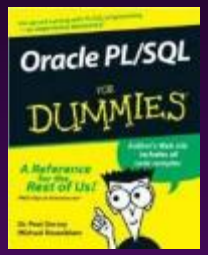
Printing Without Printers: A Database-Centric Approach to Document Exchange and Reporting

NYOUG Summer General Meeting
June 14, 2016

Michael Rosenblum & Dr. Paul Dorsey
Dulcian, Inc.
www.dulcian.com

Who Am I? – "Misha"

- Oracle ACE
- Co-author of 3 books
  - *PL/SQL for Dummies*
  - *Expert PL/SQL Practices*
  - *Oracle PL/SQL Performance Tuning Tips & Techniques* (Rosenblum & Dorsey, Oracle Press, July 2014)



- Known for:
  - SQL and PL/SQL tuning
  - Complex functionality
    - Code generators
    - Repository-based development

Documents???

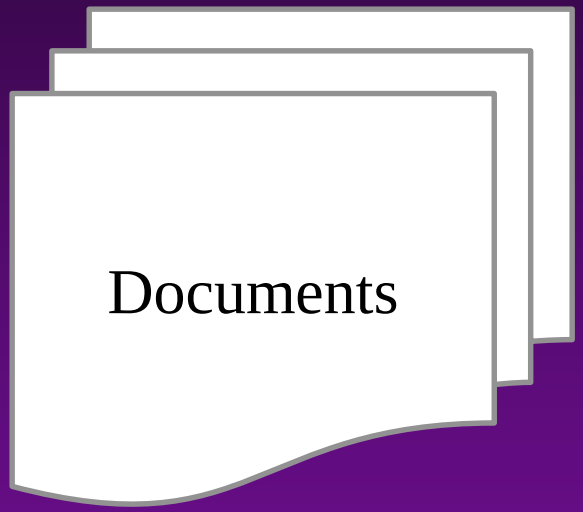
- Why bother?
  - Standards
    - For example - government, medical
  - High importance
    - Legally binding
  - Physical limitations
    - Sorry, no scrollable parchments anymore ☺

Documents…

- What are they?
  - Formal representation of available data
    - Non-interactive
    - Self-contained

- What about them?
  - Data representation on the screen is NOT the same as on paper
    - … always present is the notion of “translation” ➔ something will be “lost in translation”

Documents!!!



Data

Pre-defined format

Documents ?!?!

Topic, please!!!

Boring!

Ancient history…

Being done forever!!!

So what?

Well…

"…as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know."

© D. Rumsfeld

Known Knowns

- Data
  - Every database can store text.
    - ... And some of them are even explicitly made to be document-based
  - All scalar datatypes can be converted to text.
  - … to be fair, with some potential data loss
- Formats
  - DOC, DOCX, PDF, XLSX… (you name it!)

DULCIAN INC.

Known Unknowns

- Where to merge data and formatting
  - Schools of thought:
    - Front-end
    - Middle-tier
    - Back-end
  - Different implementations:
    - Open-source vs. commercial solutions
- Costs
- How to merge
  - Acceptable data loss when formatting
  - Performance considerations

Unknown Unknowns

- How can you predict and support future requirements?
  - … because changes can just “happen”
- Where do you draw the line between regular reporting and ad-hoc querying?
  - …because both of these areas quickly blur in actual production environments.

DULCIAN INC.

So?

- You need to solve known unknowns.
  - Choosing the wrong environment can be very costly!
- You need to build systems that are:
  - Flexible enough
  - Customizable enough

Technology Decisions

DULCIAN INC.

