Strategic Sponsor

Microsoft

Gold Sponsors

PYRAMID ANALYTICS

KRAJOWY REJESTR DŁUGÓW www.krd.pl

devart Database Interaction Solutions

Silver Sponsors

Profisee®

SQL EXPERT.pl

CREDIT SUISSE

Technical Partners

Microsoft

PTI POLSKIE TOWARZYSTWO INFORMATYCZNE ODDZIAŁ GÓRNOŚLĄSKI

Academic Partners

Wyższa Szkoła Zarządzania i Bankowości w Krakowie

Wyższe Szkoły Bankowe

Media Partners

GEEKCLUB.pl wss.pl

ERP24.pl

4programmers.net

IT professional

IN FACT PR

itnews24

virtual-it.pl

IT news POLSKA

VirtualStudy

MAGAZYN IT PL

Made in Wro Do IT here!

ERP-view.pl Portal rozwiązań IT w biznesie

Duże obiekty w SQL Server

Łukasz Grala

Architekt Platformy Danych i Rozwiązań Business Intelligence
MVP SQL Server

Łukasz Grala

MVP SQL Server | MCT | MCSE

- Architekt i trener - Data Platform & Business Intelligence Solutions
- Prelegent na licznych konferencjach
- Wykładowca i autor publikacji
- Doktorant Politechnika Poznańska – Wydział Informatyki (Bazy i hurtownie danych, Machine Learning, Mining of Massive Datasets)
- Posiada liczne certyfikaty
- Od 2010 roku co roku wyróżniany nagrodą MVP w kategorii SQL Server
- Lider PLSSUG Poznań (lukasz.grala@plssug.org.pl)

Agenda

- Co mamy na myśli mówiąc duże obiekty (ang. LOB)?
- Duże obiekty binarne i znakowe
- Duże dokumenty XML
- Dokumenty (pliki) w bazie danych
- A co z NoSQL?
- Podsumowanie

What is LOB?

