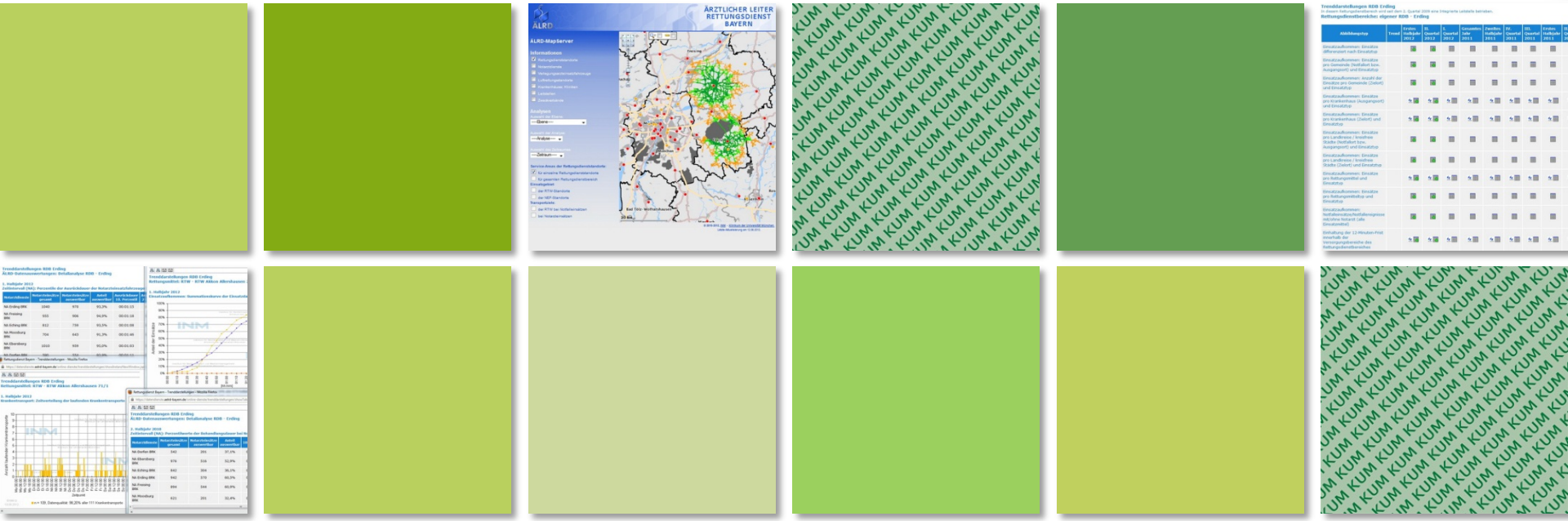
LMU KLINIKUM DER UNIVERSITÄT MÜNCHEN
CAMPUS INNENSTADT
INSTITUT FÜR NOTFALLMEDIZIN UND MEDIZINMANAGEMENT

XML / JSON IN ORACLE-RDBMS

Markus Geis

21.11.2018



AGENDA

- Institut für Notfallmedizin und Medizinmanagement – INM
- Oracle-Umgebung des INM
- XML in der Datenbank
- JSON in der Datenbank
- Fazit

AGENDA

- **Institut für Notfallmedizin und Medizinmanagement - INM**
- **Oracle-Umgebung des INM**
- **XML in der Datenbank**
- **JSON in der Datenbank**
- **Fazit**

KLINIKUM DER UNIVERSITÄT

- Campus Großhadern und Campus Innenstadt
- 2.250 Betten
- 500.000 Patienten ambulant, teilstationär und stationär
- 45 Kliniken / 60 Stationen
- 9.800 Beschäftigte:
  1.800 Mediziner / 3.400 Pflegekräfte / 4600 im medizinisch-technischen Bereich
- ca. 1 Mrd. Umsatz
- 5.000 Studenten-Wintersemester 2017/18

INSTITUT FÜR NOTFALLMEDIZIN

- Interdisziplinäre Forschungseinrichtung
- Qualitätsmanagement in der Medizin, Notfallmedizin und im Rettungswesen
- derzeit ca. 40 Mitarbeiter
- DWH: Einsatz-, medizinischen- / GEO-Daten
- HSC
- www.inm-online.de

AGENDA

- Institut für Notfallmedizin und Medizinmanagement - INM
- Oracle-Umgebung des INM
- XML in der Datenbank
- JSON in der Datenbank
- Fazit

ORACLE-UMGEBUNG INM

- 2 Knoten RAC Cluster
  - EE-Edition 12.2
    - Partitioning
    - Spatial
    - APEX
- Oracle-Linux 7.5
- Shared Filesystem NFS / Netapp-Metro-Cluster
- 768 GB HS pro Knoten

