DB2 a XML

Martin Pozdílek
MF servis s.r.o.
mpozdilek@mfservis.cz

Data v textovém souboru

47; John Doe; 58; Peter Pan; Database systems; 29; SQL; relational

Co je XML

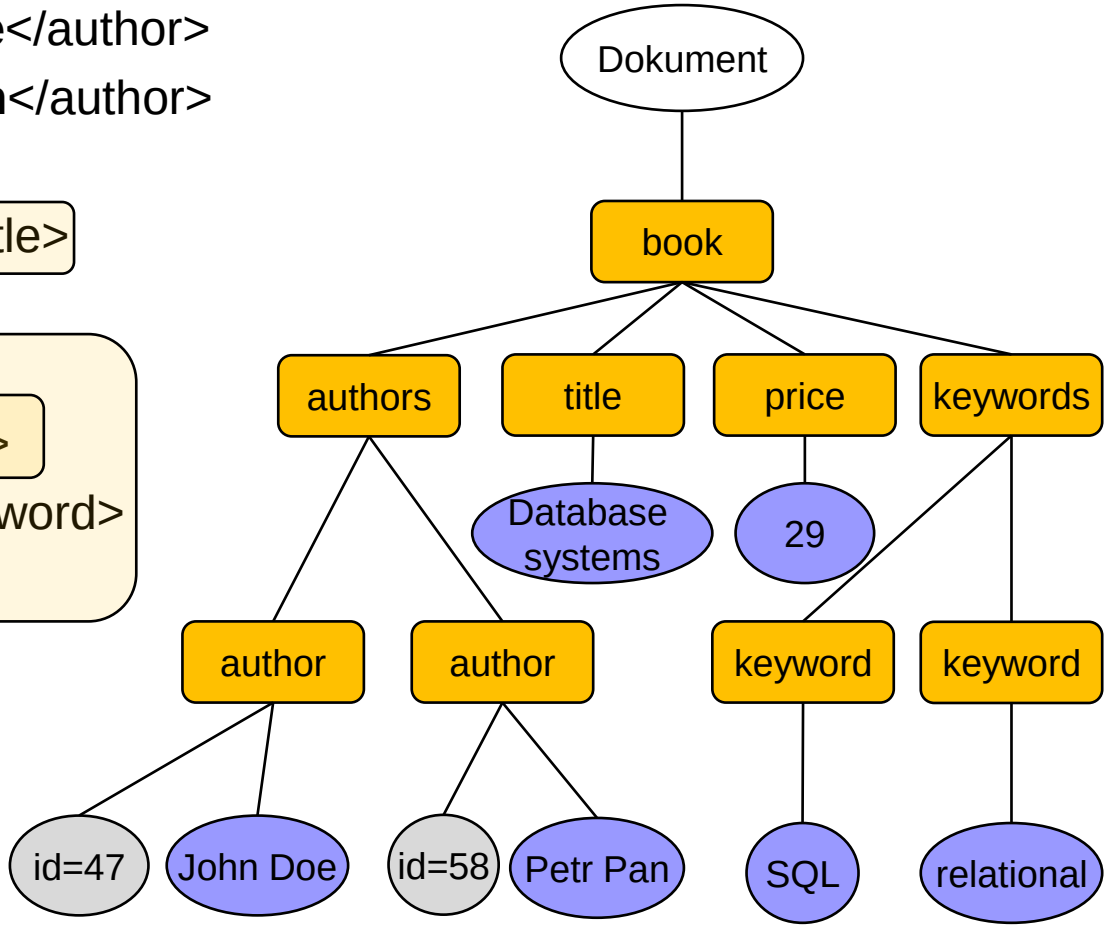
- eXtensible Markup Language
  - Flexibilní
  - Sebe popisující
  - Jednoduchý na sdílení
  - Jednoduchý na rozšiřování
  - Zvládá komplexní datové modely
  - Platformově a dodavatelsky nezávislý
- Hierarchický model dat
- Standardy pro sdílení dat založené na XML
  - Bankovní sektor - IFX, OFX, SWIFT, SPARCS, MISMO +++
  - Zdravotnictví – ACORD, XML for P&C, Life +++
  - Chemický průmysl - Chemical eStandards, CyberSecurity, PDX Standard+++
  - Telekomunikace - eTOM, NGOSS, Parlay Specification +++
  - Web Services, SOA (Service Oriented Architecture)
  - …

XML dokument

```
<book>
  <authors>
    <author id="47">John Doe</author>
    <author id="58">Peter Pan</author>
  </authors>
  <title>Database systems</title>
  <price>29</price>
  <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
  </keywords>
</book>
```

Element Atribut Data

Parsing



Serialization

Pravidla XML dokumentu

- Správně formátovaný dokument
  - Existuje pouze 1 kořen.
  - Každý element má počáteční a konečnou značku.
  - Pořadí počátečních a konečných značek si odpovídá (<a>abc).
  - Element může obsahovat další element, atribut nebo data (text node).
  - Atributy musí být ve dvojitých uvozovkách. Data nemusí.