Client-Side (1)

- Pros:
  - Shifts workload to the client
  - Minimizes network traffic

- Cons:
  - Deployment nightmare!
  - Unpredictable performance

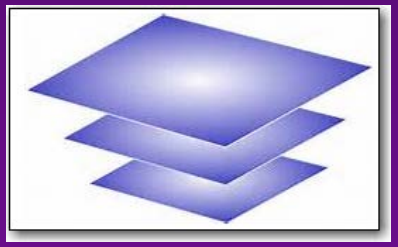
- Current state:
  - Rarely used (only in tightly controlled environments)

Client-Side (2)

- Real world example:
  - What: Client-based document viewer (now called IBM Lotus Forms)
    - Client software requires installation on every working laptop.
    - All forms must be local (and current!) on every laptop.
  - How:
    - Viewer accepts master form plus XML-formatted data.
    - Viewer allows edits which are shipped back to server as XML.
  - When: Early 2000 / 2014
- Lessons learned:
  - Real deployment nightmare, but offloading merging to clients ➔ less server resources needed

Middle Tier (1)

- Pro:
  - Convenient - if accessing multiple data sources
- Cons:
  - Frequently inefficient mechanisms for accessing data sources
    - Multiple roundtrips to database
    - High memory utilization
  - Often overkill (adds complexity)
- Current state:
  - The most commonly used solution

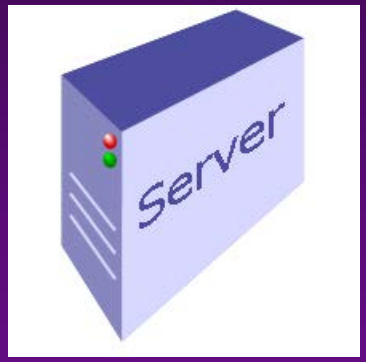


Middle Tier (2)

- Real world example:
  - What: Same client-based document viewer (now called IBM Lotus Forms), but together with Lotus Forms Server
    - Client software requires installation on every working laptop.
    - Server does the merging and returns the document.
  - How:
    - Viewer just shows final document (real-only).
    - Editing is done using a web-based application.
  - When: 2014 / 2017 (being gradually retired)
- Lessons learned:
  - Large documents (20+ pages, 600+ fields) are expensive to merge (both memory- and CPU-wise) ➔ resource tuning is very important.

Server-Side (1)

- Pros:
  - Formatting is brought to data - not vice versa
    - No extra round-trips
    - Complete access to available data
- Cons:
  - Counter-intuitive
  - Increased server workload
- Current state:
  - Deployed and working in production ☺



Server-side (2)

- Real world example:
  - What: Database generates PDFs directly.
    - Many browsers already understand PDFs directly (no need for Adobe products)
  - How:
    - Database stores fillable PDF forms, merges data (Java-based merger) and ships the final document out.
    - Viewer just shows final document (real-only).
    - Editing is done using a web-based application.
  - When: 2015+ (replacing previous implementation)
- Lessons learned:
  - Database workload increase appears to be less than expected.
  - Fillable PDFs use JavaScript internally even for basic calculations (security no-go!) → more work that expected

Why use the database?

- Fewer roundtrips mean less workload
  - … no resource waste to manage connections, environments, etc.
- Less software to configure
  - … because each piece of software has its own setting issues (+corresponding learning curve)
- Single integration mechanism (PL/SQL)
  - … plus various PL/SQL access mechanisms

DULCIAN INC.