AGENDA

- Institut für Notfallmedizin und Medizinmanagement - INM
- Oracle-Umgebung des INM
- XML in der Datenbank
- JSON in der Datenbank
- Fazit

XML IN DER DATENBANK

- Warum XML ?
  - viele Files für die Datenübertragung kommen nicht mehr als “csv”-Datei / sondern als XML-Files
  - Die Daten müssen gelesen und in die entsprechenden Tabellen geschrieben werden

XML IN DER DATENBANK

- XML: Baumstruktur mit Elementen, Attributen, Werten
  - Attribute/Elemente werden mit <TAGS> gekennzeichnet (Anfang und Ende)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Externe_Qualitaetssicherung xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="xml_schema_C1.xsd">
  <Berichtsjahr>2015</Berichtsjahr>
  <IK_Krankenhaus>260913195</IK_Krankenhaus>
  <Standort>00</Standort>
  <Land>BY</Land>
  <Dokumentationsrate>
    <Leistungsbereich>
      <Kuerzel>09/1</Kuerzel>
      <Bezeichnung>Herzschrittmacherversorgung: Herzschrittmacher-Implantation</Bezeichnung>
      <Fallzahl>99</Fallzahl>
      <Dokumentationsrate>97,0</Dokumentationsrate>
    </Leistungsbereich>
    <Leistungsbereich>
      <Kuerzel>09/2</Kuerzel>
      <Bezeichnung>Herzschrittmacherversorgung:Herzschrittmacher-Aggregatwechsel</Bezeichnung>
      <Fallzahl>22</Fallzahl>
      <Dokumentationsrate>118,2</Dokumentationsrate>
    </Leistungsbereich>
```

XML IN DER DATENBANK

- seit Oracle 9i Release 2
- Features sind zusammengefaßt unter dem Namen **Oracle XML DB**
  - Bereitstellung von relationalen Daten im XML Format
  - Abfragen und Weiterverarbeiten von XML
    - viele Funktionen zur Verarbeitung von XML (String, Number, Boolean, usw.
- XML DB steht in jeder Edition zur Verfügung (XE, SE, EE)
- XML DB ist in Oracle Database 12c obligatorisch

```
SQL> r
  1  SELECT comp_name, version, status
  2*       FROM dba_registry WHERE upper(comp_name) LIKE '%XML%'

COMP_NAME                      VERSION                        STATUS
------------------------------ ------------------------------ -----------
Oracle XML Database            12.2.0.1.0                     VALID
```

XML IN DER DATENBANK

- eigener Datentyp für die Speicherung von XML in der Datenbank
  - XMLTYPE

```
CREATE TABLE LOAD_XML
( XML_DATA      SYS.XMLTYPE,
  FILE_ID       NUMBER,
  DATEINAME     VARCHAR2(50 BYTE),
  BERICHT       VARCHAR2(50 BYTE),
  BERICHTSJAHR  NUMBER
)
```

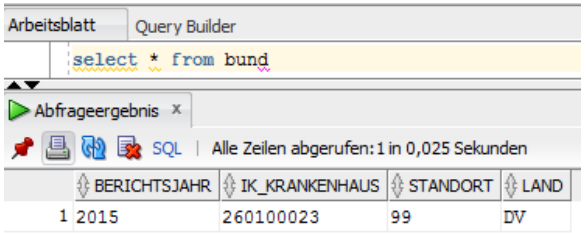
XML IN DER DATENBANK

- einlesen mit dem sql-loader

```
<?xml version="1.0" encoding="UTF-8"?>
<start>
   <Externe_Qualitaetssicherung>
        <Berichtsjahr>2015</Berichtsjahr>
        <IK_Krankenhaus>260100023</IK_Krankenhaus>
        <Standort>99</Standort>
        <Land>DV</Land>
   </Externe_Qualitaetssicherung>
</start>
```

```
[oracle@atria1 LHM]$ vi bund.ctl
load data
infile '/home/oracle/LHM/bund.xml' "str '</Externe_Qualitaetssicherung>'"
insert into table bund
fields (
     dummy1 filler       char(2000) terminated by "<Externe_Qualitaetssicherung>",
     Berichtsjahr        char(200) enclosed by "<Berichtsjahr>" and "</Berichtsjahr>"
     ,IK_Krankenhaus     char(200) enclosed by "<IK_Krankenhaus>" and "</IK_Krankenhaus>"
     ,Standort           char(200) enclosed by "<Standort>" and "</Standort>"
     ,Land               char(200) enclosed by "<Land>" and "</Land>",
     dummy2 filler       char(2000) terminated by "</start>"
)
```

```
sqlldr user/pw control=bund.ctl rows=100 bad=bund.bad
```



XML IN DER DATENBANK

- auslesen mit external-Table

