

The ADQL Parser and Its Context

Jeff Lusted
Leicester



What does the parser do?

- It turns ADQL/s into ADQL/x
- More of that later, but here is an example...
- It can turn this...

```
Select a.POS_EQ_RA, a.POS_EQ_DEC,  
a.ID_FIELD, a.CODE_QUALITY,  
a.PHOT_FLUX_PEAK From catalogue as a ;
```

- Into this...

ADQL/x Parser Output...

```
<v1:Select xmlns:v1="http://www.ivoa.net/xml/ADQL/v1.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <v1:SelectionList>
    <v1:Item Name="POS_EQ_RA" Table="a" xsi:type="v1:columnReferenceType"/>
    <v1:Item Name="POS_EQ_DEC" Table="a" xsi:type="v1:columnReferenceType"/>
    <v1:Item Name="ID_FIELD" Table="a" xsi:type="v1:columnReferenceType"/>
    <v1:Item Name="CODE_QUALITY" Table="a"
xsi:type="v1:columnReferenceType"/>
    <v1:Item Name="PHOT_FLUX_PEAK" Table="a"
xsi:type="v1:columnReferenceType"/>
  </v1:SelectionList>
  <v1:From>
    <v1:Table Name="catalogue" Alias="a" xsi:type="v1:tableType"/>
  </v1:From>
</v1:Select>
```

Or an alternative example...

- A cone search

Select * From catalogue as a

Where (a.POS_EQ_RA<100) And

((a.POS_EQ_RA>100) And (ACOS(((

SIN(a.POS_EQ_DEC) * SIN(100)) +

COS(a.POS_EQ_DEC) * (COS(100) *

COS((a.POS_EQ_RA - 100)))))) <= 100))

- Into this...

Cone Search in ADQL/x...

```
<?xml version="1.0" encoding="UTF-8"?>
<v1:Select xmlns:v1="http://www.ivoa.net/xml/ADQL/v1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <v1:SelectionList>
    <v1:Item xsi:type="v1:allSelectionItemType"/>
  </v1:SelectionList>
  <v1:From>
    <v1:Table Name="catalogue" Alias="a" xsi:type="v1:tableType"/>
  </v1:From>
  <v1:Where>
    <v1:Condition xsi:type="v1:intersectionSearchType">
      <v1:Condition xsi:type="v1:closedSearchType">
        <v1:Condition Comparison="&lt;" xsi:type="v1:comparisonPredType">
          <v1:Arg Table="a" Name="POS_EQ_RA" xsi:type="v1:columnReferenceType"/>
          <v1:Arg xsi:type="v1:atomType">
            <v1:Literal Value="100" xsi:type="v1:integerType"/>
          </v1:Arg>
        </v1:Condition>
      </v1:Condition>
      <v1:Condition xsi:type="v1:closedSearchType">
        <v1:Condition xsi:type="v1:intersectionSearchType">
          <v1:Condition xsi:type="v1:closedSearchType">
            <v1:Condition Comparison=">" xsi:type="v1:comparisonPredType">
              <v1:Arg Table="a" Name="POS_EQ_RA" xsi:type="v1:columnReferenceType"/>
              <v1:Arg xsi:type="v1:atomType">
                <v1:Literal Value="100" xsi:type="v1:integerType"/>
              </v1:Arg>
            </v1:Condition>
          </v1:Condition>
        </v1:Condition>
      </v1:Condition>
      <v1:Condition xsi:type="v1:closedSearchType">
        <v1:Condition Comparison="&lt;=" xsi:type="v1:comparisonPredType">
          <v1:Arg Name="ACOS" xsi:type="v1:trigonometricFunctionType">
            <v1:Arg xsi:type="v1:closedExprType">
              <v1:Arg Oper="+" xsi:type="v1:binaryExprType">
                <v1:Arg xsi:type="v1:closedExprType">
```

Why are we doing this?

- Answer: To aid transformations!
- Archives are being stored in relational databases:
 - Relational model
 - Increasingly capacious
 - Proven software
- Most RDBMS's support SQL:
 - But SQL was never part of the relational model and each manufacturer has competed in developing their own variant.
- MySQL, PostgreSQL, SQL Server, Oracle, DB2 are all different.

Some trivial examples...