Architectural Challenges

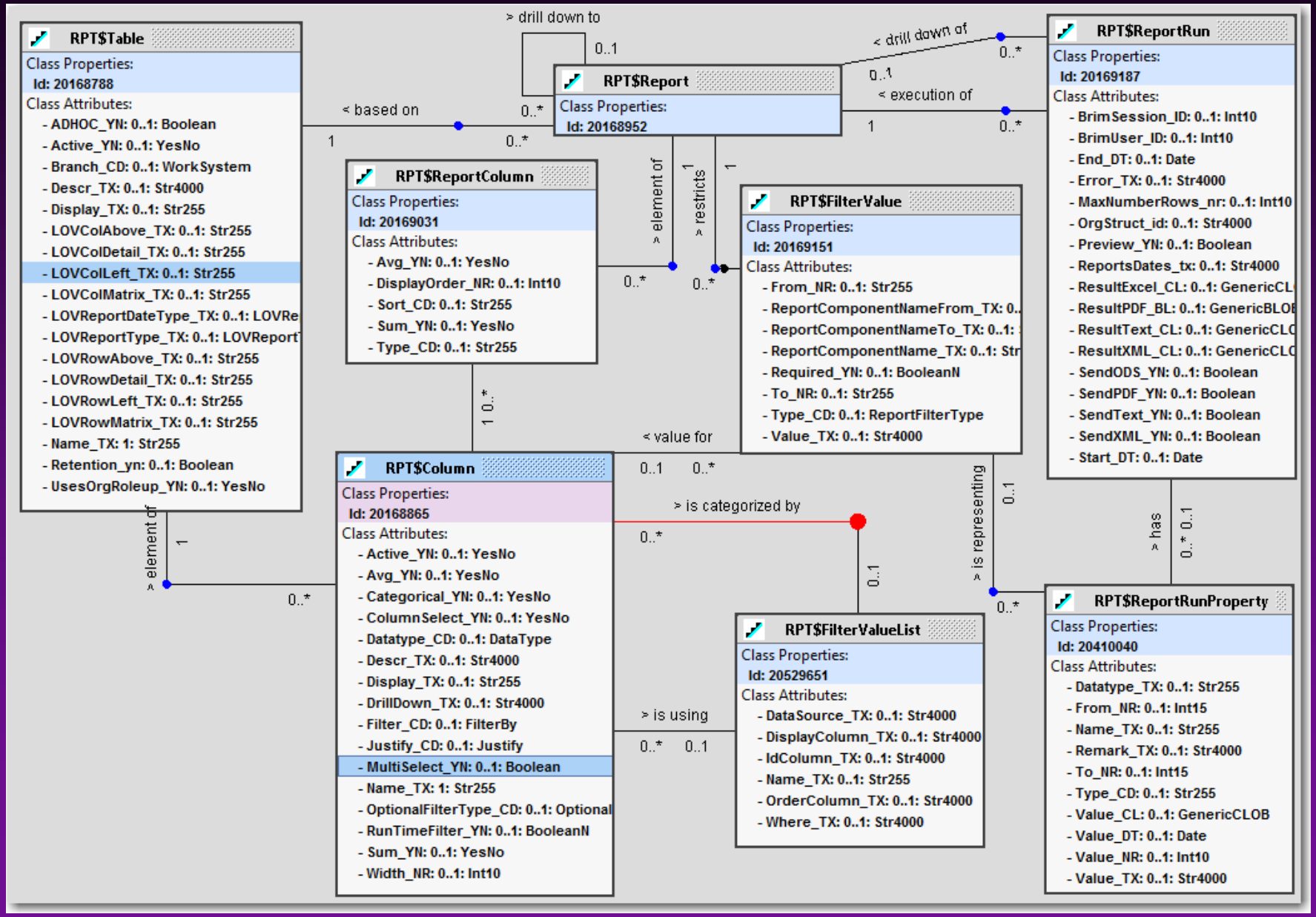
Getting Agile

- Goal:
  - Use flexible reporting
    - Parameters should be customizable enough ➔ end users feel their power
    - Documents should be well-formatted ➔ end users get what they need
  - Avoid “ad-hoc” trap
    - Adequate performance
    - Predictable (and manageable number of query permutations)