```
<?xml version="1.0"?>
<ROWSET>
        <ROW>
                <EMPID>501</EMPID>
                <FNAME>JOHN</FNAME>
                <LNAME>DOE</LNAME>
                <SEX>M</SEX>
                <SSN>500000001</SSN>
                <SALARY>30000</SALARY>
                <DEPTNO>4001</DEPTNO>
        </ROW>
        <ROW>
                <EMPID>502</EMPID>
                <FNAME>JOHN</FNAME>
                <LNAME>SMITH</LNAME>
                <SEX>M</SEX>
                <SSN>500000002</SSN>
                <SALARY>40000</SALARY>
                <DEPTNO>4001</DEPTNO>
                </ROW>
</ROWSET>
```

```
create table xml_ext (
  rw  varchar2(100),
  empid integer,
  ename varchar2(100),
  .....
  deptno integer
) organization external (
  default directory xmldir
 access parameters (
   records delimited by '</ROW>'
  badfile xmldir:'emp1.bad'
  fields (
   rw    char(100) terminated by '<ROW>',
   empid char(100) enclosed by '<EMPID>' and '</EMPID>',
   ename char(100) enclosed by '<FNAME>' and '</FNAME>',
   ...
   deptno char(100) enclosed by '<DEPTNO>' and '</DEPTNO>'
  )
 )
 location ('emp1.xml')
) reject limit unlimited;
```

Arbeitsblatt | Query Builder

```
select * from xml_ext
where empid=502
```

Abfrageergebnis

SQL | Alle Zeilen abgerufen: 1 in 0,025 Sekunden

	RW	EMPID	ENAME	LNAME	SEX	SSN	SALARY	DEPTNO
1		502	JOHN	SMITH	M	500000002	40000	4001

XML IN DER DATENBANK

- auslesen direkt aus der Datei
  - EXTRACT (deprecated 12c)
    - gibt ein XML Fragment zurück

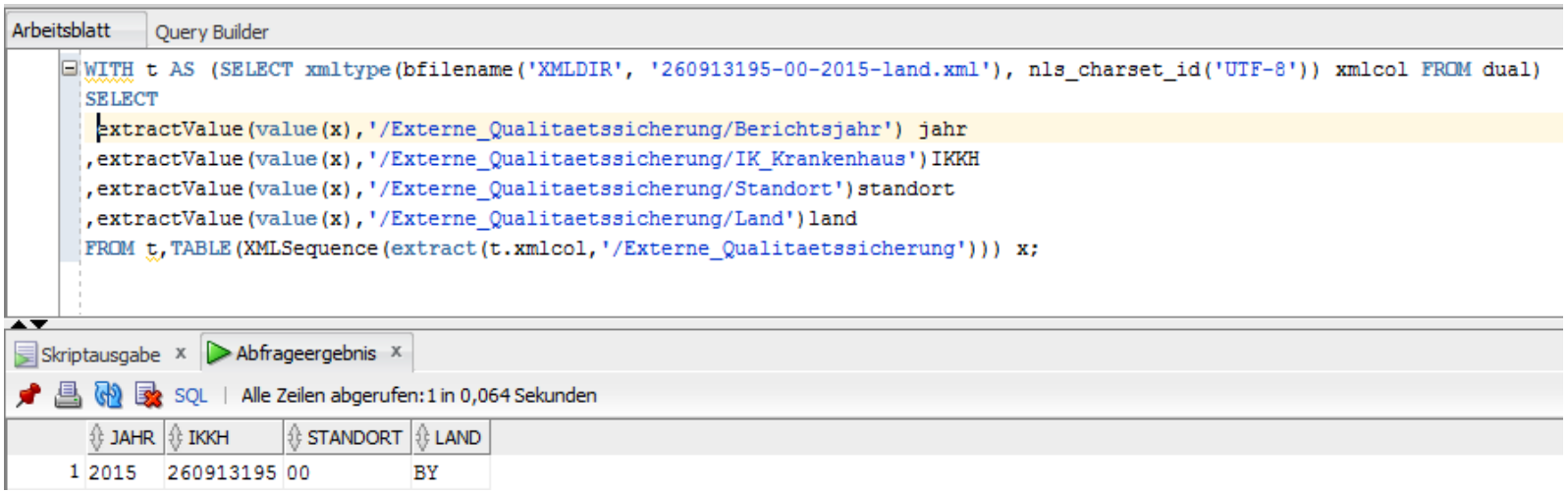
```
EXTRACT(warehouse_spec, '/Warehouse/Docks') "Number of Docks“
```

