ОНЛАЙН-ОБРАЗОВАНИЕ

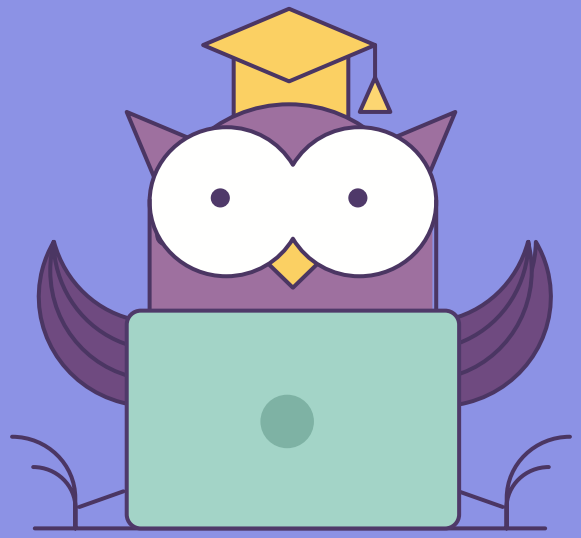
OTUS

XML - индексы

Курс “MS SQL Server разработчик”

Павел Стрекалов

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Или напишите, какие есть проблемы

Содержание

1. XML - индексы в SQL Server
2. Первичные xml-индексы
3. Вторичные xml-индексы

XML-индексы

- Для XQuery-запросов, когда запрашиваем часть xml-данных (xquery)
- Отличаются от “обычных” индексов
  - Первичный
  - Вторичный (path, value, property)
- Первичный ключ на таблице
- Кластерный индекс

ORDPATH

```
<BOOK ISBN="1-55860-438-3">
    <SECTION>
        <TITLE>Bad Bugs</TITLE>
        Nobody loves bad bugs.
        <FIGURE CAPTION="Sample bug"/>
    </SECTION>
    <SECTION>
        <TITLE>Tree Frogs</TITLE>
        All right-thinking people
        <BOLD> love </BOLD> tree frogs.
    </SECTION>
</BOOK>
```

Figure 1. Sample XML data

- Hierarchical dot-separated labels assigned to nodes
  - Compressed binary form used internally
- Positive odd integers assigned initially
- Negative, even integers used for insertions

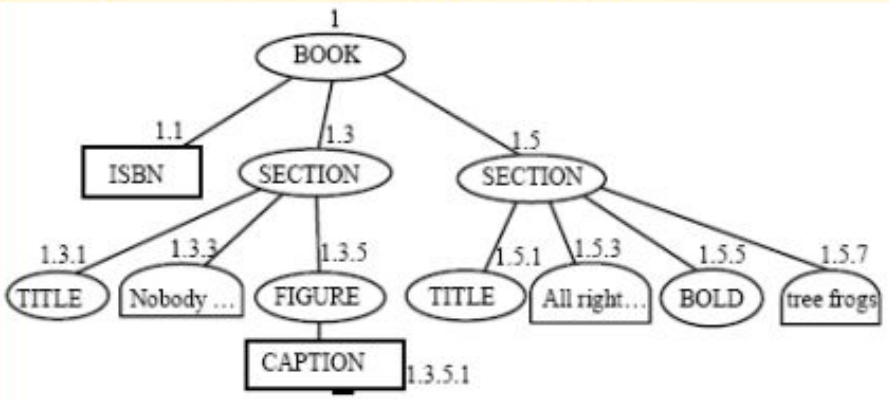


Figure 2. ORDPATH Node Label

Primary XML-индекс

OTUS

- Store "infoset" items of XML nodes in B+ tree

ID	ORDPATH	TAG	NODE_TYPE	VALUE	PATH_ID
1	1	1(BOOK)	1(Element)	Null	#1
	1.1	2(ISBN)	2(Attribute)	'1-55860-438-3'	#2#1
	1.3	3(SECTION)	1	Null	#3#1
	1.3.1	4(TITLE)	1	'Bad Bugs'	#4#3#1
	1.3.3	10(TEXT)	4(Value)	'Nobody loves Bad bugs.'	#10#3#1
	1.3.5	5(FIGURE)	1	Null	#5#3#1
	1.3.5.1	6(CAPTION)	2	'Sample bug'	#6#3#1
	1.5	3(SECTION)	1	Null	#3#1
	1.5.1	4(TITLE)	1	'Tree frogs'	#4#3#1
	1.5.3	10(TEXT)	4	'All right-thinking people'	#10#3#1
	1.5.5	7(BOLD)	1	'love'	#7#3#1
	1.5.7	10(TEXT)	4	'tree frogs'	#10#3#1