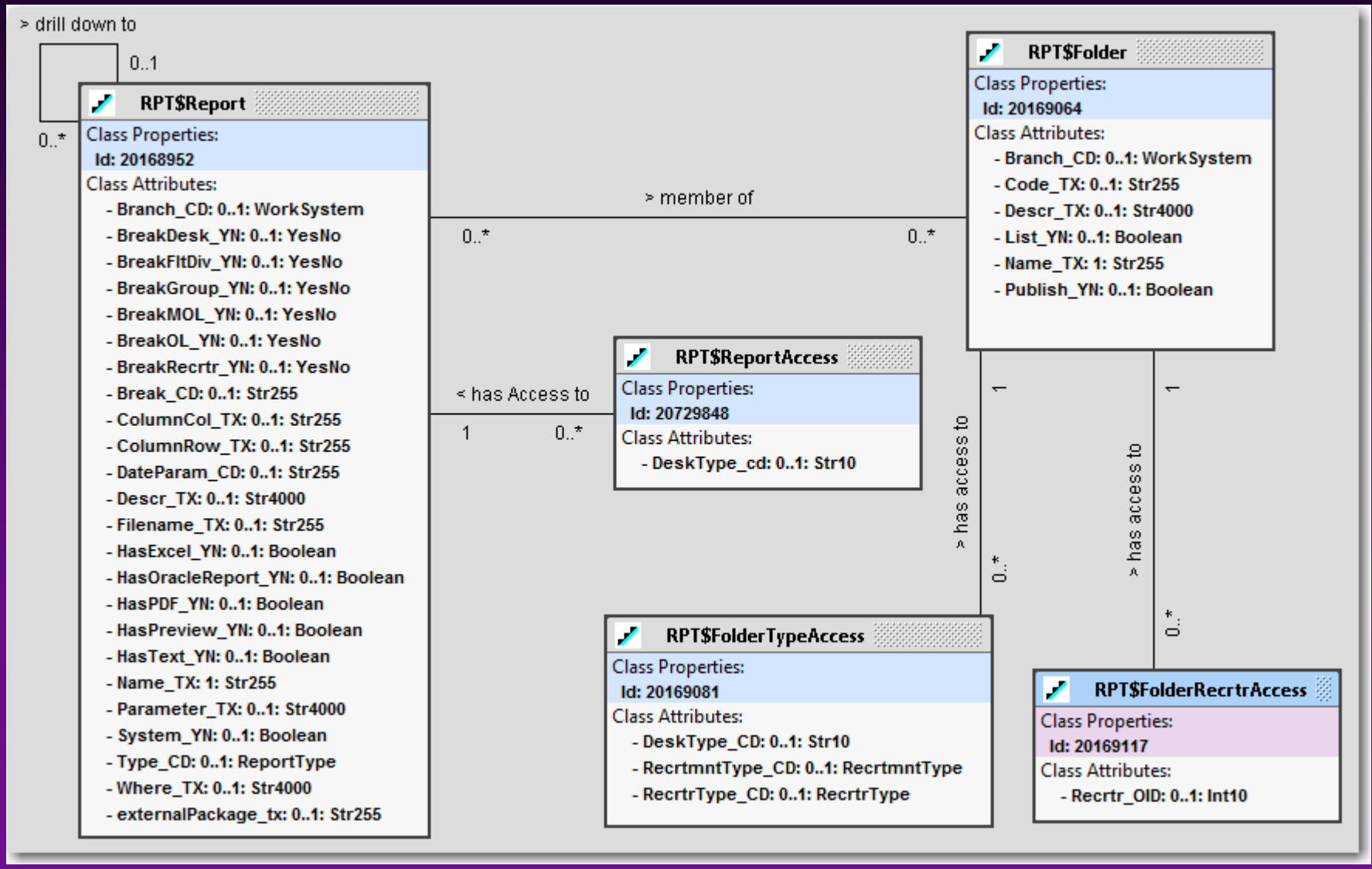
Repository-Based Development

- Solution:
  - Developers/Architects register available data sources plus all extras:
    - Columns/aggregates
    - Filters (run-time and design-time)
    - Output formats
  - End users can define customized reports (or use pre-existing ones):
    - Inputs are stored.
    - Execution is timed.
    - Errors are stored.
    - Results are stored (if needed).