- Binary files
- Binary data
- Characters data
- Documents, files
- XML file

Data types

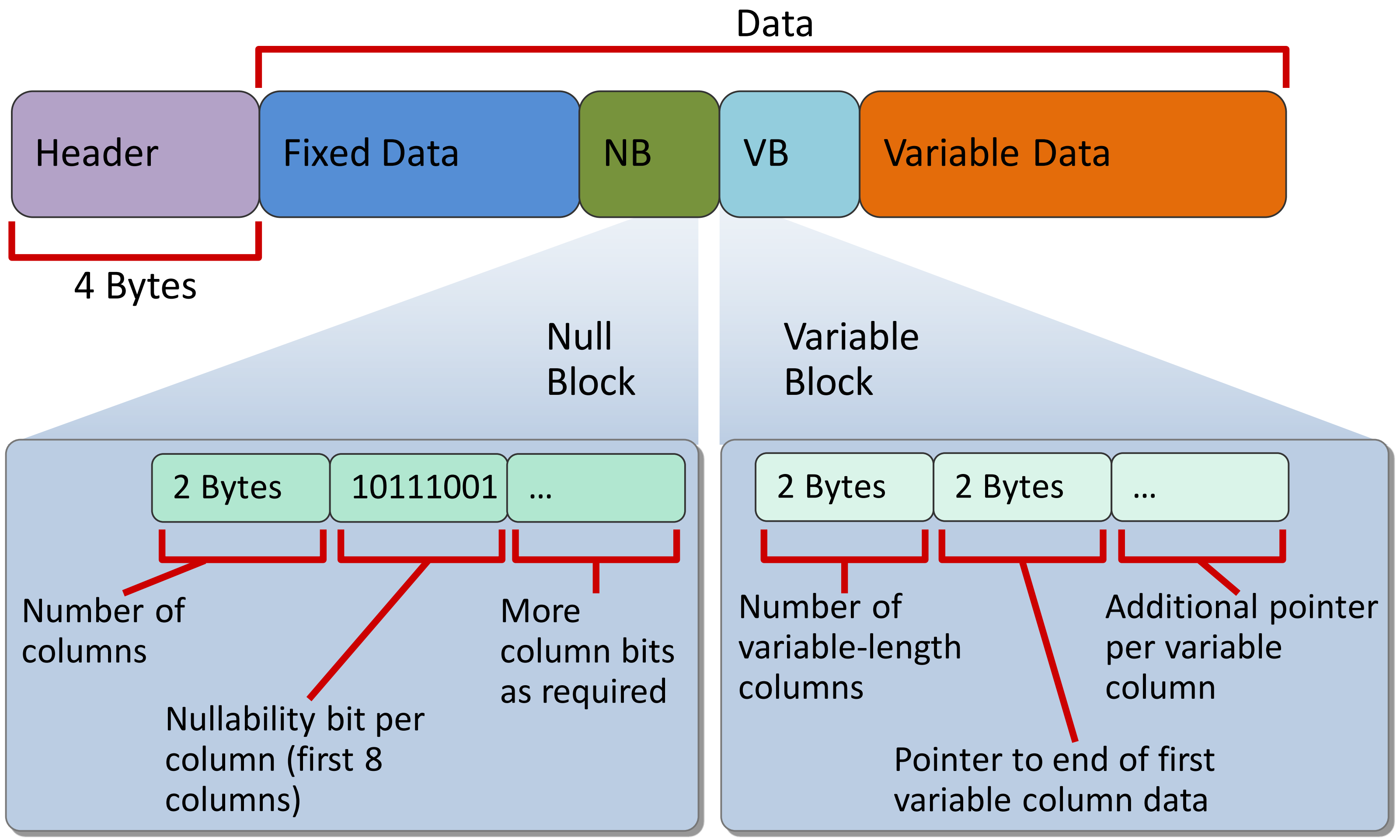
- varchar(n), nvarchar(n)
- varchar(max), nvarchar(max)
- Varbinary(n)/varbinary(max)
- text, image
- XML

How SQL Server storage data?

Table rows are stored on Disk in 8kb units, named "pages".

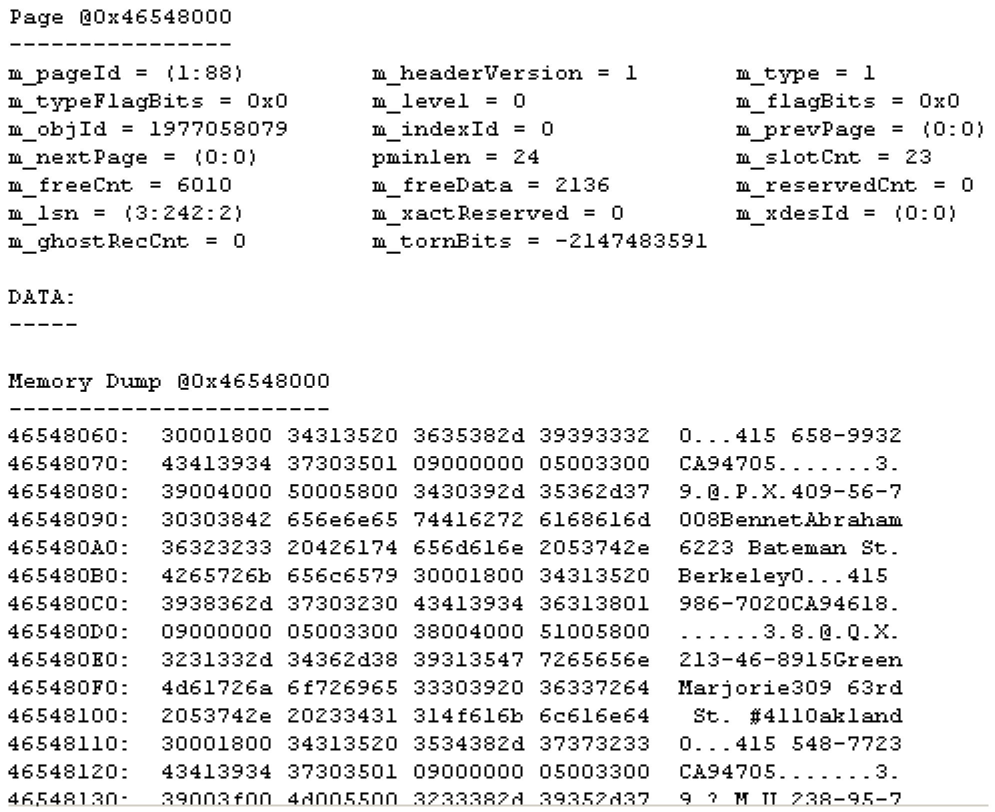
- Page – 8192 byte
- Data – 8060 byte
- Extent - 8 x Page

