

```
set long 10000;  
set linesize 100;  
set pagesize 1000;  
column Adresse format A70;
```

```
drop table Adr_List;
```

```
create table Adr_List  
  ( lfd_Nr number (10),  
    Adresse xmltype );
```

wohlgeformte Dokumente einfügen

```
insert into Adr_List values(10,xmltype('<Adr><Name>Karin</Name><Ort>Hamburg</Ort></Adr>'));
insert into Adr_List values(20,xmltype('<Adr><Name>Theo</Name><Ort>Hamburg </Ort></Adr>'));
insert into Adr_List values(30,xmltype('<Adr><Name>Peter</Name><Ort>Hamburg</Ort></Adr>'));
insert into Adr_List values(40,xmltype('<Adr><Name>Peter</Name><Ort>Bremen</Ort></Adr>'));
insert into Adr_List values(50,xmltype('<Adr><Name>Petra</Name><Ort>Bremen</Ort></Adr>'));
insert into Adr_List values(60,xmltype('<Adr><Name>Karin</Name><Ort>Hannover</Ort></Adr>'));
insert into Adr_List values(70,xmltype('<Adr><Name>Karin</Name>
<Ort Bundesland="Niedersachsen">Hannover</Ort></Adr>'));
insert into Adr_List values(80,xmltype('<Adr><Name>Anna</Name>
<Ort Bundesland="Hessen">Frankfurt</Ort></Adr>'));
insert into Adr_List values(90,xmltype('<Adr><Name>Martina</Name>
<Ort Bundesland="Hessen">Kassel</Ort </Adr>'));
insert into Adr_List values(100,xmltype('<Adr><Name>Barbara</Name>
<Ort>Hannover</Ort><Ort>Hamburg</Ort></Adr>'));
insert into Adr_List values(110,xmltype('<Adr><Name>Wibke</Name><Ort/></Adr>'));
insert into Adr_List values(120,xmltype('<Adr><Name>Paula</Name></Adr>'));
commit;
```

nicht wohlgeformte Dokumente einfügen (ergibt Fehlermeldung)

```
insert into Adr_List
values(140,xmltype('<adr><Name>Karin</Name><Ort>Hannover</Ort></Adr>'));
```

Inhalt der Tabelle anzeigen lassen

```
select * from Adr_List;

select a.Adresse.getClobVal() from Adr_List a;

select * from Adr_List a where existsNode (a.Adresse, '/Adr[Ort="Bremen"]') = 1 and lfd_Nr > 20;

select * from Adr_List a where existsNode (a.Adresse, '/Adr[count(Ort)>1]') = 1;

select * from Adr_List a where existsNode (a.Adresse, '/Adr/Ort/@Bundesland') = 1;

select * from Adr_List a where existsNode (a.Adresse, '/Adr/Ort[@Bundesland="Hessen"]') = 1;

select * from Adr_List a where extractValue(a.Adresse, '/Adr/Name') ='Anna';

select count(*) from Adr_List a;

select count(*) from Adr_List a where existsnode(a.Adresse, '/Adr/Ort') = 1;

select count(*) from Adr_List a where existsnode(a.Adresse, '/Adr[Ort="Hamburg"]') = 1;
```

```
select * from Adr_List where lfd_Nr = 30;
```

ein ganzes XML Dokument verändern

```
update Adr_List
```

```
  set Adresse = XMLTYPE
```

```
  ('<Adr><Name>Gert</Name><Ort>Frankfurt</Ort></Adr>')
```

```
  where lfd_Nr = 30;
```

```
select * from Adr_List where lfd_Nr = 30;
```

Teil eines XML Dokument verändern

```
update Adr_List
```

```
  set Adresse = UPDATEXML
```

```
  (Adresse, '/Adr/Name/text()', 'Hugo')
```

```
  where lfd_Nr = 30;
```

```
select * from Adr_List where lfd_Nr = 30;
```

weitere Spalte ergänzen und zusätzlichen Datensatz eingeben

```
column Adresse format A40;  
column Hobbies format A40;
```

```
alter table Adr_List add (Hobbies XMLType);
```

insert into Adr_List

```
  values(10,  
        xmltype('<Adr><Name>Karin</Name><Ort>Hamburg</Ort></Adr>'),  
        xmltype('<Hobby>Lesen, Theater, Tennis</Hobby>'));
```

```
select a.lfd_Nr,  
       a.Adresse as Adresse,  
       a.Hobbies as Hobbies  
from Adr_List a where lfd_Nr=10;
```