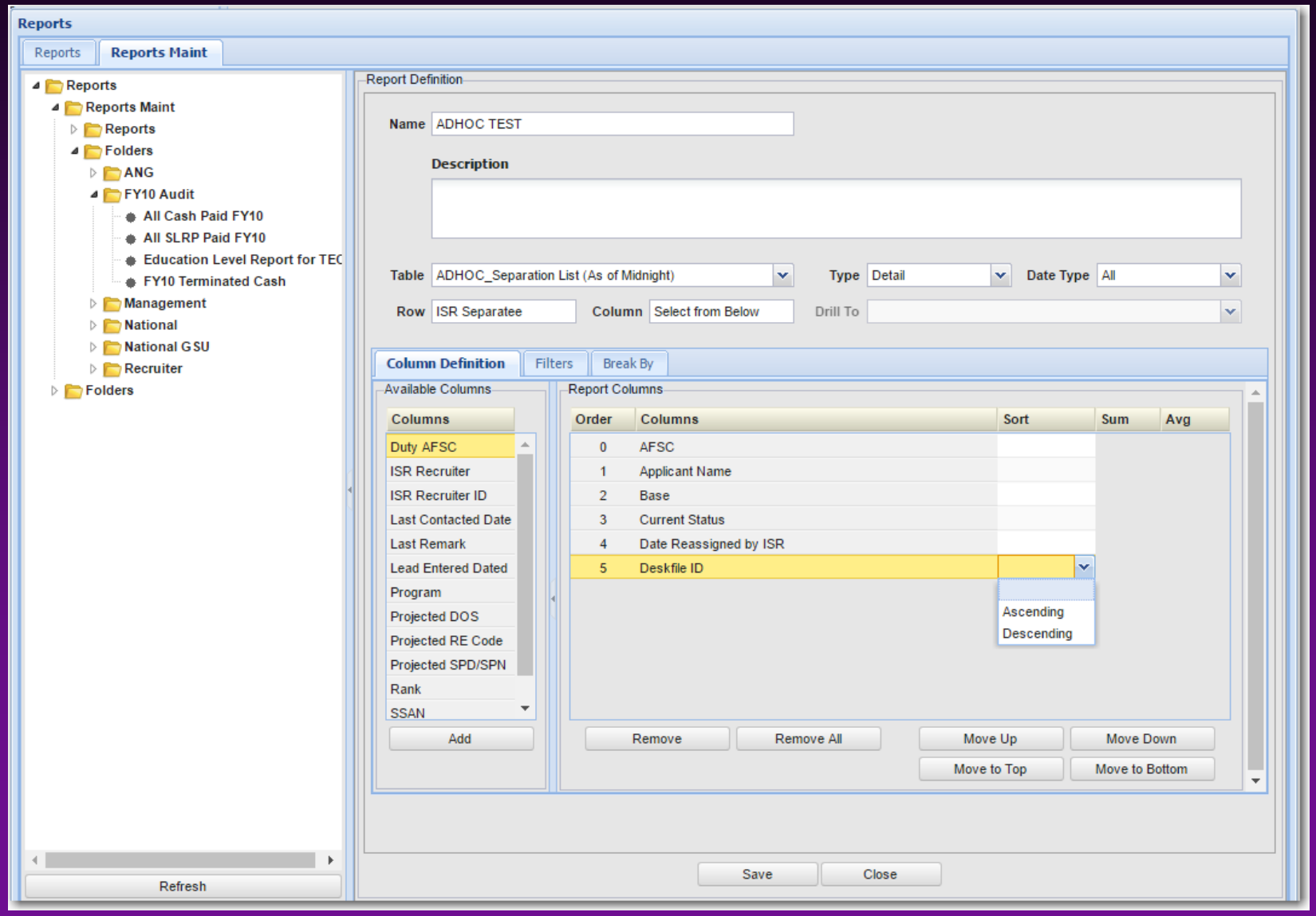
Data model (Main)