- Validní dokument
  - Správně formátovaný dokument.
  - Dokument odpovídá pravidlům popsaným pomocí XML schéma nebo Document Type Definition (DTD).
  - XML schéma a DTD jsou XML dokumenty, které definují strukturu XML dokumentu.
  - XML parser může spouštět validaci.

- Není vhodné dávat data jako jména značek.

```
<book>
 <authors>
 <John Doe>
    <age>42</author>
 </John Doe>
 <Peter Pan>
    <age>45</author>
 </Peter Pan>
 </authors>
</book>
```

Porovnání XML a relačních dat

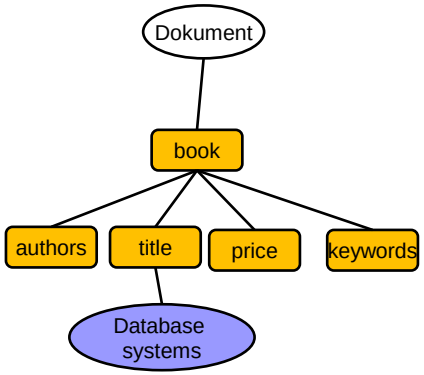
Relační data	Hierarchické XML
Data jsou plochá, uložená v tabulkách. Mezi tabulkami logické vazby.	Hierarchické uložení, princip vnořování elementů.
Data jsou sdružená do množin. V rámci množin nesetříděná.	Data jsou v dokumentu řazená. Jiný způsob řazení nebývá možný.
Pevně daná struktura.	Flexibilní sebe popisující se struktura.
Používají se NULL hodnoty.	NULL neexistuje. Jednoduše se nepřidá element.
ANSI/ISO	W3C standard

Uložení XML

- Umístění XML
  - Textové XML soubory
  - XML uloženo jako sloupec typu LOB, VARCHAR v databázích
  - Dekompozice XML do relačních tabulek
  - Nativní formát v databázích

- Reprezentace XML
  - Textová
  - Strom (DOM)
  - Fronta událostí (SAX)

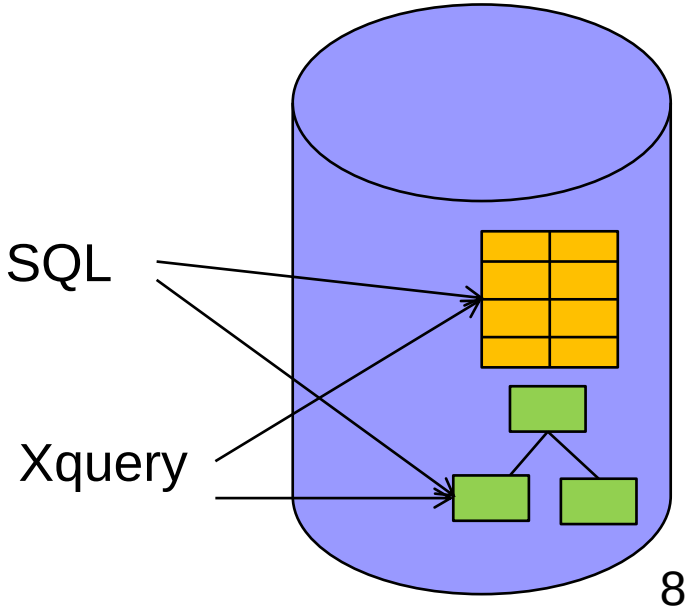
```
<book>
 <authors>
    <author id="47">John Doe</author>
    <author id="58">Peter Pan</author>
 </authors>
 <title>Database systems</title>
 <price>29</price>
 <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
 </keywords>
</book>
```



Začíná element <book>
Začíná element <authors>
Začíná element <author>
Atribut id=47
Hodnota elementu John Doe
Končí element </author>
Začíná element <authors>
…