Spalte wieder entfernen

```
alter table Adr_List drop (Hobbies);  
commit;
```

XML Type Tabelle erstellen und mit Inhalt füllen

```
drop table Adr;
```

```
create table Adr of xmltype;
```

```
insert into Adr values(xmltype('<Adr><Name>Theo</Name><Ort>Bremen</Ort></Adr>'));
```

```
insert into Adr values(xmltype('<Adr><Name>Karin</Name><Ort>Hannover</Ort></Adr>'));
```

Inhalt der Tabelle anzeigen lassen

```
select * from Adr;
```

```
select sys_nc_rowinfo$ as Adressen from Adr;
```

```
select * from Adr a  
  where existsNode (a.sys_nc_rowinfo$, '/Adr[Ort="Bremen"]') = 1;
```

Exkurs: Validierung gegen eine interne DTD

Eingabe eines gültigen XML Dokumentes

column Adresse format A80;

```
insert into Adr_List values(80,
  xmltype(' <!DOCTYPE Adr [
    <!ELEMENT Adr (Name, Ort)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT Ort (#PCDATA)>
  ]>
  <Adr><Name>Uwe</Name><Ort>Hannover</Ort></Adr>'));
```

```
select * from Adr_List where lfd_Nr=80;
```

Eingabe eines nicht gültigen XML Dokumentes

```
insert into Adr_List values(80,
  xmltype(' <!DOCTYPE Adr [
    <!ELEMENT Adr (Name, Ort)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT Ort (#PCDATA)>
  ]>
  <adr><Name>Uwe</Name><Ort>Hannover</Ort></adr>'));
```

Erzeugung von relationalen Strukturen aus XML Dokumenten

- | column Name format A20;
- | column Ort format A20;
- | column Bundesland format A20;
- | column Mitarbeiterliste format A100;

mit select statements auf Tabelle mit XML Type Informationen zugreifen

```
select a.Adresse from Adr_List a;
```

```
select lfd_nr, extract(a.Adresse,'/Adr/Name') as Name from Adr_List a;
```

```
select lfd_nr, extract(a.Adresse,'/Adr/Name/text()') as Name from Adr_List a;
```

```
select lfd_nr, extract(a.Adresse,'/Adr/Name/text()').getStringVal() as Name from Adr_List a;
```

```
select lfd_nr,  
       extractValue(a.Adresse,'/Adr/Name') as Name,  
       extractValue(value(b),'Ort') as Ort  
from Adr_List a,  
       table(xmlsequence( extract(a.Adresse, '/Adr/Ort'))) b  
order by Name;
```

relationalen View auf XML Type Strukturen erzeugen

```
drop view Pers_v;  
  
create view Pers_v as  
  select lfd_nr,  
         extractValue(a.Adresse,'/Adr/Name') as Name  
  from Adr_List a;  
  
select * from Pers_v;
```

relationale Tabelle erstellen und mit Inhalt füllen

```
drop table Pers_relational;  
  
create table Pers_relational  
  (Name varchar2(20),  
   Ort varchar2(20));  
  
insert into Pers_relational  
  select  
    extractValue(a.Adresse,'/Adr/Name'),  
    from Adr_List a;  
  
select * from Pers_relational;
```

Erzeugung von XML Dokumenten aus relationalen Tabellen

(hier wird auf die employees Tabelle des User HR zugegriffen; alle Benutzer des Workshops haben Zugriffsrechte auf diese Tabelle)

column Name format A40;

Inhalt der Ausgangstabelle ansehen

desc hr.employees;

```
select count(*) from hr.employees;
```

```
select e. department_id, e.employee_id, e.first_name, e.last_name, e.hire_date, salary  
  from hr.employees e  
  where e.employee_id >200;
```

