DATAGROUP
IT Services + Solutions

Interactive Report zu Excel und zurück

Vorstellung

- Henner Hucke
- Dipl. Ing. agr. an der Martin Luther Universität Halle
- seit 1994 Tätigkeitsschwerpunkt Datenbanken
- seit 1999 im Oracle Umfeld tätig
- Oracle RDBMS von 7.3 – 12c
- Java, C, SAP, ...
- Aktuell angestellt bei DATAGROUP Enterprise Services GmbH

DATAGROUP im Überblick

- DATAGROUP ist einer der führenden IT-Service-Anbieter Deutschlands*
- Unsere Kernkompetenz:
  We manage IT
- Wir betreuen über 600.000 IT-Arbeitsplätze weltweit
- Standorte in allen wichtigen Wirtschaftsregionen Deutschlands
- Rund 1.900 Mitarbeiter, davon rund 60 Azubis
- Wir denken anders.
  Wir arbeiten anders.
  Wir finden bessere Lösungen.

*Quelle: Lünendonk

Interactive Report zu Excel und zurück

- Ausgabe der Daten eines Interactive Reports aus APEX in eine Excel Datei
- Einlesen einer Excel Datei und Ausgabe der Daten als Interactive Report
- APEX + Oracle Datenbank als alleinige Werkzeuge
- Beispiel aus einem Kundenprojekt

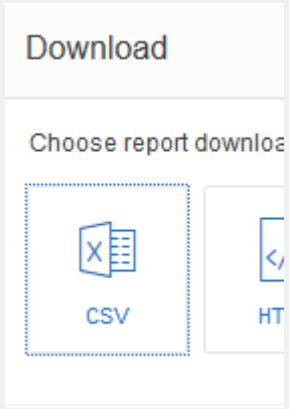
Motivation #1

- Anwender benutzen Excel und fragen daher nach Import- und Exportmöglichkeiten
- CSV-Export führt teilweise zu Verfälschungen
- CSV/ASCII Import erfordert Schulung des Anwenders

Motivation #2

1 - 17 of 17

	Teilenummer	
✎	FEB5035	Excel lügt hier ein wenig
✎	Z-700-XL	Z700 XL Frame



	A	B
1	Teilenummer	Teilebezeichnung
2	Feb 35	Excel lügt hier ein wenig
3	Z-700-XL	Z700 XL Frame

	A	B
1	Teilenummer	Teilebezeichnung
2	1145068	Excel lügt hier ein wenig
3	Z-700-XL	Z700 XL Frame

Zielstellung

- Feature für nativen Export/Import von Excel Dateien gesucht
- Neben APEX und der Oracle Datenbank möglichst keine weiteren Komponenten erforderlich

Ansatz Export

- SQL mit Sortierung und Filterkriterien aus dem Interaktiven Report extrahieren
- Ergebnismenge des SQL in XLSX Binary Large Object umwandeln
- Download des entstandenen BLOBs als Datei aus APEX heraus

SQL aus Interactive Report #1

- Package APEX_IR liefert die benötigte Funktionalität

```
function get_report (
    p_page_id          in number,
    p_region_id        in number,
    p_report_id        in number default
                       null,
    p_view             in varchar2
                       default
                       c_view_report
    )
    return t_report;
```

SQL aus Interactive Report #2

```
type t_report is record (
  sql_query      varchar2(32767),
  binds
    wwv_flow_plugin_util.t_bind_list);

type t_bind_list is table of t_bind index by
pls_integer;

type t_bind is record (
    name         varchar2(30),
    value        varchar2(32767) );
```