- Finding the current date

- Select getdate() ; SQL Server and MySQL
- Select SYSDATE from dual ; Oracle
- Select CURRENT_DATE from sysibm.sysdummy1 ; DB2

- Using Column Aliases

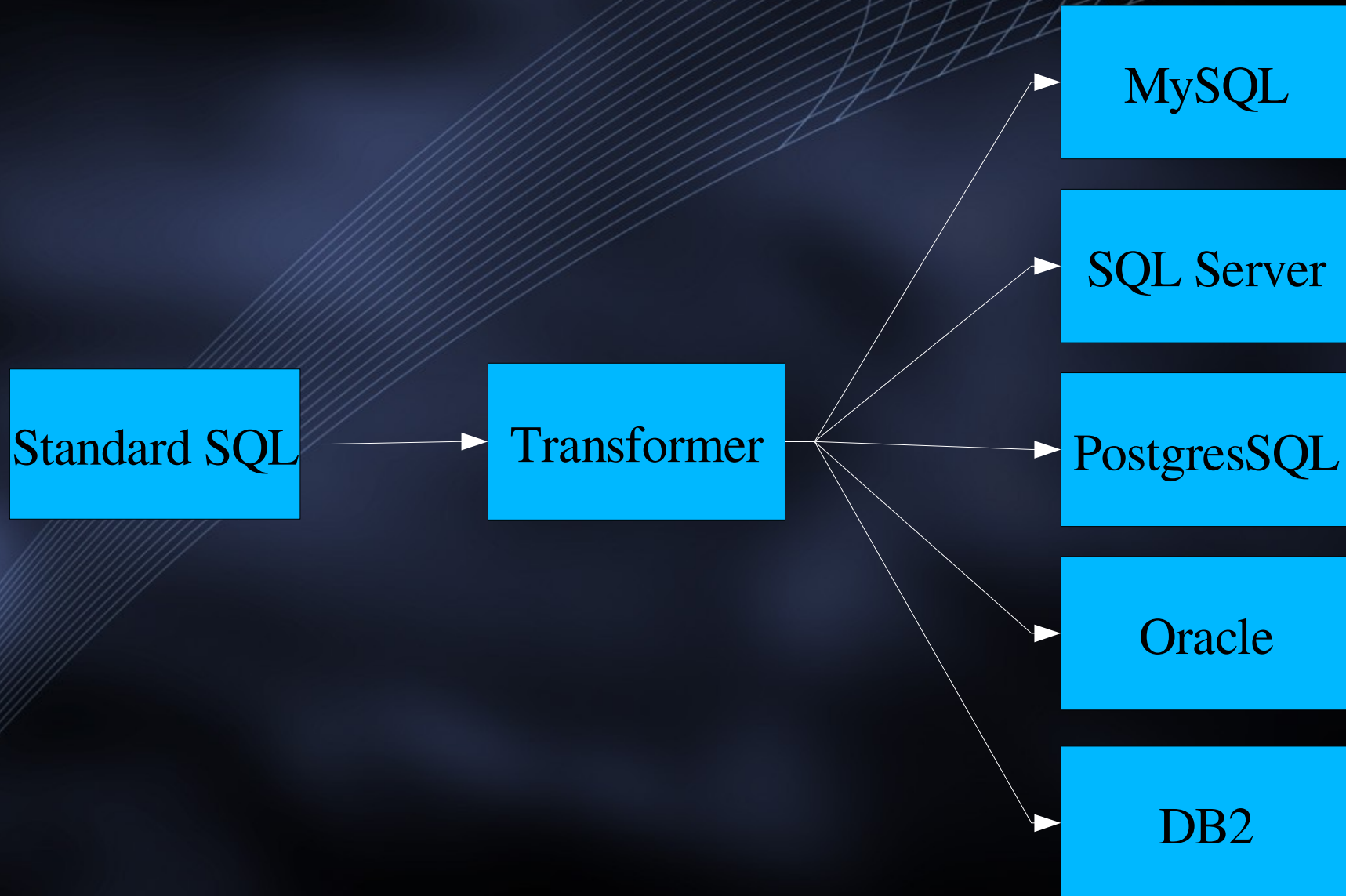
- This is valid for MySQL and other RDBMS's

```
Select MAX(a.POS_EQ_RA) as max_ra,  
       MIN(a.POS_EQ_RA) as min_ra From catalogue as a ;
```

- But this is not valid for MySQL

```
Select MAX(a.POS_EQ_RA) max_ra,  
       MIN(a.POS_EQ_RA) min_ra From catalogue as a ;
```

Transformations 1 ...



Transformations 2 ...



