ADQL Issues Revision from TAPNotes

DAL II session

TAP/ADQL Round Table

October 11, 2014

Separator Nonterminal

Description

This item proposes two options to clarify the use of whitespace and comments in ADQL.

One option for such a clarification is to amend section 2.1 of ADQL with a subsection 2.1.4, "Tokens and literals", containing text like the following (taken essentially from SQL1992).

- Any token may be followed by a separator. A nondelimiter token shall be followed by a delimiter token or a separator.

Since the full rules for the separator are somewhat more complex in ADQL, an attractive alternative could be to omit the separator nonterminal from the grammar and to just note:

- Whitespace and comments can occur wherever they can occur in SQL1992.

May 2014 Interop

Accepted as errata for the current version of the ADQL standard.

Type System

Description

This item proposes adding new section to introduce a notion of types into section 2 of the ADQL standard.

(table next slide)

May 2014 Interop

Accepted for next version Should be discussed further, with a view to including it in the next **(minor) version** of the ADQL standard.

Type System — Types

VOTable			ADQL
datatype	arraysize	xtype	type
boolean	1		implemenation defined
short	1		SMALLINT
int	1		INTEGER
long	1		BIGINT
float	1		REAL
double	1		DOUBLE
(numeric)	$\rangle 1$		implementation defined
char	1		CHAR(1)
char	n*		VARCHAR(n)
char	n		CHAR(n)
unsignedByte	n*		VARBINARY(n)
unsignedByte	n		BINARY(n)
unsignedByte	n, *, n*	adql:BLOB	BLOB
char	n, *, n*	adql:CLOB	CLOB
char	n, *, n*	adql:TIMESTAMP	TIMESTAMP
char	n, *, n*	adql:POINT	POINT
char	n, *, n*	adql:REGION	REGION

Note

Transporting TIMESTAMP, POINT and REGION as strings should not imply the ADQL string concatenation operators are applicable.

Empty Coordinate Systems

Description

This item proposes deprecating the string-valued first argument for the geometry constructors (BOX, CIRCLE, POINT, POLYGON).

May 2014 Interop

Requires further discussion. The proposal needs more work done on it before it could be included in the ADQL standard.

Explanation of optional features

Description

This item proposes adding a section of text to both the TAP and ADQL specifications that clarifies how optional features are described.

May 2014 Interop

Accepted for next version. This item should be discussed further, with a view to including it in the next **(minor) version** of the ADQL standard.

Simple Crossmatch Function

Description

This item proposes adding a simple positional crossmatch function to ADQL.

- It was agreed that this would be a useful feature for end users
- It was noted that adding this feature could be difficult to implement
- It was noted that part of the rationale for the IVOA services was to implement difficult things on the server side, making things easier for the end user

May 2014 Interop

Requires further discussion. The proposal needs more work done on it before it could be included in the ADQL standard.

No type-based decay of INTERSECTS

description

This item proposes deprecating the use of POINT as the first parameter to INTERSECTS.

May 2014 Interop

Accepted for next version. The proposed text should be included in the next **(minor) version** of the ADQL standard.

Generalized user defined functions

Description

This item proposes allowing user defined functions to return geometric types.

It was agreed that there should be no restriction on the return types of User Defined Functions.

May 2014 Interop

Accepted as errata for the current version of the ADQL standard.
It was also noted that the SimDAL working group would like to be able to define table value functions in ADQL.
It was agreed **to continue the discussion** to find a way of adding support for table value functions in a future version of the ADQL standard.

Case-Insensitive String Comparisons

Description

This item proposes adding functions and operators for to support case-insensitive string comparisons.

While support the proposed features per se, I don't think they should be optional. They're not hard to implement on all DB engines I'm aware of, and optional features are always a pain in so many ways. You'd at least have to say how clients are expected to discover support for them if you go for optional (Markus).

May 2014 Interop

Accepted for next version. It was agreed that the UPPER and LOWER functions and the ILIKE operator should be included as an optional feature in the next **(minor) version** of the ADQL standard.

Set operators

Description

This item proposes adding support for the UNION, EXCEPT and INTERSECT operators.

We need to confirm which platforms support what operators (Dave).
The text describing the set operators in the ADQL standard should include the following caveats

- The set operands MUST produce the same number of columns
- The corresponding columns in the operands MUST have the same data types
- The corresponding columns in the operands SHOULD have the same metadata
- The metadata for the results SHOULD be generated from the left-hand operand

The Oracle database platform includes support for UNION and INTERSECT, but not EXCEPT. However, Oracle's MINUS operator appears to be equivalent to EXCEPT. If we make these operators as a required feature of ADQL then it means service implementations based on Oracle will have to parse and translate ADQL queries. This may not be an issue, but it will be the first time we have (knowingly) added a feature to ADQL that requires service implementations to parse and translate ADQL queries.
Need to map which platforms support which operators. (see next slide)

May 2014 Interop

Accepted for next version.The following operators should be included as required operators in the next **(minor) version** of the ADQL standard: UNION, EXCEPT, INTERSECT.