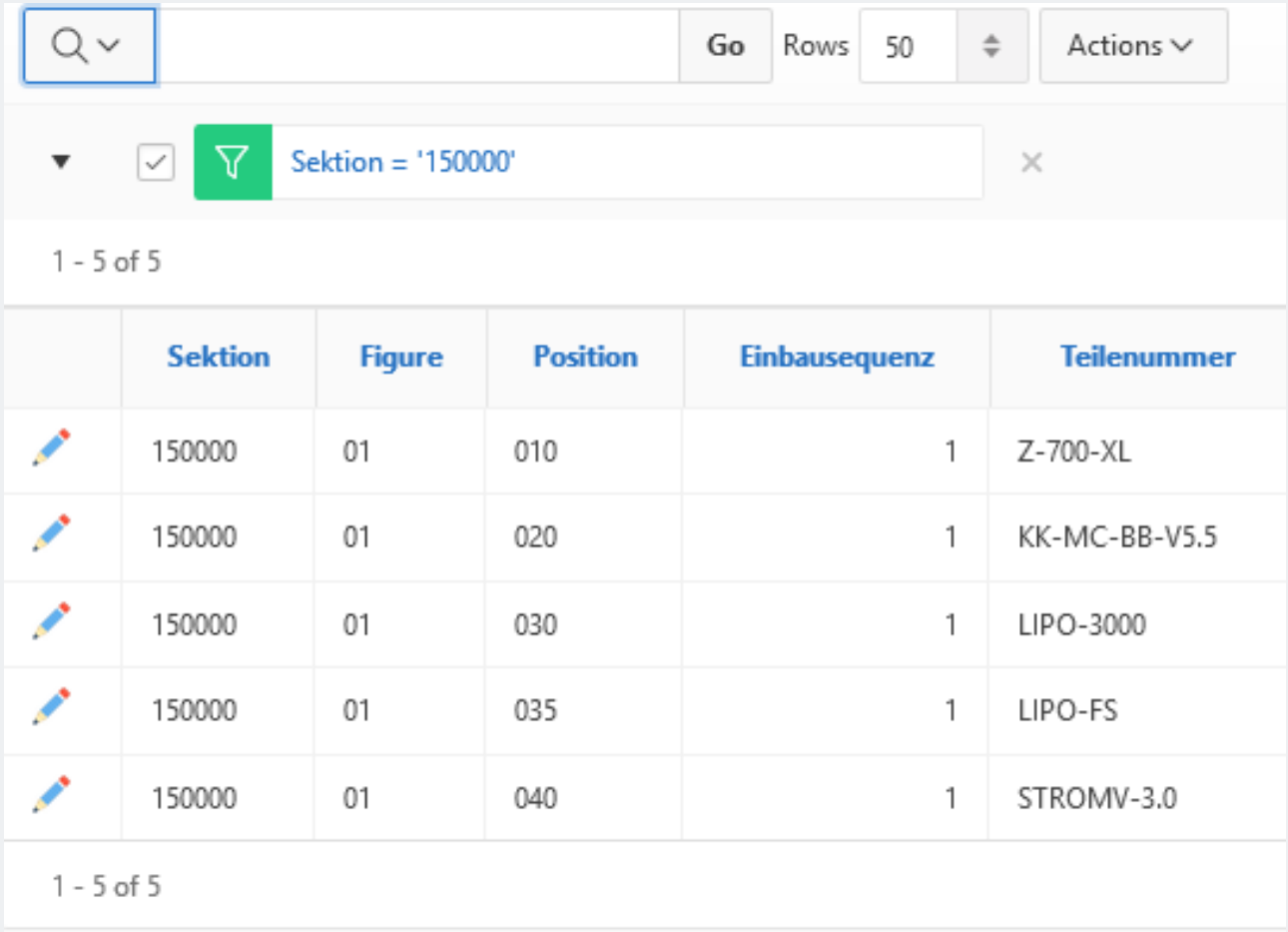
SQL aus Interactive Report #3

```
-- region_id bestimmen
SELECT region_id INTO l_region_id
FROM apex_application_page_regions
WHERE application_id = p_app_id
AND page_id = p_page_id
AND static_id = p_static_id
AND source_type = 'Interactive Report';
-- report_id bestimmen
l_report_id := apex_ir.get_last_viewed_report_id (
     p_page_id => p_page_id,
     p_region_id => l_region_id
);
```