DB2 pureXML

- DB2 je hybridní databáze
  - Tabulka může obsahovat jak relační tak XML data.
    - Relační data jsou uložena v tabulkách.
    - XML data jsou uložena v XML uložišti v nativním formátu. Ne jako řetězce v tabulkách.
  - XML je v DB2 podporováno na všech úrovních (INSERT, IMPORT, RUNSTATS, ...).
  - Na relační data lze přistupovat pomocí SQL a XQuery.
  - Na XML data lze přistupovat pomocí SQL a XQuery.
  - ```
    CREATE TABLE ABC (
        id INT,
        name VARCHAR(200),
        doc XML)
    ```
  - ```
    SELECT * FROM ABC
    ```
  - ```
    XQUERY db2-fn:xmlcolumn("ABC.doc")
    ```


    ```

Vlastnosti pureXML

- Při dotazování není třeba parsování.
- Akceptují se pouze správné dokumenty.
- Používá se SAX místo DOM parsování.
  - DOM zabírá 2x – 10x více místa než textové XML.
  - DOM je jednodušší, ale pomalejší.

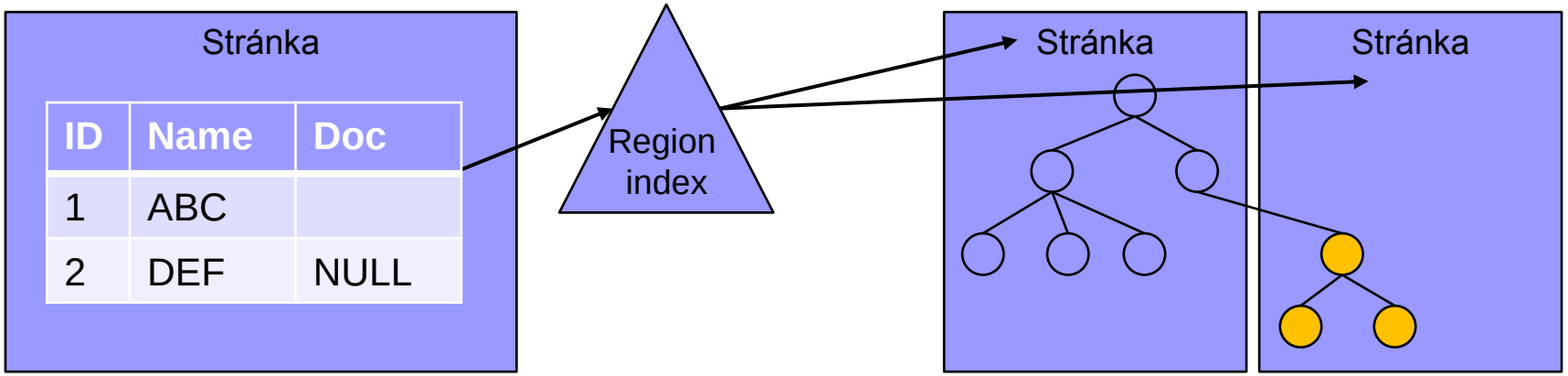
- Podpora jmenných prostorů.
- XPATH 2.0, Xquery, SQL/XML.
- Aktualizace částí XML dokumentů.

- Validace dokumentů je dobrovolná.
- Podpora XML schéma i více na jednom sloupci.
- XML schéma lze měnit.

- Indexovat lze libovolný element nebo atribut.

Interní uložení XML

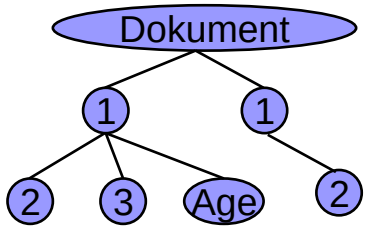
- XML dokument je parsován a uložen do Xquery Data Model (XDM).
- XML, který se nevejde na stránku je rozdělen do regionů.
- V relační části tabulky je uložen ukazatel do Region indexu. Region index obsahuje ukazatele na stránky s XML dokumentem.
- Relační stránky se u SMS tablespace ukládají do DAT souboru, XML stránky do XDA souboru.
- DB2 při práci nerozlišuje mezi relačními a XML stránkami – Bufferpool, Prefetch, logování, …
- Dostatečně malé XML dokumenty mohou být uloženy INLINE na stejných stránkách jako relační data.
- Je vhodné, aby XML dokument byl rozdělen na minimální počet regionů.



Interní uložení XML

- Interní formát je komprimován. Neukládá se jméno elementu, ale odkaz na seznam všech elementů.
- Při kompresi tabulky má každý XML dokument svůj slovník.
- Díky ukazatelům a indexům je XML dokument v DB2 větší než textová podoba XML.
- Maximální velikost XML dokumentu 2GB.

ID	Name
1	Person
2	Phone
3	Name



- U DMS lze oddělit tablespaces pro relační a XML data - IN DATATBS LONG IN XMLTBS.

Dotazování na XML

- DB2 podporuje 2 jazyky
  - XQUERY
    - Funkcionální programovací jazyk navržený W3C pro dotazování XML dat.
    - Struktura XML na rozdíl od relačních dat nemusí být předvídatelná a hodně se mění.
      - Vyhledávání dat na různých úrovních dokumentu.
      - Výsledek může mít různé datové typy.
      - Umí měnit XML data.
      - Umí měnit strukturu XML dat.
    - Vždy začíná xquery.
    - Dvě podstatné funkce pro získání dat z databáze.
      - db2-fn:sqlquery ('SQL')
      - db2-fn:xmlcolumn ('XML column name')
  - SQL/XML
    - Rozšiřuje SQL o možnosti práce s XML.
    - Obsahuje funkce pro současnou práci s relačním a XML obsahem v rámci jednoho SQL dotazu.

- XPATH
  - Způsob navigace v rámci XML dokumentu.
  - Platí pro XQUERY a SQL/XML.

XPath