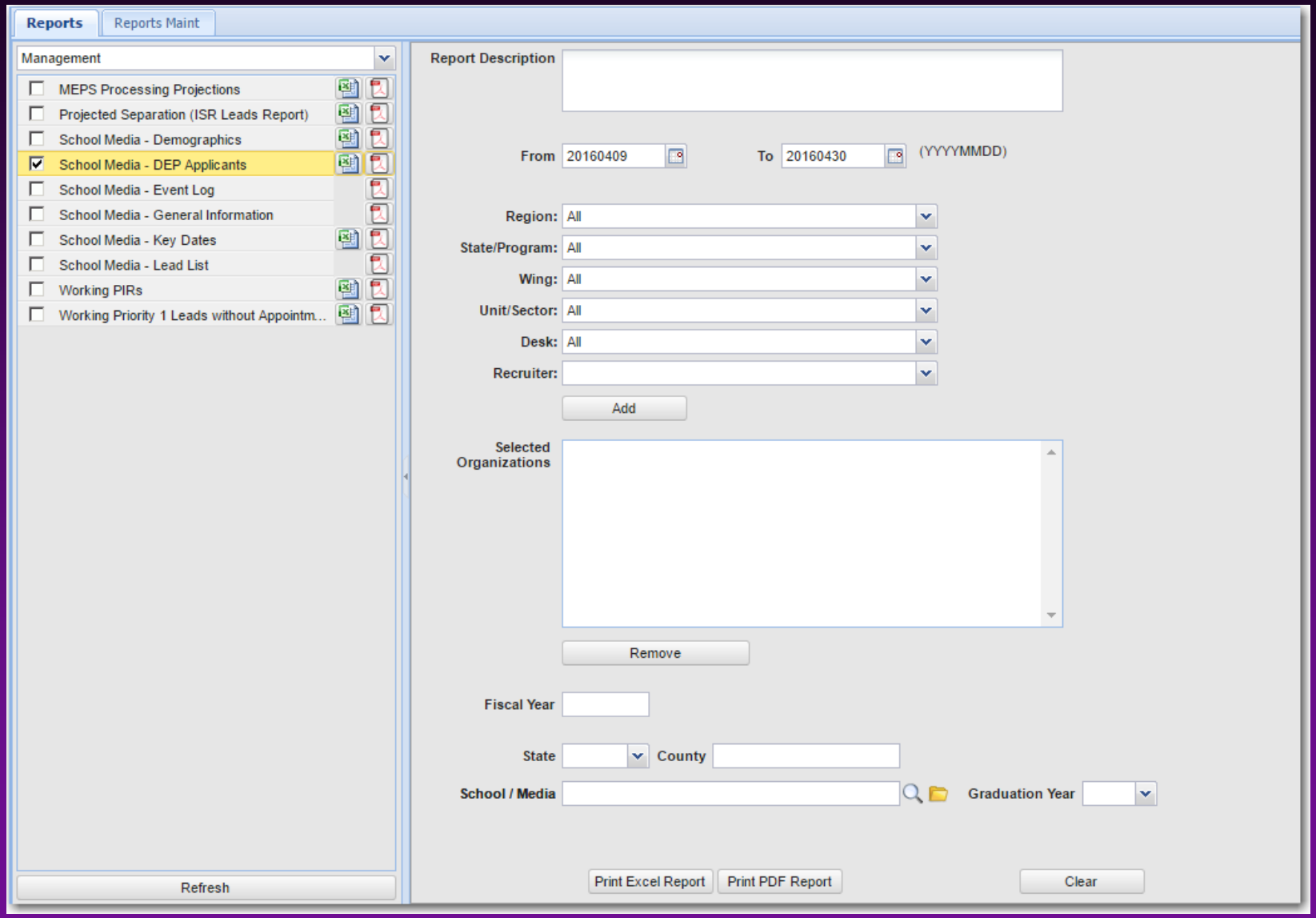
Data model (Access)



Maintenance UI Example



Running UI Example



Important Lessons

- Avoid over-generalizations:
  - … because your own developers should be able to understand the repository.
- Don’t be afraid of project-specific data elements
  - … because it makes your code much more maintainable.
- Never ever reuse attributes from a generic model for alternative purposes
  - … because at some point you may have to migrate to a different solution.