SQL aus Interactive Report #4

```
l_report t_report;
...
-- region_id bestimmen
... into l_region_id ...
...
-- report_id bestimmen
l_report_id := ...
...
-- das Report-Objekt holen
l_report := apex_ir.get_report (
     p_page_id => p_page_id,
     p_region_id => l_region_id,
     p_report_id => l_report_id
);
```

SQL aus Interactive Report #5



SQL aus Interactive Report #5

```
select null as apxws_row_pk, "KAPITEL", "FIGURE",
"POSITION", ..., "R_ID" from ( select * from ( select
s.KAPITEL, s.FIGURE, s.POSITION, ... from STUECKLISTE
s, teilestamm t where s.teilenummer = t.teilenummer
order by 1,2,3,4 ) r where ("KAPITEL" =
:APXWS_EXPR_1) ) r where rownum <=
to_number(:APXWS_MAX_ROW_CNT)

-------- BINDS --------
NAME: APXWS_EXPR_1 VALUE:       150000
NAME: APXWS_MAX_ROW_CNT VALUE: 1000000
```

Ansatz Export

- SQL mit Sortierung und Filterkriterien aus dem Interaktiven Report extrahieren
- Ergebnismenge des SQL in XLSX Binary Large Object umwandeln
- Download des entstandenen BLOBs als Datei aus APEX heraus

Ergebnismenge in XLSX-BLOB

- BI-Publisher als zusätzliches Produkt
- Apache POI Framework plus Java Stored Procedure
- PL/SQL Packages AS_XLSX und AS_READ_XLSX von Anton Scheffer

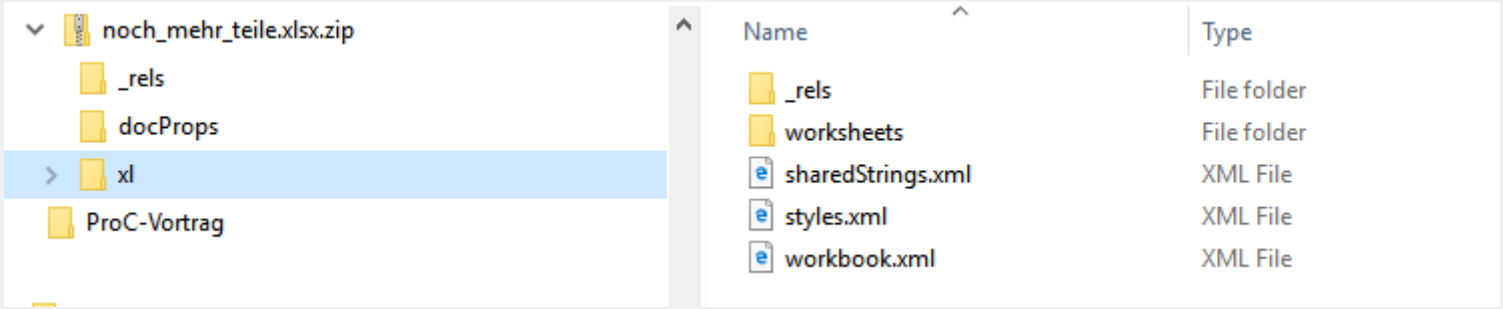
Package AS_XLSX #1

- SQL in Datenbank laden und es funktioniert
- MIT Lizenz
- Beachten: Abhängigkeit von UTL_FILE

  select * from user_dependencies where name in ('AS_READ_XLSX', 'AS_XLSX')

XLSX ist auch eine Art von XML

- XLSX Format ist ein gepacktes Verzeichnis mit XML Dateien
- Standardisiert und offen gelegt, siehe https://de.wikipedia.org/wiki/Office_Open_XML