```
<customerInfo>
   <cusotmer id ="1">
      <name>Victor</name>
      <sex>M</sex>
      <phone type="work">739-1274</phone>
  </customer>

  <customer id ="2">
     <name>April</name>
     <sex>F</sex>
     <phone type="home">983-2179</phone>
  </customer>
</customerInfo>
```

Cesty
/
/customerInfo
/customerInfo/customer/@id
/customerInfo/customer/name
/customerInfo/customer/sex
/customerInfo/customer/phone
/customerInfo/customer/phone/@type

- Každý element má svoji cestu.
- / slouží pro oddělení hierarchie elementů

XPath

```
<customerInfo>
  <customer id="1">
    <name>Victor</name>
    <sex>M</sex>
    <phone type="work"> 739-1274</phone>
  </customer>
  <customer id ="2">
    <name>April</name>
    <sex>F</sex>
    <phone type="home"> 983-2179 </phone>
  </customer>
</customerInfo>
```

Znak	Význam
/	Vybere kořenový element
//	Sám sebe nebo potomka. Nemusí se definovat cesta. Berou se všechny potomci.
text()	Vrátí hodnotu aktuálního elementu
@	Atribut
*	Všechny elementy
.	Aktuální element.
..	Rodičovský element
@*	Všechny atributy
[…]	Podmínky. [n] – n-tý element. Začíná se od 1. [element > hodnota]

Popis	Dotaz	Výsledek
Vyber všechna telefonní čísla	/customerInfo/*/phone/text/()	739-1274, 983-2179
Vyber typy telefonních čísel	/customerInfo//phone/@type	work, home
Telefonní číslo prvního zákazníka	/customerInfo/customer[1]/phone/text()	739-1274
Elementy s atributem hodnoty home	/customerInfo//phone[@type='home']	<phone type="home"> 983-2179 </phone>
Vyber všechna jména	//name/text()	Victor April

XPath

Popis	Dotaz	Výsledek
Druhý zákazník.	/customerInfo/customer**[2]**	<customer id ="2"><name>**April**</name> <sex>**F**</sex><phone type = "home"> **983**</phone></customer>
Zákaznice	/customerInfo/customer**[sex="F"]**	<customer id ="2"><name>**April**</name> <sex>**F**</sex><phone type = "home"> **983**</phone></customer>
Zákazník s id 1	customerInfo/customer[@**id="1"**]	<customer id="1"> <name>**Victor**</name> <sex>**M**</sex><phone type="work"> **123**</phone></customer>
Id všech zákazníků s telefonem větším než 200	***//***customer[phone > "200"]/@id	2
Telefon > 200	/customerInfo/customer/phone[**.** > "200"]	><phone type = "home"> **983**</phone>
ID zákaznic	/customerInfo/customer/sex[.="F"]/**../** @id	2

XQuery

- Dotaz na XML sloupec
  - Na vyhodnocení jsou poslány všechny XML dokumenty ze sloupce tabulky.
  - Nad každým XML dokumentem je proveden XPATH dotaz.
  - `xquery db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo`

- SQL dotaz, který vrací XML sloupec
  - Nejprve se vyhodnotí SQL dotaz, který vrátí množinu XML dokumentů.
  - Nad každým vráceným XML dokument je proveden XPATH dotaz.
  - `xquery db2-fn:sqlquery('SELECT INFO FROM CUSTOMER')/customerinfo`
  - `xquery db2-fn:sqlquery('SELECT INFO FROM CUSTOMER WHERE CID=1000')/customerinfo//name/text()`

XQuery - FLWOR

- Plná syntaxe XQuery dotazů.
  - FOR – prochází sekvenci elementů a postupně je propojuje s proměnnou
  - LET – svazuje proměnnou se sekvencí
  - WHERE – podmínka pro zpracování pouze části elementů
  - ORDER – přerovnání prvků v iteraci
  - RETURN – vytvoří výstup dotazu

- ```
  xquery
    for $c in db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo
    let $pc := $c/addr/pcode-zip
    where $c/addr/@country = "Canada"
    order by $c/@Cid
    return <contact>{$c/@Cid}{$pc}</contact>
  ```

- Spojení dvou XML dokumentů

```
for $book in db2-fn:xmlcolumn('BOOKS.DOC')/book
    $for entry in db2-fn:xmlcolumn('REVIEWS.DOC')/entry
    where $book/title=$entry/title
    return <review>{$entry/review/text()} </review>;
```
```

