Negative when date 2 is before date 1. Positive otherwise.

Introduced in 17.32.



referenced by:

- functionExpression

base64_decode:

Converts the base64_encoded value back to the binairy value as defined on Wikipedia.
Parameters:

- Input: value to convert back to the original.

Returns: the input decoded back to the binairy value.

referenced by:

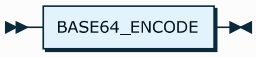
- functionExpression

base64_encode:

Converts a binairy value to base64_encoded characters as defined on Wikipedia.
Parameters:

- Input: value to convert to base64 characters.

Returns: the input encoded to base64 characters.



referenced by:

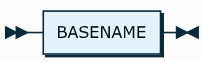
- functionExpression

basename:

Extract file name from a file path.
Parameters:

- File path: Full path of a file, consisting of drive, folders, file name and possible extension.
- Extension: Optional extension to remove, including leading '.' when necessary.

Returns: the file name from the file path.



referenced by:

- functionExpression

camel:

Converts provided string to Camel case.
Parameters:

- Input: the string that will be converted to Camel case.

Returns: A string converted to Camel case.

referenced by:

- functionExpression

ceil:

Rounds the input to the largest following integer. Unless an amount of decimals is defined, in which case it rounds to the largest integer number with the amount of decimals or date with the amount of positions.
Parameters:

- Input: A number or datetime to ceil.
- Decimals [optional]: A number to specify how many decimals it may ceil to in case of a number. In case of a datetime, it reflects the number of time positions, ranging from -2 for years to 2 for minutes.

Returns: the ceiling of the input.

referenced by:

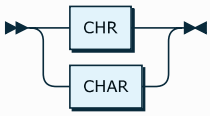
- functionExpression

chr:

Get a character from database character set.
Parameters:

- Input: a numeric value of a character.

Returns: A character from the database character set.



referenced by:

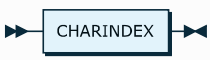
- functionExpression
- stringOrChar

charindex:

Get a number which is a position of the first occurrence of a search string in another string, starting looking at an optional start location.
Parameters:

- Search string: Text to search for.
- Full string: String to be searched in.
- Start Position [optional]: Position of string to start searching.

Returns: the position of the search string inside the full string.



referenced by:

- functionExpression

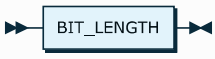
bit_length:

Get the number of bits needed to represent a value. For a blob, this is the number of bits for the bytes of the blob. For all other data types, the value is first converted to a string and then the number of bits of the UTF8 representation is determined.

Parameters:

- Value: value to determine length in bits for.

Returns: number of bits needed to represent the value.



referenced by:

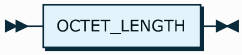
- functionExpression

octet_length:

Get the number of bytes needed to represent a value. For a blob, this is the number of bytes of the blob. For all other data types, the value is first converted to a string and then the number of bytes of the UTF8 representation is determined.
Parameters:

- Value: value to determine length in bytes for.

Returns: number of bytes needed to represent the value.



referenced by:

- functionExpression

repeat:

Get a concatenation of the text by a number of times.
Parameters:

- Text: text to repeat.
- Times: number of time to repeat the text.

Returns: the text repeated a number of times.

referenced by:

- functionExpression

raise_error:

referenced by:

- functionExpression

clock_offset_avg:

referenced by:

- functionExpression

clock_offset_last:

referenced by:

- functionExpression

coalesce:

Performs a coalescing operation.
Parameters:

- Left: an object.
- Right: an object.

Returns: the left value if right is empty, otherwise the right value.

referenced by:

- functionExpression

concat:

Concatenate the left and right values together as a text. The values must all be either a BLOB or a type that can be cast to a string.

referenced by:

- expression

concat_func:

Concatenate a list of values together as a text or BLOB. The values must all be either a BLOB or a type that can be cast to a string.

referenced by:

- functionExpression

cos:

Returns the cosine of the provided angle.
Parameters:

- Input: the angle to get the cosine of.

Returns: A number which represents the cosine of the provided angle.

referenced by:

- functionExpression

covfefify:

referenced by:

- functionExpression

compress:

referenced by:

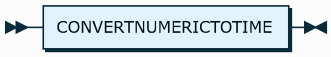
- functionExpression

convertnumerictotime:

ConvertNumericToTime converts hours with decimal numbers to time
Parameters:

- Input: number to be converted to time.
- Show Seconds [optional]: accepts boolean input.
- Zero Change [optional]: when enabled and input is 0, shows Zero Text.
- Zero Text [optional]: text to display when input is 0 and Zero Change is true.
- Show Days [optional]: accepts boolean input.

Returns: the trimmed text.



referenced by:

- functionExpression

csvdecode:

Returns the CSV decoded input.
Parameters:

- Input: the input which will be decoded into CSV.

Returns: An object which is the CSV decoded input.

referenced by:

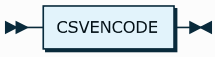
- functionExpression

csvencode:

Returns the CSV encoded input.
Parameters:

- Input: the input which will be encoded into CSV.

Returns: An object which is the CSV encoded input.



referenced by:

- functionExpression

uncompress:

referenced by:

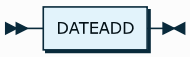
- functionExpression

dateadd:

Adds an amount of time to a date.
Parameters:

- Interval: the date interval to be added.
- Number: the number of intervals to add.
- Date: the date to wich the interval should be added.

Returns: the original date with the number of intervals added.



referenced by:

- functionExpression

datepart:

Get the specified datepart from a datetime.
Parameters:

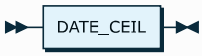
- datepart: a part of a date.
- date: a datetime to get the datepart from.

Returns: a part of a datetime.

referenced by:

- functionExpression

date_ceil:



referenced by:

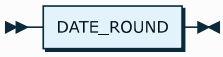
- functionExpression

date_floor:

referenced by:

- functionExpression

date_round:



referenced by:

- functionExpression

date_trunc:

referenced by:

- functionExpression

day:

Collect the day from a date.
Parameters:

- Input: A dateTime.

Returns: the day as an integer.

referenced by:

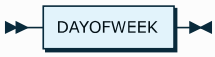
- functionExpression

dayofweek:

Collect the day of a week from a date.
Parameters:

- Input: A dateTime.

Returns: the day of a week as an integer.



referenced by:

- functionExpression

dayofyear:

Collect the day of a year from a date.
Parameters:

- Input: A dateTime.

Returns: the day of a year as an integer.

referenced by:

- functionExpression

decode:

Decode function is a function expression if...then...then...else. The expression is compared with all search values. Upon first match, the function evaluates to the result following the search. If no match found, the function evaluates to the optional default. When no default is present, the function evaluates to NULL.
Parameters:

- Expression: the value to compare with. Is automatically converted to the datatype of the first search value.
- Search[n]: if Expression is equal to Search, the function evaluates to the result.
- Result[n]: the value returned if expression equals to search
- Default: if no matches are found, default will be returned.

Returns: the result of the first match and otherwise the default value.

referenced by:

- functionExpression

dense_rank:

referenced by:

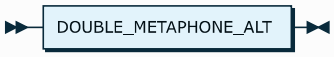
- functionExpression

double_metaphone:

referenced by:

- functionExpression

double_metaphone_alt:



referenced by:

- functionExpression

divide:

Divide one number by the second number.
Parameters:

- first: a number to divide.
- second: a number to divide with.

Returns: the divided output.

referenced by:

- expression

exists:

referenced by:

- expression

exp:

Returns the provided number raised to the specified power.
Parameters:

- Input: the number to raise by the specified power.

Returns: A number which is the provided number raised to the specified power.

no references

exp_func:

referenced by:

- functionExpression

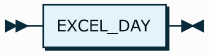
excel_day:

Get the number which in Excel represents the date. The leap year bug of Excel is included, so the day number increments by 2 between February 28, 1900 and March 1, 1900.
Parameters:

- Date: datetime to get number for.

Returns: the number which in Excel represents the date, with 1 meaning January 1, 1900.

Introduced in 17.32.



referenced by:

- functionExpression

floor:

Rounds the input to the smallest following integer. Unless an amount of decimals is defined, in which case it rounds to the smallest integer with the amount of decimals or date with the amount of positions.
Parameters:

- Input: A number or datetime to floor.
- Decimals [optional]: A number to specify how many decimals it may floor to in case of a number. In case of a datetime, it reflects the number of time positions, ranging from -2 for years to 2 for minutes.

Returns: the floor of the input.

referenced by:

- functionExpression

from_unixtime:

Get the date/time from an integer representing a UNIX epoch time.
Parameters:

- Input: An integer.

Returns: the date/time which the UNIX epoch time represents.

referenced by:

- functionExpression

gzip:

Compress a BLOB using the GZIP-algorithm.
Parameters:

- input: The BLOB to compress.

Returns: a GZIP-compressed version of the input.

referenced by:

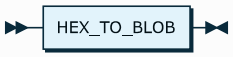
- functionExpression

hex_to_blob:

Cast a text in hexadecimal notation to the corresponding BLOB. Reverse of to_hex.
Parameters:

- Text: text in hexadecimal notation with two characters per byte.

Returns: BLOB.



referenced by:

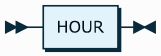
- functionExpression

hour:

Collect the hour from a date.
Parameters:

- Input: A dateTime.

Returns: the hour as an integer.



referenced by:

- functionExpression

htmldecode:

referenced by:

- functionExpression

htmlencode:

referenced by:

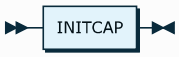
- functionExpression

initcap:

Changes the first letter of each word in uppercase, all other letters in lowercase.
Parameters:

- Input: Text to convert.

Returns: the input with the first letter of each word in uppercase.



referenced by:

- functionExpression

instr:

Get a number which is a position of the first occurrence of substring in the string.
Parameters:

- String: String to be searched.
- Substring: Text to search for.
- StartPosition [optional]: Position of string to start searching.
- occurrence [optional]: Return the position of the occurrence.

Returns: the position of the substring inside the original string.

referenced by:

- functionExpression

is_vies_eu_vat:

Evaluates to boolean true when the value provided is a valid EU VAT account that is active and approved for cross-country use in the EU. Each validation takes approximately 250 ms and uses the VIES service offered by the European Union.
Parameters:

- Value: value to test.

Returns: True when the value provided is a valid and approved EU vat account.

referenced by:

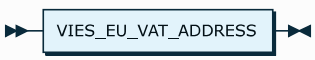
- functionExpression

vies_eu_vat_address:

Evaluates to text of the address linked to the EU VAT account provided (when the EU VAT account is valid and active). Each validation takes approximately 250 ms and uses the VIES service offered by the European Union.
Parameters:

- Value: value to test.

Returns: address associated with the EU VAT account. Null when not a valid/active EU VAT account.



referenced by:

- functionExpression

vies_eu_vat_country:

Evaluates to text of the country linked to the EU VAT account provided (when the EU VAT account is valid and active). Each validation takes approximately 250 ms and uses the VIES service offered by the European Union.
Parameters:

- Value: value to test.

Returns: country associated with the EU VAT account. Null when not a valid/active EU VAT account.

referenced by:

- functionExpression

vies_eu_vat_name:

Evaluates to text of the name linked to the EU VAT account provided (when the EU VAT account is valid and active). Each validation takes approximately 250 ms and uses the VIES service offered by the European Union.
Parameters:

- Value: value to test.

Returns: name associated with the EU VAT account. Null when not a valid/active EU VAT account.

referenced by:

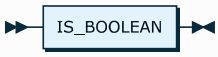
- functionExpression

is_boolean:

Determines whether a value is a valid boolean. When true, the value can be converted by to_boolean.
Parameters:

- Input: value to convert.

Returns: true when the value can be converted to a boolean, false otherwise.



referenced by:

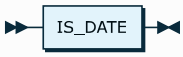
- functionExpression

is_date:

Determines whether a value is a valid number according to the specified format. When true, the value can be converted by to_date.
Parameters:

- Input: value to convert.
- Format: format mask. Optional; defaults to local style. List of format masks is available in a separate section.

Returns: true when the value can be converted to a date, false otherwise.



referenced by:

- functionExpression

is_email:

Evaluates to boolean true when the value provided is a valid email address according to RFC 822.
Parameters:

- Email: email address.
- Allow plus sign: indicator whether the use of a plus sign ('+') to introduce an alias is allowed (defaults to false).
- Check TLD: indicator whether the TLD of the domain name must be validated against the current list (defaults to true).

Returns: True when the value provided is a valid email address.

referenced by:

- functionExpression

is_eu_vat:

Evaluates to boolean true when the value provided can be a valid EU VAT account using a fixed algorithm. The fixed algorithm accepts VAT numbers that are valid according to the country's rules for assigning tax numbers. It does NOT check whether the VAT account is in use. Use `is_vies_eu_vat` to check whether it is actually active and approved for cross-country EU use.
Parameters:

- Value: value to test.

Returns: True when the value provided can be a valid EU vat account.

referenced by:

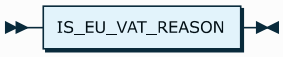
- functionExpression

is_eu_vat_reason:

Evaluates to text expressing the reason why the value can not be a valid EU VAT account using a fixed algorithm.
Parameters:

- Value: value to test.

Returns: text indicating reason why the value can not be a valid EU VAT account.



referenced by:

- functionExpression

is_guid:

Determines whether a value is a valid GUID. When true, the value can be converted by to_guid.
Parameters:

- Input: value to convert.

Returns: true when the value can be converted to a GUID, false otherwise.

referenced by:

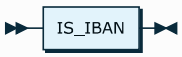
- functionExpression

is_iban:

Evaluates to boolean true when the value provided is a valid IBAN account.
Parameters:

- Value: value to test.
- Remove space: indicator whether spaces should be removed before IBAN validation.

Returns: True when the value provided is a valid IBAN account.



referenced by:

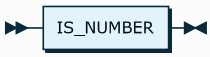
- functionExpression

is_number:

Determines whether a value is a valid number according to the specified format. When true, the value can be converted by to_number.
Parameters:

- Input: value to convert.
- Format: format mask. Optional; defaults to local style. List of format masks is available in a separate section.

Returns: true when the value can be converted to a number, false otherwise.



referenced by:

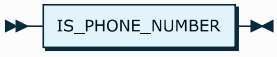
- functionExpression

is_phone_number:

Evaluates to boolean true when the value provided is a valid phone number for the country indicated. In case no country is specified, an E.164 notation is required.
Parameters:

- Phone: phone number as text.
- Country: country code as ISO 3166-1 alpha-2 code.

Returns: True when the value provided is a valid phone number for the country indicated.



referenced by:

- functionExpression

is_uri:

referenced by:

- functionExpression

jsondecode:

referenced by:

- functionExpression

jsonelement:

referenced by:

- functionExpression

jsonencode:

referenced by:

- functionExpression

length:

Gets the number of characters in provided string.
Parameters:

- Input: the string to get the length of.

Returns: A number which represents the number of characters in the provided string.

referenced by:

- functionExpression

levenshtein:

Determine the Levenshtein distance between two values as defined on Wikipedia.

referenced by:

- functionExpression

ln:

Get the natural logarithm of a number.
Parameters:

- Input: a number to get the natural logarithm from.

Returns: the natural logarithm of the input.

referenced by:

- functionExpression

log:

Get the natural logarithm of a number in a specified base.
Parameters:

- Input: a number to get the natural logarithm from.
- Base [optional]: the base to get the natural logarithm from.

Returns: the natural logarithm of the input in the specified base.

referenced by:

- functionExpression

lower:

Converts provided string to lowercase.
Parameters:

- Input: the string that will be converted to lowercase.

Returns: A string converted to lowercase.

referenced by:

- functionExpression

lpad:

Pad a string to the left to make it a specified length.
Parameters:

- Input: string to be padded.
- Lenght: the length the string should be padded to.
- Characters [optional]: Characters to pad with.

Returns: A string padded to the left to a given length with the optional specified characters.

referenced by:

- functionExpression

ltrim:

Trims characters from the left side of a string.
Parameters:

- Input: the string from to trim characters from the left side.
- (Optional) Characters to trim: all character(s) to trim. Default is " ".