Standards: ADQL/x

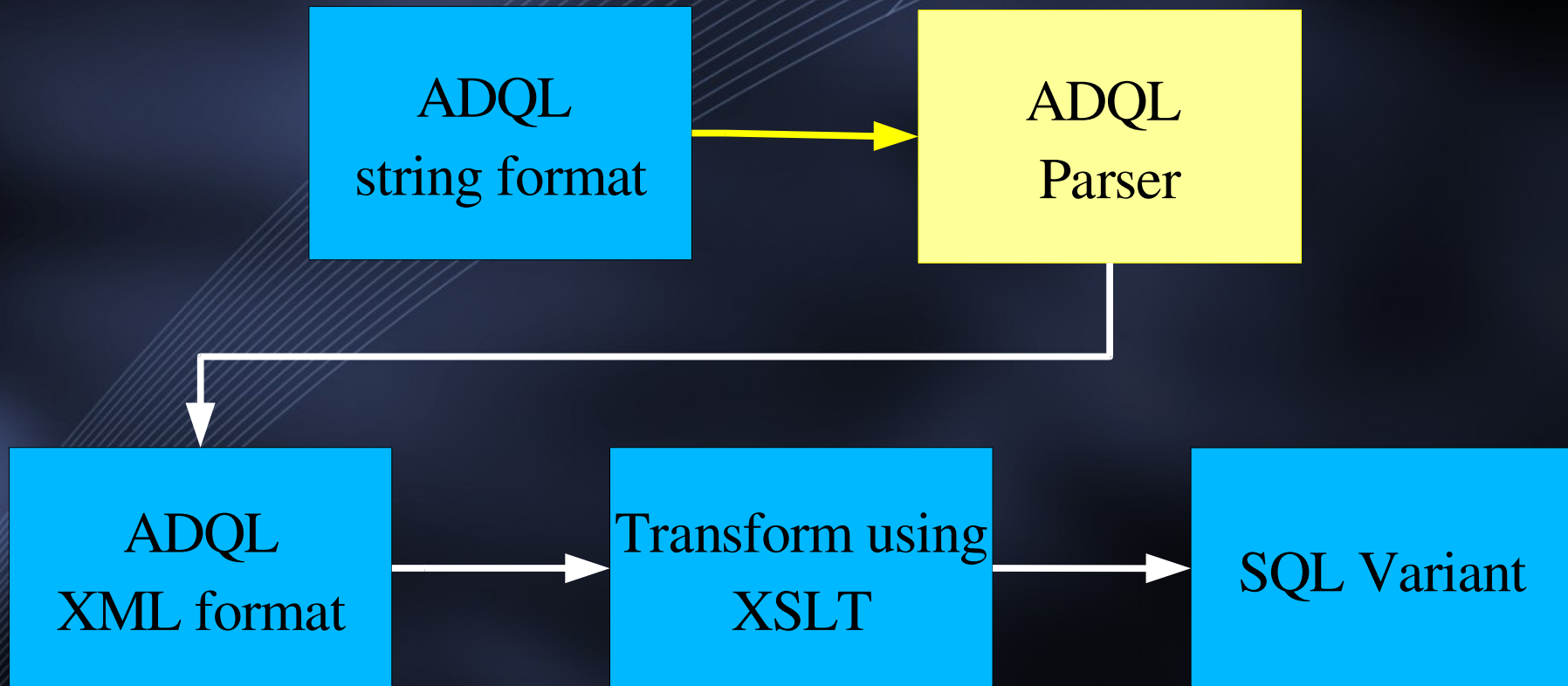
The first ADQL spec was orientated to XML:

- There was an XML schema (ADQL/x)
- And also a syntax for a human readable string format (ADQL/s)
- The two were difficult to keep in line (even in the first spec)
- Using XML for transporting queries has not made for transparency.