```
WAREHOUSE_NAME            Number of Docks
------------------------- -------------------------
New Jersey
San Francisco             <Docks>1</Docks>
```

  - EXTRACTVALUE (deprecated 12c)
  - XMLTABLE
    - zerlegt XML-Infos in Zeilen und Spalten / relationale Tabelle

XML IN DER DATENBANK

- EXTRACTVALUE: gibt den einzelnen Wert eines XML Fragments zurück / XML-Instance + Xpath-String + Variable
  absoluter XPath_string initial slash (/)
  relativer XPath_string ohne initial slash ('*/Personalerfassung)



XML IN DER DATENBANK

- Verarbeitung innerhalb der Datenbank
  - auslesen aus einer Tabelle mit dem Datentype “XMLTYPE”

1. Table für Aufnahme der Daten erzeugen:

```
CREATE TABLE LOAD_XML
( XML_DATA      SYS.XMLTYPE,
  FILE_ID       NUMBER,
  DATEINAME     VARCHAR2(50 BYTE),
  BERICHT       VARCHAR2(50 BYTE),
  BERICHTSJAHR  NUMBER
)
```

2. Directory auf dem Server erzeugen:

```
CREATE OR REPLACE DIRECTORY XMLDIR AS '/home/oracle/LHM';
```

3. Daten in Tabelle einlesen:

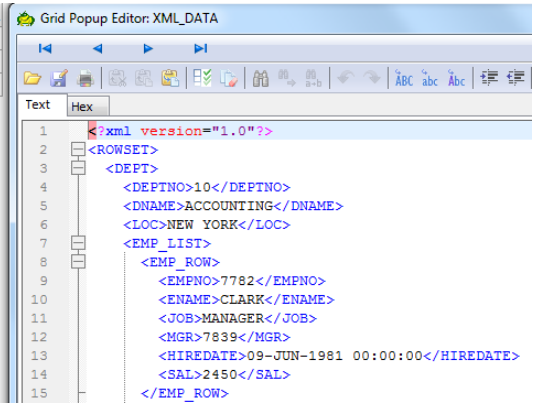
```
insert into baby_lhm.load_xml values ((XMLType(bfilename('XMLDIR','260913195-00-
2015-xml.xml'),nls_charset_id('UTF-8'))),21,'260913195-00-2015-xml.xml','AB',2015);
```

XML IN DER DATENBANK

- INSERT into Table

```
insert into baby_lhm.load_xml values (
        '<?xml version="1.0"?>
<ROWSET>
  <DEPT>
    <DEPTNO>10</DEPTNO>
    <DNAME>ACCOUNTING</DNAME>
    <LOC>NEW YORK</LOC>
    <EMP_LIST>
      <EMP_ROW>
        <EMPNO>7782</EMPNO>
        <ENAME>CLARK</ENAME>
        <JOB>MANAGER</JOB>
        <MGR>7839</MGR>
        <HIREDATE>09-JUN-1981 00:00:00</HIREDATE>
        <SAL>2450</SAL>
      </EMP_ROW>
      <EMP_ROW>
        <EMPNO>7839</EMPNO>
        <ENAME>KING</ENAME>
        <JOB>PRESIDENT</JOB>
        <HIREDATE>17-NOV-1981 00:00:00</HIREDATE>
        <SAL>5000</SAL>
      </EMP_ROW>
      <EMP_ROW>
        <EMPNO>7934</EMPNO>
        <ENAME>MILLER</ENAME>
        <JOB>CLERK</JOB>
        <MGR>7782</MGR>
        <HIREDATE>23-JAN-1982 00:00:00</HIREDATE>
        <SAL>1300</SAL>
      </EMP_ROW>
    </EMP_LIST>
  </DEPT>
</ROWSET>',23,'test.xml','X',2018
);
```

XML_DATA	FILE_ID	DATEINAME	BERICHT	BERICHTSJAHR
(HUGECLOB)	21	260913195-00-2015-xml.xml	AB	2015
(HUGECLOB)	22	260913195-00-2015-land.xml	AB	2015
(HUGECLOB)	23	test.xml	X	2018



XML IN DER DATENBANK

- auslesen aus einer Tabelle mit “XMLTABLE”

```
SELECT xt.*
FROM  baby_lhm.load_xml x,
       XMLTABLE('/Externe_Qualitaetssicherung'
         PASSING x.xml_data
         COLUMNS
           Berichtsjahr number PATH 'Berichtsjahr',
           IKKH number PATH  'IK_Krankenhaus',
           Standort VARCHAR2(10) PATH 'Standort',
           Land VARCHAR2(10) PATH 'Land'
         ) xt
```

	BERICHTSJAHR	IKKH	STANDORT	LAND
1	2015	260913195	00	BY

XML IN DER DATENBANK

- auslesen aus einer Tabelle “master-detail” / xmltable / passing