Returns: A string with characters trimmed from the left.

referenced by:

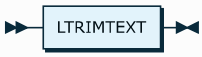
- functionExpression

ltrimtext:

Ltrimtext trims the left side of a text by a list of trim texts. When the text starts with a trim text, the trim text is removed.
Parameters:

- Text: text to be trimmed.
- Trim Text [optional]: text(s) we want to left trim from Text.

Returns: the trimmed text.



referenced by:

- functionExpression

md5:

Converts a value to a 128-bit hash value as defined on Wikipedia.
Parameters:

- Input: Text to convert with MD5.

Returns: the input converted with MD5.

referenced by:

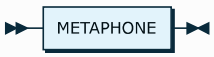
- functionExpression

metaphone:

Converts a value to the Metaphone code as defined on Wikipedia.
Parameters:

- Input: value to convert to metaphone.
- Length: maximum output length of the given input.

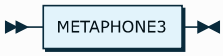
Returns: the input converted to metaphone, with a given output length.



referenced by:

- functionExpression

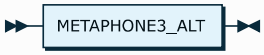
metaphone3:



referenced by:

- functionExpression

metaphone3_alt:



referenced by:

- functionExpression

mod:

Get the remainder of a divide calculation.
Parameters:

- dividend: a number.
- divider: a number.

Returns: the remainder.

referenced by:

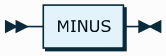
- functionExpression

minus:

Subtracts a value from another.
Parameters:

- Value: a number or datetime.
- Subtract: a number or datetime.

Returns: the value minus the subtraction.



referenced by:

- expression

minute:

Collect the minute from a date.
Parameters:

- Input: A dateTime.

Returns: the minute as an integer.

referenced by:

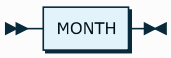
- functionExpression

month:

Collect the month from a date.
Parameters:

- Input: A dateTime.

Returns: the month as an integer.



referenced by:

- functionExpression

new_time:

referenced by:

- functionExpression

newid:

Creates a new Guid id.
Returns: the new Guid id.

referenced by:

- functionExpression

nvl:

Coalesce all values together.
Returns: All values coalesced together.

referenced by:

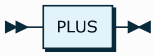
- functionExpression

plus:

Adding a value to another.
Parameters:

- Value: a number or datetime.
- add: a number or datetime.

Returns: A new value with both values added to eachother.



referenced by:

- expression

power:

Gets a value of a number raised to another.
Parameters:

- Value: a number.
- exponent: a number.

Returns: the value of a number raised to another.

referenced by:

- functionExpression

object_exists:

referenced by:

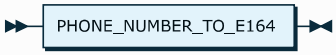
- functionExpression

phone_number_to_e164:

Evaluates to the phone number in E.164 notation format. In case no country is specified, an E.164 notation is required.
Parameters:

- Phone: phone number as text.
- Country: country code as ISO 3166-1 alpha-2 code.

Returns: Phone number in E.164 notation format.



referenced by:

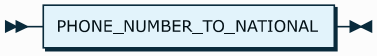
- functionExpression

phone_number_to_national:

Evaluates to the phone number in national notation format. In case no country is specified, an E.164 notation is required.
Parameters:

- Phone: phone number as text.
- Country: country code as ISO 3166-1 alpha-2 code.

Returns: Phone number in national notation format.



referenced by:

- functionExpression

phone_number_to_international:

Evaluates to the phone number in international notation format. In case no country is specified, an E.164 notation is required.
Parameters:

- Phone: phone number as text.
- Country: country code as ISO 3166-1 alpha-2 code.

Returns: Phone number in international notation format.

referenced by:

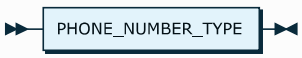
- functionExpression

phone_number_type:

Evaluates to the type of phone number: FIXED_LINE: fixed landline, FIXED_LINE_OR_MOBILE: either fixed landline phone or mobile number, MOBILE: mobile phone number, PAGER: pager, PERSONAL_NUMBER: personal phone number, PREMIUM_RATE: premium rate, SHARED_COST: shared cost, TOLL_FREE: toll-free, UAN: Universal Access Number; a phone number that can be accessed without dialing geographic area codes, UNKNOWN: unknown type, VOICEMAIL: voicemail, VOIP: VoIP. In case no country is specified, an E.164 notation is required.
Parameters:

- Phone: phone number as text.
- Country: country code as ISO 3166-1 alpha-2 code.

Returns: Type of phone number.



referenced by:

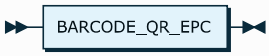
- functionExpression

barcode_qr:

referenced by:

- functionExpression

barcode_qr_epc:



referenced by:

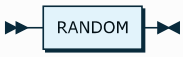
- functionExpression

random:

Generates a random number between 0 and 1.
Parameters:

- Seed: Produce a repeatable sequence of random numbers each time that seed value is provided.

Returns: A random number between 0 and 1.



referenced by:

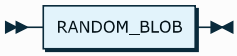
- functionExpression

random_blob:

Generates a blob with pseudo-random values.
Parameters:

- Length: Produce a blob with this length in terms of bytes.

Returns: A blob with pseudo-random values.



referenced by:

- functionExpression

rand:

referenced by:

- functionExpression

rank:

referenced by:

- functionExpression

regexp_substr:

Extracts a substring from the given value using regular expression. The regular expression must follow the Microsoft.net regular expression language.
Parameters:

- Input: The text to get the substring from.

- Pattern: Regular expression pattern.
- Start position [optional]: The start index from the input.
- Appearance [optional]: Indicating the appearance of the substr operation.
- Match_parameter [optional]: A text literal that lets you change the default matching behavior of the function.

Returns: the substring from the input.

referenced by:

- functionExpression

regexp_instr:

Determine the position of the regular expression in the given value. Returns 0 when the regular expression is not contained in the given value. The regular expression must follow the Microsoft.net regular expression language.
Parameters:

- Input: The text to get the regular expression position from.
- Pattern: Regular expression pattern.
- Start position [optional]: The start index from the input.
- Appearance [optional]: Indicating the appearance of the instr operation.
- ReturnOption [optional]: Select either the first character found or the first character after the occurrence of the pattern.
- Match_parameter [optional]: A text literal that lets you change the default matching behavior of the function.

Returns: the location of a regular expression pattern in the input.

referenced by:

- functionExpression

regexp_replace:

Replaces all occurrences matching the regular expression with the replacement value. The replacement value may contain references to matches in the regular expression by using the dollar-sign ('$') plus the reference number. The regular expression must follow the Microsoft.net regular expression language.

Parameters:

- Input: The text to get the substring from.
- Pattern: Regular expression pattern.
- Replacement [optional]: Text to replace with.
- Start position [optional]: The start index from the input.
- Appearance [optional]: Indicating the appearance of the replace operation (use '0' for all appearances).
- Match_parameter [optional]: A text literal that lets you change the default matching behavior of the function. The available options are 'c' for case-sensitive, 'i' for ignore case, 'n' for single-line, 'm' for multi-line and 'x' for ignore pattern white space.

Returns: the input with every occurrence of the regular expression pattern replaced with the replacement.

referenced by:

- functionExpression

remainder:

Get the remainder of a divide calculation.
The REMAINDER function uses the round function in its formula, whereas the MOD function uses the floor function in its formula.
Parameters:

- Number1: a number.
- Number2: a number.

Returns: the remainder.

referenced by:

- functionExpression

replace:

Changes a given text by replacing all occurrences from an ordered list of texts by their associated replacement texts.
Parameters:

- Input: the text to change.
- Old text 1: the text to be replaced.
- New text 1: the replacement text.
- Old text n: additional texts to be replaced.
- New text n: additional replacement texts.

Returns: A text with all old texts replaced by their new text in the order they occur in the list.

referenced by:

- functionExpression

reverse:

Flips the input around.
Parameters:

- Input: text to flip around.

Returns: the text with it's characters in reversed order.

referenced by:

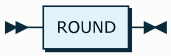
- functionExpression

round:

Rounds the input to the closest following integer. Unless an amount of decimals is defined, in which case it rounds to the closest integer number with the amount of decimals or date with the amount of positions.
Parameters:

- Input: A number or datetime to round.
- Decimals [optional]: A number to specify how many decimals it may round to in case of a number. In case of a datetime, it reflects the number of time positions, ranging from -2 for years to 2 for minutes.

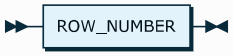
Returns: the rounded input.



referenced by:

- functionExpression

row_number:



referenced by:

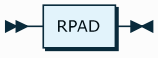
- functionExpression

rpad:

Rightpad function pads the right-side of a string with a specific set of characters to the given length. When no set of characters given, it will pad with a whitespace.
Parameters:

- Input: Text to be padded.
- Length: The length to make the input to.
- Pad text [optional]: Text to add to the input if the length is larger then the input.

Returns: the padded text, or null if the string cannot be padded.



referenced by:

- functionExpression

rtrim:

Trims characters from the right side of a string.
Parameters:

- Input: the string from which to trim characters from the right side.
- (Optional) Characters to trim: the character to trim. Default is " ".

Returns: A string with characters trimmed from the right.

referenced by:

- functionExpression

rtrimtext:

Ltrimtext trims the right side of a text by a list of trim texts. When the text ends with a trim text, the trim text is removed.
Parameters:

- Text: text to be trimmed.
- Trim Text [optional]: text(s) we want to right trim from Text.

Returns: the trimmed text.

referenced by:

- functionExpression

microsecond:

Collect the microsecond from a date.
Parameters:

- Input: A dateTime.

Returns: the microsecond as an integer.

referenced by:

- functionExpression

millisecond:

Collect the millisecond from a date.
Parameters:

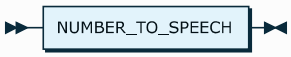
- Input: A dateTime.

Returns: the millisecond as an integer.

referenced by:

- functionExpression

number_to_speech:



referenced by:

- functionExpression

normalize:

Normalize a file path by replacing all invalid and non-ASCII characters for use in a file path by underscore. After that, the file path is made more readable by various operations such as removal of duplicate whitespace and underscore characters.
Parameters:

- Original file path: path of the file.
- Maximum file name length: length in characters into which the normalized file name must fit.
- Allow path separator: whether to allow the path separator '\' in the normalized file name. When not, occurrences are replaced.

Returns: a normalized file path.

referenced by:

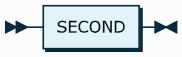
- functionExpression

second:

Collect the second from a date.
Parameters:

- Input: A dateTime.

Returns: the second as an integer.



referenced by:

- functionExpression

sha1:

Calculate hash value for parameter according to SHA1 algorithm.
Parameters:

- Value: text or BLOB to calculate SHA1 hash value for.

Returns: Hash value.

referenced by:

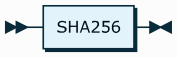
- functionExpression

sha256:

Calculate hash value for parameter according to SHA256 algorithm.
Parameters:

- Value: text or BLOB to calculate SHA256 hash value for.

Returns: Hash value.



referenced by:

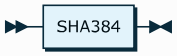
- functionExpression

sha384:

Calculate hash value for parameter according to SHA384 algorithm.
Parameters:

- Value: text or BLOB to calculate SHA384 hash value for.

Returns: Hash value.



referenced by:

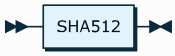
- functionExpression

sha512:

Calculate hash value for parameter according to SHA512 algorithm.
Parameters:

- Value: text or BLOB to calculate SHA512 hash value for.

Returns: Hash value.



referenced by:

- functionExpression

shorten:

Reduces a text to a maximum length. When the text exceeds the maximum length, text is removed from the middle by replacing the removed text by a shorter text.
Parameters:

- Input: Text to shorten.
- Length: Maximum length of the shortened text.
- Replacement: Text to replaced the removed text by; defaults to ellipsis ('...').

Returns: Input shortened to length.

referenced by:

- functionExpression

soundex:

Converts a value to the Soundex code as defined on Wikipedia.
Parameters:

- Input: Text to that retrieve the soundex value from.

Returns: A text started with a number and followed by 3 digits.

referenced by:

- functionExpression

sign:

Returns the sign of a number, with -1 representing a negative number, 1 a positive number and 0 a zero.
Parameters:

- Input: A number.

Returns: decimal.

referenced by:

- functionExpression

sin:

Returns the sine of the provided angle.
Parameters:

- Input: the angle to get the sine of.

Returns: A number which represents the sine of the provided angle.

referenced by:

- functionExpression

split_part:

referenced by:

- functionExpression

sql_variant:

referenced by:

- functionExpression

sqlrowcount:

referenced by:

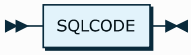
- functionExpression

sqlerrm:

referenced by:

- functionExpression

sqlcode:



referenced by:

- functionExpression

sqrt:

Returns the square root of the provided number.
Parameters:

- Input: the number to get the square root of.

Returns: A number which represents the square root of the provided number.

referenced by:

- functionExpression

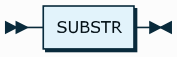
substr:

Gets a substring from the input.
Parameters:

- Input: text to gather the substring from.
- Start: start position.

- Lenght: maximum lenght of the substring.

Returns: the substring from the original input.



referenced by:

- functionExpression

sys_context:

Text value of a parameter associated with a context.
Parameters:

- context: a namespace.
- parameter: name of the parameter.
- (Optional) alias: name of the active data container to query. The most high ranked data container is queried when not specified or null.

The namespace DATACONTAINERATTR can have all data container attributes as parameter name. The namespace USERENV has the following parameter names:

- APPLICATION_VERSION: version of the client application.
- APPLICATION_VERSION1: first number of version of the client application.
- APPLICATION_VERSION2: first two numbers of version of the client application.
- APPLICATION_VERSION3: first three numbers of version of the client application.
- APPLICATION_FULL: name and version of the client application.
- APPLICATION_BUILD_EXPIRATION_DATE: build expiration date of the client application.
- AUTHENTICATION_METHOD: current authentication method.
- BILLING_ID: billing ID deviating from agreement code.
- BILLING_REFERENCE: additional billing reference.
- CLIENT_IP_ADDRESS_INTERNAL: internal IP address of the client device.
- CLIENT_IP_ADDRESS_EXTERNAL: external IP address of the client device.
- CLIENT_LOGICAL_CORE_COUNT: number of logical processor cores in the client device.
- CLIENT_MACHINE_NAME: machine name of the client device.
- CLIENT_SYSTEM_64_BIT: whether the OS is 64-bit on the client device.
- CLIENT_SYSTEM_NAME: full OS name running on the client device.
- CLIENT_SYSTEM_DIRECTORY: system directory of the client device.
- CLIENT_SYSTEM_PAGE_SIZE: system page size of the client device.
- CLIENT_VIRTUAL_MACHINE: whether the client device is a virtual machine.
- CLR_VERSION_BUILD: build version of the Common Language Runtime.
- CLR_VERSION_MAJOR: major version of the Common Language Runtime.
- CLR_VERSION_MAJOR_REVISION: major revision of the Common Language Runtime.
- CLR_VERSION_MINOR: minor version of the Common Language Runtime.
- CLR_VERSION_MIN_REVISION: minor revision of the Common Language Runtime.
- COMPANY_ID: ID of the company of current user.
- COMPANY_NAME: name of the company of current user.
- COMPANY_PHONE: phone of the company of current user.
- COMPANY_IBAN: IBAN of the company of current user.
- COMPANY_BIC: BIC of the company of current user.
- COMPANY_WEB_SITE: web site of the company of current user.
- COMPANY_STATE: state of the company of current user.
- COMPANY_POSTAL_CODE: postal code of the company of current user.
- COMPANY_COUNTRY: country of the company of current user.
- DATA_CONTAINER_ALIAS: alias of active data container.
- DATA_CONTAINER_ID: ID of active data container.
- DATABASE_DESCRIPTION: description of database.
- DATABASE_FULL_NAME: full name of database.
- DATABASE_VERSION: version of database.
- DIRECTORY_SEPARATOR: OS-specific separator for directory elements.
- DIRECTORY_SEPARATOR_ALT: Alternative OS-specific separator for directory elements.
- IIID: Invantive installation ID.
- IUID: Invantive user ID.
- LANG: ISO abbreviation for the language name of the user. Alternative: USER_LANGUAGE_CODE.
- MODULE: name of the client application. Alternative: APPLICATION_NAME.
- PATH_SEPARATOR: OS-specific separator for path.
- POOL_IDENTITY_ID: pool identity ID.
- PROCESS_64_BIT: whether the OS process on the client device runs as 64-bit.
- PROCESS_COMMAND_LINE: command line used to start the OS process.
- PROCESS_CURRENT_DIRECTORY: current directory of the OS process.
- PROCESS_STACK_TRACE: current stack trace of the OS process.
- PROCESS_WORKING_SET: working set of the OS process.
- PROVIDER_DESCRIPTION: description of active data container.
- PROVIDER_DOCUMENTATION_URL: documentation (URL) of active data container.
- PROVIDER_DOWNLOAD_IMPLEMENTATION_URL: download driver (URL) of active data container.
- PROVIDER_INSTANCE_UID: UID of active data container.
- PROVIDER_NAME: name of active data container.
- PROVIDER_SHORT_NAME: short name of active data container.
- PROVIDER_TECHNICAL_DOCUMENTATION_URL: technical documentation (URL) of active data container.
- SESSION_USER: log on code of the current user. Alternatives: CURRENT_USER and USER.
- SESSION_TIME_OFFSET_SEC: difference in seconds between UTC and the date/time representation of the session. A positive number means that the represented date/time of the session is higher than the UTC time of the date/time.
- SESSIONID: session ID of current session.
- UI_LANGUAGE_CODE: language code of the user interface.
- USER_DOMAIN_NAME: Windows domain name of current user.
- USER_EMAIL_ADDRESS: email address of current user.
- USER_FIRST_NAME: first name of current user.
- USER_FULL_NAME: full name of current user.