How SQL Server Organizes Data in Rows?



Data
Header
Fixed Data
NB
VB
Variable Data
4 Bytes
Null Block
Variable Block
2 Bytes
10111001
…
Number of columns
Nullability bit per column (first 8 columns)
More column bits as required
2 Bytes
2 Bytes
…
Number of variable-length columns
Pointer to end of first variable column data
Additional pointer per variable column

Page



```
Page @0x46548000
----------------
m_pageId = (1:88)            m_headerVersion = 1         m_type = 1
m_typeFlagBits = 0x0         m_level = 0                 m_flagBits = 0x0
m_objId = 1977058079         m_indexId = 0               m_prevPage = (0:0)
m_nextPage = (0:0)           pminlen = 24                m_slotCnt = 23
m_freeCnt = 6010             m_freeData = 2136           m_reservedCnt = 0
m_lsn = (3:242:2)            m_xactReserved = 0          m_xdesId = (0:0)
m_ghostRecCnt = 0            m_tornBits = -2147483591

DATA:
-----

Memory Dump @0x46548000
-----------------------
46548060:  30001800 34313520 3635382d 39393332  0...415 658-9932
46548070:  43413934 37303501 09000000 05003300  CA94705.......3.
46548080:  39004000 50005800 3430392d 35362d37  9.@.P.X.409-56-7
46548090:  30303842 656e6e65 74416272 6168616d  008BennetAbraham
465480A0:  36323233 20426174 656d616e 2053742e  6223 Bateman St.
465480B0:  4265726b 656c6579 30001800 34313520  Berkeley0...415
465480C0:  3938362d 37303230 43413934 36313801  986-7020CA94618.
465480D0:  09000000 05003300 38004000 51005800  ......3.8.@.Q.X.
465480E0:  3231332d 34362d38 39313547 7265656e  213-46-8915Green
465480F0:  4d61726a 6f726965 33303920 36337264  Marjorie309 63rd
46548100:  2053742e 20233431 314f616b 6c616e64   St. #411Oakland
46548110:  30001800 34313520 3534382d 37373233  0...415 548-7723
46548120:  43413934 37303501 09000000 05003300  CA94705.......3.
46548130:  39003f00 4d005500 3233382d 39352d37  9.?.M.U.238-95-7
```

Page

DEMO 1

HOW SQL SERVER STORAGE DATA LOB?

varchar, text, …

DEMO 2

XML Data Type

- Is a native data type for XML
- Lets you store XML documents and fragments
- Is used for columns, variables, or parameters
- Exposes methods to query and modify XML

XML Schema Collections

- SQL Server provides native storage of XML data with xml data type
- Ypu can optionally associate XSD schemas with an xml data type through an XML schema collection
- The XML schema collection:
  - Stores the imported XML schemas
  - Validates XML instances
  - Types the XML data as it stored in the database

Untyped XML

You may choose to store any well-formed XML. One reason is that you might not have a schema for the XML data. Another reason is that you might want to avoid the processing overhead involved in validating the XML against the XML schema collection. For complex schemas, validating the XML can involve substantial work. The following example shows a table being created with an untyped XML column:

```
CREATE TABLE App.Settings
(   SessionID int PRIMARY KEY,
    WindowSettings xml
);
```

Any well-formed XML could be stored in the WindowSettings column up to the maximum size of a SQL Server XML object which is currently 2GB.