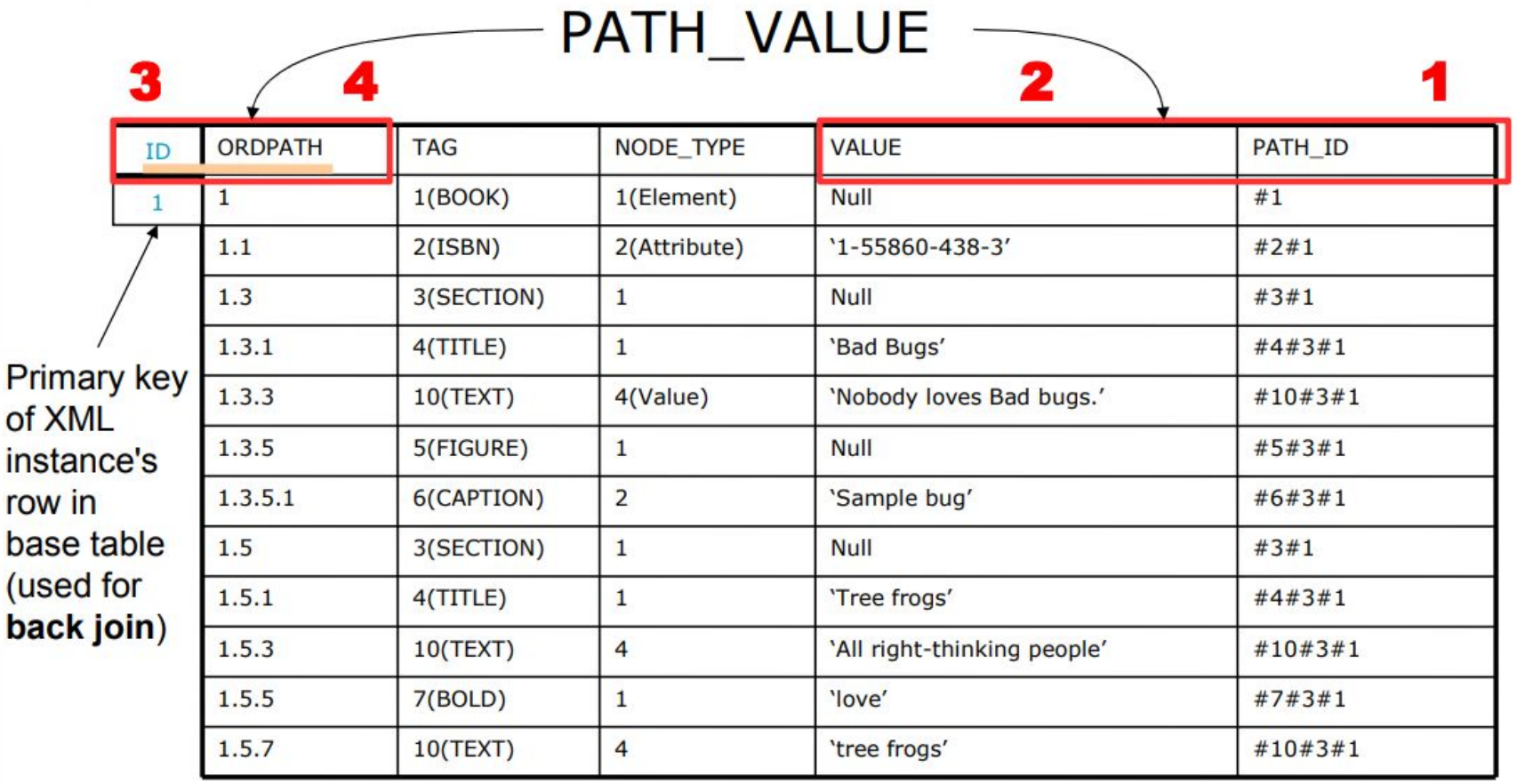
Primary key of XML instance's row in base table (used for **back join**)

- The primary XML index is clustered in document order
- Each path expression is evaluated by scanning all rows in the primary XML index for a given XML instance
- Performance slows down for large XML values