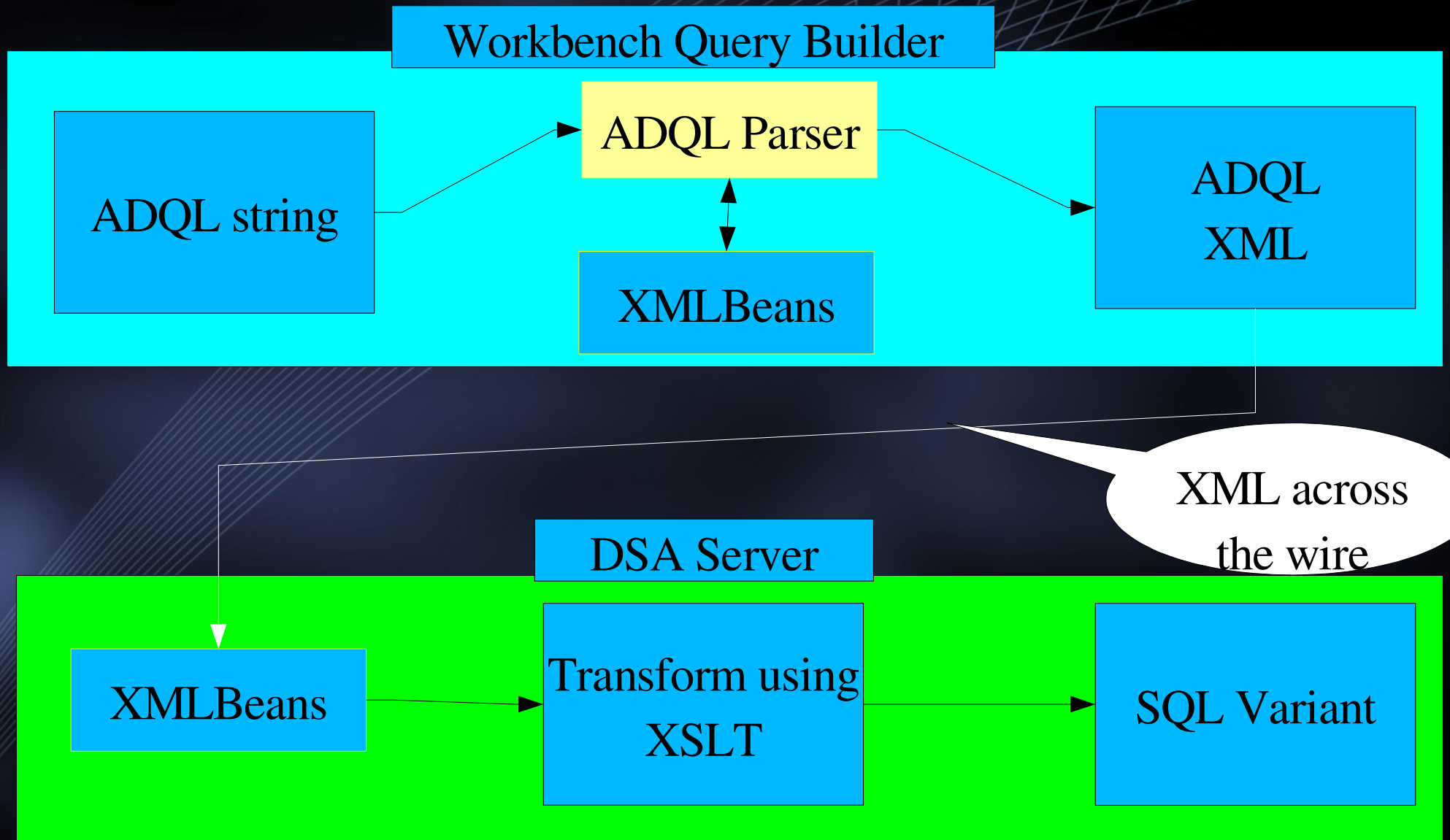
Standards in transition: ADQL/s

- The basis of the standard has changed.
- The string form of ADQL has become the focus (ADQL/s).
- TAP will use ADQL/s to transport query requests.
- The syntax of ADQL/s is BNF based upon SQL92 with astronomical extensions.
- ADQL/x is henceforth an implementation issue.