Inhalt der Ausgangstabelle in XML umwandeln

```
select XMLELEMENT(Name, e.first_name || ' ' || e.last_name) as Name,  
       XMLELEMENT ( EinstellungsDatum, e.hire_date) as Datum  
FROM hr.employees e  
WHERE e.employee_id > 200;  
  
select XMLELEMENT  
  (Abteilung,  
   XMLATTRIBUTES (department_id as Abt_Nr),  
   XMLAGG  
     (XMLELEMENT  
      (Mitarb,  
       XMLATTRIBUTES (e.employee_id as Pers_Nr),  
       XMLELEMENT (Name, e.last_name)  
      ) order by e.last_name  
     )  
  ) as Personaldaten  
FROM hr.employees e where e.department_id < 50 group by e.department_id ;
```

XMLType Tabelle erstellen, mit Inhalt füllen und anzeigen

```
drop table pers_data;  
  
create table pers_data of xmltype;  
  
desc pers_data;
```

```
insert into pers_data  
  SELECT XMLELEMENT(Mitarb,  
    XMLATTRIBUTES (e.employee_id as Pers_Nr),  
    XMLELEMENT(Name, e.first_name || ' ' || e.last_name),  
    XMLELEMENT (Einst_Datum, e.hire_date))  
  FROM hr.employees e  
  WHERE e.employee_id > 200 ;
```

```
select p.sys_nc_rowinfo$.getClobVal() as Mitarbeiterliste from pers_data p;  
  
select extractvalue  
  (value(p), '/MITARB/NAME') as Name  
  from pers_data p where existsNode(value(p), '/MITARB[@PERS_NR="205"])=1;
```

XMLType View erzeugen auf relationler Tabelle - mit zusätzlich relationaler Spalte

column Mitarbeiterliste format A80;

```
drop view employees_view;
```

```
CREATE OR REPLACE VIEW employees_view as select  
  employee_id as Pers_Nr,  
  XMLELEMENT  
    ("Mitarb",  
    XMLForest(e.first_name || ' ' || e.last_name AS "Name",  
    e.hire_date AS "hiredate")) as Mitarbeiterliste  
  FROM hr.employees e  
  WHERE salary > 15000;
```

```
select * from employees_view;
```

```
select * from employees_view e  
  where existsNode(e.Mitarbeiterliste, '/Mitarb[Name="Steven King"])=1;
```

XML DB Repository

(Basisformatierungen für SQL/Plus)

```
set long 10000;  
set pagesize 1000;  
set linesize 160;  
column any_path format a40;  
column path(1) format a40;  
column Repository_Struktur format a40;  
column resourcen format a40;  
column Bezeichnung format a30;  
column Erstelldatum format a40;
```

Anzeige der XML DB Konfig Datei

```
select XDBUriType('/xdbconfig.xml').getXML() as XMLDB_ConfigDatei  
from dual;
```

Repository Struktur + Ressourcen Merkmale

```
select path as Repository_Struktur
  from path_view
  where under_path (res,2,'/') =1;

select count(*) as Anzahl_Ressourcen
  from resource_view r
  where under_path(r.res, 2, '/') = 1;

select res
  from resource_view
  where equals_path (res,'/public/WS')>0;

select
  r.any_path,
  extractValue(r.res, '/Resource/DisplayName') as Bezeichnung,
  extractvalue(r.res, '/Resource/CreationDate') as Erstelldatum
  from resource_view r
  where under_path(r.res,'/public/WS')>0
  order by extractValue (r.res, '/Resource/DisplayName');
```

neuen Folder im XML DB Repository anlegen

```
call dbms_xdb.deleteResource('/public/WS/WSMaster/neu');  
call dbms_xdb.deleteResource('/public/WS/WSMaster/heu');  
call dbms_xdb.deleteResource('/public/WS/WSMaster');
```

```
DECLARE  
    retBool BOOLEAN;  
BEGIN  
    retBool:=DBMS_XDB.createfolder('/public/WS/WSMaster/heu');  
END;
```

```
select any_path as Pfad  
from resource_view  
where equals_path (res, '/public/WS/WSMaster/heu')>0;
```

Dokumente in dem Repository hinterlegen (ohne Schemaprüfung)

```
call dbms_xdb.deleteResource('/public/WS/WSMaster/presentation.xml');
commit;
```