Secondary индекс

- **Secondary indexes** on the primary XML index optimize for certain classes of queries
  - Four introduced in this paper:
    PATH/PATH_VALUE, PROPERTY, VALUE, Content indexing
- Secondary XML indexes help with bottom-up evaluation
  - Nodes found in secondary XML indexes can be back joined with primary XML index, enabling continuation of query execution with those nodes
  - Leads to significant performance gains

Secondary - PATH

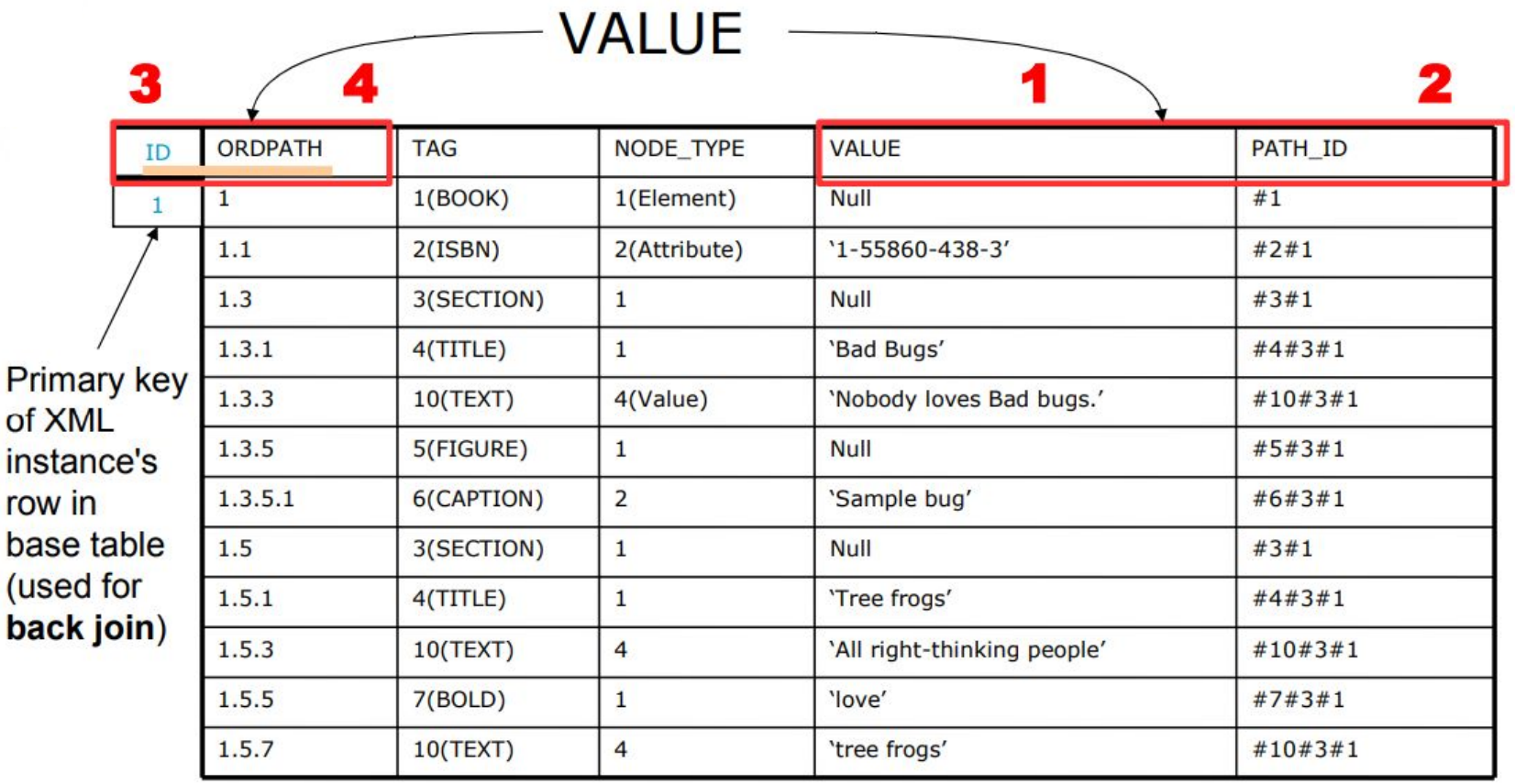


ID	ORDPATH	TAG	NODE_TYPE	VALUE	PATH_ID
1	1	1(BOOK)	1(Element)	Null	#1
	1.1	2(ISBN)	2(Attribute)	'1-55860-438-3'	#2#1
	1.3	3(SECTION)	1	Null	#3#1
	1.3.1	4(TITLE)	1	'Bad Bugs'	#4#3#1
	1.3.3	10(TEXT)	4(Value)	'Nobody loves Bad bugs.'	#10#3#1
	1.3.5	5(FIGURE)	1	Null	#5#3#1
	1.3.5.1	6(CAPTION)	2	'Sample bug'	#6#3#1
	1.5	3(SECTION)	1	Null	#3#1
	1.5.1	4(TITLE)	1	'Tree frogs'	#4#3#1
	1.5.3	10(TEXT)	4	'All right-thinking people'	#10#3#1
	1.5.5	7(BOLD)	1	'love'	#7#3#1
	1.5.7	10(TEXT)	4	'tree frogs'	#10#3#1

- Helps with searches for a path + value match (e.g. /BOOK/SECTION[TITLE="Tree frogs"])

Creates a secondary XML index on columns built on path values and node values in the primary XML index. In the PATH secondary index, the path and node values are key columns that allow efficient seeks when searching for paths.