- USER_GENDER: gender of current user.
- USER_HOME_DIRECTORY: home directory of current user on client device.
- USER_INTERACTIVE: whether the current user works interactive.
- USER_PICTURES_DIRECTORY: pictures directory of current user on client device.
- USER_FAVORITES_DIRECTORY: favorites directory of current user on client device.
- USER_DESKTOP_DIRECTORY: desktop directory of current user on client device.
- USER_DOCUMENTS_DIRECTORY: documents directory of current user on client device.
- USER_PROFILE_DIRECTORY: profile directory of current user on client device.
- USER_LAST_LOG_ON: time of last log on of current user.
- USER_LAST_NAME: last name of current user.
- USER_LINKED_IN: LinkedIn name of current user.
- USER_MIDDLE_NAME: middle name of current user.
- USER_MOBILE_NUMBER: mobile number of current user.
- USER_NATIONALITY: nationality of current user.
- USER_PHONE_NUMBER: phone number of current user.
- USER_PICTURE_URL: picturel (URL) of current user.
- USER_SKYPE: Skype name of current user.
- USER_TITLE: title of current user.
- USER_TWITTER: Twitter name of current user.
- USER_WEB_SITE: personal web site of current user.
- VOLUME_SEPARATOR: OS-specific separator for volume.

Returns: Value of the parameter in the context namespace.

referenced by:

- functionExpression

tan:

Returns the tangent of the provided angle.
Parameters:

- Input: the angle to get the tangent of.

Returns: A number which represents the tangent of the provided angle.

referenced by:

- functionExpression

times:

Multiplies one number by the second number.
Parameters:

- First: a number to multiply.
- Second: a number to multiply with.

Returns: the first number multiplied by the second number.

referenced by:

- expression

translate:

Translate replaces all occurrences of each character in from_string to its corresponding character in to_string.
Parameters:

- input: The string to replace a sequence of characters with another set of characters.
- from_string: The string that will be searched for in the input.
- to_string: All characters in the from_string will be replaced with the corresponding character in the to_string

Returns: the input with all occurrences of each character in from_string replaced by its corresponding character in to_string.

referenced by:

- functionExpression

translate_resources:

Replace all Invantive-style resources ('{res:...}') by their translation in the specified user interface language.
Parameters:

- txt: The string to replace resources in.
- language: ISO 639-1 language code. Optional; defaults to current user interface language.

Returns: the input with all resources replaced by their translation in the specified language.

referenced by:

- functionExpression

trim:

Trims whitespaces from both sides of the provided string.
Parameters:

- Input: the string from which to trim characters.

Returns: A string trimmed from whitespaces from both sides.

referenced by:

- functionExpression

trunc:

Calculates the integral part of a number. Unless an amount of decimals is defined, in which case it calculates to the integer with the amount of decimals or date with the amount of positions.
Parameters:

- Input: A number or datetime to truncate.
- Decimals [optional]: A number to specify how many decimals it may truncate to in case of a number. In case of a datetime, it reflects the number of time positions, ranging from -2 for years to 2 for minutes.

Returns: the truncated input.

referenced by:

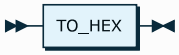
- functionExpression

to_hex:

Converts a value into a hexadecimal number.
Parameters:

- Input: value to convert.

Returns: the input converted to a hexadecimal number.



referenced by:

- functionExpression

unicode_to_blob:

Converts a value into a BLOB. Text is interpreted as UTF-16.
Parameters:

- Input: value to convert.

Returns: the input converted to a BLOB.

referenced by:

- functionExpression

unistr:

Converts a text with unicodes to regular characters.
Parameters:

- Input: text with unicodes.

Returns: the input converted to all regular characters.

referenced by:

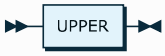
- functionExpression

upper:

Converts provided string to uppercase.
Parameters:

- Input: the string that will be converted to uppercase.

Returns: A string converted to uppercase.



referenced by:

- functionExpression

urldecode:

Decodes a url.
Parameters:

- Url: url to decode.

Returns: the decoded url.

referenced by:

- functionExpression

urlencode:

Encodes a url.
Parameters:

- Url: url to encode.

Returns: the encoded url.

referenced by:

- functionExpression

unix_timestamp:

Get the UNIX epoch time of a date/time.
Parameters:

- Input: A dateTime. Current date/time is used when no value is specified.

Returns: the UNIX epoch time.

referenced by:

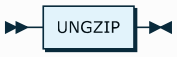
- functionExpression

ungzip:

Decompress a BLOB using the GZIP-algorithm.
Parameters:

- input: The BLOB to decompress.

Returns: the decompressed BLOB.



referenced by:

- functionExpression

zip:

referenced by:

- zipAggregateFunction

xmlcomment:

Format a text as an XML comment.
Parameters:

- Input: the input which will be formatted as XML comment.

Returns: A text with the input as XML comment.

referenced by:

- functionExpression

xmldecode:

Returns the XML decoded input.
Parameters:

- Input: the input which will be decoded into XML.

Returns: An object which is the XML decoded input.

referenced by:

- functionExpression

xmlencode:

Returns the XML encoded input.
Parameters:

- Input: the input which will be encoded into XML.

Returns: An object which is the XML encoded input.

referenced by:

- functionExpression

xmlelement:

referenced by:

- functionExpression

xmltransform:

Applies an XSL style sheet to the XML instance.
Parameters:

- XML: XML type instance to be transformed with the XSL style sheet.
- Style sheet: The XSL style sheet to apply.

Returns: the XML instance with the style sheet applied to it.

referenced by:

- functionExpression

xmlformat:

Pretty-print xml text.
Parameters:

- Xml: xml to pretty-print.

Returns: the pretty-printed XML text.

referenced by:

- functionExpression

httpget:

Collects all data from the URL as binary data.
The URL must be publicly accessible. Use the NativePlatformScalarRequest view on cloud applications to directly access their web APIs.
Parameters:

- URL: the URL to collect the data from.

Returns: the collected data as an byte array.

referenced by:

- functionExpression

httpget_text:

Collects all data from the URL as text.
The URL must be publicly accessible. Use the NativePlatformScalarRequest view on cloud applications to directly access their web APIs.
Parameters:

- URL: the URL to collect the data from.
- Encoding: the encoding from the data to receive, which is by default UTF8.

Returns: the collected data as text.

referenced by:

- functionExpression

httppost:

referenced by:

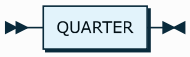
- functionExpression

quarter:

Collect the quarter from a date.
Parameters:

- Input: A dateTime.

Returns: the quarter as an integer.



referenced by:

- functionExpression

quote_ident:

referenced by:

- functionExpression

quote_literal:

referenced by:

- functionExpression

quote_nullable:

referenced by:

- functionExpression

user:

Gets the user log on code.
Returns: the user log on code.

referenced by:

- functionExpression

utcoffset:

referenced by:

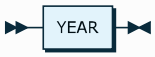
- functionExpression

year:

Collect the year from a date.
Parameters:

- Input: A dateTime.

Returns: the year as an integer.



referenced by:

- functionExpression

to_array:

referenced by:

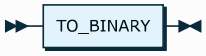
- functionExpression

to_binary:

Converts a value into a BLOB. Text is interpreted as UTF-8.
Parameters:

- Input: value to convert.

Returns: the input converted to a BLOB.



referenced by:

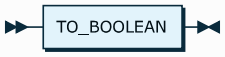
- functionExpression

to_boolean:

Converts a value into a boolean. Boolean expressions keep their value, for numbers 0 is considered false and all other values true and all dates are false. For texts, the following are considered true: 'true', 't', 'yes', 'y', 'on', '1', 'True', 'TRUE', 'T', 'YES', 'Y', 'ON'. False are 'false', 'f', 'no', 'n', 'off', '0', 'False', 'FALSE', 'F', 'NO', 'N' and 'OFF'. In all other scenarios, the default boolean conversion of Invantive holds.
Parameters:

- Input: value to convert.

Returns: the input converted to a boolean.



referenced by:

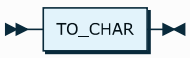
- functionExpression

to_char:

Converts a value into text.
Parameters:

- Input: value to convert.
- Format: format mask. Optional; defaults to American style. List of format masks is available in a separate section.

Returns: the input converted to text.



referenced by:

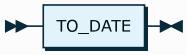
- functionExpression

to_date:

Converts a value into a datetime.
Parameters:

- Input: value to convert.
- Format: format mask. Optional; defaults to American style. List of format masks is available in a separate section.

Returns: the input converted to a datetime.



referenced by:

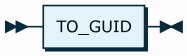
- functionExpression

to_guid:

Converts a value into a guid.
Parameters:

- Input: value to convert.

Returns: the input converted to a guid.



referenced by:

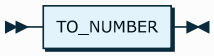
- functionExpression

to_number:

Converts a value into a number. An error is raised when the value can not be converted to a number.
Parameters:

- Input: value to convert.
- Format: format mask. Optional; defaults to local style. List of format masks is available in a separate section.

Returns: the input converted to text.



referenced by:

- functionExpression

zero_blob:

Generates a blob with 0-byte values.
Parameters:

- Length: Produce a blob with this length in terms of bytes.

Returns: A blob with 0-byte values.

referenced by:

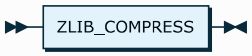
- functionExpression

zlib_compress:

Compress a BLOB using the Zlib-algorithm.
Parameters:

- input: The BLOB to compress.

Returns: a Zlib-compressed version of the input.



referenced by:

- functionExpression

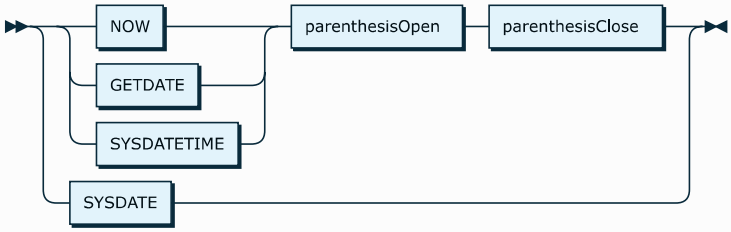
zlib_decompress:

referenced by:

- functionExpression

now:

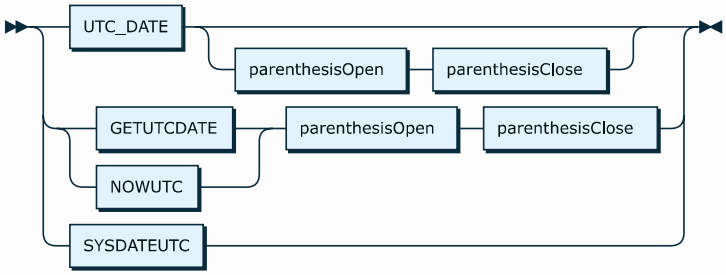
The time of the system clock in local time at the device where Invantive UniversalSQL runs.
Returns: current date/time.



referenced by:

- functionExpression

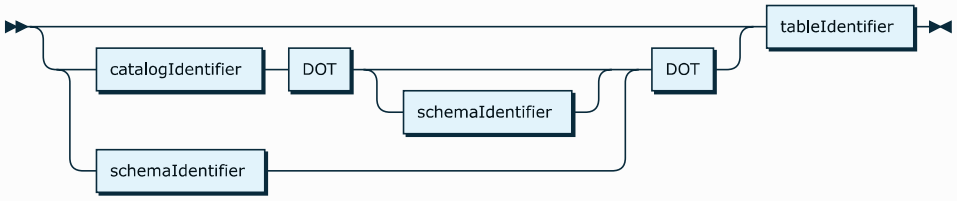
utc:



referenced by:

- functionExpression

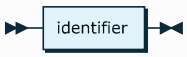
fullTableIdentifier:



referenced by:

- tableOrFunctionSpec
- tableOrFunctionSpecWithRotator
- tableSpec
- tableSpecWithRotator

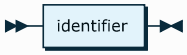
catalogIdentifier:



referenced by:

- fullTableIdentifier

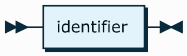
schemaIdentifier:



referenced by:

- fullTableIdentifier

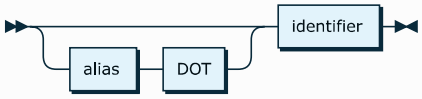
tableIdentifier:



referenced by:

- createOrReplaceViewStatement
- dropViewStatement
- fullTableIdentifier

fieldIdentifier:



referenced by:

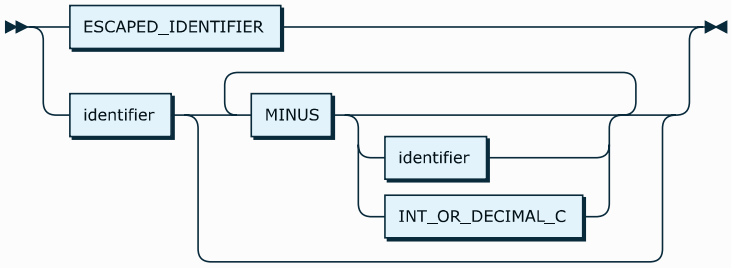
- expression

attributeIdentifier:

referenced by:

- setIdentifier

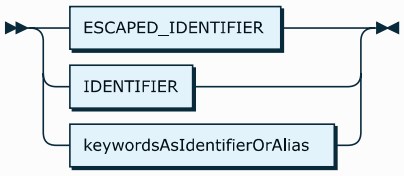
identifierWithMinus:



referenced by:

- attributeIdentifier

identifier:

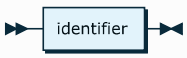


referenced by:

- catalogIdentifier
- column

- columnNoAlias
- createTableArgument
- csvTableColumnSpec
- dataContainerAlias
- excelTableColumnSpec
- fieldIdentifier
- htmlTableColumnSpec
- identifierWithMinus
- joinSet
- jsonTableColumnSpec
- namedExpression
- noJoinSet
- pSqlFunctionOrProcedureStatementNS
- pSqlPackageProcedureStatementNS
- pSqlVariableName
- parameterExpression
- partitionIdentifier
- partitionSimpleIdentifier
- roleIdentifier
- schemaIdentifier
- tableIdentifier
- triggerRecordVariableExpression
- xmlTableColumnSpec