Manipulace s XML

- Vkládání
  - `INSERT INTO ABC (1,'ABC','<book>…</book>');`
  - `INSERT INTO ABC (1,'ABC',xmlparse (dokument '<book>…</book>'));`
  - `INSERT INTO ABC (1,'ABC',xmlparse (dokument '<book>…</book>' preserve whitespace));`
- Mazání
  - `DELETE FROM ABC WHERE ID=1`
  - `DELETE FROM ABC WHERE XMLEXISTS ('$d//book[title="Database systems"]' passing book as b)`
  - `UPDATE ABC SET doc = NULL WHERE ID=2`
  - Pro smazání části XML dokumentu je třeba použít UPDATE
- IMPORT a EXPORT
  - `IMPORT FROM /data/data.del OF DEL  XML FROM /data/xmlfiles INSERT INTO ABC`
  - `EXPORT TO /data/data.del OF DEL XML TO XMLFILE X /data/xmlfiles MODIFIED BY XMLINSEPFILES SELECT * FROM ABC`

/data/
  data.del
    1,'A',"<XDS FIL='X1.xml' />"
    2,'B',"<XDS FIL='X2.xml' />"
    3,'C',"<XDS FIL='X3.xml' />"

/data/xmlfiles
  X1.xml
  X2.xml
  X3.xml

Aktualizace XML dokumentu

- Aktualizace XML dokumentu v rámci SQL UPDATE příkazu
  - Vložení, smazání, nahrazení a přejmenování elementu.
  - Aktualizace hodnoty.
  - Pořadí změn: Označení elementů ke smazání, vložení elementů bez udání pozice, přejmenování, náhrada hodnot v value of, vložení s udáním pozice, náhrada hodnot bez value of, náhrada value of pro elementy, smazání označených nodů.

- U zákazníka změn město na Toronto

```
UPDATE customer SET info = XMLQUERY('
  copy $c := $INFO
  modify do replace value of $c/customerinfo/addr/city with "Toronto"
  return $c')
WHERE CID =1001
```

- Smaž element city

```
UPDATE customer SET info = XMLQUERY('
  copy $c := $INFO
  modify do delete $c/customerinfo/addr/city
  return $c')
 WHERE CID =1001
```

Aktualizace XML dokumentu

- Vlož element city do elementu addr
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do insert <city>Toronto</city> into $c/customerinfo/addr
      return $c')
    WHERE CID =1001
    ```

- Vlož element phone **za** element phone
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do insert <phone type="cell">905-555-7272</phone> after
    $c/customerinfo/phone
      return $c')
     WHERE CID =1001
    ```

- Vlož element company-name **jako první** element addr
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do insert <company-name>ACME Co</company-name> as first into
    $c/customerinfo/addr
      return $c')
     WHERE CID =1001
    ```
    ```

Aktualizace XML dokumentu

- Vlož element company-name **jako poslední** element do addr
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do insert <company-name>ACME Co</company-name> as last into
    $c/customerinfo/addr
      return $c')
     WHERE CID =1001
    ```

- Vlož element email před element addr
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do insert <email>ksmith@acme.com</email> before $c/customerinfo/addr
      return $c')
     WHERE CID =1001
    ```

- Změn jméno elementu addr na address
  - ```
    UPDATE customer SET info = XMLQUERY('
      copy $c := $INFO
      modify do rename $c/customerinfo/addr as "address"
      return $c')
     WHERE CID =1001
    ```
    ```

SQL/XML

- Spuštění Xquery dotazu v rámci SQL dotazu.
  - Při použití pouze čistého SQL lze pracovat pouze s XML dokumentem jako celkem.
  - S SQL/XML lze pracovat i s částí XML dokumentu.

- XMLQUERY()
  - Skalární funkce vrací výsledek XQuery dotazu jako jednu hodnotu typu XML.
  - XQuery dotaz se spouští proti každému řádku s XML dokumentem.
  - Proměnná $i slouží pro spojení relační a XML části dotazu.
  - `SELECT XMLQUERY('$i/customerinfo/name' PASSING INFO AS "i") FROM CUSTOMER`

```
<name>Kathy Smith</name>
<name>Kathy Smith</name>
<name>Jim Noodle</name>
<name>Robert Shoemaker</name>
<name>Matt Foreman</name>
<name>Larry Menard</name>
```

SQL/XML

- XMLTABLE()
  - Tabulková funkce vrací výsledek Xquery dotazu jako relační tabulku.
  - Spustí se Xquery dotaz a jeho výsledek je přemapován z elementů na relační sloupce s datovými typy.
  - ```
    SELECT T.* FROM XMLTABLE(
      'db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo'
      COLUMNS "NAME" VARCHAR (20) PATH 'name',
              "STREET" VARCHAR (50) PATH 'addr/street',
              "CITY" VARCHAR (20) PATH 'addr/city'
      ) AS T
    ```