Set operators — DBMS Support

Vendor	UNION	INTERSECT	EXCEPT
SQLServer	YES	YES	YES
Sybase			
Oracle	YES	YES	(see below)
PostgreSQL	YES	YES	YES
MySQL	YES	NO	NO
Derby			
HSQLBD			
SQLLite			

- The Oracle database platform supports the UNION and INTERSECT, operators directly, and the MINUS operator is equivalent to EXCEPT.
- The MySQL DBMS doesn't support EXCEPT and INTERSECT. Query translations are needed in both cases.

Adding a Boolean Type

Description

This item proposes adding a BOOLEAN type to ADQL.

It was agreed that although making these changes would be a good thing, more work needs to be done on identifying and solving potential compatibility issues before the changes can be included in the ADQL standard. It was agreed that BOOLEAN data type would be a useful feature to add It was agreed that changing the return type of CONTAINS() to be a BOOLEAN would make it easier to use It was agreed that making these changes could cause compatibility issues which could not be addressed in a (minor) increment of the std:ADQL standard

May 2014 Interop

Split this into two separate proposals.
Both of these changes should be considered for a future **(major) revision** of the ADQL standard.

Adding a Boolean Type

Description

Add support for a BOOLEAN type to ADQL, without changing the return type of CONTAINS().

This would avoid the potential compatibility issues raised at the May 2014 Interop. The general concensus at the May 2014 Interop was that a BOOLEAN data type would be useful, and that adding it as a new type would not cause compatibility issues.
I'm fairly skeptical on allowing boolean-typed column references ("WHERE my_col") on grounds that parse errors will probably be harder to understand for users with them (Markus).

changing the return type of CONTAINS()

Description

Change the return type of CONTAINS().

This is the part that would potentially cause compatibility issues with existing services and clients. If we want this, it's definitely a matter for a major version, and I'd first like to see the impact of boolean-valued expressions outside of comparisons (Markus).

Casting to Unit

Description

This item proposes adding a new function IN_UNIT(expr,) which would convert values in one unit into another.

It was agreed that scaling conversions would not be difficult to implement It was agreed that conversion between wavelength and frequency would be difficult to implement consistently It was agreed that unit conversions would be most useful in a SELECT list It was agreed that unit conversions would be most difficult to implement in a WHERE clause.
Implementations should be given some leeway in what they'd accept; I'd consider it fine if they were only required to allow unit casts on single columns, and restrict what pairs of unit strings they support (Markus).

May 2014 Interop

Requires further discussion. The proposal needs more work done on it before it could included in the ADQL standard.

Column References with UCD Patterns

Description

This item proposes adding a pre-processing macro that locates columns based on their UCD.

I believe the main issue here is what do to when no or multiple columns match (Markus).

May 2014 Interop

Requires further discussion. The proposal needs more work done on it before it could be considered ready to be included in the ADQL standard.

Modulo operator

Description

This item proposes adding the modulus operator x % y to the std:ADQL grammar.

If this is the only required operator we are adding, then it is probably not worth the potential compatibility issues. However, if we are updating the std:ADQL grammar to add UNION, EXCEPT and INTERSECT as operators, does adding the x % y operator cause any additional compatibility issues? (Dave)
I'm unconvinced new syntax that doesn't add capabilities is worth the effort. (Markus)

May 2014 Interop

Rejected The benefits of adding the x % y operator syntax were outweighed by cost of compatibility issues caused by adding a new required operator to the ADQL grammar.

Bitwise operators

Description

This item proposes adding support for the AND, OR, XOR and NOT bitwise operations and hexadecimal literals to the ADQL grammar.

May 2014 Interop

Accepted for next version. Hexadecimal literal values should be included in the next **(minor) version** of the ADQL standard.
Accepted for next version. The following functions should be included as an optional feature in the next **(minor) version** of the ADQL standard.: BIT_AND(x, y), BIT_OR(x, y), BIT_XOR(x, y) and BIT_NOT(x)
Rejected. The benefits of adding the operator, [exp] op [exp], syntax for each operation were outweighed by cost of compatibility issues caused by adding new operators to the ADQL grammar.

However, if we are updating the std:ADQL grammar to add UNION, EXCEPT and INTERSECT as operators, does adding the [exp] op [exp] for each of the bitwise operations cause any additional compatibility issues? (Dave)
The functions seem useful and implementable to me; I don't see a big benefit in making the functionality available through operators. (Markus)

CAST operator

Description

This item proposes adding a limited form of the CAST operator to the ADQL grammar. This proposal specifically limited the CAST operation to numeric data types only, and excluded CAST to and from character strings.

May 2014 Interop

Accepted for next version.The CAST operator should be including as a required operator in the next **(minor) version** of the ADQL standard.
To be discussed further. The set of type conversions should be discussed further, with a view to finalizing the set of conversions supported in the next **(minor) version** of the ADQL standard.