Package AS_XLSX #2

```
procedure query2sheet (
    p_sql varchar2,
    p_column_headers boolean := true,
    p_directory varchar2 := null,
    p_filename varchar2 := null,
    p_sheet pls_integer := null
);

function finish return blob;
```

Package AS_XLSX #3

- Überladung der Procedure as_xlsx.query2sheet
- Anstelle des SQL-Statements wird ein Parameter vom Type apex_ir.t_report übergeben

```
as_xlsx.query2sheet (
  p_report apex_ir.t_report
) as ...
```

Package AS_XLSX #4

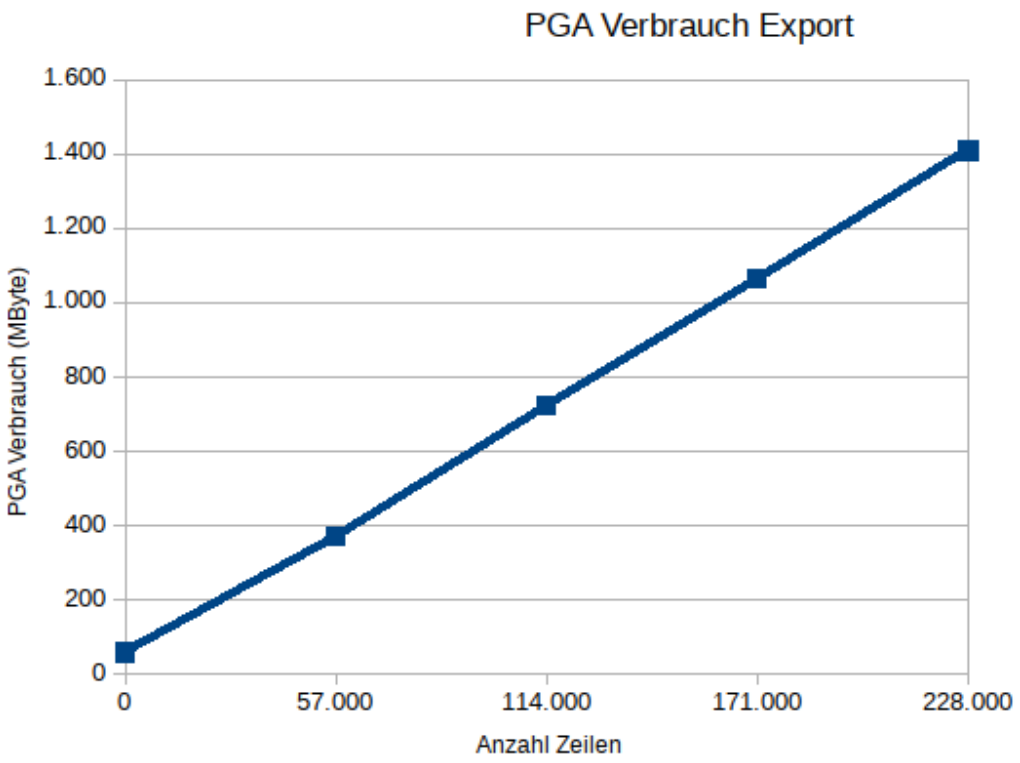
- Der folgende Code wird ergänzt

```
dbms_sql.parse(t_c,
     p_report.sql_query,
     dbms_sql.native);
for i in 1 .. p_report.binds.count loop
  dbms_sql.bind_variable( t_c,
    ':' || p_report.binds (i).name,
    p_report.binds (i).value
  );
end loop;
```

```
t_c := dbms_sql.open_cursor;
dbms_sql.parse( t_c, p_sql, dbms_sql.native );
dbms_sql.describe_columns2( t_c, t_col_cnt, t_desc_tab );
```

PGA-Verbrauch (Export)

- Daten aus Cursor werden vor Export in PL/SQL Tabellen abgelegt



Anzahl exportierter Zeilen all_objects	Session PGA (MByte)
0	59
57.000	371
114.000	724
171.000	1.065
228.000	1.409