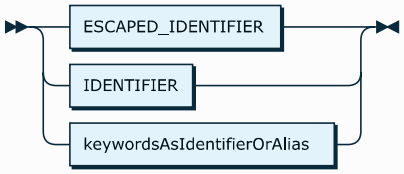
roleIdentifier:



referenced by:

- alterSessionSetRoles

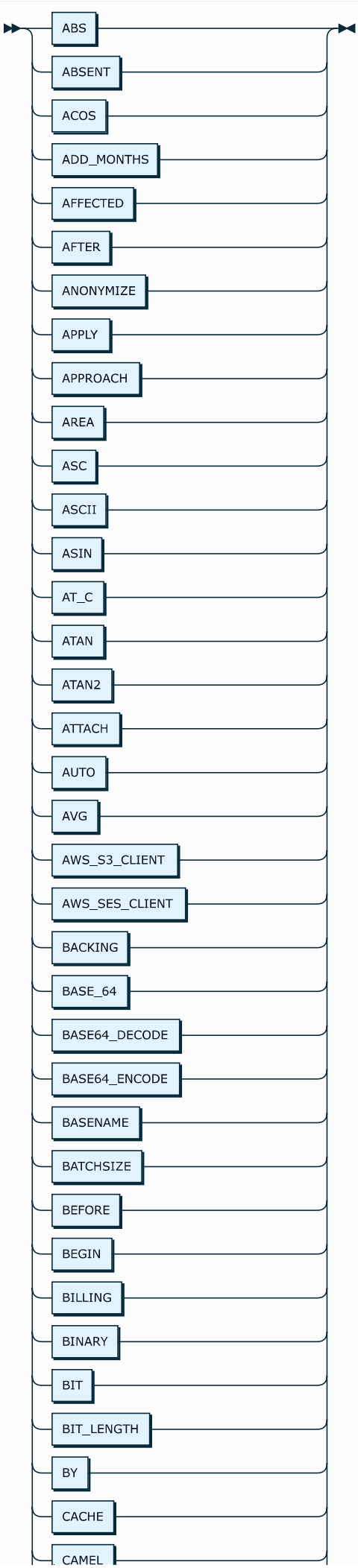
alias:

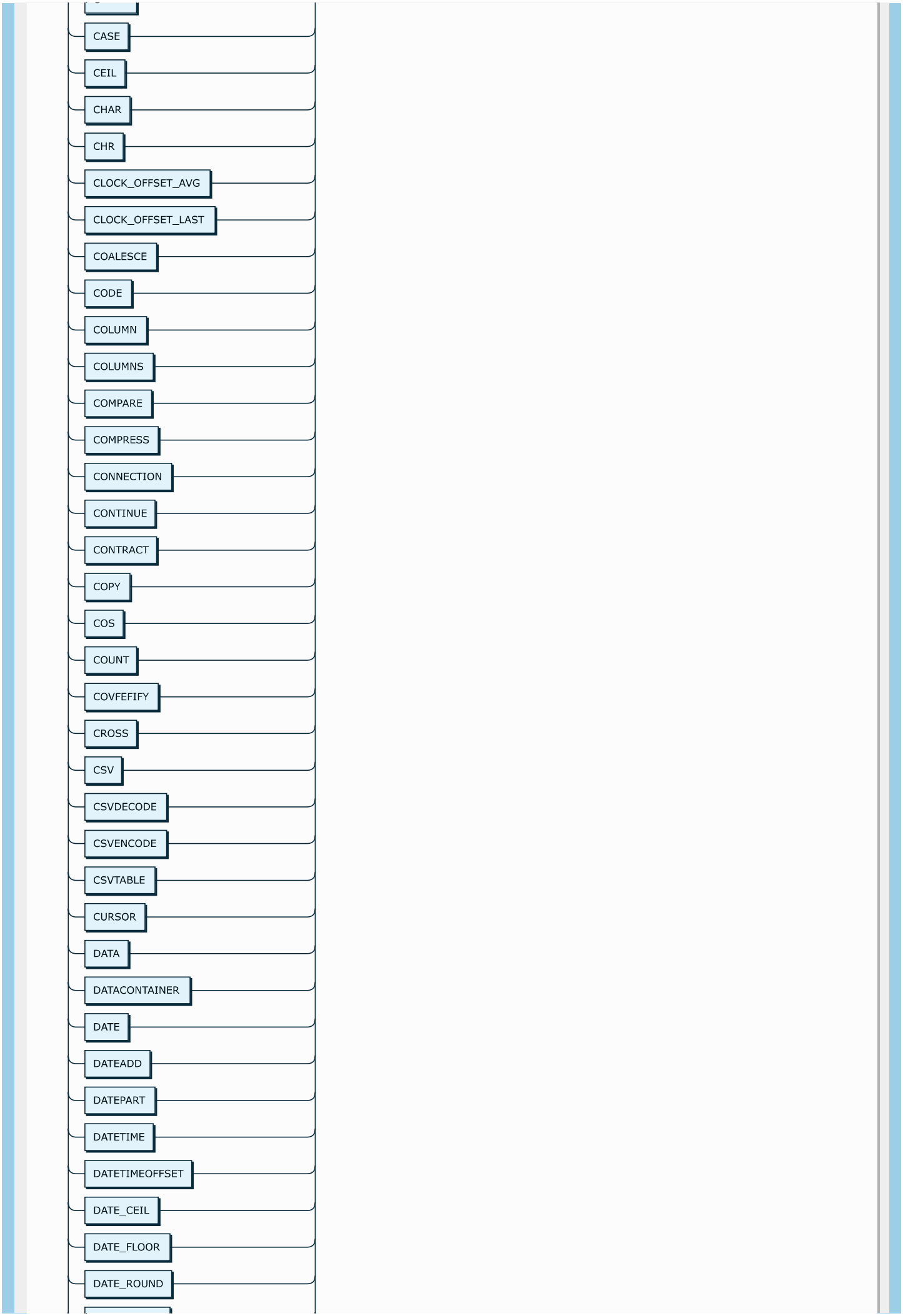


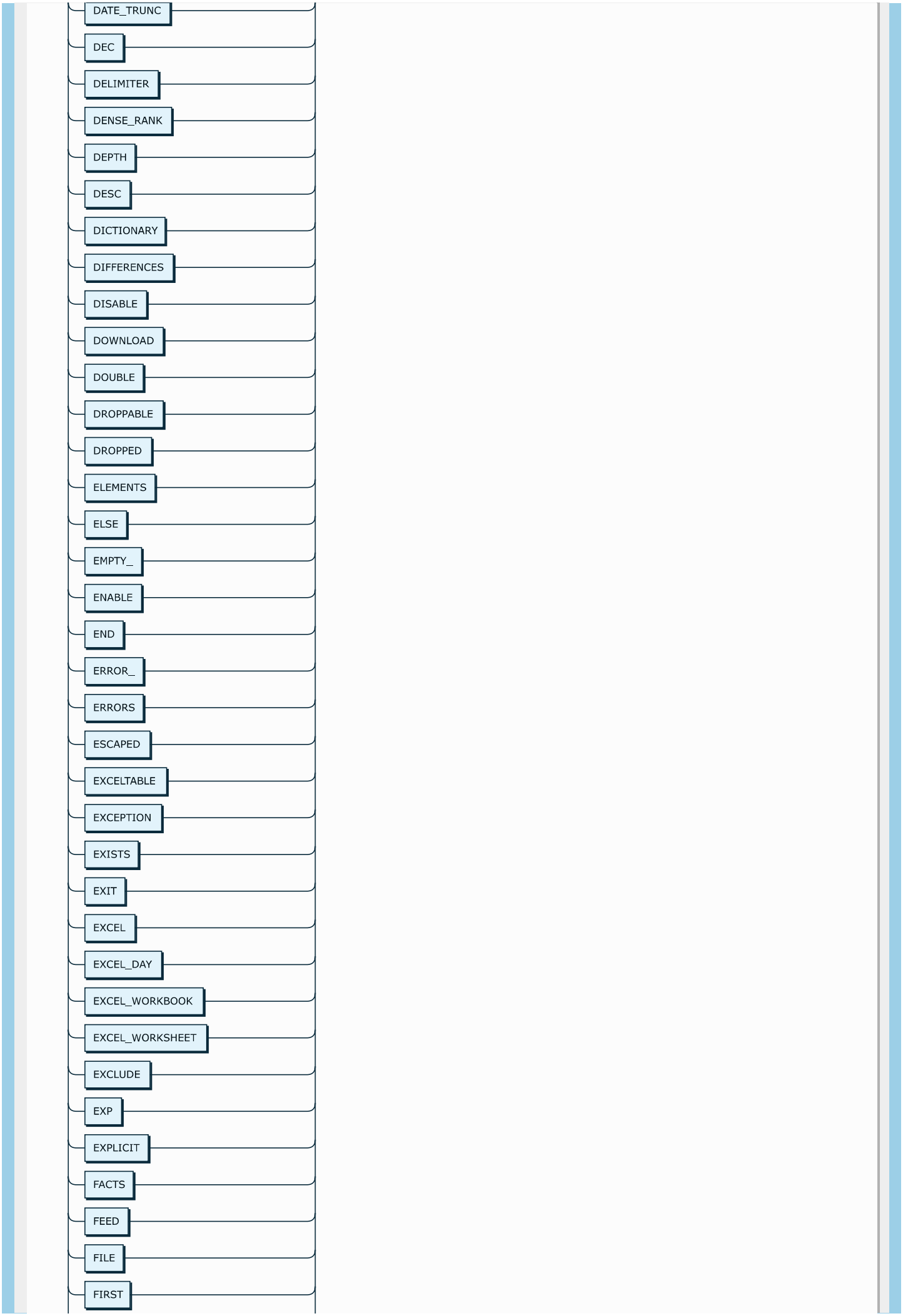
referenced by:

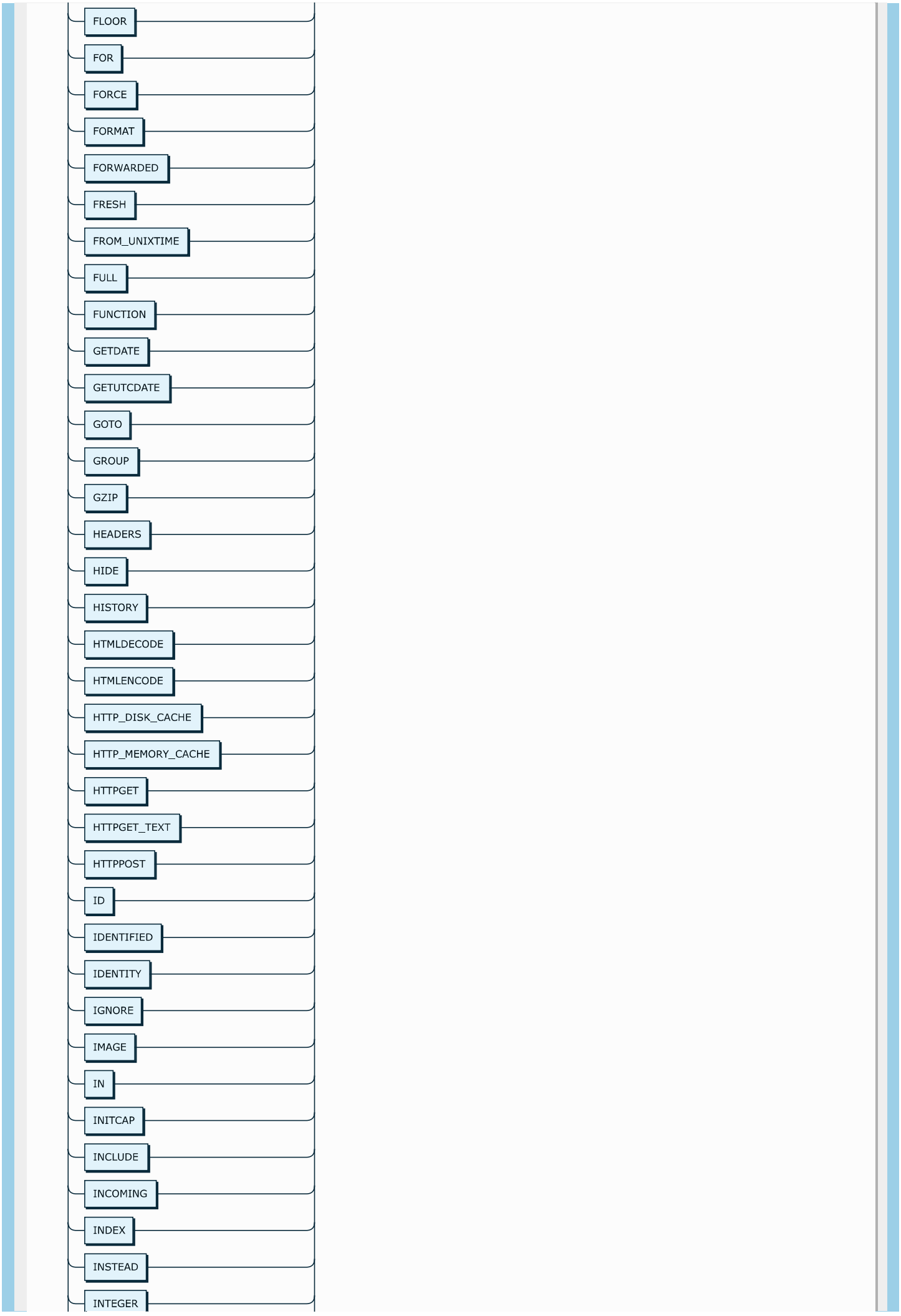
- aliased
- allColumnsSpecId
- fieldIdentifier

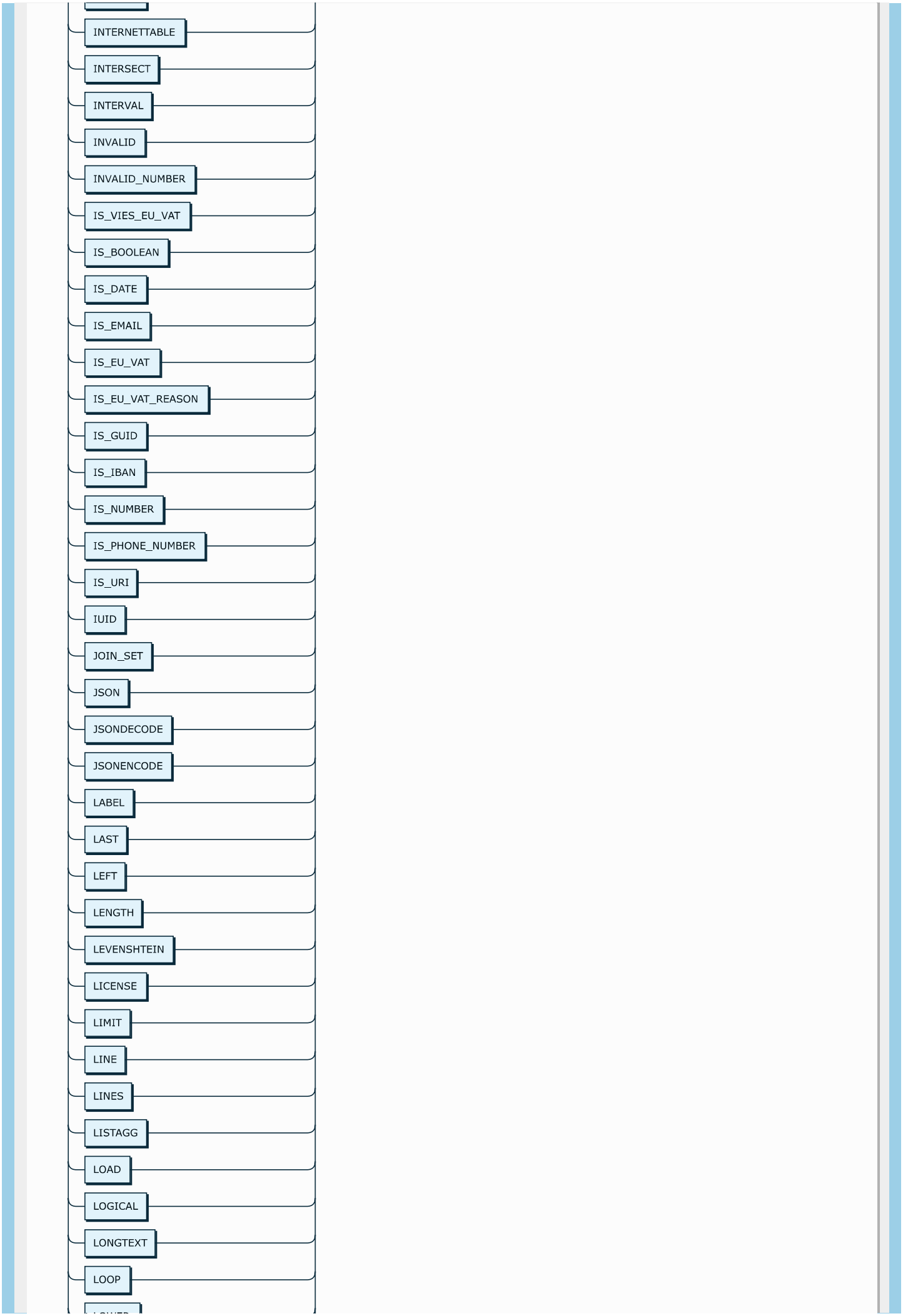
keywordsAsIdentifierOrAlias:

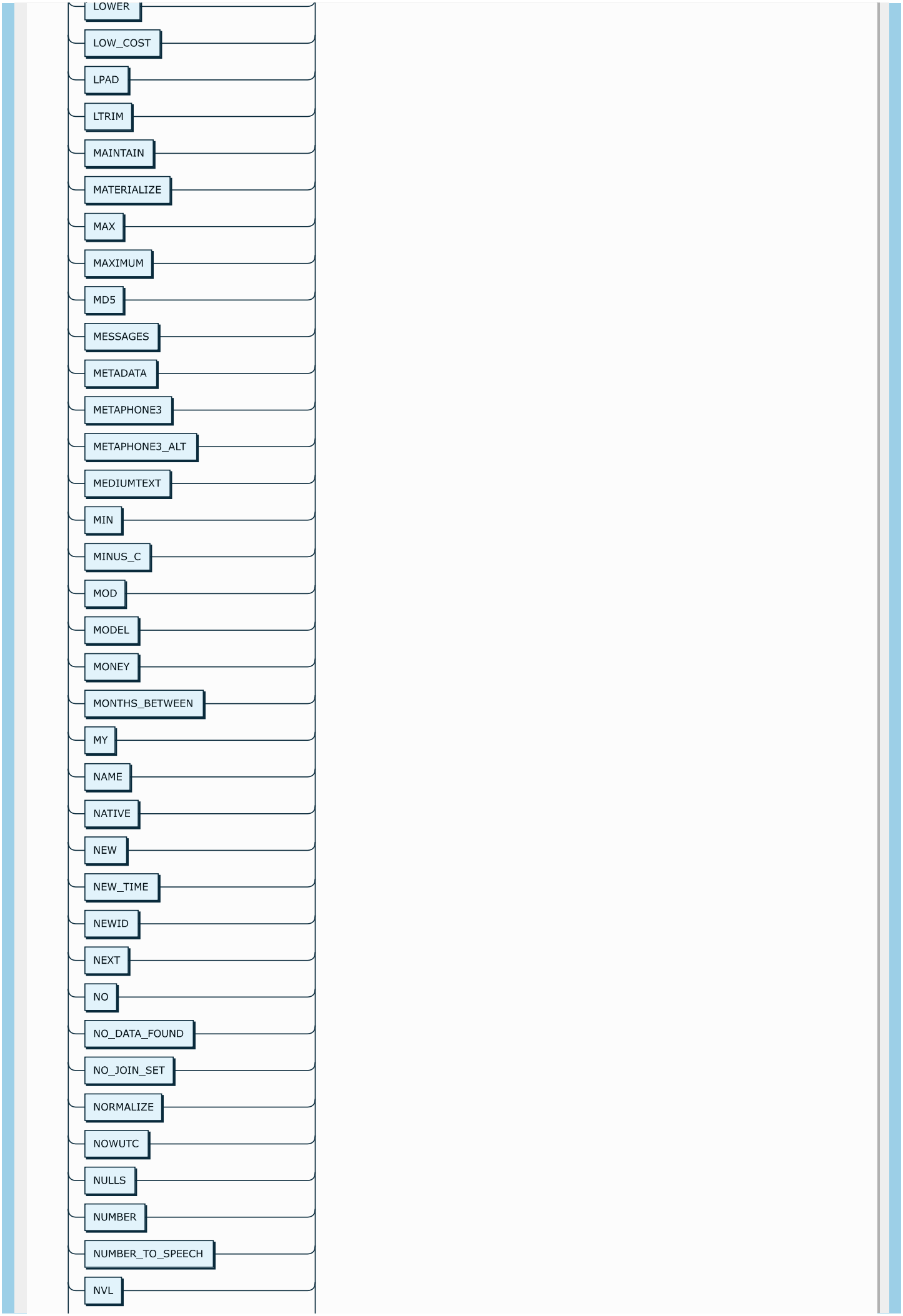

ABS
ABSENT
ACOS
ADD_MONTHS
AFFECTED
AFTER
ANONYMIZE
APPLY
APPROACH
AREA
ASC
ASCII
ASIN
AT_C
ATAN
ATAN2
ATTACH
AUTO
AVG
AWS_S3_CLIENT
AWS_SES_CLIENT
BACKING
BASE_64
BASE64_DECODE
BASE64_ENCODE
BASENAME
BATCHSIZE
BEFORE
BEGIN
BILLING
BINARY
BIT
BIT_LENGTH
BY
CACHE
CAMEL

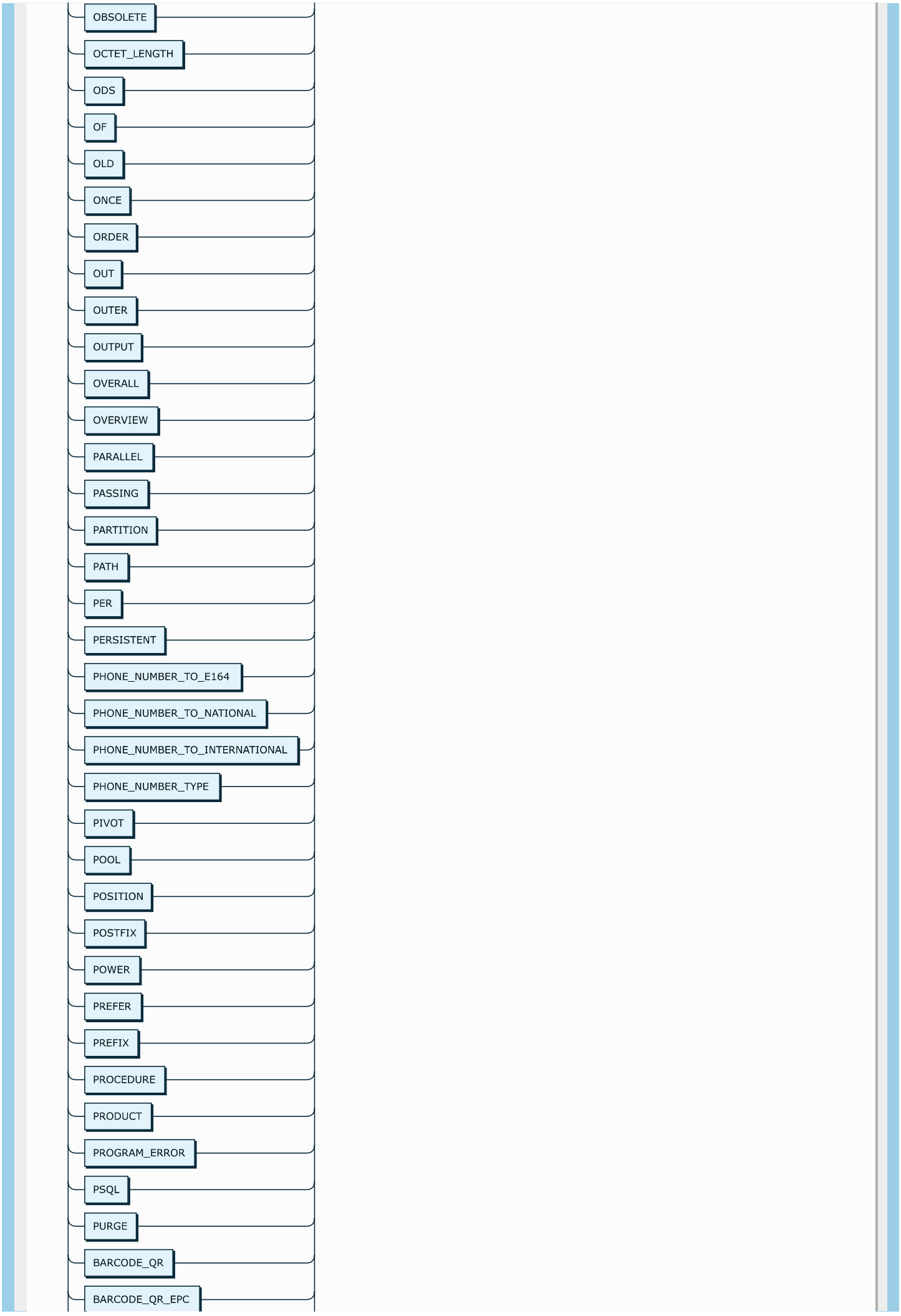

CASE
CEIL
CHAR
CHR
CLOCK_OFFSET_AVG
CLOCK_OFFSET_LAST
COALESCE
CODE
COLUMN
COLUMNS
COMPARE
COMPRESS
CONNECTION
CONTINUE
CONTRACT
COPY
COS
COUNT
COVFEFIFY
CROSS
CSV
CSVDECODE
CSVENCODE
CSVTABLE
CURSOR
DATA
DATACONTAINER
DATE
DATEADD
DATEPART
DATETIME
DATETIMEOFFSET
DATE_CEIL
DATE_FLOOR
DATE_ROUND

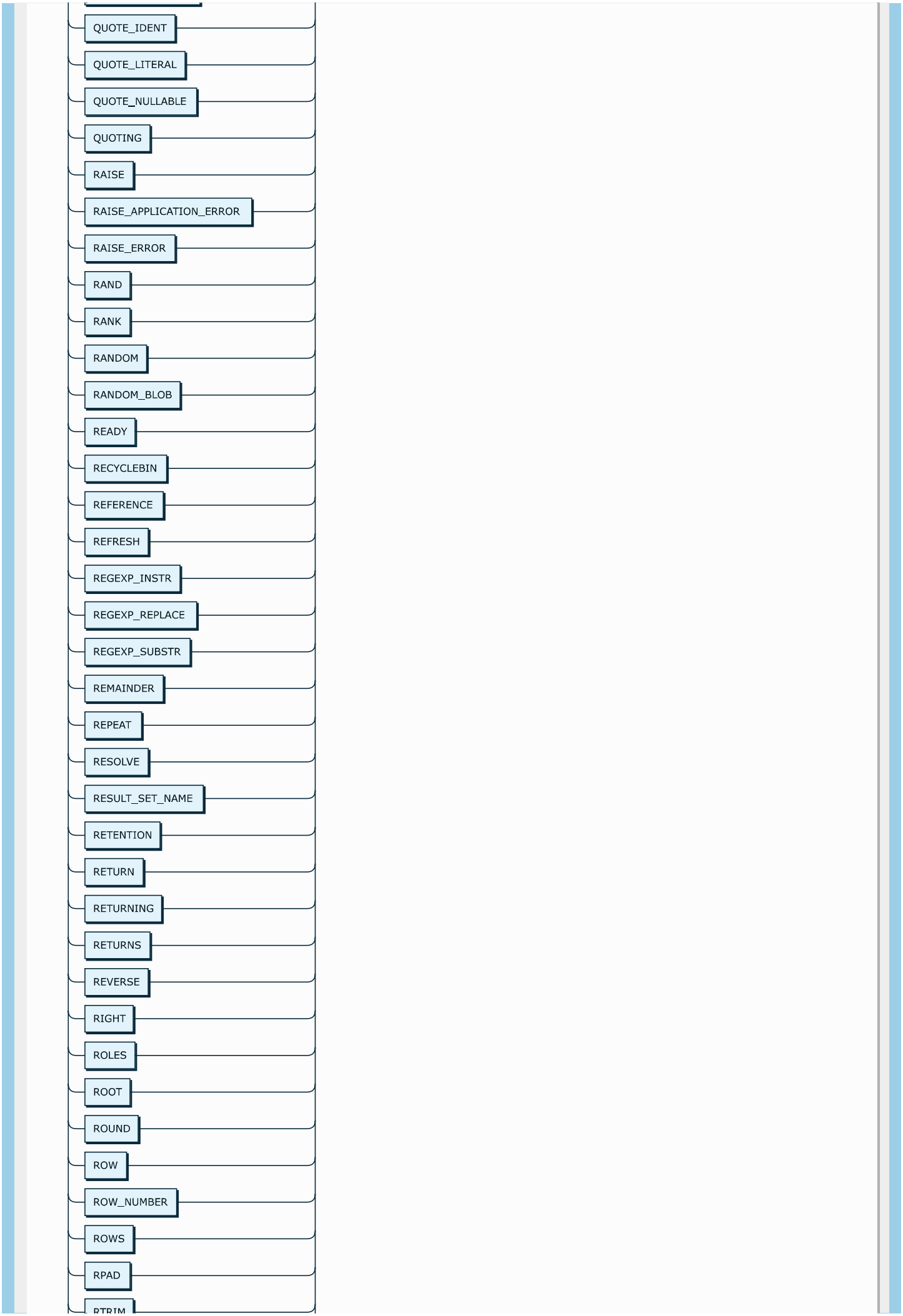

DATE_TRUNC
DEC
DELIMITER
DENSE_RANK
DEPTH
DESC
DICTIONARY
DIFFERENCES
DISABLE
DOWNLOAD
DOUBLE
DROPPABLE
DROPPED
ELEMENTS
ELSE
EMPTY_
ENABLE
END
ERROR_
ERRORS
ESCAPED
EXCELTABLE
EXCEPTION
EXISTS
EXIT
EXCEL
EXCEL_DAY
EXCEL_WORKBOOK
EXCEL_WORKSHEET
EXCLUDE
EXP
EXPLICIT
FACTS
FEED
FILE
FIRST