DECLARE

```
retBool BOOLEAN;
BEGIN
retBool:=DBMS_XDB.createresource
('/public/WS/WSMaster/presentation.xml',
 '<presentation>
 <veranstaltung>Workshop</veranstaltung>
 <termin>Mai 05</termin>
 <ort>Berlin</ort>
 <thema>Workshop Oracle XML DB</thema>
 <referentin>
   <name>Annegret Warnecke</name>
   <firma>Oracle Deutschland GmbH, Berlin</firma>
   <funktion>Senior Systemberaterin</funktion>
 </referentin>
 </presentation>');
END;
```

```
select r.res.getClobVal() as presentation
from resource_view r
where any_path = '/public/WS/WSMaster/presentation.xml';
```

XML Schema Registrierung

(Basisformatierungen für SQL/Plus)

```
set long 10000;  
set pagesize 1000;  
set linesize 160;  
column Author format A15;  
column column_name format a20;  
column element_name format a20;  
column owner format a20;  
column Publisher format A10;  
column object_name format A30;  
column qual_schema_url format a100;  
column schema_owner format a20;  
column schema_url format a60;  
column table_name format a30;  
column Title format A40;  
column view_name format a20;  
column xmlschema format a50;
```

```
select owner, table_name, column_name from dba_xml_tab_cols order by owner, table_name;
```

```
select table_name, column_name from user_xml_tab_cols order by table_name;
```

```
select * from dba_xml_views;
```

```
select owner, view_name, column_name from dba_xml_view_cols;
```

Scripte zur Präsentation Oracle XML DB

```
EXEC dbms_xmlschema.deleteSchema('http://www.WS.com/A1_presentation.xsd',  
dbms_xmlschema.DELETE_CASCADE_FORCE);
```

cascade alle zu dem registrierten Schema vorab generierten Typen und default Tabellen
 werden gelöscht

force das Schema wird auch gelöscht, wenn es von dem Schema abhängige Objekte in
 der DB gibt; alle abhängigen Objekte werden als ungültig markiert

```
call dbms_xdb.deleteResource('/public/WS/WSMaster/A1_presentation.xsd');  
commit;
```

Input eines XML Schema Dokumentes in den Public Folder des Repositories

```
DECLARE
  retBool BOOLEAN;
BEGIN
  retBool:=DBMS_XDB.createresource
  ('/public/WS/WSMaster/A1_presentation.xsd',
  '<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="presentation">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="veranstaltung"/>
        <xs:element name="thema"/>
        <xs:element name="referent"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:schema>');
END;
```

Registrieren des eingestellten XML Schema Dokumentes an der DB

```
BEGIN
  DBMS_XMLSCHEMA.REGISTERSCHEMA
  (schemaurl=>'http://www.WS.com/A1_presentation.xsd',
  schemadoc=>sys.UriFactory.getUri('/public/WS/WSMaster/A1_presentation.xsd'));
END;
```

Scripte zur Präsentation Oracle XML DB

```
select table_name from user_xml_tables where  
    xmlschema like 'http://www.WS.com/%';
```

```
desc "presentationXXXX_TAB";
```

```
desc "presentationXXXX_T";
```

```
select owner, schema_url from all_xml_schemas;
```

```
EXEC dbms_xmlschema.deleteSchema('http://www.WS.com/A2_presentation.xsd',  
dbms_xmlschema.DELETE_CASCADE_FORCE);
```

registrieren eines Schemas

```
declare  
doc varchar2(4000) :=  
'<?xml version="1.0"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
  xmlns:xdb="http://xmlns.oracle.com/xdb" >  
  <xs:element name="presentation"  
    xdb:defaultTable= "A2_presentation_TAB"  
    xdb:SQLType="CLOB">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element name="veranstaltung"/>  
        <xs:element name="thema"/>  
        <xs:element name="referent"/>  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
</xs:schema>;'  
begin  
  dbms_xmlschema.registerSchema('http://www.WS.com/A2_presentation.xsd', doc);  
end;
```

```
desc "A2_presentation_TAB";
```

