iDB Command Syntax
QQL Command Documentation

Overview

Quick Query language (QQL) supports the following commands.

Accessing an iDB Database usually involves specifying a Database Name, an Accessor and a Model.

Glossary

Database Name - the name of the iDB database file on the iDB server. Should be a single word with no extension.

Accessor - A single word identifier that names the privileges and default behavior for a database session.

Model - The Model is an identifier that acts as a shortcut to an initial stored Query. It generally results in an HTML page that represents a root view of the user's data.

Identifier - Identifiers represent three different types of items within iDB. Closure Numbers (CN), Association Numbers (AN), and Literal Values (LV). Field Names and Field Values are both forms of Literal Values (LV).

Closure Number (CN) - Closure Numbers are monotonically increasing integers that act as synchronization points within the database. They may be specified for time travel and Query Predicates.

Association Number (AN) - Unique numeric identifier for each Association. Used to group Field Name / Field Value pairs into logical Associations.

Literal Value (LV) - Text or binary string that may be used as a Field Name or Field Value. Every Literal Value has a corresponding Literal Value Identifier (LVI). Literal Values are maintained in a Literal Value Table (LVT). Each unique Literal Value is guaranteed to be placed into the Literal Value Table exactly once.

Literal Value Identifier (LVI) -

Literal Value Table (LVT) -

Field Name (FN) -

Field Name Identifier (FNI) -

Field Value (FV) -

Field Value Identifier (FVI) -

Sort Order Number (SON) - Every Literal Value has a corresponding Sort Order Number. Sort Order Numbers are 64-bit binary integers that provide the basis for comparing the ordering of any two Literal Values. Internally, Sort Order Numbers are directly related to the traversal of the B-Tree index structure associated with the Literal Value Table.

Natural Sort Order - All Literal Values are treated as ASCII character strings. Any two Literal Values can be compared using the **`StandardCompare()`** function. StandardCompare breaks the strings down into a sequence of tokens and compares the logical values of the tokens. Each token has a logical type: null,

whitespace, punctuation, negative number, negative zero, zero, positive number or alphabetic. Comparing tokens allows the StandardCompare result to indicate whether two values are less than, greater than, equivalent to or identical to each other.

Selector - Ordered list of FN *relop* FV query segments used to select Associations to be operated on by subsequent Query Segments.

Hidden Selector - A Selector used as part of the Accessor to restrict access to a subset of the database.

Creator - A sequence of FN=FV field assignments which is invoked each time a new Association is created. The Creator is part of the Accessor and is used to ensure that records are created within a controlled subset of the database. The Creator may also include specialized FVs that create timestamps

Query - Sequence or Query Segments contained within parentheses and returning a result in one of various formats.

Query Segment -

Time Travel -

Result -

Base64 Encoding - iDB uses a specialized character set and ordering for representing binary values. The 64 characters in order are **`0-9, A-Z, a-z, $, #`**. One feature of this sequence is that the zero (0) represents binary zero, so leading-zero suppression for binary integers works as expected. Also, everything is nicely ordered so it is generally possible to tell at a glance which ID is larger.

Six-Bit - iDB form of Base-64 encoding. Used for converting binary ID numbers to a human-readable form, as well as for encoding longer binary data (such as disk files) for inclusion in Queries.

Predicate - A list including a Closure Number (CN), optional Association Number(s) (ANs) and Field Name(s) (FNs) used to determine whether an Update Query will conflict with previous database operations. The predicate is included at the beginning of a Query and is enclosed in curly braces { }. If the predicate conflicts, the Query terminates immediately and returns a null result.

Supervisory Commands

These commands can be run through the web interface at http://a-0.us/iDB.html.

Fill in the **Database:** field with the name of a file on the server.

Fill in the **Access:** field with the name of an Accessor, if desired. When security is implemented, an accessor will be required. The test implementation waives this requirement to simplify actual editing of Accessors.