```
insert into baby_lhm.load_xml values (
        '<?xml version="1.0"?>
<ROWSET>
  <DEPT>
    <DEPTNO>10</DEPTNO>
    <DNAME>ACCOUNTING</DNAME>
    <LOC>NEW YORK</LOC>
    <EMP_LIST>
      <EMP_ROW>
        <EMPNO>7782</EMPNO>
        <ENAME>CLARK</ENAME>
        <JOB>MANAGER</JOB>
        <MGR>7839</MGR>
        <HIREDATE>09-JUN-1981 00:00:00</HIREDATE>
        <SAL>2450</SAL>
      </EMP_ROW>
      <EMP_ROW>
        <EMPNO>7839</EMPNO>
        <ENAME>KING</ENAME>
        <JOB>PRESIDENT</JOB>
        <HIREDATE>17-NOV-1981 00:00:00</HIREDATE>
        <SAL>5000</SAL>
      </EMP_ROW>
      <EMP_ROW>
        <EMPNO>7934</EMPNO>
        <ENAME>MILLER</ENAME>
        <JOB>CLERK</JOB>
        <MGR>7782</MGR>
        <HIREDATE>23-JAN-1982 00:00:00</HIREDATE>
        <SAL>1300</SAL>
      </EMP_ROW>
    </EMP_LIST>
  </DEPT>
</ROWSET>',23,'test.xml','X',2018
);
```

```
1  SELECT x1.deptno,x1.loc, x2.ename, x2.job
2  FROM baby_lhm.load_xml x,
3         XMLTABLE('/ROWSET/DEPT'
4           PASSING x.xml_data
5           COLUMNS
6             DEPTNO        number  PATH 'DEPTNO',
7             LOC           varchar2(30) PATH 'LOC',
8             abteilung     xmltype path 'EMP_LIST/EMP_ROW'
9              ) x1
10        ,XMLTABLE('/EMP_ROW'
11          PASSING x1.abteilung
12          COLUMNS
13            ENAME varchar2(20) PATH 'ENAME',
14            JOB varchar2(20) PATH 'JOB'
15             ) x2
16 where file_id=24
```

Data Grid

Messages | Data Grid | Trace | DBMS Output | Query Viewer | Explain Plan | Script Output

DEPTNO	LOC	ENAME	JOB
10	NEW YORK	CLARK	MANAGER
10	NEW YORK	KING	PRESIDENT
10	NEW YORK	MILLER	CLERK

XML IN DER DATENBANK

- “master-detail” / xmltable
- Problematisch: falsche Spaltennamen / outer-join

```
1  SELECT x1.deptno,x1.loc, x2.ename, x2.job
2  FROM baby_lhm.load_xml x,
3         XMLTABLE('/ROWSET/DEPT'
4           PASSING x.xml_data
5           COLUMNS
6             DEPTNO        number  PATH 'DEPTNO',
7             LOC           varchar2(30) PATH 'LOC1',
8             abteilung     xmltype path 'EMP_LIST/EMP_ROW'
9              ) x1
10        ,XMLTABLE('/EMP_ROW'
11          PASSING x1.abteilung
12          COLUMNS
13            ENAME varchar2(20) PATH 'ENAME',
14            JOB varchar2(20) PATH 'JOB'
15             ) (+)x2
16 where file_id=24
```

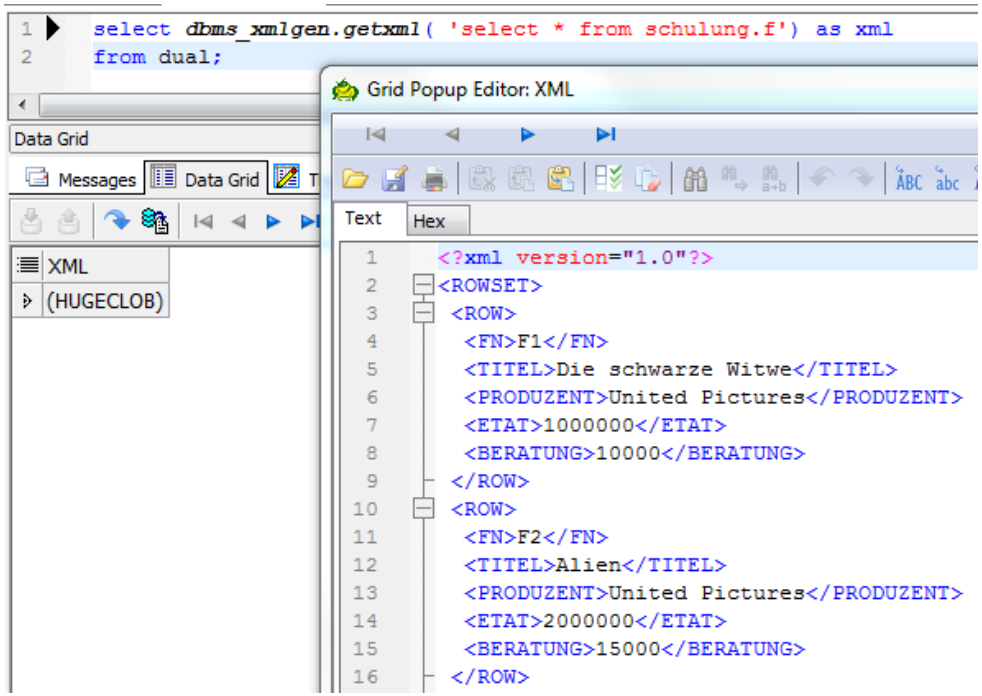
Data Grid

Messages | Data Grid | Trace | DBMS Output | Query Viewer | Explain Plan | Sc

DEPTNO	LOC	ENAME	JOB
10		CLARK	MANAGER
10		KING	PRESIDENT
10		MILLER	CLERK
20			

XML IN DER DATENBANK

- Tabelle in XML auslesen
  - dbms_xmlgen.getxml (Rückgabe: CLOB)
  - dbms_xmlgen.getxmltype (Rückgabe: XMLType)



XML IN DER DATENBANK

- Tabelle in XML und in eine Datei auslesen
  - DBMS_XSLPROCESSOR.clob2file()
  - DBMS_XMLDOM.writetofile()
    - sys_refcursor / RETURNS an ARRAY
    - DBMS_XMLDOM package / access XMLType (DOMDOCUMENT)
      - NEWDOMDOCUMENT
      - WRITETOFILE