Dokument mit Verweis auf das Schema eingeben

```
insert into "A2_presentation_TAB"  
values (xmltype ('<presentation  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:noNamespaceSchemaLocation=  
  "http://www.WS.com/A2_presentation.xsd">  
    <veranstaltung>XML DB Presentation</veranstaltung>  
    <thema>XML in der Oracle DB Basics</thema>  
    <name>XYZ</name>  
  </presentation>'));
```

```
select * from "A2_presentation_TAB";
```

hier wird eine Fehlermeldung ausgeworfen

```
insert into "A2_presentation_TAB"  
values (xmltype ('<presentation  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:noNamespaceSchemaLocation=  
  "http://www.WS.com/A2_presentation.xsd">  
    <veranstaltung>XML DB Kundentag</veranstaltung>  
    <thema>XML in der Oracle DB Basics</thema>  
    <Name>Annegret Warnecke</name>  
  </presentation>'));
```

```
EXEC dbms_xmlschema.deleteSchema('http://www.WS.com/A3_presentation.xsd',  
dbms_xmlschema.DELETE_CASCADE_FORCE);
```

registrieren eines Schemas

BEGIN

DBMS_XMLSCHEMA.REGISTERSCHEMA

```
(schemauri=>'http://www.WS.com/A3_presentation.xsd',  
schemadoc=>'<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
xmlns:xdb="http://xmlns.oracle.com/xdb"  
xdb:storeVarrayAsTable="true" >  
<xs:element name="presentation"  
xdb:defaultTable="A3_presentation_TAB">  
  <xs:complexType  
    xdb:SQLType= "A3_presentation_T1" xdb:maintainDOM= "false">  
    <xs:sequence>  
      <xs:element name="Veranstaltung">  
        <xs:complexType xdb:SQLType= "A3_presentation_T2">  
          <xs:sequence>  
            <xs:element name="Bezeichnung"  
              xdb:SQLInline = "false"  
              xdb:defaultTable="A3_Bez_TAB"/>  
            <xs:element ref= "X" minOccurs= "0" maxOccurs= "5"/>  
          </xs:sequence>  
        </xs:complexType>  
      </xs:element>  
      <xs:element name="Referent" type="Referent_Type"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
</xs:sequence>  
</xs:complexType>
```

```
</xs:element>
<xs:element name= "X" xdb:defaultTable="A3_X_TAB">
  <xs:complexType xdb:SQLType= "A3_presentation_T3">
    <xs:sequence>
      <xs:element name="Ort" type="xs:string"/>
      <xs:element name="Termin" type="xs:date"/>
      <xs:element name="max_Anz_Teiln" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="Referent_Type"
xdb:SQLType="A3_presentation_T4">
  <xs:sequence>
    <xs:element name = "Anrede">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Frau"/>
          <xs:enumeration value="Herr"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name = "Name"/>
    <xs:element ref= "Tel." maxOccurs= "unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name= "Tel." type= "xs:string" xdb:defaultTable=""/>
</xs:schema>);
```

END;

```
desc "A3_presentation_TAB";
```

```
desc user_xml_tables;
```

Anlegen einer Tabelle mit Bezug auf das registrierte Schema

```
drop table Present;  
drop table Present_3;
```

```
create table Present of XMLType  
XMLTYPE store as CLOB  
XMLSCHEMA  
"http://www.WS.com/A1_presentation.xsd"  
ELEMENT "presentation";  
create table Present_3  
(NR number(10) not null, pres XMLTYPE not null)  
XMLTYPE column pres XMLSCHEMA  
"http://www.WS.com/A1_presentation.xsd"  
ELEMENT "presentation";
```

Zugriffe auf DB Inhalte mittels Browser

*Umformung von relationalen Strukturen in XML Form mittels dbURIType
(Großschreibung erforderlich)*

```
select * from hr.departments;
```

```
select dbURIType('/HR/DEPARTMENTS').getXML()  
from dual;
```

```
select dbURIType('/HR/DEPARTMENTS/ROW[DEPARTMENT_ID="10"]').getXML()  
FROM dual;
```

```
select dbURIType('/HR/DEPARTMENTS/ROW[DEPARTMENT_ID="10"]/DEPARTMENT_NAME').getXML()  
FROM dual;
```

*Zugriff auf relationale Tabellen mittels http (und Umwandlung in XML Form)
(ebenfalls Großschreibung erforderlich);*