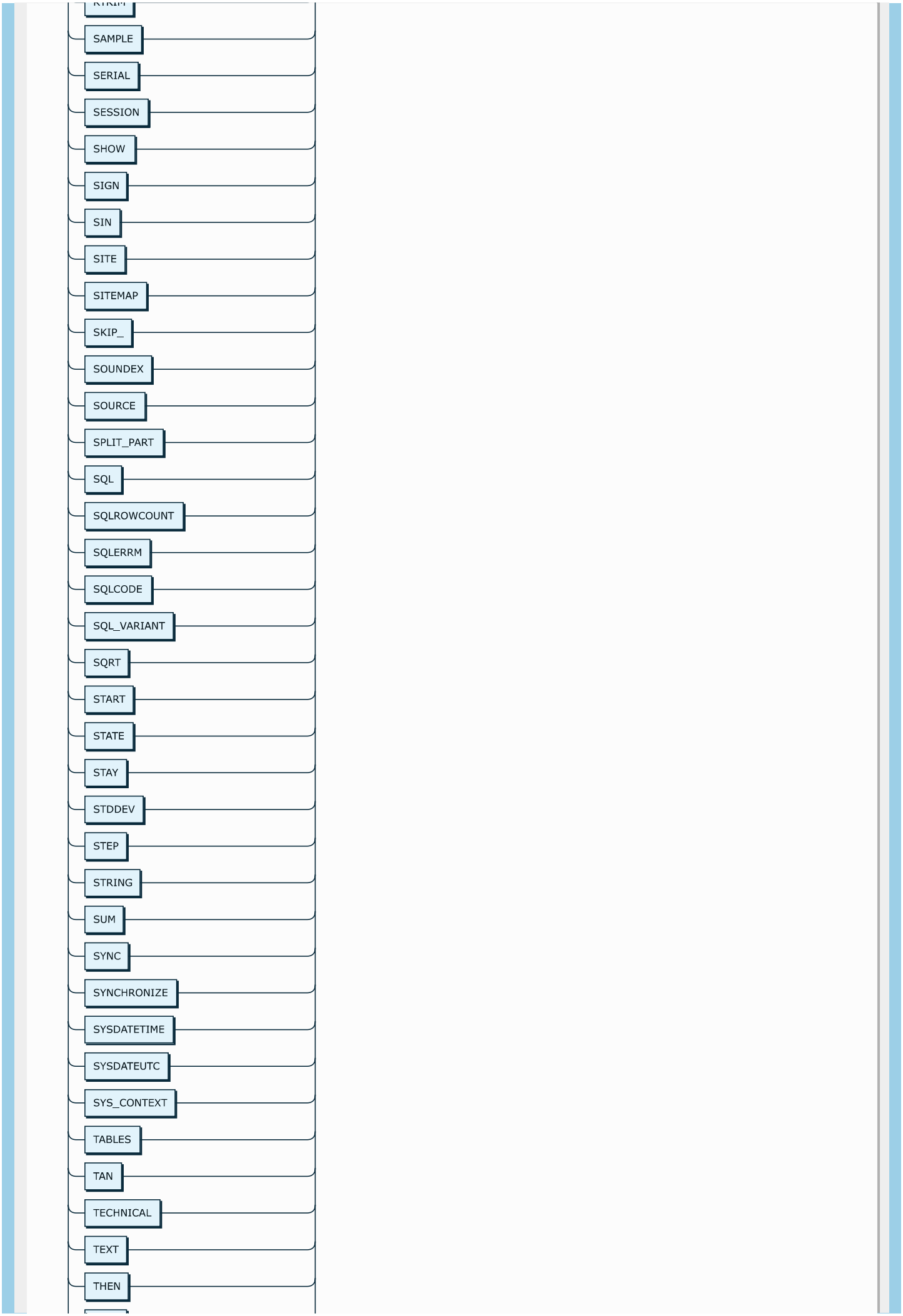

FLOOR
FOR
FORCE
FORMAT
FORWARDED
FRESH
FROM_UNIXTIME
FULL
FUNCTION
GETDATE
GETUTCDATE
GOTO
GROUP
GZIP
HEADERS
HIDE
HISTORY
HTMLDECODE
HTMLENCODE
HTTP_DISK_CACHE
HTTP_MEMORY_CACHE
HTTPGET
HTTPGET_TEXT
HTTPPOST
ID
IDENTIFIED
IDENTITY
IGNORE
IMAGE
IN
INITCAP
INCLUDE
INCOMING
INDEX
INSTEAD
INTEGER

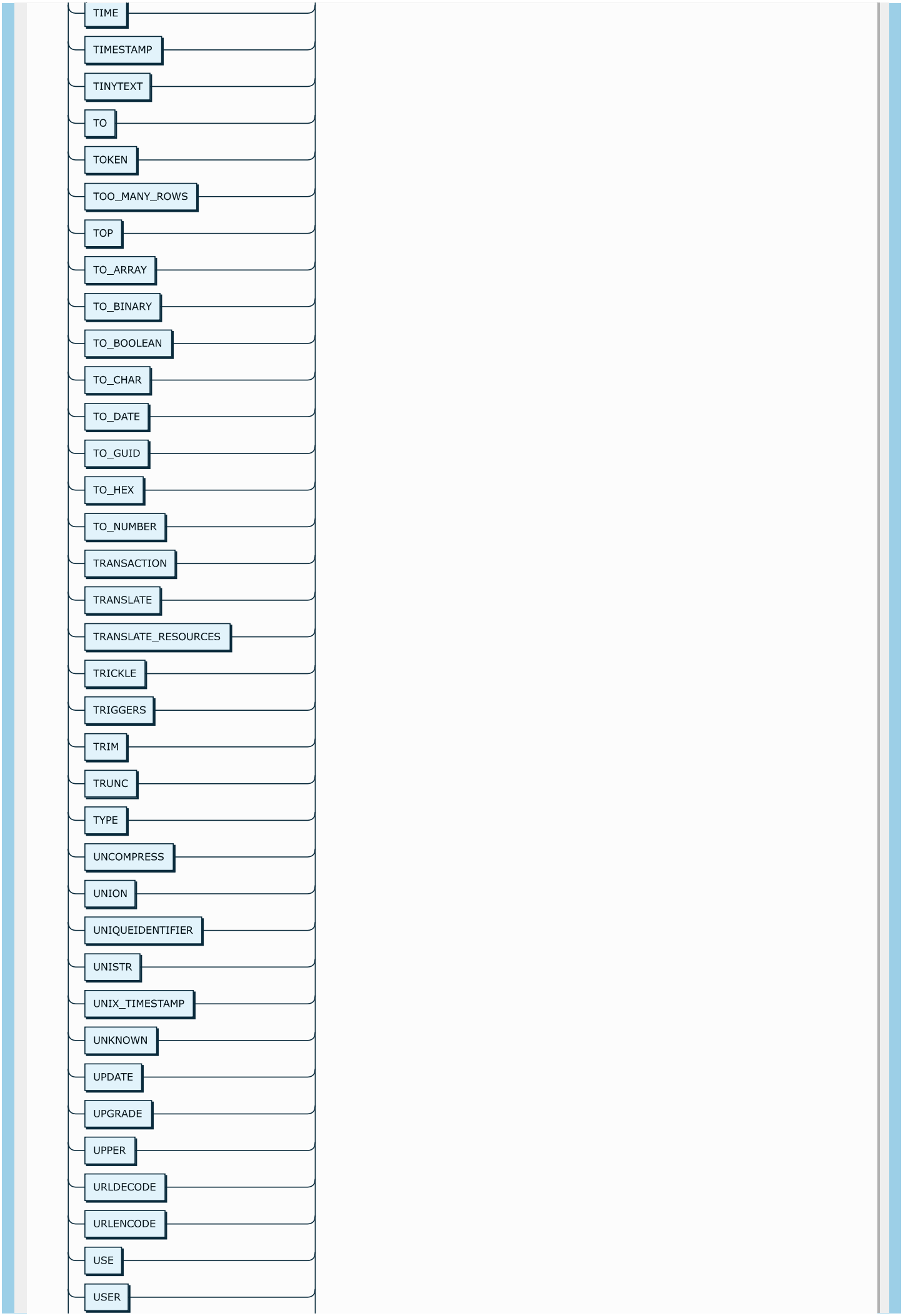

INTERNETTABLE
INTERSECT
INTERVAL
INVALID
INVALID_NUMBER
IS_VIES_EU_VAT
IS_BOOLEAN
IS_DATE
IS_EMAIL
IS_EU_VAT
IS_EU_VAT_REASON
IS_GUID
IS_IBAN
IS_NUMBER
IS_PHONE_NUMBER
IS_URI
IUID
JOIN_SET
JSON
JSONDECODE
JSONENCODE
LABEL
LAST
LEFT
LENGTH
LEVENSHTEIN
LICENSE
LIMIT
LINE
LINES
LISTAGG
LOAD
LOGICAL
LONGTEXT
LOOP

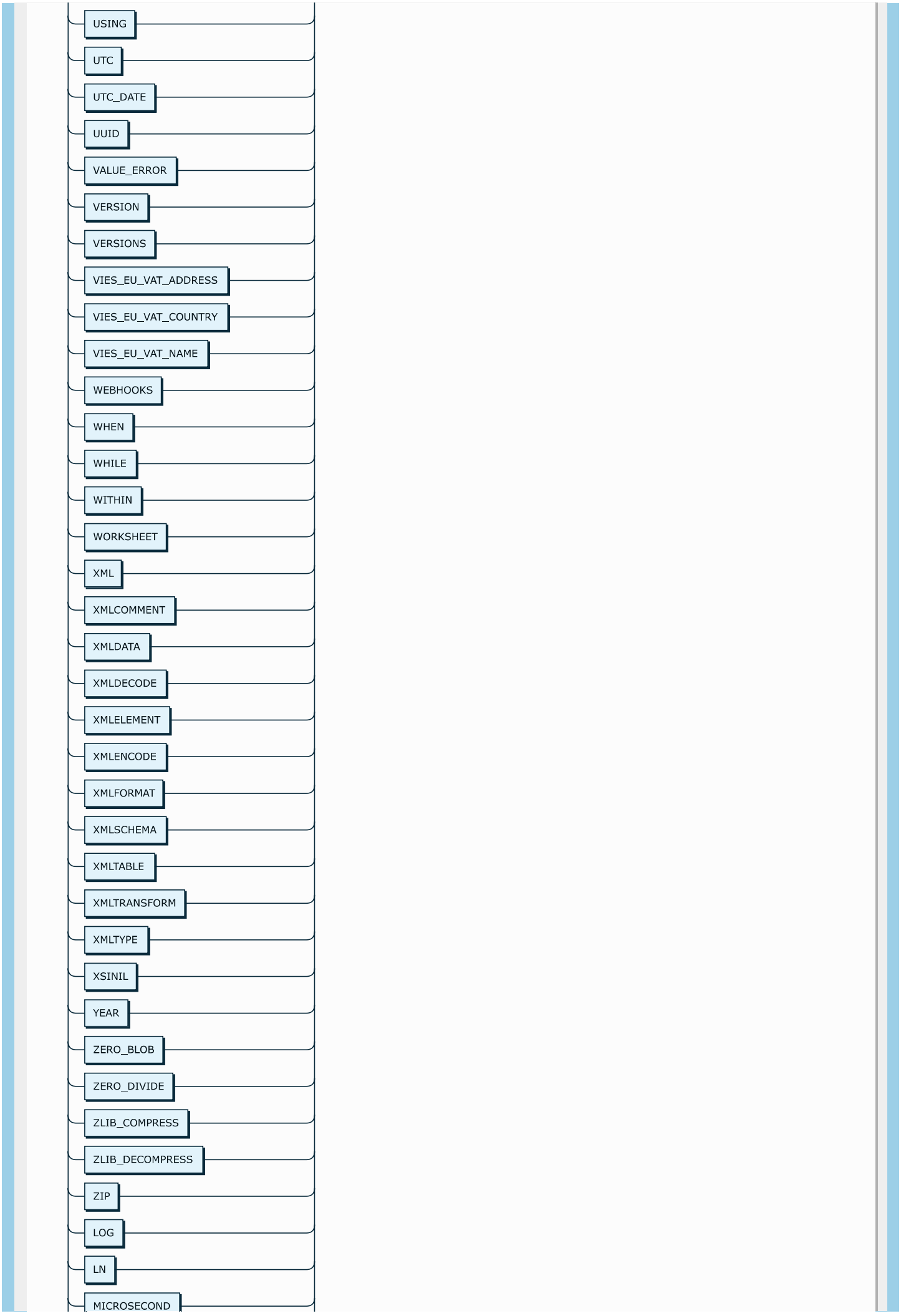

LOWER
LOW_COST
LPAD
LTRIM
MAINTAIN
MATERIALIZE
MAX
MAXIMUM
MD5
MESSAGES
METADATA
METAPHONE3
METAPHONE3_ALT
MEDIUMTEXT
MIN
MINUS_C
MOD
MODEL
MONEY
MONTHS_BETWEEN
MY
NAME
NATIVE
NEW
NEW_TIME
NEWID
NEXT
NO
NO_DATA_FOUND
NO_JOIN_SET
NORMALIZE
NOWUTC
NULLS
NUMBER
NUMBER_TO_SPEECH
NVL

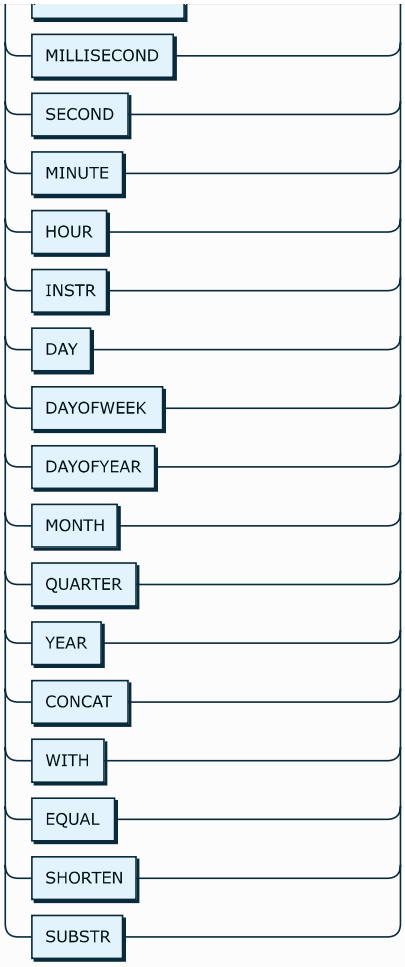

OBSOLETE
OCTET_LENGTH
ODS
OF
OLD
ONCE
ORDER
OUT
OUTER
OUTPUT
OVERALL
OVERVIEW
PARALLEL
PASSING
PARTITION
PATH
PER
PERSISTENT
PHONE_NUMBER_TO_E164
PHONE_NUMBER_TO_NATIONAL
PHONE_NUMBER_TO_INTERNATIONAL
PHONE_NUMBER_TYPE
PIVOT
POOL
POSITION
POSTFIX
POWER
PREFER
PREFIX
PROCEDURE
PRODUCT
PROGRAM_ERROR
PSQL
PURGE
BARCODE_QR
BARCODE_QR_EPC


QUOTE_IDENT
QUOTE_LITERAL
QUOTE_NULLABLE
QUOTING
RAISE
RAISE_APPLICATION_ERROR
RAISE_ERROR
RAND
RANK
RANDOM
RANDOM_BLOB
READY
RECYCLEBIN
REFERENCE
REFRESH
REGEXP_INSTR
REGEXP_REPLACE
REGEXP_SUBSTR
REMAINDER
REPEAT
RESOLVE
RESULT_SET_NAME
RETENTION
RETURN
RETURNING
RETURNS
REVERSE
RIGHT
ROLES
ROOT
ROUND
ROW
ROW_NUMBER
ROWS
RPAD
RTRIM


SAMPLE
SERIAL
SESSION
SHOW
SIGN
SIN
SITE
SITEMAP
SKIP_
SOUNDEX
SOURCE
SPLIT_PART
SQL
SQLROWCOUNT
SQLERRM
SQLCODE
SQL_VARIANT
SQRT
START
STATE
STAY
STDDEV
STEP
STRING
SUM
SYNC
SYNCHRONIZE
SYSDATETIME
SYSDATEUTC
SYS_CONTEXT
TABLES
TAN
TECHNICAL
TEXT
THEN


TIME
TIMESTAMP
TINYTEXT
TO
TOKEN
TOO_MANY_ROWS
TOP
TO_ARRAY
TO_BINARY
TO_BOOLEAN
TO_CHAR
TO_DATE
TO_GUID
TO_HEX
TO_NUMBER
TRANSACTION
TRANSLATE
TRANSLATE_RESOURCES
TRICKLE
TRIGGERS
TRIM
TRUNC
TYPE
UNCOMPRESS
UNION
UNIQUEIDENTIFIER
UNISTR
UNIX_TIMESTAMP
UNKNOWN
UPDATE
UPGRADE
UPPER
URLDECODE
URLENCODE
USE
USER

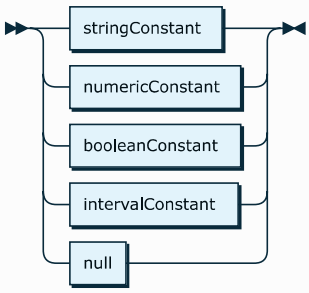

USING
UTC
UTC_DATE
UUID
VALUE_ERROR
VERSION
VERSIONS
VIES_EU_VAT_ADDRESS
VIES_EU_VAT_COUNTRY
VIES_EU_VAT_NAME
WEBHOOKS
WHEN
WHILE
WITHIN
WORKSHEET
XML
XMLCOMMENT
XMLDATA
XMLDECODE
XMLELEMENT
XMLENCODE
XMLFORMAT
XMLSCHEMA
XMLTABLE
XMLTRANSFORM
XMLTYPE
XSINIL
YEAR
ZERO_BLOB
ZERO_DIVIDE
ZLIB_COMPRESS
ZLIB_DECOMPRESS
ZIP
LOG
LN
MICROSECOND



referenced by:

- alias
- attributeIdentifier
- identifier

constantExpression:

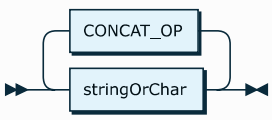


referenced by:

- expression

stringConstant:

A constant text value with varchar2 data type.