```
declare
        refc sys_refcursor;
        xdoc DBMS_XMLDOM.DOMDocument;
begin
   open refc for
      SELECT * from schulung.f ;
      xdoc := DBMS_XMLDOM.NewDOMDocument(xmltype( refc ));
      DBMS_XMLDOM.WRITETOFILE(xdoc, 'XMLDIR/xml2.xml','UTF-8');
end;
```

```
[oracle@atria1 LHM]$ less xml2.xml
<?xml version="1.0" encoding="UTF-8"?>
<ROWSET>
  <ROW>
    <FN>F1</FN>
    <TITEL>Die schwarze Witwe</TITEL>
    <PRODUZENT>United Pictures</PRODUZENT>
    <ETAT>1000000</ETAT>
    <BERATUNG>10000</BERATUNG>
  </ROW>
  <ROW>
    <FN>F2</FN>
    <TITEL>Alien</TITEL>
    <PRODUZENT>United Pictures</PRODUZENT>
    <ETAT>2000000</ETAT>
    <BERATUNG>15000</BERATUNG>
  </ROW>
```

XML IN DER DATENBANK

- Update XML
  - das gesamte XML-Dokument
  - einzelne XML Nodes / mit der 12.2 wurden diverse Funktionen deprecated: z.B. appendChildXML, deleteXML, updateXML

  - Ersatz durch „XMLQuery“

AGENDA

- Institut für Notfallmedizin und Medizinmanagement - INM
- Oracle-Umgebung des INM
- XML in der Datenbank
- JSON in der Datenbank
- Fazit

JSON IN DER DATENBANK

- JSON: JavaScript Object Notation
- strukturieren von Daten (einfacher als XML ->TAG)
  - Platzverbrauch / kompaktes Datenformat
  - <zeit>06:11</zeit>
- JSON selbst gültiges Javascript
- Zugriff über normalen Attributzugriff
- Datenaustauschs zwischen Anwendungen

JSON IN DER DATENBANK

- Beispiel:

```
{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Waehrung": "EURO",
  "Inhaber":
  {
    "Name": "Mustermann",
    "Vorname": "Max",
    "maennlich": true,
    "Hobbys": ["Reiten", "Golfen", "Lesen"],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}
```

JSON IN DER DATENBANK

- Format

Objektart	Format	Beispiel
String	Text innerhalb von Anführungszeichen	"text"
Zahlenwert	Folge von 0 bis 9, optional mit Dezimalpunkt und/oder Exponent	1.6
boolscher Wert	true oder false	true
Liste / Array	Aufgelistete Werte innerhalb von eckigen Klammern, durch Komma getrennt	[wert, wert2, ...]
Objekt / "assoziativer Array"	Index/Wert-Zuordnung mithilfe eines Doppelpunkts innerhalb von geschweiften Klammern, durch Komma getrennt	{"attname": wert, "attname2": wert2, ...}

JSON IN DER DATENBANK

- Differenz: XML <-> JSON

