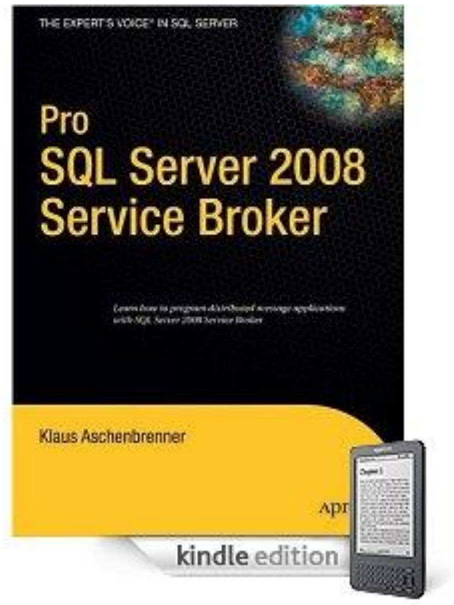
SQL Server 2008 Database Internals

Klaus Aschenbrenner
SQL Server Consultant
http://www.csharp.at
http://twitter.com/Aschenbrenner

Bring your SQL Server installations to a new level of excellence!

About me

- Independent SQL Server Consultant
- International Speaker, Author
- „Pro SQL Server 2008 Service Broker“
- http://www.csharp.at
- http://twitter.com/Aschenbrenner



Agenda

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions

Agenda

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions

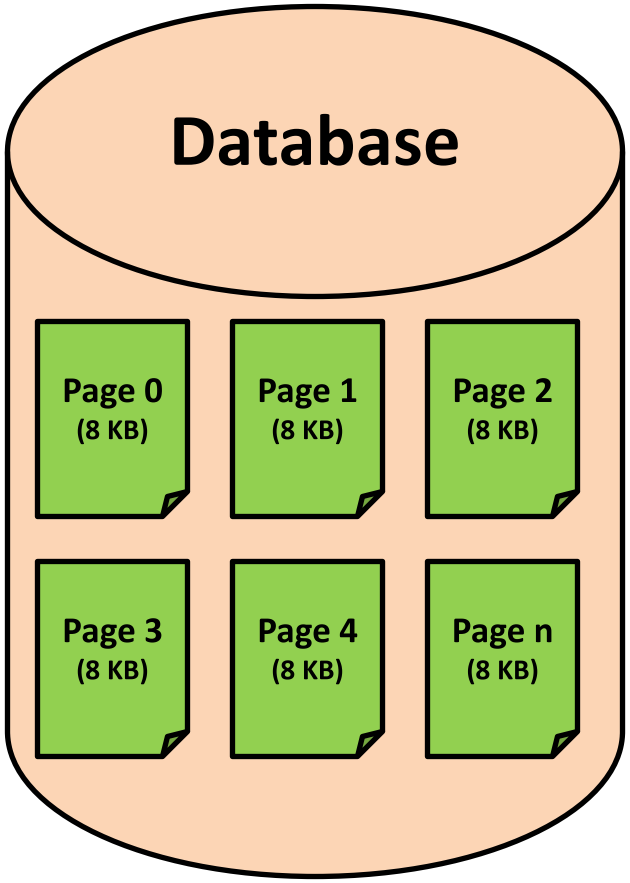
Why such a topic?

- SQL Server is a black-box to ***some*** users
  - SELECT, INSERT, UPDATE, DELETE – what else?
- No awareness about data pages
- Bad database designs
  - Fixed vs. variable column length
- Bad index design
  - Wrong clustering key, wide keys
  - Bookmark Lookups
  - Dependency Clustered/Non-Clustered Index

Database Basics

- Stores data & index
- Splitted into so-called „pages“
  - 8 KB (8.192 bytes) chunks
  - Different kinds of pages
- Unit of I/O
  - SQL Server does I/O on the page-level
  - Reads data page by page
  - Writes data page by page