Name	Street	City
Kathy Smith	5 Rosewood	Toronto
Kathy Smith	25 EastCreek	Markham
Jim Noodle	25 EastCreek	Markham
Robert Shoemaker	1596 Baseline	Aurora
Matt Foreman	1596 Baseline	Toronto
Larry Menard	223 NatureValley	Toronto
    ```

SQL/XML

- XMLEXISTS()
  - SQL predikát ověřuje, zda Xquery vrací prázdnou sekvenci nebo ne.

  - ```
    SELECT CID, INFO
      FROM XMLCUSTOMER WHERE
      XMLEXISTS('$d/customerinfo[name = "John Smith"],
      passing INFO as "d")
    ```
    ```

SQL/XML

- XMLPARSE()
  - převede řetězec na datový typ XML
- XMLSERIALIZE()
  - převede XML na řetězec
- XMLVALIDATE()
  - ověří dokument proti XML schéma
- XMLELEMENT(jméno element, SQL sloupce)
  - vytvoří XML element ze sloupce
- XMLFOREST (sloupec as elementname, sloupec2 as elementname, …)
  - Vytvoří sekvenci XML elementů z SQL sloupců
- XMLATTRIBUTES()
  - Vytvoří atribut
- XMLCONCAT()
  - Spojení XML dokumentů
- XMLNAMESPACES()
  - Vytvoření jmenného prostoru
- XMLAGG()
  - Seskupení XML dokumentů
- XMCOMMENT()
- XMDOCUMENT()
- XMLGROUP()
- XML_NAMESPACE()
- XML_PI()
- XML_ROW()
- XML_TEXT()
- XML_TRANSFORM()

SQL/XML

```
select xmlserialize( content
        xmlelement(name "tr", xmlelement(name "td", xmlattributes('center' as
"align"), e.empno),
        xmlelement(name "td", e.firstnme),
        xmlelement(name "td", e.lastname),
        xmlelement(name "td", xmlattributes('center' as "align"), e.phoneno),
        xmlelement(name "td", xmlattributes('center' as "align"), d.deptno),
        xmlelement(name "td", substr(d.deptname, 1, 24)) ) as clob(180) ) as "result"
  from employee e, department d
  where e.workdept = d.deptno and year(hiredate) < 1970
```

```
<tr><td align="center">000010</td><td>CHRISTINE</td><td>HAAS</td>
<td align="center">3978</td><td align="center">A00</td>
<td>SPIFFY COMPUTER SERVICE </td></tr> ...
```

XML index

- Nad XML dokumenty lze vytvářet indexy pro rychlejší vyhodnocování dotazů.
- Index obsahuje odkazy na elementy nebo atributy, které odpovídají XML vzoru.
- XML index lze vytvořit pouze nad jedním XML sloupcem.
- Jeden klíč může odkazovat na více elementů v XML dokumentu.
- XML sloupec může mít více indexů.
- XML indexy zpomalují operace INSERT, UPDATE, DELETE.
- XML indexy zabírají místo na disku.
- Není vhodné pokaždé indexovat všechny elementy.

- Index pro rychlou práci s platy

```
CREATE INDEX IDX1 ON TB1(XMLDOC)
  GENERATE KEY USING XMLPATTERN
  '/company/emp/salary'
  AS SQL DOUBLE;
```

- Index pro rychlé hledání podle atributu id

```
CREATE INDEX IDX2 ON TB1(XMLDOC)
  GENERATE KEY USING XMLPATTERN
   '//@id' AS SQL VARCHAR(20);
```

XML indexy

- Systémové
  - Region index
    - Vytváří se při vložení prvního XML do sloupce.
    - Ukládá informace, jak je XML index rozdělen na regiony.
    - Jeden index na všechny XML sloupce v tabulce.
  - XML Path index
    - Pro každý XML sloupec samostatný index.
    - Uchovává všechny unikátní cesty pro všechny XML dokumenty ve sloupci.

- Uživatelský index
  - Vytváří se na základě vzoru.
  - Klíč může odkazovat na více elementů.
  - Může být detailnější než systémové.

XML indexy

- Unikátní index na ID zákazníka
  - `CREATE UNIQUE INDEX idx ON table(col) GENERATE KEY USING XMLPATTERN '/customerInfo/@id' as sql integer;`
- Všechny elementy name
  - `… XMLPATTERN '//name' as sql varchar(35)`
- Všechny číselné atributy
  - `… XMLPATTERN '//@' as sql double`
- Všechny textové hodnoty elementů
  - `… XMLPATTERN '//text()' as sql varchar(hashcode)`
- Všechny elementy pod customer
  - `… XMLPATTER '/customerInfo/customer//text()'`
- Pro AS INTEGER, AS DECIMAL se vytváří rychlejší celočíselné indexy.