Secondary - VALUE

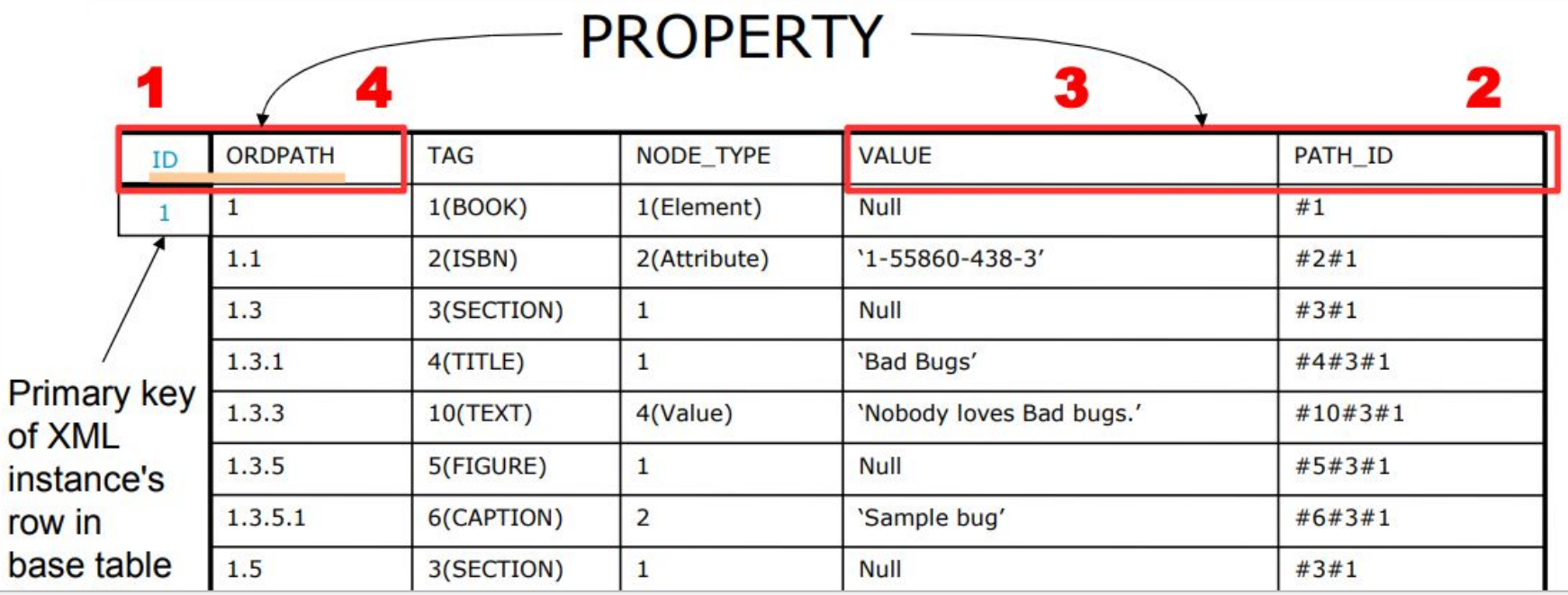


ID	ORDPATH	TAG	NODE_TYPE	VALUE	PATH_ID
1	1	1(BOOK)	1(Element)	Null	#1
	1.1	2(ISBN)	2(Attribute)	'1-55860-438-3'	#2#1
	1.3	3(SECTION)	1	Null	#3#1
	1.3.1	4(TITLE)	1	'Bad Bugs'	#4#3#1
	1.3.3	10(TEXT)	4(Value)	'Nobody loves Bad bugs.'	#10#3#1
	1.3.5	5(FIGURE)	1	Null	#5#3#1
	1.3.5.1	6(CAPTION)	2	'Sample bug'	#6#3#1
	1.5	3(SECTION)	1	Null	#3#1
	1.5.1	4(TITLE)	1	'Tree frogs'	#4#3#1
	1.5.3	10(TEXT)	4	'All right-thinking people'	#10#3#1
	1.5.5	7(BOLD)	1	'love'	#7#3#1
	1.5.7	10(TEXT)	4	'tree frogs'	#10#3#1

- Helps when we're looking for a data value but have a wildcard in the path (e.g. `/BOOK/SECTION[FIGURE/@*="Sample Bug"]`)

Creates a secondary XML index on columns where key columns are (node value and path) of the primary XML index.

Secondary - PROPERTY



ID	ORDPATH	TAG	NODE_TYPE	VALUE	PATH_ID
1	1	1(BOOK)	1(Element)	Null	#1
	1.1	2(ISBN)	2(Attribute)	'1-55860-438-3'	#2#1
	1.3	3(SECTION)	1	Null	#3#1
	1.3.1	4(TITLE)	1	'Bad Bugs'	#4#3#1
	1.3.3	10(TEXT)	4(Value)	'Nobody loves Bad bugs.'	#10#3#1
	1.3.5	5(FIGURE)	1	Null	#5#3#1
	1.3.5.1	6(CAPTION)	2	'Sample bug'	#6#3#1
	1.5	3(SECTION)	1	Null	#3#1

```
SELECT CatalogDescription.value('(/ProductDescription/@ProductModelID)[1]', 'int'),
       CatalogDescription.value('(/ProductDescription/@ProductModelName)[1]', 'varchar(30)')
FROM Production.ProductModel
WHERE ProductModelID = 19
```

- Helps find (possibly multi-valued) properties of object with known ID and PATH_ID

Creates a secondary XML index on columns (PK, path and node value) of the primary XML index where PK is the primary key of the base table.

Secondary xml-index

Ниже приведены некоторые рекомендации по созданию вторичных индексов.