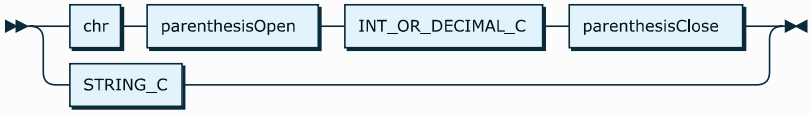


referenced by:

- allColumnsSpecColumnNamePostfix
- allColumnsSpecColumnNamePrefix
- allColumnsSpecLabelPostfix
- allColumnsSpecLabelPrefix
- alterDataDictionarySetBackingConnectionString
- alterPersistentCacheConfigureWebhooksStatement

- alterPersistentCacheDownloadStatement
- alterPersistentCacheDropStatement
- alterPersistentCachePurgeStatement
- alterPersistentCacheSetBackingConnectionString
- alterPersistentCacheSetDataContainerOptions
- alterPersistentCacheSetStatement
- alterPersistentCacheSetTableOptions
- beginTransactionStatement
- commitTransactionStatement
- constantExpression
- csvTableColumnSpec
- forCsvClause
- forJsonClause
- forXmlClause
- forXmlClauseCommonDirectives
- htmlTableColumnSpec
- htmlTableExpression
- intervalConstant
- jsonTableColumnSpec
- labeled
- pSqlExecuteNativeStatementNS
- partitionIdentifier
- resultSetName
- rollbackTransactionStatement
- xmlTableColumnSpec

stringOrChar:



referenced by:

- stringConstant

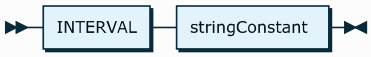
intervalConstant:

A constant interval value, reflecting the time span between two dates. The string constant consists of an integer number and unit of time, taken from the following list:

- Millisecond,
- second,
- minute,
- hour,
- day,
- week, and
- year.

The units may be postfixed with an 's' without changing meaning, like 'years'.

Valid interval values are for example: "5 seconds", "20 hours" and "1 year". There is no support for combined intervals such as "30 minutes and 30 seconds".

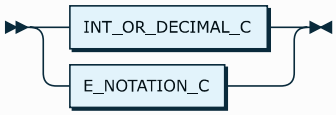


referenced by:

- alterPersistentCachePartitionRefreshStatement
- alterPersistentCacheRefreshStatement
- alterPersistentCacheTableRefreshStatement
- constantExpression
- httpDiskCache
- httpMemoryCache
- ods

numericConstant:

A constant numeric value with numeric data type.

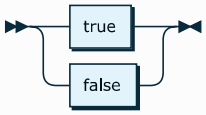


referenced by:

- alterPersistentCacheDownloadStatement
- alterPersistentCachePartitionRefreshStatement
- alterPersistentCacheRefreshStatement
- alterPersistentCacheSetStatement
- alterPersistentCacheTableRefreshStatement

- constantExpression
- createTableStatement
- csvTableColumnSpec
- excelTableColumnSpec
- excelTableOptions
- forCsvClause
- forJsonClause
- insertStatement
- internetTableOptions
- joinSet
- limitClause
- partitionIdentifier
- partitionSimpleIdentifier
- sqlDataTypeExtended
- synchronizeMaxRowsStatement
- synchronizeStatement
- topClause

booleanConstant:



referenced by:

- alterPersistentCacheSetStatement
- alterPersistentCacheSetTableOptions
- constantExpression
- httpDiskCache
- httpMemoryCache
- lowCost
- ods

null:

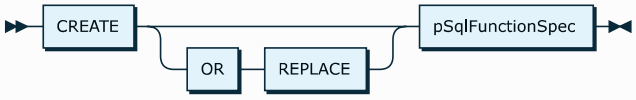
The "unknown" value null.

referenced by:

- constantExpression
- jsonTableSpec
- ndjsonTableSpec
- xmlTableSpec

pSqlCreateFunction:

The PSQL create function statement creates a PSQL block that executes the code and returns a value to the caller using the 'return' statement. An error is raised when a function, procedure or package with the same name already exists. When 'or replace' is specified, a previously existing function with the same name is dropped before creating the new function.



referenced by:

- sqlOrPSqlStatement

pSqlAlterFunction:

referenced by:

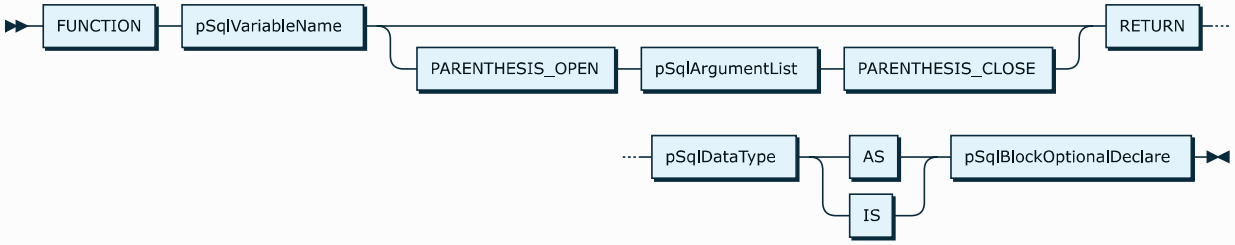
- sqlOrPSqlStatement

pSqlDropFunction:

referenced by:

- sqlOrPSqlStatement

pSqlFunctionSpec:



referenced by:

- pSqlAlterFunction
- pSqlCreateFunction

pSqlCreateProcedure:

The PSQL create function statement creates a PSQL block that executes code when called. An error is raised when a function, procedure or package with the same name already exists. When 'or replace' is specified, a previously existing procedure with the same name is dropped before creating the new procedure.

referenced by:

- sqlOrPSqlStatement

pSqlAlterProcedure:

referenced by:

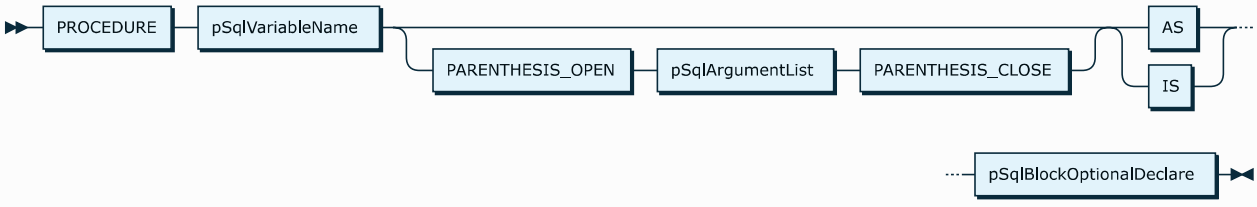
- sqlOrPSqlStatement

pSqlDropProcedure:

referenced by:

- sqlOrPSqlStatement

pSqlProcedureSpec:



referenced by:

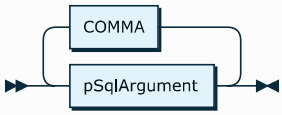
- pSqlAlterProcedure
- pSqlCreateProcedure

pSqlArgument:

referenced by:

- pSqlArgumentList

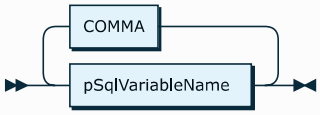
pSqlArgumentList:



referenced by:

- pSqlFunctionSpec
- pSqlProcedureSpec

pSqlVariableList:



referenced by:

- pSqlExecuteImmediateStatementNS
- pSqlExecuteNativeStatementNS
- uniqueSelectStatement

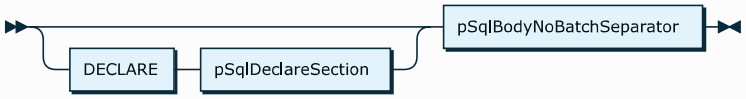
pSqlBlock:

A PSQL block is a structure to define procedural logic. It can contain both procedural logic as well as SQL statements like "select".

referenced by:

- pSqlBlockOrStatement
- pSqlStatement

pSqlBlockNoBatchSeparator:



referenced by:

- sqlOrPSqlStatement
- synchronizeTrigger
- synchronizeUsingPSqlBlock

pSqlBlockOptionalDeclare:

referenced by:

- pSqlFunctionSpec
- pSqlProcedureSpec

pSqlDeclareSection:

A PSQL declare section defines one or more local variables, which are available in the block and nested blocks.

referenced by:

- pSqlBlock
- pSqlBlockNoBatchSeparator
- pSqlBlockOptionalDeclare

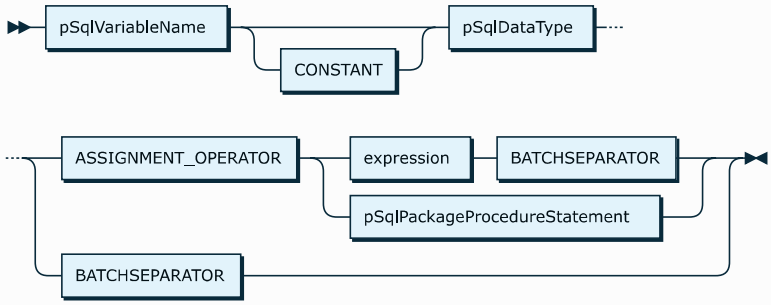
pSqlDeclaration:

referenced by:

- pSqlDeclareSection

pSqlItemDeclaration:

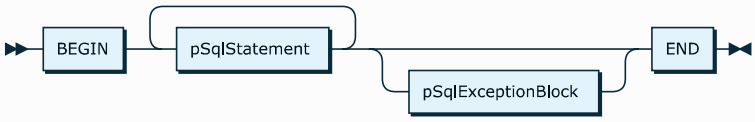
An item declaration defines one named variable, based upon data type. The initial value can be added as a constant.



referenced by:

- pSqlDeclaration

pSqlBodyNoBatchSeparator:



referenced by:

- pSqlBlockNoBatchSeparator
- pSqlBody

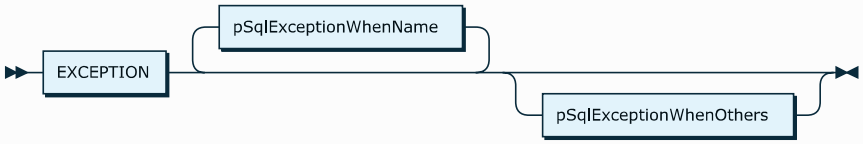
pSqlBody:

A PSQL body contains the procedural logic as well as SQL statements. Variables must have been declared beforehand.

referenced by:

- pSqlBlock
- pSqlBlockOptionalDeclare

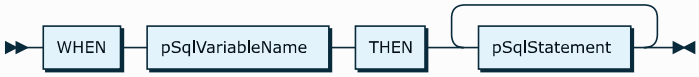
pSqlExceptionBlock:



referenced by:

- pSqlBodyNoBatchSeparator

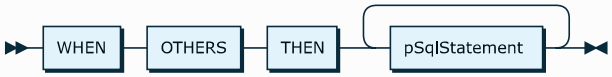
pSqlExceptionWhenName:



referenced by:

- pSqlExceptionBlock

pSqlExceptionWhenOthers:

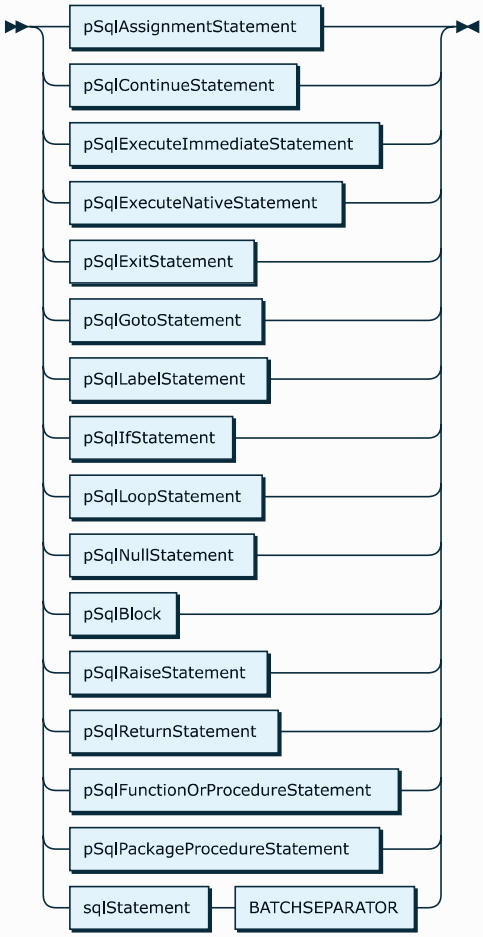


referenced by:

- pSqlExceptionBlock

pSqlStatement:

A number of basic PSQL statements are available.



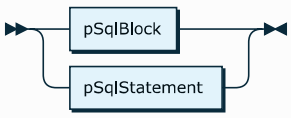
referenced by:

- pSqlBlockOrStatement
- pSqlBodyNoBatchSeparator
- pSqlExceptionWhenName

- pSqlExceptionWhenOthers
- sqlOrPSqlStatement

pSqlBlockOrStatement:

A PSQL block or statement defines a procedural step or a SQL statement to be executed.



referenced by:

- pSqlBlockOrStatements

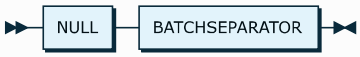
pSqlBlockOrStatements:

referenced by:

- pSqlElsIfExpression
- pSqlForNumberLoopStatement
- pSqlForRecordLoopStatement
- pSqlIfStatement
- pSqlRepeatUntilLoopStatement
- pSqlWhileLoopStatement

pSqlNullStatement:

The null-statement is a NOP-statement (No Operator). The use of the null-statement is necessary when a PSQL statement is needed, but no activity needs to be performed such as with an if statement. The null-statement also makes explicit that a developer has considered the actions needed and found that no action applies to a specific scenario. This leads to improved code documentation.



referenced by:

- pSqlStatement

pSqlReturnStatement:

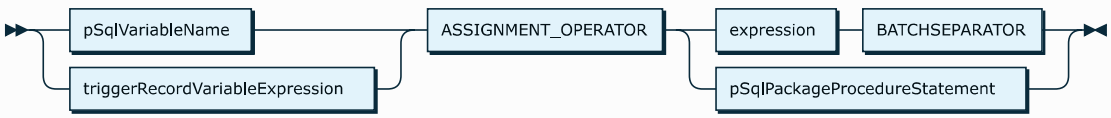
Return a value from a function to the calling code.

referenced by:

- pSqlStatement

pSqlAssignmentStatement:

The assignment statement assign a new value to a variable. To assign the results of a SQL query to a value, use a select ... into ... statement.

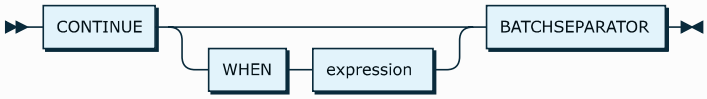


referenced by:

- pSqlStatement

pSqlContinueStatement:

This PSQL continue statement continues execution of the innermost loop at the start of the loop. When a boolean expression is specified, the innermost loop is only started at the start when the boolean expression evaluates to true.