Database Basics - Illustriated



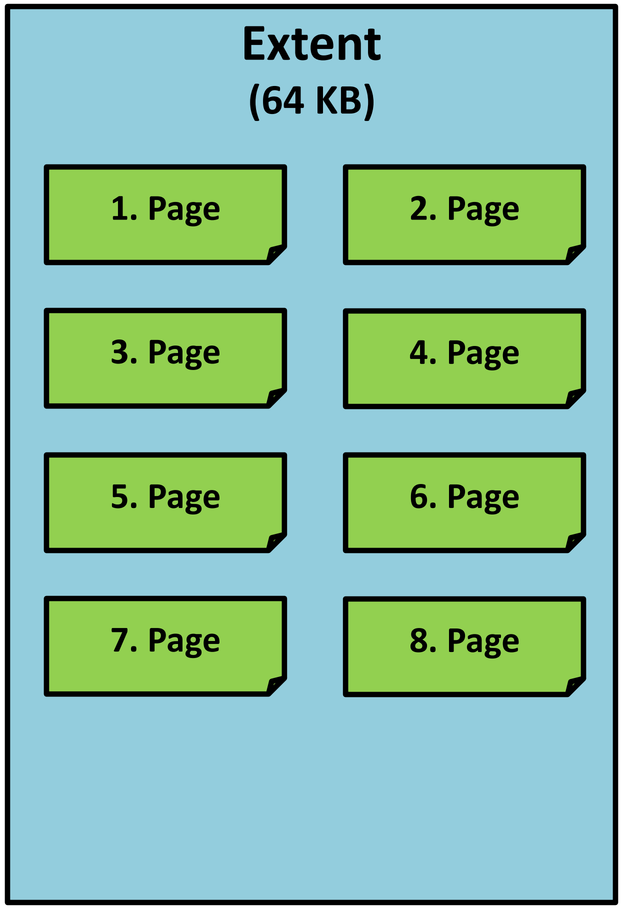
Extents

- Pages are grouped into extents
  - Consists of 8 logically grouped pages
  - 64 KB (65.536 bytes) chunk (8 x 8 KB)
- 2 kinds of extents
  - Uniform Extent
  - Mixed Extent

Uniform/Mixed Extent

- Uniform Extent
  - Belongs to **one** table/index
  - All 8 pages belongs to the same database object
- Mixed Extent
  - Belongs to **different** tables/indexes
  - Up to 8 different database objects per mixed extent

Extent - Illustriated



Uniform/Mixed Extent

- New tables/indexes
  - Always created in mixed extents
  - At no time an uniform extent is allocated
- Existing tables/indexes
  - If occupies 8 pages: all new pages (page 9, 10, 11, ...) are allocated in an new uniform extent
- Conclusion
  - Minimum/Maximum: 1-8 mixed extent
  - n uniform extents

Extent Management

- Done with 2 special kinds of pages
  - Global Allocation Map Pages (GAM)
  - Shared Global Allocation Map Pages (SGAM)
  - Pages are (again) 8 KB large

GAM page

- Stores if an extent is used or not
- Each extent is represented with 1 bit
- Bit not set: extent is used
- Bit set: extent is free
- 8.000 bytes of net space per GAM page available (excluding page header)
  - 64.000 bits available
  - One GAM covers a 4 GB data range (64.000 x 64 / 1024 / 1024)
- One GAM page per 4 GB of data needed

SGAM page

- Stores if an extent is used as an mixed extent...
- ... and has at least one free page available
- Each extent is represented with 1 bit
- Bit not set
  - Uniform Extent or no more free pages available in the mixed extent
- Bit set
  - Mixed Extent with at least one free page
- 64.000 bits available (same as GAM page)

GAM/SGAM Bit Settings

Description	GAM Bit	SGAM Bit
Free, not used	1	0
Uniform Extent or full Mixed Extent	0	0
Mixed Extent with at least one free page	0	1

GAM/SGAM Pages

- GAM page is the third page within a database file
- SGAM page is the fourt page within a database file
- GAM/SGAM pages occuring after 511.230 pages

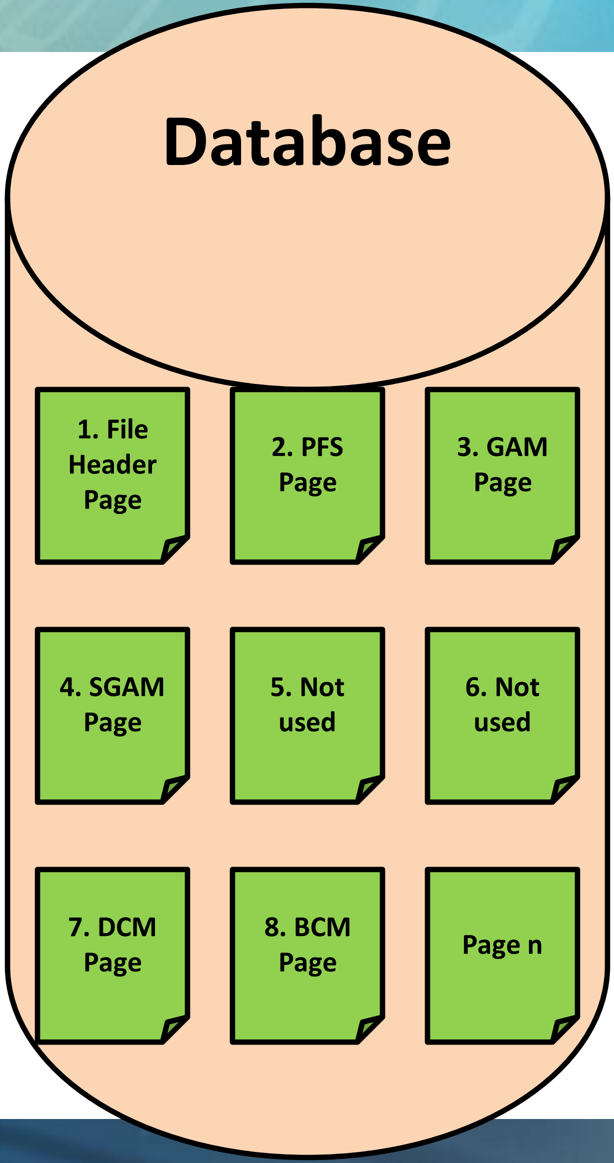
Database Page Structure

- 1st page: File Header Page
- 2nd page: Page Free Space Page (PFS)
  - Occurs after every 8.088 page
  - Stores how much space is left in each page
  - Used by INSERT and UPDATE operations
  - Each data page is represented by 1 byte
- 3rd/4rd page: GAM/SGAM page
  - Occurs after every 511.230 page
- 5th/6th page: not used

