-- XML IN ORACLE

-- This example shows how to create a table that stores XML data.

drop table addr_book;

create table addr_book(
    name varchar2(20),
    card sys.xmltype,
    creationDate Date
)
/

-- This shows that Objects of Type XML can be stored in Tables.

insert into addr_book values ('James',
    sys.XMLType.createXML('<acard createdby="Jim">
        <email>geller@njit.edu</email>
        <wphone>596-3383</wphone>
        <wphone>596-3366</wphone>
        <cell>794-4399</cell>
    </acard>'),
    sysdate)
/

-- The constructor takes the string that contains XML code and stores it as an object.

select * from addr_book;

-- This shows that Objects of Type XML can be stored in Tables.

insert into addr_book values ('James',
    sys.XMLType.createXML('<acard createdby="Jim">
        <email>geller@njit.edu</email>
        <wphone>596-3383</wphone>
    </acard>')
-- In order to do any decent querying, we need a larger database.
-- I had an unexplainable error. I fixed it by deleting and retyping.

insert into addr_book values (
   'James',
   sys.XMLType.createXML("<acard createdby="Jim">
      <email>geller@njit.edu</email>
      <wphone>596-3383</wphone>
      <wphone>596-3366</wphone>
      <cell>794-4399</cell>
      <address>
         <line1>323 Dr. King Blvd</line1>
         <city>Newark</city>
         <state>NJ</state>
         <zip>07102</zip>
      </address>
   </acard>'
),
sysdate)
/

insert into addr_book values (
   'Roger',
   sys.XMLType.createXML("<acard createdby="Jim">
      <email>roger12@yahoo.com</email>
      <wphone>111-1234</wphone>
      <wphone>111-5678</wphone>
      <cell>111-3342</cell>
      <address>
         <line1>123 Main St.</line1>
         <city>Atlanta</city>
         <state>Georgia</state>
         <zip>33333</zip>
      </address>
   </acard>'
),
sysdate)
/

insert into addr_book values ('Bobby',
   sys.XMLType.createXML(
      '<acard createdby="Jim">'
         '<email>bob@aol.com</email>
         <wphone>111-2900</wphone>
         <address>
            <line1>200 Oak St.</line1>
            <city>Atlanta</city>
            <state>Georgia</state>
            <zip>33330</zip>
         </address>
      </acard>
   ),
   sysdate)
/

insert into addr_book values ('Tommy',
   sys.XMLType.createXML(
      '<acard createdby="Jim">'
         '<email>tom200@yahoo.com</email>
         <wphone>333-8000</wphone>
         <cell>111-9000</cell>
      </acard>
   ),
   sysdate)
/

insert into addr_book values ('Abe',
   sys.XMLType.createXML(
      '<acard createdby="Jim">'
         '<email>abe1212@yahoo.com</email>
         <wphone>111-2000</wphone>
         <cell>111-3344</cell>
         <address>
            <line1>125 Main St.</line1>
            <city>Savannah</city>
            <state>GA</state>
            <zip>33300</zip>
         </address>
         <comment>Kids: John, Jamie</comment>
      </acard>
   ),
   sysdate)
/

insert into addr_book values ('Tony',
   sys.XMLType.createXML(
      '<acard createdby="Raj">'
         '<email>anthony1@yahoo.com</email>
         <wphone>111-1000</wphone>
         <cell>111-3000</cell>
         <address>
<line1>123 Elm St.</line1>
<city>Savannah</city>
<state>GA</state>
<zip>33300</zip>
</address>
</acard>

-- AS OF FALL 2008, in AQUA I did **NOT** have to use an alias for this query.
-- HOWEVER in more complicated queries I still needed the alias!!!
-- CARD was the name of the column that contains XML objects.
-- You MUST use an alias. Otherwise it does not work for more
-- complicated queries.

p39

select Name, a.card.extract('/').getstringval()
from addr_book a
where Name like 'R%'
/

-- We added a standard WHERE clause.
-- This does NOT use the XML data.
-- To get subtrees out of an XML expression, we use the method
-- extract().
-- extract() takes as argument a path expression.

-- Now we have to learn a whole new language. It is called
-- XPath.
-- It is a query language for XML.

-- (1) A '/' extracts the whole XML object.
-- This is called the ROOT object.
-- Because XML forms trees.
-- And in UNIX '/' stands for root.
-- So that name makes sense.

-- As in UNIX, / is also used as a path separator.
-- The "right thing" is a child of the "left thing":

-- Like this: left_thing/right_thing

select Name, a.card.extract('/').getstringVal()
from addr_book a;

-- "Give me the name and all the roots in the column card
-- and all descendants."
-- So far this does exactly the same as:

select Name, j.card
from addr_book j;

-- "Give me Name and everything in the column card"

p41

-- In the middle column I can put anything. Not just <acard>

-- I did this.
insert into addr_book values ('Joe',
sys.XMLType.createXML('<test>here</test>'),
sysdate)
/

-- The last select statement will extract the <test> also:

-- select Name, a.card.extract('/')
-- from addr_book a;

-- Now try these two:

select Name, a.card.extract('/acard').getstringVal()
from addr_book a
/
select Name, a.card.extract('/test').getstringVal() from addr_book a
/

-- The first one means: Return the root expressions
-- that are <acard> with all descendants.

-- The second one means: Return the roots with the root tag
-- test and all descendants.

p42

-- Now we refine this:
-- We need a new function. A boolean. It's like extract().
-- But it returns true (1) if something is there.
-- It is called existsNode().
-- It does NOT return the the XML structure, only the truth value.

select Name, a.card.extract('/test').getstringval()
from GELLER.addr_book a
where a.card.existsNode('/test') = 1

-- Now every row except "Joe" is gone completely!

-- ----------------------------------

-- This drops the row with Joe completely.

select Name, a.card.extract('/acard').getstringval()
from addr_book a
where a.card.existsNode('/acard') = 1;

p43

-- Now we will study how to get different parts of an /acard with XPath
-- Get the <email> elements that are under <acard> roots.

select Name, a.card.extract('/acard/email').getstringval()
from addr_book a
where a.card.existsNode('/acard') = 1
/

-- Joe has no email address. So, try this again:

select Name, a.card.extract('/').getstringval()
from addr_book a
where a.card.existsNode('/acard/email') = 0;

-- Show name and all XML for people that do NOT have
-- an email tag inside of an acard tag. 0 = false,