```
<customerInfo>
  <customer id="1">
    <name>Victor</name>
    <sex>M</sex>
    <phone type="work"> 739-1274</phone>
  </customer>
  <customer id ="2">
    <name>April</name>
    <sex>F</sex>
    <phone type="home"> 983-2179 </phone>
  </customer>
</customerInfo>
```

Podmínky pro použití XML indexu

```
<customerInfo cid="1">
  <name>Victor</name>
  <phone> 123 </phone>
</customerInfo>
```

- Index obsahuje predikát, který je stejný nebo méně restriktivní než predikát v dotazu.

	xmlpattern 'customerInfo/phone'	xmlpattern '//phone'	xmlpattern '/customerInfo/*'
where $i//phone = "123"	Ne. Phone může být i mimo customerInfo.	Ano	Ne. Phone může být i mimo customerInfo.
where $i/customerInfo/phone = "123"	Ano	Ano	Ano
where $i/customerInfo/* = "123"	Ne	Ne	Ano

- Predikát v indexu a dotazu musí mít stejný datový typ.

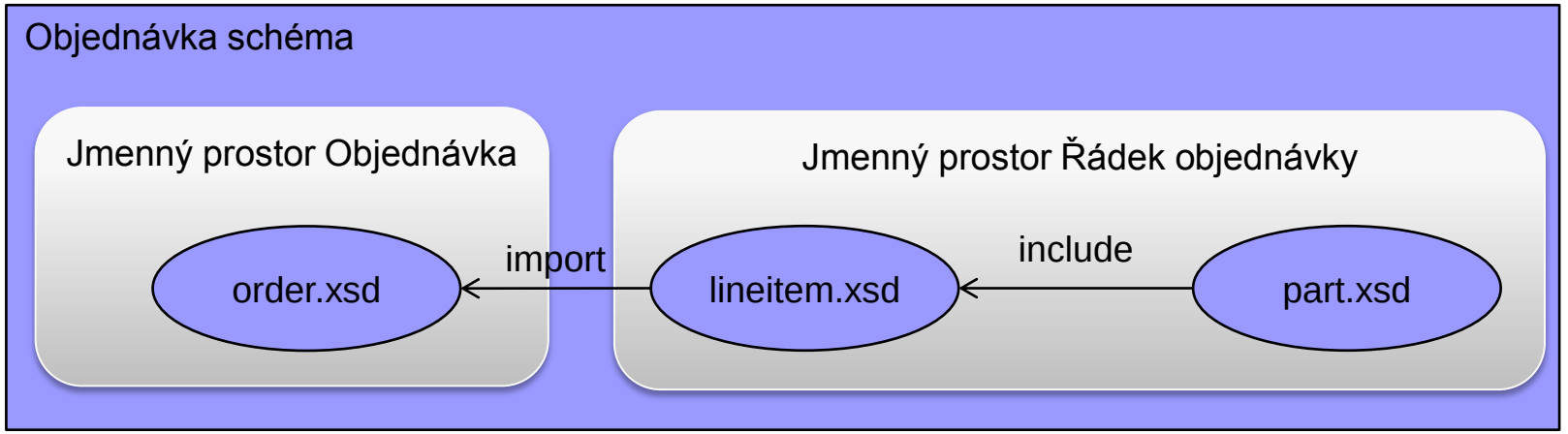
	xmlpattern 'customerInfo/@cid' as sql varchar(10)	xmlpattern 'customerInfo/@cid' as sql double
for $i in db2-fn:sqlquery (…) where $i/customerInfo/@cid = "1"	Ano	Ne
for $i in db2-fn:sqlquery (…) where $i/customerInfo/@cid = 1	Ne	Ano

- /text() je použit stejně v dotazu a v indexu. Spíše text() v indexu nepoužívat.

	xmlpattern 'customerInfo/name'	xmlpattern 'customerInfo/name/text()
where $i/customerInfo/name = "A"	Ano	Ne
where $i/customerInfo/name/text() = "A"	Ne	Ano

XML schéma

- Definuje zda je XML validní po stránce
  - Struktury
  - Datových typů
    - Základní: String, boolean, float, double, decimal, integer, positiveInteger, byte, date, datetime, any
    - Odvozené: omezení základního, výčet, spojení základních (string + integer), povolené vzory, délka
    - Komplexní: definice elementů, atributů, sekvence elementů
- Skládá se z 1 nebo více schéma dokumentů.
- Schéma dokument může definovat jmenný prostor.



XML schéma

XML dokument