```
<parkdaten>¶
···<zeit>06:11</zeit>¶
···<parkhaus>P5</parkhaus>¶
···<personalnummer>10058975</personalnummer>¶
</parkdaten>¶
```

```
{¶
··zeit:·"06:11",¶
··parkhaus:·"P5",¶
··personalnummer:·10058975¶
}¶
```

- <parkdaten -> es handelt sich um Parkdaten
- XML im Gegensatz zu JSON deklarativ
- Schnittstelle klar definiert -> JSON
- Unklare Datenlage -> Deklarationen von XML

JSON IN DER DATENBANK

- Oracle: JSON
  - Tabelle erstellen / kein eigener Datentyp

```
create table json_documents
(    id                  number not null,
     daten               clob,
     CONSTRAINT json_documents_pk PRIMARY KEY (id),
     constraint check_json check (daten is json))
```

```
create table json_documents
(    id                  number not null,
     daten               clob,
     CONSTRAINT json_documents_pk PRIMARY KEY (id),
     constraint check_json check (daten is json(STRICT)))
```

JSON IN DER DATENBANK

- Oracle: JSON
  - Insert JSON document aus einer Transaktion

```
INSERT INTO json_documents (id, daten)
          VALUES (1,
             '{
              "FirstName"      : "John",
              "LastName"       : "Doe",
               "Job"            : "Clerk",
               "Address"        : {
                                  "Street"   : "99 My Street",
                                  "City"     : "My City",
                                  "Country"  : "UK",
                                  "Postcode" : "A12 34B"
                                 },
               "ContactDetails" : {
                                  "Email"    : "john.doe@example.com",
                                  "Phone"    : "44 123 123456",
                                  "Twitter"  : "@johndoe"
                                 },
               "DateOfBirth"    : "01-JAN-1980",
               "Active"         : true
          }');
```

JSON IN DER DATENBANK

- Oracle: JSON
  - SQLLOAD (local oder Server)

```
File: trans.ctl
load data into table json_load
terminated by ','
(  trans_id    sequence(max,1),
   fname       filler char(80),
   trans_body  lobfile(fname) terminated by EOF)
File: trans.data
/load_json/trans1.txt
/load_json/trans2.txt
/load_json/trans3.txt
```

JSON IN DER DATENBANK

- Oracle: JSON
  - External Table / XMLTAG)

```
CREATE TABLE ext_xml2 (xml_text VARCHAR2(2000) )
ORGANIZATION external
(
  TYPE oracle_loader
  DEFAULT DIRECTORY home
  ACCESS PARAMETERS
  (
    RECORDS
    XMLTAG ("DepartmentName", "employeeName" )
    READSIZE 1024 SKIP 0 FIELDS NOTRIM
    MISSING FIELD VALUES ARE NULL
  )
  location ('emp.xml')
)
REJECT LIMIT UNLIMITED
/
```

JSON IN DER DATENBANK

- Oracle: JSON
  - Oracle: CHECK-JSON document

```
UPDATE json_documents a
SET    a.daten = '{"FirstName" : "Invalid Document"'
WHERE  a.daten.FirstName = 'Jayne';
```

```
Fehlerbericht -
ORA-02290: CHECK-Constraint (CORETEST.CHECK_JSON)
verletzt
```

JSON IN DER DATENBANK

- Oracle: JSON
  - Oracle: Select-JSON document

```
SELECT *
FROM   json_documents a
```

Skriptausgabe | Abfrageergebnis

SQL | Alle Zeilen abgerufen:2 in 0,004 Sekunden

	ID	DATEN
1	1	{ "FirstName" : "John", "LastName" : "Doe", ...
2	2	{ "FirstName" : "Jayne", "LastName" : "Do...

JSON IN DER DATENBANK

- Oracle: JSON
  - Oracle: Select-JSON document

```
SELECT a.daten.ContactDetails
FROM   json_documents a;
```

	CONTACTDETAILS
1	{"Email":"john.doe@example.com","Phone":"44 123 123456","Twitter":"@johndoe"}
2	{"Email":"jayne.doe@example.com","Phone":""}

JSON IN DER DATENBANK

- Oracle: JSON
  - is JSON / JSON_VALUE (extrahiert einzelne, skalare Werte)
    - JSON_QUERY: "Gegenstück" zu JSON_VALUE: JSON-Fragmente aus dem Dokument auszuschneiden
    - JSON_VALUE using NULL ON ERROR returns NULL for non-JSON data

```
SELECT JSON_VALUE(a.data, '$.FirstName') AS first_name
FROM json_documents_no_constraint a;
```

	FIRST_NAME
1	John
2	(null)

JSON IN DER DATENBANK

- Oracle: JSON
  - Is JSON
  - Nur Zeilen mit „JSON“ werden ausgegeben