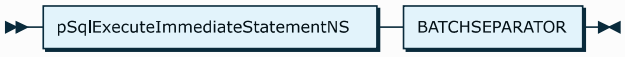


referenced by:

- pSqlStatement

pSqlExecuteImmediateStatement:

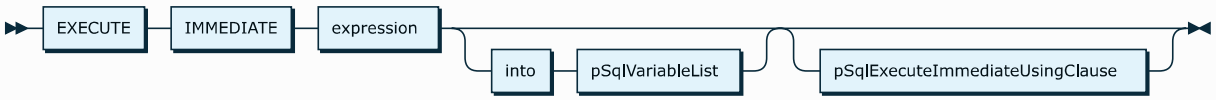
The execute immediate PSQL statement enables the use of SQL statements that are compiled at runtime. For instance dynamic DDL statements can not always be executed on compiled time and the execute immediate enables these.



referenced by:

- pSqlStatement

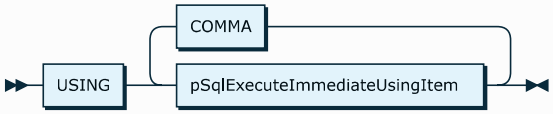
pSqlExecuteImmediateStatementNS:



referenced by:

- pSqlExecuteImmediateStatement
- pSqlForRecordLoopStatement

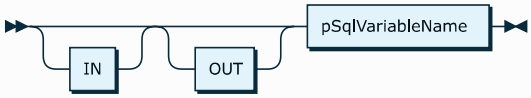
pSqlExecuteImmediateUsingClause:



referenced by:

- pSqlExecuteImmediateStatementNS

pSqlExecuteImmediateUsingItem:



referenced by:

- pSqlExecuteImmediateUsingClause

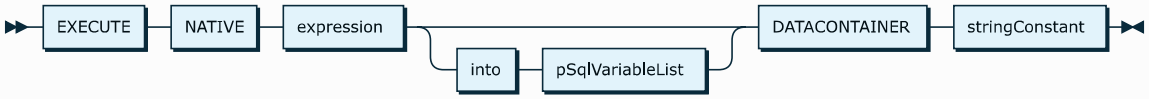
pSqlExecuteNativeStatement:

Execute a statement directly on a specific data container. The statement must match the native SQL or API code for the data container specified. Output of the statement is retrieved into PSQL variables.

referenced by:

- pSqlStatement

pSqlExecuteNativeStatementNS:

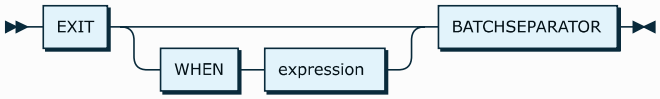


referenced by:

- pSqlExecuteNativeStatement
- pSqlForRecordLoopStatement

pSqlExitStatement:

This PSQL exit statement exits execution of the innermost loop. When a boolean expression is specified, the innermost loop is only exited when the boolean expression evaluates to true.



referenced by:

- pSqlStatement

pSqlGotoStatement:

Continue execution at the location of the specified label.

referenced by:

- pSqlStatement

pSqlLabelStatement:

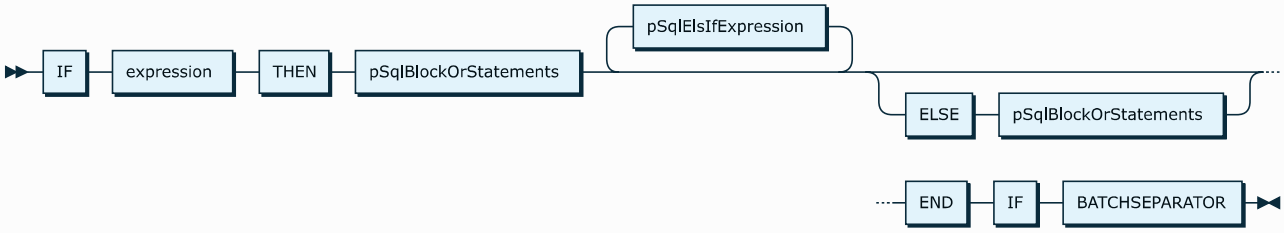
Specificy location of a label.

referenced by:

- pSqlStatement

pSqlIfStatement:

The if-statement performs conditional logic. When the boolean expression after if holds, the PSQL block after the 'then' will be executed. Other branches can be specified using an elsif. Otherwise, and only when specified, the logic after the else is executed.



referenced by:

- pSqlStatement

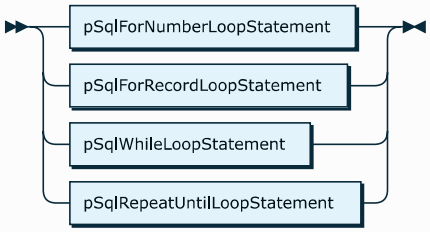
pSqlElsIfExpression:

referenced by:

- pSqlIfStatement

pSqlLoopStatement:

A variety of PSQL statements for loops are available.

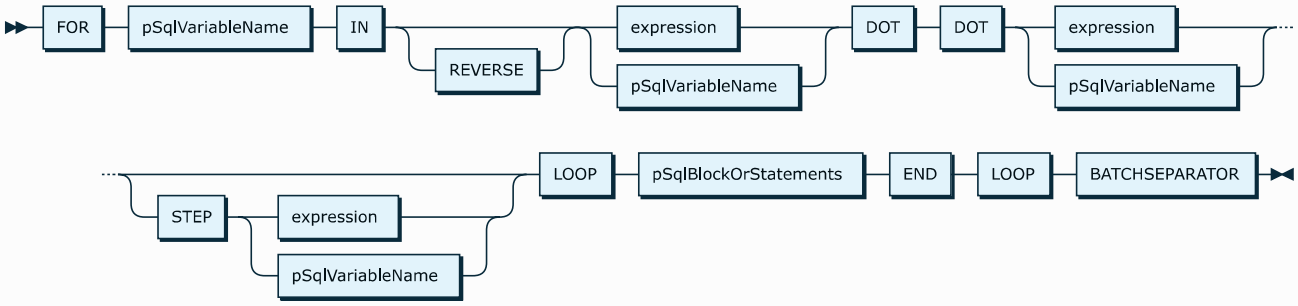


referenced by:

- pSqlStatement

pSqlForNumberLoopStatement:

This PSQL integer loop statement iterates over a range of integer values, executing PSQL statements for each iterated value. The iterations goes from the first value to the last value in increments of specified step size. Default step size is 1. The iterations go backward in decrements of 1 when 'reverse' is specified.

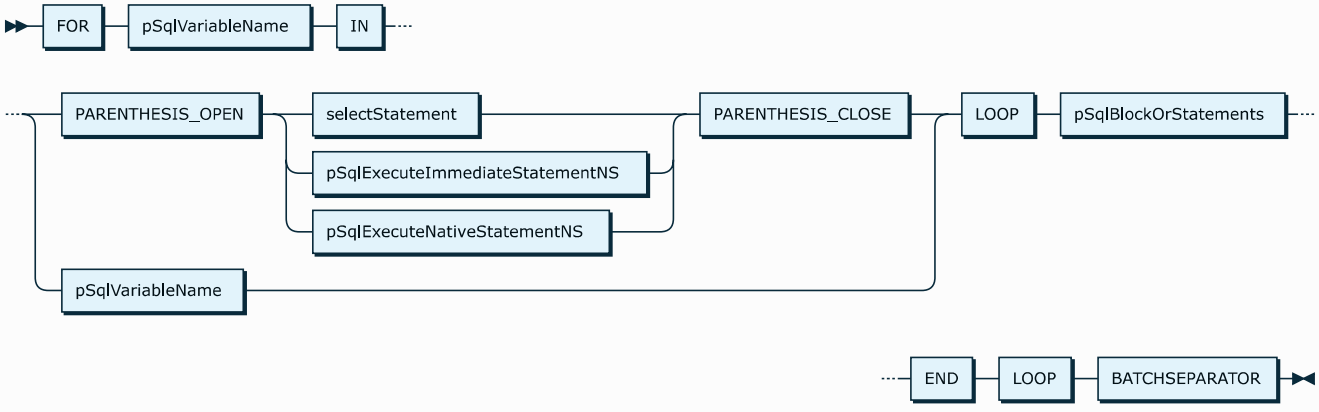


referenced by:

- pSqlLoopStatement

pSqlForRecordLoopStatement:

This PSQL result set loop statement iterates over a result set returned by an Invantive UniversalSQL query. The PSQL statements are executed for each record. The record's specific values can be retrieved using the variable.



referenced by:

- pSqlLoopStatement

pSqlRepeatUntilLoopStatement:

This PSQL loop statement (or 'repeat until') executes PSQL statements as long as the specified loop is not terminated using an 'exit' statement.

referenced by:

- pSqlLoopStatement

pSqlWhileLoopStatement:

This PSQL while loop statement executes PSQL statements as long as the specified boolean condition evaluates to true at loop end.

referenced by:

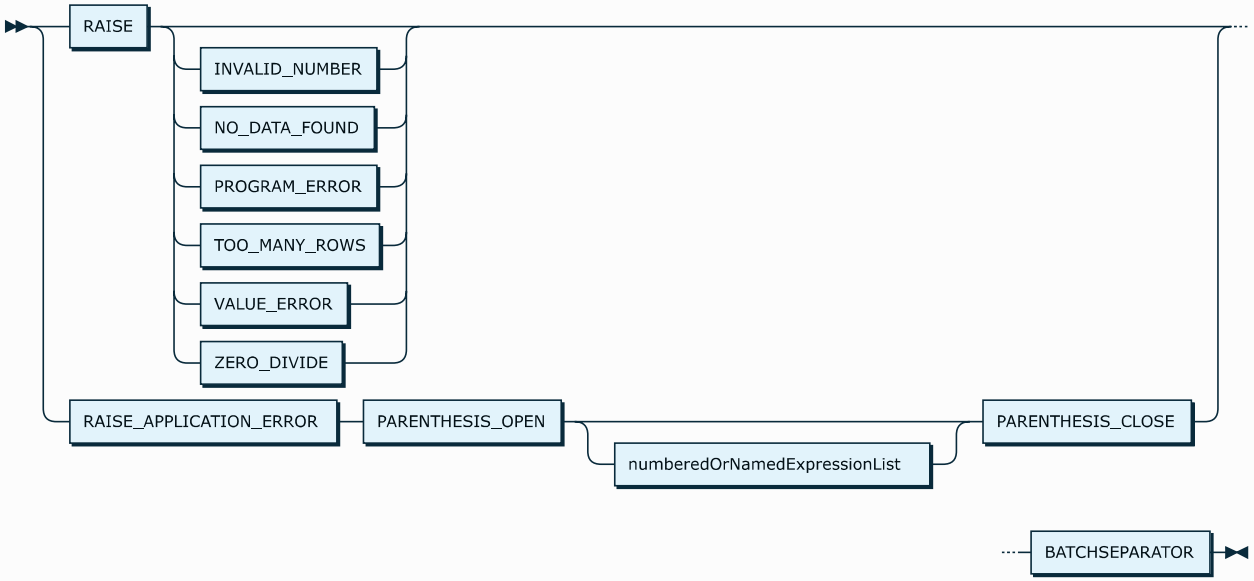
- pSqlLoopStatement

pSqlRaiseStatement:

The raise statement raises one from a list of pre-defined exceptions:

- INVALID_NUMBER: typically used when a string is converted to a number, but does not contain a valid number.
- NO_DATA_FOUND: typically used when a 'select into' statement returns zero rows.
- PROGRAM_ERROR: typically used to signal an (internal) system error.
- TOO_MANY_ROWS: typically used when a 'select into' statement returns two or more rows.
- VALUE_ERROR: typically used when during evaluation of an expression an error occurs.
- ZERO_DIVIDE: typically used when the right-hand side of a divide is zero.

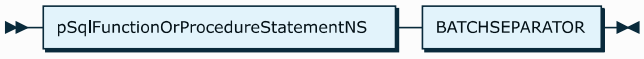
The raise_application_error statement raises an exception, which consists of a non-unique text or numeric identification plus a message text. Optionally, you can add a kind request and the natural key as parameters.



referenced by:

- pSqlStatement

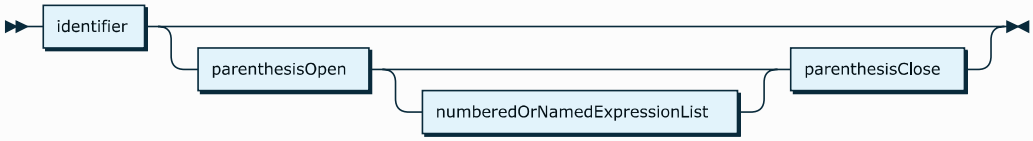
pSqlFunctionOrProcedureStatement:



referenced by:

- pSqlStatement

pSqlFunctionOrProcedureStatementNS:



referenced by:

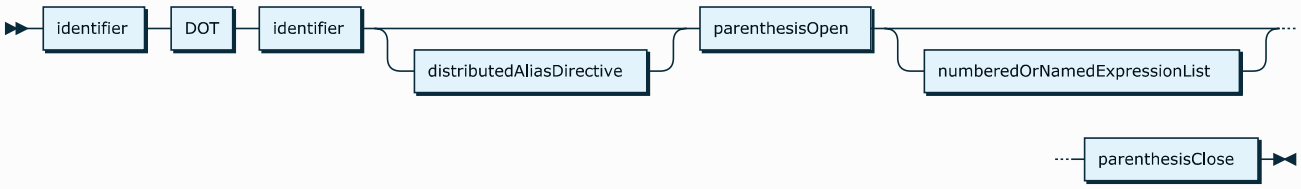
- pSqlFunctionOrProcedureStatement
- pipelinedTableFunctionSpec

pSqlPackageProcedureStatement:

referenced by:

- pSqlAssignmentStatement
- pSqlItemDeclaration
- pSqlStatement

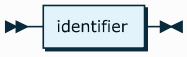
pSqlPackageProcedureStatementNS:



referenced by:

- expression
- pSqlPackageProcedureStatement
- pipelinedTableFunctionSpec

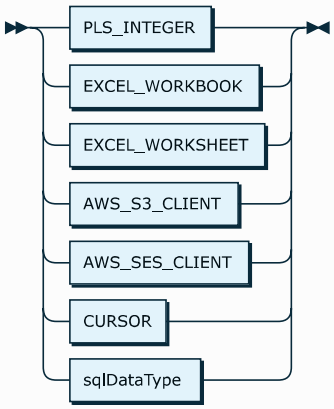
pSqlVariableName:



referenced by:

- pSqlArgument
- pSqlAssignmentStatement
- pSqlDropFunction
- pSqlDropProcedure
- pSqlExceptionWhenName
- pSqlExecuteImmediateUsingItem
- pSqlForNumberLoopStatement
- pSqlForRecordLoopStatement
- pSqlFunctionSpec
- pSqlGotoStatement
- pSqlItemDeclaration
- pSqlLabelStatement
- pSqlProcedureSpec
- pSqlVariableList

pSqlDataType:



referenced by:

- pSqlArgument
- pSqlFunctionSpec
- pSqlItemDeclaration

EOF:

referenced by:

- sqlBatch

Contact

- Invantive ® BV
- Biesteweg 11
- 3849 RDHierden
- Nederland

- Sales: +31 88 00 26 500
- Support: forums.invantive.com
- Office hours: 09:00 - 17:00 CET
- E-mail: sales@invantive.com
- Web: invantive.com

Report security incident
Telephone: +31 88 00 26 598
Email: security@invantive.com
More information

- Information for use in Outlook
- Chamber of Commerce : 130 31 406
- Managing Director: Guido Leenders
- Company domiciled in Roermond.
- VAT: NL812602377B01
- Founded: 1992
- 2012 NAICS: 511210

- Bank: IBAN NL25 BUNQ 2098 2586 07, BIC BUNQNL2A