```
<wikipedista uid="Novak">
    <jmeno>Jirka</jmeno>
    <prijmeni>Novák</prijmeni>
    <pocetEditaci>152</pocetEditaci>
</wikipedista>
```

XSD dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="wikipedista">
  <xs:complexType>
      <xs:sequence>
          <xs:element name="jmeno" type="xs:string"></xs:element>
          <xs:element name="prijmeni" type="xs:string"></xs:element>
          <xs:element name="pocetEditaci" type="xs:integer"></xs:element>
      </xs:sequence>
      <xs:attribute name="uid" type="xs:string"></xs:attribute>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Připojení XSD k dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
<wikipedista xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="wikipedista.xsd"
             uid="Novak">
    <jmeno>Jirka</jmeno>
    <prijmeni>Novák</prijmeni>
    <pocetEditaci>152</pocetEditaci>
</wikipedista>
```

XML schéma

XSD dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xsd:simpleType name="ProductCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}[A-Z]{2}"/>
   <xsd:restriction>
</xsd:simpleType>

<xsd:element name="ProductCode" type="ProductCodeType"/>
```

DB2 a XML schéma

- XML schéma je nepovinné. Vhodné používat, pouze pokud je to nutné (CPU).
- Kontrola se provádí na úrovni XML dokumentu.
- XML sloupec může mít i více XML schémat.
- Lze používat i konfliktní XML schémata.
- XML sloupec může obsahovat ověřené i neověřené XML dokumenty.

- XML Schéma Repository (XSR)
  - XML schéma musí být před použitím registrováno.

```
REGISTER XMLSCHEMA uri FROM 'file://cesta/soubor.xsd' [WITH 'file://cesta/propoerties.xml'] AS relational_id
ADD XMLSCHEMA DOCUMENT TO relational_id ADD uri FROM 'file://cesta/soubor.xsd' [WITH 'file://cesta/propoerties.xml']
[ENABLE DECOMPOSITION]
COMPLETE XMLSCHEMA relational_id [WITH 'file://cesta/propoerties.xml'] [ENABLE DECOMPOSITION]
DROP XSROBJECT relational_id
SELECT objectname, targetnamespace, schemalocation FROM syscat.xsrobjects WHERE objecttype='S' and status='C'
```

```
INSERT INTO DEPT VALUES (1,'XML')
INSERT INTO DEPT VALUES (2, XMLVALIDATE ('XML' according to xmlschema id "dbschema.id"))
INSERT INTO DEPT VALUES (2, XMLVALIDATE ('XML' according to xmlschema uri "http://..."))
ALTER TABLE ADD CONSTRAINT xml_chk CHECK (xmldoc IS VALIDATED ACCORING TO sch.doc)
SELECT xmldoc FROM DEPT WHERE xmldoc IS VALIDATED
```

Evoluce XML schéma

- Většina schémat lze postupně vyvíjet.
  - Přidání dalších elementů a atributů.
  - Zvýšení počtu opakovaných elementů.

- Aktualizace XML schéma
  - Příkaz UPDATE XML SCHEMA.
  - Zaregistrování obou schémat.
  - Kontrola jejich kompatibility.
  - Nahrazení starého schématu novým.
  - Existující XML dokumenty se tváří, že byly zkontrolovány novým XML schématem. Existující dokumenty nejsou dotčeny.
  - Nové schéma má stejné objectID jako staré schéma. Dokumenty odkazují do původního místa.

- Příklad
  - `REGISTER XMLSCHEMA schema_v1`
  - `INSERT INTO T VALUES (XMLVALIDATE('xml')`
  - `REGISTER XMLSCHEMA schema_v2`
  - `UPDATE XMLSCHEMA schema_v1 WITH schema_v2 DROP NEW SCHEMA`
  - `INSERT INTO T VALUES (XMLVALIDATE('xml')`

Mapování XML na relační strukturu

- Mapování je definované pomocí XML Schéma
  - XML jde rozložit do více sloupců
  - Lze definovat uživatelské výrazy pro transformaci (UDF, podmínky, podmínky na strukturu XML,
  - XML lze uložit paralelně i do XML sloupce.
  - Lze spustit i validaci.
  - Lze definovat referenční integrity.
  - Mapování může směřovat do různých tabulek a sloupců.
  - <xsd:element name="phone" type="xsd:string" db2-xdb:rowSet="employee_tab" db2-xdb:column="phone"/>
- Případy, kdy je vhodné XML uložit do relační struktury
  - Stávající SQL aplikace potřebují data z XML
  - Aplikace má pouze SQL API pro připojení k databázi
  - XML již není pro další zpracování dále potřeba
  - XML je jednoduché a velmi pravidelné
  - Změny XML schématu je málo časté.
- Nevhodné
  - XML je velmi komplexní.
  - XML schéma se často mění.
  - Zpracování přes XML může být rychlejší než SQL přes mnoho tabulek.

Děkuji za pozornost