Ansatz Export

- SQL mit Sortierung und Filterkriterien aus dem Interaktiven Report extrahieren

- Ergebnismenge des SQL in XLSX Binary Large Object umwandeln
- Download des entstandenen BLOBs als Datei aus APEX heraus

Download des BLOBS in APEX #1

- XLSX Blob wird „On the fly“ erzeugt und kann mit Hilfe der Packages htp, owa_util und wpg_docload zum Browser geschickt werden
- Neue APEX Seite mit On Load/Before Header Prozedur, welche einen HTML Header und den XLSX-BLOB zum Browser schickt
- danach wird die Verarbeitung gestoppt

Download des BLOBS in APEX #2

```
l_blob := ...
sys.htp.init;
sys.owa_util.mime_header(
  'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet',
  FALSE, 'UTF-8'
);
sys.htp.p('Content-length: ' ||
sys.dbms_lob.getlength( l_blob ));
sys.htp.p('Content-Disposition: inline; filename="'
|| l_filename || '"' );
sys.owa_util.http_header_close;
sys.wpg_docload.download_file(l_blob);
apex_application.stop_apex_engine;
```

Ansatz Export

- SQL mit Sortierung und Filterkriterien aus dem Interaktiven Report extrahieren

- Ergebnismenge des SQL in XLSX Binary Large Object umwandeln

- Download des entstandenen BLOBs als Datei aus APEX heraus

AS_READ_XLSX #1

- SQL in Datenbank laden und es funktioniert
- MIT-Lizenz
- Beachten: Abhängigkeit von UTL_FILE

```
select * from user_dependencies where name in
('AS_READ_XLSX', 'AS_XLSX')
```

AS_READ_XLSX #2

```
function read(
  p_xlsx blob,
  p_sheets varchar2 := null,
  p_cell varchar2 := null
) return tp_all_cells pipelined;
```