"oradb" ist nicht der Name der DB sondern eine Funktion des DBURI Servlets)

```
http://awarneck-de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[DEPARTMENT_NAME="Public  
Relations"]/LOCATION_ID/text()
```

```
http://awarneck-de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[LOCATION_ID="1700"]
```

```
select count (*) from hr.departments  
where location_id=1700;
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[LOCATION_ID="1700"]?rowsettag=locations
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[LOCATION_ID="1700"]?contenttype=text/xml&rowset  
tag=locations
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[LOCATION_ID="1700"]?contenttype=text/html&rowset  
tag=locations
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/HR/DEPARTMENTS/ROW[LOCATION_ID="1700"]?contenttype=text/plain&rowse  
ttag=locations
```

weitere Beispiele für den Zugriff auf relationale Tabellen mittels Browser

```
select * from emp;
```

```
http://awarneck-de.de.oracle.com:8080/oradb/XDB/EMP/ROW[EMPNO=7369 and DEPTNO=20]/ENAME
```

```
http://awarneck-de.de.oracle.com:8080/oradb/XDB/EMP/ROW[SAL>2000]/ENAME?rowsettag=hoheGehalt2000
```

Aufbereitung einer relationalen Tabelle mittels eines Stylesheets, das im Repository abgelegt ist

```
http://awarneck-de.de.oracle.com:8080/public/Stylesheets/emp.xml
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/XDB/EMP?transform=/public/Stylesheets/emp.xml&contentType=text/html
```

Aufbereitung einer relationalen Tabelle mittels eines Stylesheets, das in einer Tabelle abgespeichert ist (create und insert statements s. Ende des Textes)

```
select * from stylesheet_tab;
```

```
http://awarneck-de.de.oracle.com:8080/oradb/XDB/STYLESHEET_TAB
```

```
http://awarneck-  
de.de.oracle.com:8080/oradb/XDB/EMP?transform=/oradb/XDB/STYLESHEET_TAB/ROW[LFD_NR  
="1"]/STYLESHEETS/text()&contentType=text/html
```

Constraints und Trigger

not null constraint auf xmltype column (ohne Schemabindung)

```
drop table AAA;  
  
create table AAA  
  (NR VARCHAR2(10),  
   INHALT XMLTYPE NOT NULL);  
-----  
insert into AAA  
  values(1, XMLTYPE('<Tag>xxx</Tag>'));  
insert into AAA  
  values (2);
```

check constraint auf xmltype column (ohne Schemabindung)

```
grant create any Trigger to WSx

desc user_triggers;
-----
drop table po_XML;
create table po_xml (id number (10), podoc XMLType);

drop trigger default_po_xml_date;
create or replace trigger default_po_xml_date
  BEFORE INSERT ON po_xml
  FOR EACH ROW
  DECLARE
    xmldata XMLType;
    tmpxml XMLType;
  BEGIN
    xmldata := :new.podoc;
    IF xmldata.existsnode('/PurchaseOrder/Actions/Action/Date') = 1
    THEN
      SELECT UPDATEXML(xmldata, '/PurchaseOrder/Actions/Action/Date',
        '<Date>'|| to_char(sysdate, 'DD-MON-YYYY') || '</Date>')
        INTO tmpxml FROM dual;
      :new.podoc := tmpxml;
    END IF;
  END;
```

```
insert into po_xml
values
(
  1,
  XMLTYPE('<?xml version="1.0"?>
    <PurchaseOrder>
      <Reference>Savitha</Reference>
      <Actions>
        <Action>
          <User>SCOTT</User>
          <User>ADAMS</User>
          <Date></Date>
        </Action>
      </Actions>
    </PurchaseOrder>'
)
);

select * from po_xml;
```

```
insert into po_xml
VALUES
(
  2,
  XMLTYPE('<?xml version="1.0"?>
    <PurchaseOrder>
      <Reference>Savitha</Reference>
      <Actions>
        <Action>
          <User>HR</User>
          <Date>30-Jan-2005</Date>
        </Action>
      </Actions>
    </PurchaseOrder>'
)
);

select * from po_xml;
```

unique constraint (nur für XML Schema basierende Tabellen)

drop table pres;

create table pres of XMLType

XMLSCHEMA

"http://www.WS.com/X_presentation.xsd"