Existing Solutions and Case Studies

Tools Overview

- JSON generation
  - PL/SQL-based [PL/JSON]
    - Free
- PDF generation
  - PL/SQL-based [PL/PDF]
    - Commercial
  - Java-based [ITEXT]
    - “Sort of” free (AGPL)
- XSLX/XML generation
  - PL/SQL-based
    - Our own (meaning free ☺)

PL/JSON (1)

- Used for:
  - Very granular manipulation of JSON data
  - 100% PL/SQL-based
  - Various extra APIs for printing/converting to XML/table access etc.
- Info:
  - Developed by Jonas Krogsboell and Lewis R. Cunningham
  - Github: https://github.com/pljson/pljson
  - Current version: 1.0.5
  - Works even in Oracle 12c, but some people prefer customization of names to avoid clashing with Oracle's own JSON datatype

PL/JSON (2)

- Code sample

```
declare
  obj pljson.json:=pljson.json();
begin
  obj.put('A', 'a little string');
  obj.put('B', 123456789);
  obj.put('C', true);
  obj.put('D', false);
  obj.put('F', json_value.makenull);
  obj.remove('C');

  obj.put('Nested JSON',
              pljson.json('{"lazy construct": true}'));
  obj.put('An array', pljson.json_list('[1,2,3,4,5]'));
  obj.print;
end;
```

PL/JSON (3)

- Our story:
  - Currently in production to manipulate reasonably large (up to 1 MB) JSON documents from a medical provider
    - JSON is multi-level (and often very deep).
    - Manipulations are very intense.
  - Because PL/JSON is using Oracle objects and object-collections, it has a very noticeable impact on server memory usage ➔ needed to make sure that server has enough resourses.
- Overall:
  - Saved us from reinventing the wheel!
  - Written by people who REALLY know the business

PL/PDF (1)

- Used for:
  - Procedural constructor of PDF documents
  - 100% PL/SQL-based (source code is wrapped ☹)
  - Various extensions
- Info:
  - Commercial product (but prices are reasonable)
  - Company website: http://plpdf.com/plpdf.php
  - Current version: 4.1.0
  - Important feature: works even on XE!
  - Also includes PL/DOCX (but we didn’t evaluate it)

PL/PDF (2)

- Code sample

```
procedure InitDefault (l_blob OUT blob) is
begin
  -- create new doc (with default settings) and first page
  plpdf.Init;
  plpdf.NewPage();

  -- set font and add text
  plpdf.SetPrintFont(
    p_family => 'Arial', -- Font family: Arial
    p_style  => null, -- Font style: Regular
    p_size   => 12 -- Font size: 12
    );
  plpdf.PrintoutText(p_x => 20,p_y => 30,p_txt => 'Hello!');

  -- close document and create PDF as BLOB
  plpdf.SendDoc(p_blob => l_blob);
end;
```

PL/PDF (3)

- Our story:
  - Currently in production and used for multiple purposes:
    1. Server-generated emails with attached documents
    2. Highly customizable reports (where defining a single template is too difficult)
    3. Transmitting sensitive information (PDFs can be encrypted and password-protected)
  - Very limited support of fillable fields ☹
- Overall evaluation:
  - Same issue as PL/JSON – significant memory footprint
  - Very convenient substitution of extremely complex reporting tools
  - Somewhat time consuming to develop because every line has to be scripted.