Typed XML

You may wish to have SQL Server validate your data against a schema. You might want to take advantage of storage and query optimizations based on the type information or want to take advantage of this type information during the compilation of your queries. The following example shows the same table being created with a typed XML column:

```
CREATE TABLE App.Settings
(   SessionID int PRIMARY KEY,
    WindowSettings xml (SettingsSchemaCollection)
);
```

In this case, a schema collection called SettingsSchemaCollection has been defined and SQL Server will not allow data to be stored in the WindowSettings column if it does not meet the requirements of at least one of the XML schemas in the SettingsSchemaCollection.

Content

```
CREATE TABLE App.Settings
( SessionID int PRIMARY KEY,
WindowSettings xml (SettingsSchemaCollection)
);
```

This is equivalent to defining the table by the following code as the keyword CONTENT is the default value for typed XML declarations:

```
CREATE TABLE App.Settings
( SessionID int PRIMARY KEY,
WindowSettings xml (CONTENT
SettingsSchemaCollection)
);
```

Document

```
CREATE TABLE App.Settings
( SessionID int PRIMARY KEY,
WindowSettings xml (DOCUMENT
SettingsSchemaCollection)
);
```

In this case, XML fragments would not be able to be stored in the WindowSettings column. Only well-formed XML documents could be stored. For example, a column that is intended to store a customer order can then be presumed to actually hold a customer order and not some other type of XML document.

XML Indices

Create XML index on XML column

```
CREATE PRIMARY XML INDEX idx_1 ON docs (xDoc)
```

Create secondary indexes on tags, values, paths

Creation:

- Single-threaded only for primary XML index
- Multi-threaded for secondary XML indexes

Uses:

- Primary Index will always be used if defined (not a cost based decision)
- Results can be served directly from index
- SQL's cost based optimizer will consider secondary indexes

Maintenance:

- Primary and Secondary Indices will be efficiently maintained during updates
- Only subtree that changes will be updated
- No online index rebuild ☹
- Clustered key may lead to non-linear maintenance cost ☹

Schema revalidation still checks whole instance

XML Indices

- **PRIMARY XML Index** – Use when lots of Xquery

  FOR VALUE – Useful for queries where values are more selective than paths such as //*[.="Seattle"]

  FOR PATH – Useful for Path expressions: avoids joins by mapping paths to hierarchical index (HID) numbers. Example: /person/address/zip

  FOR PROPERTY – Useful when optimizer chooses other index (for example, on relational column, or FT Index) in addition so row is already known

Selective Index

CREATE SELECTIVE INDEX

xml, …

DEMO 3

Filestream Implementation

- Filestream is a database/file system hybrid
- Enable Filestream at an OS level (per-instance)
- Filestream filegroups declared using DDL
  - Filestream storage is tied to a database
- The filegroup is mapped to a directory
  - Sparse tx log, metadata, data live here
  - Must be NTFS file system
  - Caution: Files deletable from file system with appropriate permissions
- VARBINARY(MAX) columns defined with FILESTREAM attribute
  - Filestream storage not available for other large types
  - Column's data is stored in the file system

Filestream Enhancements

- SQL Server 2008 R2
  - Snapshot Isolation Level Support
- SQL Server 2012
  - Multiple Containers per Filegroup
    - Performance Improvement for T-SQL and File I/O
    - Up to 5x read improvement
    - Linear scaling with multiple threads
  - Full Always-On Support
    - Access through virtual network name
    - Transparent reconnect
    - No support for Filestream and Database Mirroring
  - MAXSIZE can be specified for Filestream containers

Filestream Enhancements

- SQL Server 2014

FILESTREAM

DEMO 4

FileTable

- FileTable
  - Built over the Filestream feature
  - Allows access by existing applications
  - Win32 API access
    - Provides data read and write capabilities
    - Remote access using SMB protocol
    - Metadata manipulation handled by FileTable
    - Other capabilities handled by the file system
  - Hierarchical namespace