ELEMENT "presentation";

alter table pres

add **constraint** VERANSTALTUNG_IS_UNIQUE

unique (xmldata."veranstaltung");

insert into pres

```
values (xmltype ('<presentation
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "http://www.WS.com/X_presentation.xsd">
  <veranstaltung>Kundentag</veranstaltung>
  <termin>Mai 05</termin>
  <ort>Berlin</ort>
  <thema>Oracle XML DB</thema>
  <referent>
    <name>Tilo Henke</name>
    <firma>Oracle</firma>
    <funktion>Oracle</funktion>
  </referent>
</presentation>'));
```

Input wiederholen

primary key constraint (nur für XML Schema basierende Tabellen)

```
alter table pres
  drop constraint VERANSTALTUNG_IS_UNQIUE;

alter table pres
  add constraint VERANSTALTUNG_IS_PK
  primary key(xmldata."veranstaltung");

commit;
```

check constraint auf Gültigkeit gem. registriertem Schema

```
drop table PX;
```

```
create table PX of XMLType
```

```
(CHECK (XMLIsValid(sys_nc_rowinfo$) = 1))
```

```
XMLSCHEMA
```

```
"http://xmlns.oracle.com/xdb/schemas/XDB/www.WS.com/X_presentation.xsd"
```

```
ELEMENT "presentation";
```

```
insert into PX
```

```
values (xmltype ('<presentation
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="http://www.WS.com/X_presentation.xsd">
```

```
<veranstaltung>Kundentag</veranstaltung>
```

```
<termin>Mai 05</termin>
```

```
<ort>Berlin</ort>
```

```
<thema>Oracle XML DB</thema>
```

```
<referent>
```

```
<Name>Tilo Henke</Name>
```

```
<firma>Oracle</firma>
```

```
<funktion>Oracle</funktion>
```

```
</referent>
```

```
</presentation>');
```

insert into PX

```
values (xmltype ('<presentation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
"http://www.WS.com/X_presentation.xsd">
  <veranstaltung>Kudentag</veranstaltung>
  <termin>Mai 05</termin>
  <ort>Berlin</ort>
  <thema>Oracle XML DB</thema>
  <referent>
    <name>Tilo Henke</name>
    <firma>Oracle</firma>
    <funktion>Oracle</funktion>
  </referent>
</presentation>'));
```

XML Schemata in der DB erstellen

```
drop type Mitarbeiter_t;  
  
create or replace type Mitarbeiter_t as object  
(  
  PersNr number(2),  
  Name varchar2(20),  
  Strasse varchar2(20),  
  Ort varchar2(20)  
);  
  
select dbms_xmlSchema.generateSchema('XDB','MITARBEITER_T') as result  
from dual;
```

```
drop type emp1_t;
drop type dep1t_t;

create or replace type dept1_t as object
(
  DEPTNO number(2),
  DNAME varchar2(14),
  LOC varchar2(13)
);

create or replace type emp1_t as object
(
  EMPNO number(4),
  ENAME varchar2(10),
  JOB varchar2(9),
  MGR number(4),
  HIREDATE date,
  SAL number(7,2),
  COMM number(7,2),
  DEPT dept1_t
);

select dbms_xmlSchema.generateSchema('XDB','EMP1_T') as result
from dual;
```

```
select * from USER_OBJECT_TABLES;
select * from USER_VARRAYS;
```

```
select * from USER_NESTED_TABLES;  
select * from USER_TYPES;  
select type_name, attr_name, attr_type_name, attr_no, inherited from USER_TYPE_ATTRS;  
select * from user_type_methods;
```

Kommentar: Object Enhancements

In prior releases, an object type referenced by a table or user-defined type could be changed only by adding methods to it. In Oracle9i, objects can evolve more freely and any changes made to a type can propagate to the schema objects that reference it. Oracle9i provides the ability to add, drop, and modify attributes of an object type. Collections were introduced in Oracle8.0. They provide an alternate way of modeling data in one-to-many relationships. There are two kinds of collection types: varying-length arrays (VARRAYs) and nested tables. User-defined collection types can be used to specify columns in database tables, attributes of object types, and PL/SQL variables. Oracle9i allows collections to be nested within other collections.

The Oracle8 and Oracle8i models did not directly support inheritance. In Oracle9i, object type definitions can be organized into inheritance hierarchies.