```
SELECT JSON_VALUE(a.data, '$.FirstName') AS first_name
FROM json_documents_no_constraint a
WHERE a.data IS JSON;
```

	FIRST_NAME
1	John

JSON IN DER DATENBANK

- Oracle: JSON_VALUE
  - Ausgabe eines Wertes , keine z.B. nested records or arrays

```
SELECT JSON_VALUE(a.daten,
'$.ContactDetails') AS contact_details
FROM json_documents a
ORDER BY 1;
```

	CONTACT_DETAILS
1	(null)
2	(null)

```
SELECT JSON_VALUE(a.daten,
'$.ContactDetails' ERROR ON ERROR) AS
contact_details
FROM   json_documents a
ORDER BY 1;
```

```
ORA-40456: JSON_VALUE wurde als nicht-skalarer Wert ausgewertet
40456. 00000 - "JSON_VALUE evaluated to non-scalar value"
*Cause:    The provided JavaScript Object Notation (JSON) path expression
           selected a non-scalar value.
*Action:   Correct the JSON path expression or use JSON_QUERY.
```

JSON IN DER DATENBANK

- Oracle: JSON
  - JSON_EXISTS / Phone-Element ist vorhanden, aber hat ein null value

```
SELECT a.daten.FirstName,
       a.daten.LastName,
       a.daten.ContactDetails.Email AS Email
FROM   json_documents a
WHERE  JSON_EXISTS(a.daten.ContactDetails, '$.Phone' FALSE ON ERROR)
AND    a.daten.ContactDetails.Phone IS NULL;
```

	FIRSTNAME	LASTNAME	EMAIL
1	Jayne	Doe	jayne.doe@example.com

JSON IN DER DATENBANK

- Oracle: JSON from Table
- Die meisten Verbesserung kamen erst ab/mit der Version 12.2
  - JSON_OBJECT
  - JSON_ARRAY
  - JSON_ARRAYAGG

JSON IN DER DATENBANK

- Oracle: JSON from Table
  - JSON_OBJECT
  - JSON Ausgabe für jeden Datensatz

```
SELECT JSON_OBJECT ('id2' VALUE cust_id,'name' VALUE
(first ||' '|| last),
'joined' VALUE join_date)
FROM customers;
```

{"id2":1,"name":"Eric Cartman","joined":"2017-07-10T16:26:15"}
{"id2":2,"name":"Kenny McCormick","joined":"2017-07-10T16:26:15"}
{"id2":3,"name":"Kyle Brofloski","joined":"2017-07-10T16:26:15"}
{"id2":4,"name":"Stan Marsh","joined":"2017-07-10T16:26:15"}

JSON IN DER DATENBANK

- Oracle: JSON from Table
  - JSON_ARRAY
  - JSON ein ARRAY für jeden Datensatz

```
SELECT JSON_ARRAY(first, last)
FROM customers;
```

```
["Eric","Cartman"]
["Kenny","McCormick"]
["Kyle","Brofloski"]
["Stan","Marsh"]
```

JSON IN DER DATENBANK

- Oracle: JSON from Table
  - JSON_ARRAYAGG
  - ARRAY für variable Größen mit mehreren Datensätzen

```
SELECT JSON_ARRAYAGG(
       JSON_OBJECT('id' VALUE item_id,
                   'name' VALUE name,
                   'quantity' VALUE stock_quantity))
FROM items
WHERE stock_quantity > 10;
```

[{"id":345,"name":"Border Patrol Costume","quantity":20},{"id":429,"name":"Air Guitar","quantity":14}]

JSON IN DER DATENBANK

- Oracle: JSON from Table
  - Update JSON
    - geändertes Dokument kann nur komplett auf die DB zurückgeschrieben werden
    - JSON Daten liegen in Standard SQL Datentypen
      VARCHAR2 /LOB
    - Standard Oracle API‘s

AGENDA

- Institut für Notfallmedizin und Medizinmanagement - INM
- Oracle-Umgebung des INM
- XML in der Datenbank
- JSON in der Datenbank
- Fazit

FAZIT

- das INM benutzt XML hauptsächlich für Datenlade-Operationen
  - Klinikdaten
  - Reanimationspuppen / Medizin-Geräte
  - Einsatzdaten

- JSON wird hauptsächlich genutzt um Informationen zwischen verschiedenen Applikation auszutauschen (GEO-Server)

- Oracle RDBMS verfügt über eine große Funktionalität bei der Verarbeitung von XML und JSON (auch mit PL/SQL)

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!

M. Geis

Klinikum der Universität München

Institut für Notfallmedizin
und Medizinmanagement

Phone: 089 / 4400-57101

E-Mail: markus.geis@med.uni-muenchen.de

Internet: www.inm-online.de