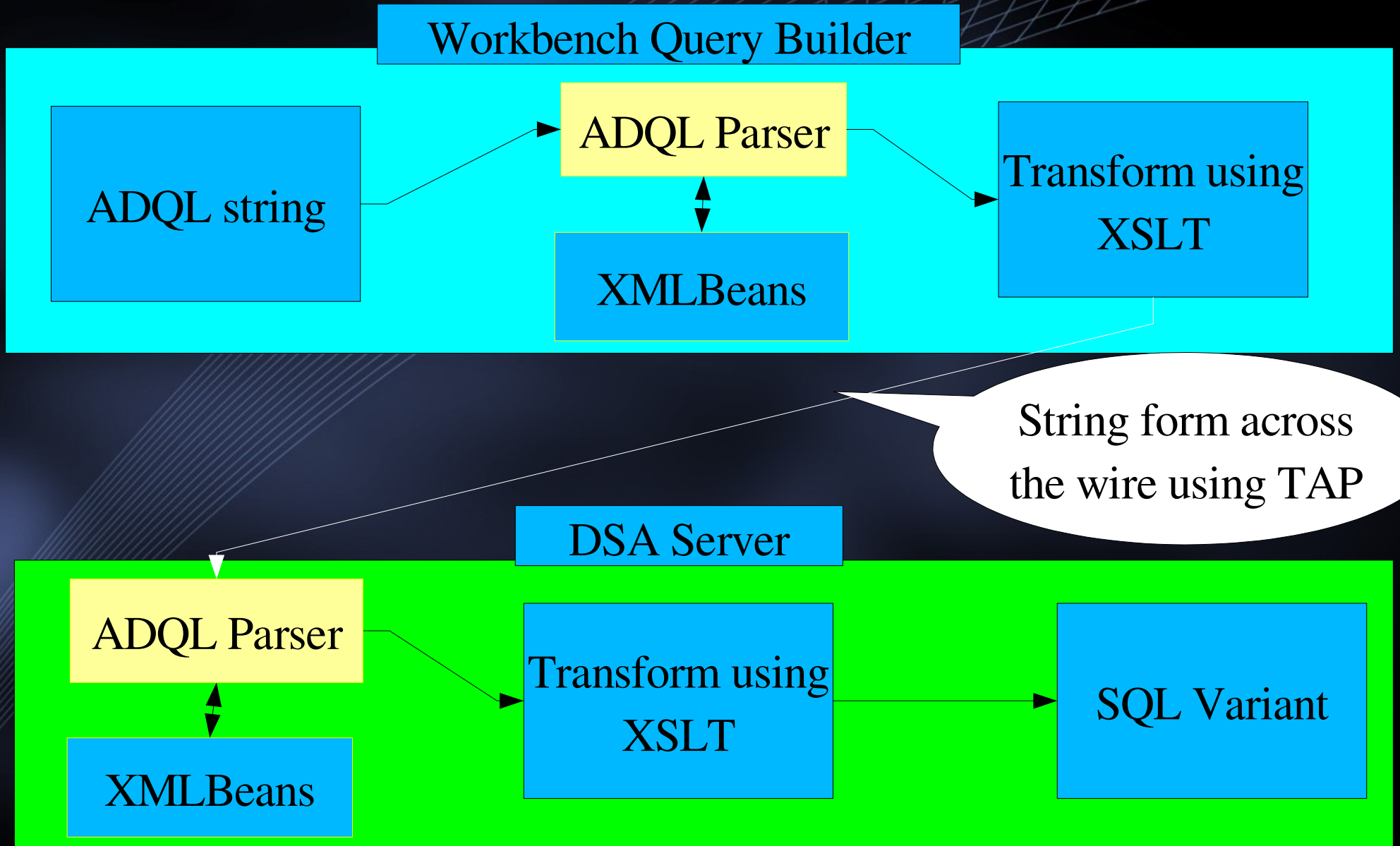
Where the Parser sits logically.



Where the Parser sits physically:



Where the Parser will sit physically:



Specification: BNF Diagrams

<query_specification> ::=

SELECT [<set_quantifier>][<set_limit>] <select_list>
<table_expression>

<table_expression> ::=

<from_clause>

[<where_clause>]

[<group_by_clause>]

[<having_clause>]

[<order_by_clause>]

Technologies used

- JavaCC to design/generate a parser that conforms to a BNF syntax
 - Open Source (Apache Foundation)
 - Java based
 - Tokens are defined using regular expressions and BNF-aligned syntax
- XmlBeans to produce the resulting XML

But How?

- Take the ADQL/x schema
- Back port the schema to a BNF grammar
- At the same time compare with SQL92
- Make sure the BNF grammar is in a format suitable for JavaCC
- Make sure I don't go too far: Region, STC, Crossmatch
- Write a utility (different from the parser) to generate official-looking BNF from the result.

Result: AdqlStoX

- Basically syntactical BNF
- Semantics beginning to creep in.
 - XmlBeans are used to interrogate the result for semantical correctness
 - For example, the uniqueness of a column alias is enforced. The following is invalid although syntactically correct:

```
Select MAX(a.POS_EQ_RA) as max_ra,  
       MIN(a.POS_EQ_RA) as max_ra From catalogue as a ;
```

- One of the pleasing aspects is the development seems capable of incremental improvement.

Other Features

- It is capable of compiling a fragment of ADQL
 - For example:
 - “Where a.ra between 1.2 and 1.3”
 - or...
 - “a.ra between 1.2 and 1.3”
 - or even...
 - “1.2”
- Rather surprisingly, it can be setup to compile (or at least check the syntax) as you type.

Future Development.

- Supporting comments (usability issue)
- Multiple error tracking
- Better error messages and reporting
- Folding in of table meta data for better semantics
- Specialized version to sit within DSA
- Offered as a service (SOAP or REST).