Fill in the **Model:** field with the name of a model, if desired.

The **Command:** field should contain a Supervisory Command or a Query or a Query List.

The **Data:** field may contain literal data to be inserted using the Supervisory Import commands. It may also be used to specify the first record of a CSV file to give field names to records loaded from a disk file.

The **File:** field may be used to specify a disk file to be imported.

Command	Description
!NEW	Create a new, empty database with the specified name. Will overwrite an existing database.
!IMPORT NEW TXT	Loads a text file into a new database by creating numbered records from the individual lines of the text file. Will overwrite an existing database.
!IMPORT TXT	Loads numbered text lines into individual records.
!IMPORT NEW CSV	Creates a new database using the data from the CSV file. The first record contains the field names that will be used. Will overwrite an existing database.
!IMPORT ALL CSV	Adds records to the database using the data from the CSV file. The first record contains the field names that will be used. Every record in the CSV file will create a new record in the database.
!IMPORT CSV	Adds records to the database using the data from the CSV file. The first record contains the field names that will be used. Duplicate records database will be suppressed.
!IMPORT NEW XML	Creates a new database using the data from the XML file. XML tags or properties will define the field names that will be used. Will overwrite an existing database.
!IMPORT ALL XML	Adds records to the database using the data from the XML file. XML tags or properties will define the field names that will be used. Every record in the CSV file will create a new record in the database.

Command	Description
!IMPORT XML	Adds records to the database using the data from the XML file. XML tags or properties will define the field names that will be used. Duplicate records database will be suppressed.
!IMPORT NEW HTML	Creates a new database using the data from a **<table>** in the HTML file. The first **<th>** or **<tr>** defines the field names that will be used. Will overwrite an existing database.
!IMPORT ALL HTML	Adds records using the data from a **<table>** in the HTML file. The first **<th>** or **<tr>** defines the field names that will be used. Every record in the CSV file will create a new record in the database.
!IMPORT HTML	Adds records using the data from a **<table>** in the HTML file. The first **<th>** or **<tr>** defines the field names that will be used. Duplicate records database will be suppressed.
!IMPORT CSNV	Adds records using Comma Separated Name=Value pairs. Does not honor Quoted Strings. Every record in the input file will create a new record in the database.
!IMPORT SSS	Adds records using Semicolon Separated Name=Value pairs. Does not honor Quoted Strings. Every record in the input file will create a new record in the database.

QQL Commands

Command	Example	Description
+	("Kind"=="A" + "Val"=="2 ?)	Let find add to the list of Associations. This is essentially equivalent to an OR operation.
*	(* "Kind"="test")	Create a new record if there are no records selected.
?-		Do not show any Fields
?	("Kind"=="A" ?)	Show all unique fields in selected records
@		Perform Script. NOT IMPLEMENTED
!ALL	(!ALL ? !HTML)	Select all Associations from the database
!Balance	(!balance)	Force re-balance of the Literal Value Table B-Tree.
!PutClosure	(!putclosure)	Explicitly insert a closure marker in the database.
!PutTreeWebPage	(!PutTreeWebPage)	Display the Literal Value Table as a Tree. Only useful for very small databases.
!Compact	("Kind"=="A" ? !compact)	Show selected Records/Fields in compact HTML
!Compact-all		Show selected Records/Fields in compact form
!Compact-once		Show results in compact form with values once
!Compact-first		Show all literal values first, then compact records
!Compact-none		Show all selected Associations, Fields and Values in compact form as IDs, without literal values.
!CSV		
!+Log		Turn on operation logging
!-Log		Turn off operation logging
!+Save		Enable save of database after Query completes
!-Save		Disable save of database after Query completes
!+Edit		Build results into an editable form
!-Edit		Return standard result format
!HTML		Return results in HTML
!XML		Return results in XML
!Value-NCount		Number of selected numeric values