FileTable

- FileTable is a fixed format table (SQL 2012)
  - Uses filestream storage and container
  - Includes file system properties as columns
  - HierarchyID columns presents data as "synthesized" hierarchical file system share
- Available as file system share or T-SQL table
- Can be maintained through the share directly
  - Some columns maintainable through T-SQL
  - Preserves filename and suffix
- Allows non-transactional access through share

FileTable Setup

- Enable Filestream feature - prerequisite
- ALTER database to
  - Allow non-transactional access
  - Specify directory name for share
- CREATE tablename AS FILETABLE
  - Specify filetable_directory (share subdirectory)
  - Specify collation for Name column
  - No partitioned FileTables
- File access at:
  - \\machine\share\dbdirectory\ftdirectory
    - Share is instance share name
    - Dbdirectory is name specified on ALTER DATABASE
    - Ftdirectory is name specified on CREATE [tablename] AS FILETABLE

FileTable Share Behaviors

- Share is real Windows Share
  - Even antivirus programs think it's real
  - Not available if SQL Server not running
  - Uses SMB protocol – open TCP 139, 445 for remote file access
  - Cannot layer FileTable on existing share
- Some limitations
  - Memory mapped files not supported
    - Can't open with Notepad, Paint files locally
  - 15 level subdirectory limit
  - Windows Explorer limit 260 byte names
    - NTFS (and FileTable) allow more

FileTable Columns

Name	Type	Description
path_locator	hierarchyid	Position of this node in the hierarchical file namespace.
parent_path_locator	hierarchyid	HierarchyID of the parent directory (computed column)
stream_id	uniqueidentifier	Unique Id for Filestream data
file_stream	varbinary(max) filestream	Filestream data
file_type	nvarchar(255)	Type of the file. Usable for full-text index
cached_file_size	bigint	Size of the filestream(cached value)
name	nvarchar(255)	File/Folder Name
creation_time	datetime2	Creation Time
last_write_time	datetime2	LastWriteTime
last_access_time	datetime2	LastAccessTime
is_directory	bit	TRUE for directories.
is_offline	bit	Offline attribute
is_hidden	bit	Hidden attribute
is_readonly	bit	Read Only attribute
is_archive	bit	Archive attribute
is_system	bit	System attribute
is_temporary	bit	Temporary attribute

T-SQL Access

- Insert
  - Many columns use defaults
- Update
  - Can update stream without changing datetimes
  - No WRITE method
- Delete
  - Fails on directories containing directories or files
- BCP
  - Can disable system constraints and use BCP without constraints
- SELECT INTO does not create a FileTable

Programming Functions

- T-SQL built-ins allow portable code
  - GetFileNamespacePath()
    - Accessor on each row
    - GetFileNamespacePath(1) gets full UNC pathname
  - FileTableRootPath()
    - Gets root path of the table
    - FileTableRootPath('tablename') gets root path with table name included
  - GetPathLocator()
    - Retrieves hierarchyid value for UNC name

FILETABLE

DEMO 5

NoSQL

- varchar(max) / FILESTREAM ?

- Azure DocumentDB
- MongoDB
- Hbase (HDInsight)
- Blob Storage

Pytania?

Kontakt:

lukasz@grala.biz

lukasz.grala@plssug.org.pl

lukasz@sqlexpert.pl

Blogi:

blog.sqlexpert.pl (en/pl)

SQLResearch.com (pl)

MVP Microsoft® Most Valuable Professional

SQL EXPERT.pl

PTI ODDZIAŁ WIELKOPOLSKI POLSKIE TOWARZYSTWO INFORMATYCZNE

POLITECHNIKA POZNAŃSKA POZNAN UNIVERSITY OF TECHNOLOGY

Strategic Sponsor

Gold Sponsors

Silver Sponsors

Technical Partners

Academic Partners

Media Partners