AS_READ_XLSX #3

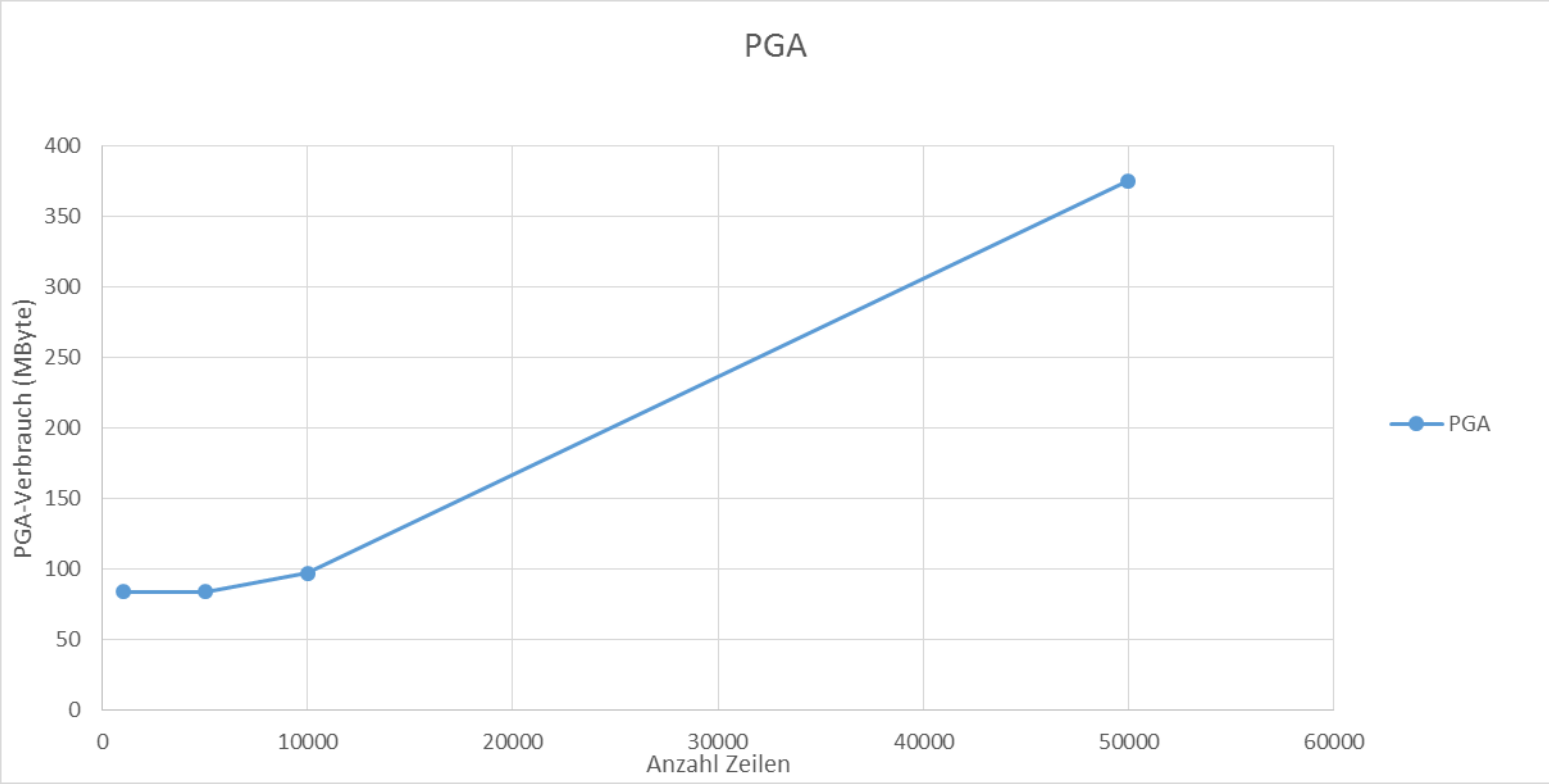
```
type tp_one_cell is record (
    sheet_nr number(2),
    sheet_name varchar(4000),
    row_nr number(10),
    col_nr number(10),
    cell varchar2(100),
    cell_type varchar2(1),
    string_val varchar2(4000),
    number_val number,
    date_val date,
    formula varchar2(4000)
);

type tp_all_cells is table of tp_one_cell;
```

AS_READ_XLSX #3

```
cursor csr_xlsx(cp_blob blob) is
   select sheet_nr, row_nr, col_nr,
     cell_type,
       decode(cell_type,
       'S', string_val,
       'N', to_char(number_val),
       'D', to_char(date_val, 'dd.mm.yyyy'),
       'N/A') as cell_value
    from table( as_read_xlsx.read( cp_blob ) )
    where sheet_nr = 1
    and row_nr > 1
    order by row_nr, col_nr;
```

PGA-Verbrauch (Import)

- auch hier ist der PGA-Bedarf nicht zu unterschätzen



Quellen

- **AS_XLSX und AS_READ_XLSX**
  - https://technology.amis.nl/2011/02/19/create-an-excel-file-with-plsql/
  - https://technology.amis.nl/2013/01/19/read-a-excel-xlsx-with-plsql/
- **APEX_IR**
  - https://docs.oracle.com/database/121/AEAPI/apex_ir.htm#AEAPI29337
  - http://deneskubicek.blogspot.de/2013/05/getting-interactive-report-query.html
- **Download Link**
  - http://davidsgale.com/apex-how-to-download-a-file

Fragen / Anregungen

Vielen Dank für Ihre Aufmerksamkeit!

DATAGROUP Enterprise Services GmbH
Auf den Tongruben 3 ▪ 53721 Siegburg
Tel. +49 1520 165 5394 ▪ henner.hucke@datagroup.de ▪ www.datagroup.de

Sie finden uns auch auf