Command	Example	Description
!Value-Count		Number of selected values
!Value-NUM		Show the list of selected values in numeric format
!Value-SUM		Compute Sum
!Value-MIN		Compute Minimum
!Value-MAX		Compute Maximum
!Value-AVG		Compute Average
!Value-STV		Compute Standard Variance
!Value-STD		Compute Standard Deviation
!Value-SOP		Compute Sum-of-Products
!Value		Show the list of selected values
!Value-ACount		Number of selected Associations
!Value-FCount		Number of selected Field Names
[1..9] [-10..-1]		Select subset of records. Starting and Ending record. Negative numbers count from the last record
[55:3] [-5:1]		Select range of records. Starting record and count. Negative numbers count from the last record
!Extract		Show database in HTML Extract form
!VerifyLVT		Dump and Verify Literal Value Table
!LVT		Dump entire Literal Value Table in HTML form
!AT		Dump full Association Table in HTML form.
"Field"~"regex"		Select records with matching Regular Expression
"Field"==="value"		Select records with Field identically equal to Value
"Field"=="value"		Select records with Field logically equal to Value
"Field"=0;"value"		Set Field to Value in every selected record
"Field"="value"		Add new Field Value for every selected record
"Field"="v1";"v2"		Set Field to multiple Values in every selected record

Command	Example	Description
"Field"<"value"		Select records with Field logically less than Value
"Field"<="value"		Select records with Field logically less than or equal to Value
"Field">"value"		Select records with Field logically greater than Value
"Field">="value"		Select records with Field logically greater than or equal to Value
AssocNum		Explicitly add Association Number to selected list
"Field"?-		Remove Field from list of fields to show
"Field"?		Add Field to list of fields to show
"Field"@		Pivot to select all Associations that contain any of the Field Value pairs
"Field"*		Sort by the selected Field
"Field"=!now		Special Form: Include a timestamp in a field value. Usually used in a Creator.
"Field"=!query		Special Form: Allow saving the current query for logging purposes.
"Field"=!Referer		Special Form: Include the HTTP Referer in a field value.
"Field"=!RemoteAddr		Special Form: Include the HTTP Remote IP Address in a field value.
"Field"=!RemoteHost		Special Form: Include the HTTP Remote Host Name in a field value.
"Field"=!UserAgent		Special Form: Include the HTTP User Agent in a field value.
"Field"=!URL		Special Form: Include the URL in a field value.

Syntax Notes:
A "Field" or "value" may consist of a literal value contained within double quotes or accent marks.

A literal contained in single quotes will be expected to be in six-bit form. This allows transport of binary values

over normal communication channels.

A "`Field`" or "`value`" may be an iDB identifier. This is a six-bit-encoded integer that represents a literal value. You can get ID values for literals by selecting **`!Compact-first`** result format.

A "`Field`" or "`value`" may take the form of a localID followed by a quoted string. The localID becomes a synonym for the iDB identifier of the literal string. This allows repeated uses of a literal to only require entry once in a given Query, and to only need a single lookup in the LVT.

Specialty operators beginning with ! are not case sensitive.

Error Indications

The following error messages may be generated by the iDB Server.

-MaxAT Overflow is n,nnn

-MaxLV Overflow is n,nnn

-MaxID Overflow is n,nnn

-PutAT AN is already Literal Value: ssss

-PutAT FN is not Literal Value: ssss

-PutAT LV is not Literal Value: ssss

-Invalid Time Travel

-Error: Missing ")".

-No Create

-Missing "]" in Range [...]

-Unknown Command: ..."*command*"

-Unrecognized Relational: ..."*relop*"

-At n,nnn of n,nnn, Query Exception: *exception message*

-Unrecognized Supervisory Command: ...“*command*”...

-Unrecognized Query Segment: ...“*command*”...

-Illegal Predicate: ...“*command*”...

-Predicate missing CN: ...“*command*”...

Example Command Sequences