IText (1)

- Used for:
  - Procedural constructor of PDF documents
  - 100% Java-based
- Info:
  - “Sort of” free (AGPL), but also there are commercial implementations (http://itextpdf.com/)
  - SourceForge: https://sourceforge.net/projects/itext/
  - Current version: 5.5.9

IText (2)

- Code sample

```
public class HelloWorldNarrow {
    /** Path to the resulting PDF file. */
    public static final String RESULT
        = "results/part1/chapter01/hello_narrow.pdf";
    public static void main(String[] args)
         throws DocumentException, IOException {
        // step 1
        // Using a custom page size
        Rectangle pagesize = new Rectangle(216f, 720f);
        Document document = new Document(pagesize, 36f, 72f, 108f, 180f);
        // step 2
        PdfWriter.getInstance(document, new FileOutputStream(RESULT));
        // step 3
        document.open();
        // step 4
        document.add(new Paragraph(
            "Hello World! Hello People! " +
            "Hello Sky! Hello Sun! Hello Moon! Hello Stars!"));
        // step 5
        document.close();
    }
}
```

IText (3)

- Our story:
  - Currently in production as a part of PL/SQL-based solution (no wrapped code)
  - Still under AGPL (or at least that's what I've been told ☺)
    - Creates PDF documents with fillable fields (in DEV)
    - Merges data to created PDFs (in PROD)
- Overall evaluation:
  - Very convenient and flexible
  - AGPL license may be a show-stopper for many organizations
  - Reasonably low performance impact (definitely cheaper than roundtrips to an application server)

XLSX (1)

- Our own solution
  - … yes, this time we reinvented the wheel, but:
    - XLSX is generic enough to be generated
    - All built-in computations are reasonably straightforward.
- Used for:
  - Procedural constructor of Excel reports in XLSX format (including multi-worksheet)
  - Also needed for reports that are wider than normal pages
  - 100% PL/SQL-based

XLSX (2)

- Code sample

```
--header
dbms_lob.append(v_XML_cl,vc_style_tx);
dbms_lob.append(v_XML_cl,
'<Worksheet ss:Name="EA NETRES EAD REPORT"><Table>
<Column ss:Width="200"/>
<Column ss:StyleID="RptDetail" ss:AutoFitWidth="0" ss:Width="42" ss:Span="9"/>
<Column ss:Index="12" ss:StyleID="RptDetail" ss:Width="50.25"/>
<Column ss:AutoFitWidth="0" ss:Width="5.25"/>
<Column ss:StyleID="RptDetail" ss:AutoFitWidth="0" ss:Width="42"
ss:Span="3"/>');

--Title and Header
dbms_lob.append(v_XML_cl,
'<Row><Cell ss:MergeAcross="16" ss:StyleID="Title">
    <Data ss:Type="String">EA NEC/EAD REPORT</Data></Cell>
</Row>
<Row><Cell ss:MergeAcross="16" ss:StyleID="RptDetail" >
    <Data ss:Type="String">Organization:'||v_org_tx||'</Data></Cell>
</Row>
<Row>
<Cell ss:MergeAcross="11" ss:StyleID="BorderB">
    <Data ss:Type="String">NEW ENLISTMENT CONTRACT STATUS</Data></Cell>
<Cell ss:Index="14" ss:MergeAcross="3" ss:StyleID="BorderB">
    <Data ss:Type="String">EAD STATUS</Data></Cell>
</Row>');
...
```

XLSX (3)

- Our story:
  - Currently in production in reporting module as alternative to PDF output (especially whenever PDF formatting would be unacceptable)
- Overall evaluation:
  - Lots of manual coding, but solved exactly what needed to be solved
  - Very functional because lots of people still prefer to do their own calculations in Excel (especially senior level management folks)

Summary

- It is possible to manage documents directly in the database
  - … although, it may often be counterintuitive.
- It is possible to have flexible reporting
  - … by utilizing repository-based solutions.

- There are even some third-party tools (commercial and free) to make your life simpler
  - … so you don't need to reinvent the wheel. ☺

Contact Information

- Michael Rosenblum – mrosenblum@dulcian.com
- Dulcian, Inc. website - www.dulcian.com
- Blog: wonderingmisha.blogspot.com