- Если при работе с XML-столбцами часто используются выражения пути, вторичный XML-индекс PATH, скорее всего, ускорит обработку данных. Типичный пример — выполнение метода **exist()** для XML-столбцов в предложении WHERE инструкции Transact-SQL.

- Если с использованием выражений пути извлекаются множественные значения из отдельных экземпляров XML, может принести пользу кластеризация путей в пределах каждого экземпляра XML в индекс PROPERTY. Этот сценарий обычно имеет место при работе с наборами свойств, когда извлекаются свойства объекта и известно значение его первичного ключа.

- Если запрашиваются значения экземпляров XML, не зная имен элементов или атрибутов, содержащих эти значения, следует подумать о создании индекса VALUE. Как правило, это имеет место при уточняющем запросе по осям нижних уровней, например //author[last-name="Howard"], где элементы <author> могут встречаться на любом уровне иерархии. Кроме того, такая ситуация встречается при обработке запросов с символами-шаблонами (например, /book [@* = "novel"], где в запросе выполняется поиск элементов <book>, имеющих некоторый атрибут со значением "novel").

[Документация SQL Server — XML Indexes]

Селективные XML-индексы

```
CREATE SELECTIVE XML INDEX SXI_index
ON Tbl(xmlcol)
FOR
(
    pathTitle = '/book/title/text()' AS XQUERY 'xs:string',
    pathAuthors = '/book/authors' AS XQUERY 'node()',
    pathId = '/book/id' AS SQL NVARCHAR(100)
)
```

Selective XML Indexes (SXI)

Precision Indexing: Basics of Selective XML Indexes in SQL Server 2012

Материалы

- Документация SQL Server — XML Indexes
- Статья "Indexing XML Data Stored in a Relational Database" (презентация)
- Getting Started With XML Indexes

ДЕМО

XML индексы