Database Page Structure

- 7th page: Differential Changed Map Page (DCM page)
  - Stores which Extents have been changed until the last database backup
  - Represents a 4 GB interval
  - Occurs after every 511.230 page
- 8th page: Bulk Changed Map Page (BCM page)
  - Stores which Extents have been used in a minimally or bulk-logged operation
  - Represents a 4 GB internal
  - Occurs after every 511.230 page

Database Page Structure



Agenda

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions

Table Metadata

- Provided System-Views
  - sys.tables
  - sys.columns
  - sys.indexes
  - sys.check_constraints
  - sys.default_constraints
  - sys.key_constraints
  - sys.foreign_keys
  - sys.partitions
  - sys.allocation_units

sys.indexes

- One record per index per table
- 3 possible scenarios
  - Clustered Index
  - Non-Clustered Index
  - Table without an index (Heap)
- Heap
  - Unordered table data without any index
  - Column name: NULL
  - Column index_id: 0

sys.indexes

- Non-Clustered Index
  - index_id: 2 to 250, 256 – 1005
  - 251 – 255 are internally reserved by SQL Server
  - Maximum: 999 possible Non-Clustered Index per table
- Maximum: 1.000 records in sys.indexes per table
  - 999 Non-Clustered Indexes record
  - 1 Clustered Index/Heap record (index_id 1 or NULL)

sys.partitions

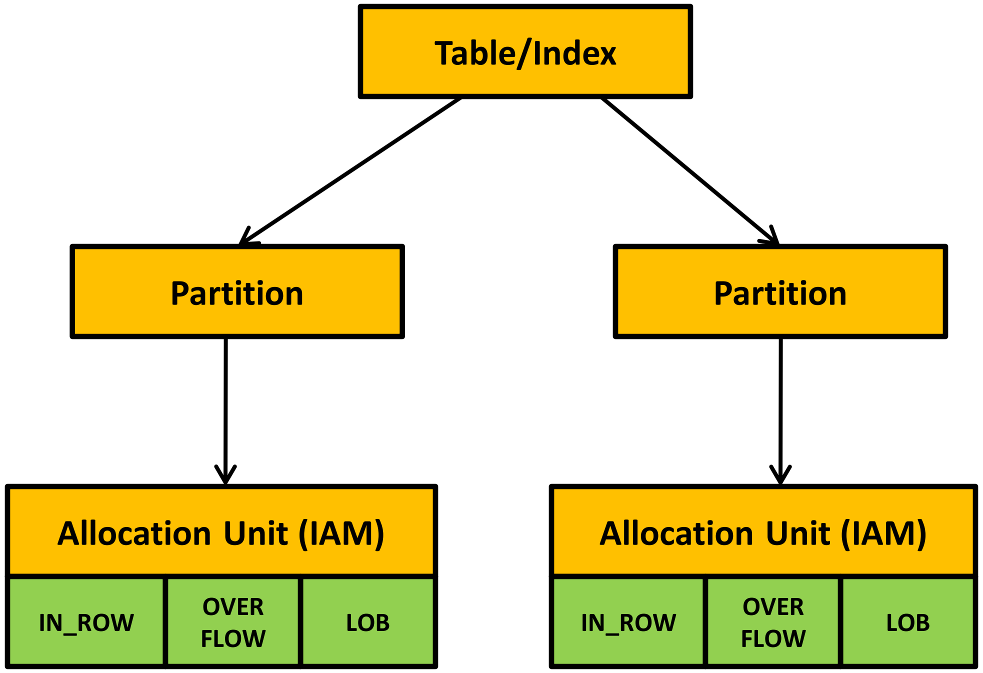
- New in SQL Server 2005
- A table/index can be splitted into several partitions
- Partitions can be spread across file groups
- At least one record for each heap table/index
- Maximum: 1.000 partitions per table/index
- Internal name of such a record
  - Hobt – *Heap or B-Tree*
  - *„Hobbit“*

Data stored in partitions

- In-Row Data Pages
  - IN_ROW_DATA
  - Fixed and variable length columns
- Row-Overflow Data Pages
  - ROW_OVERFLOW_DATA
  - varchar, nvarchar, varbinary, sql_variant – if necessary
- LOB Data Pages
  - LOB_DATA
  - text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max), CLR user-defined data types

sys.allocation_units

- Allocation Unit
  - Pages of the same kind within a single partition
- Contains 1, 2, or 3 records per partition



IAM Page

- Each allocation unit contains an Index Allocation Map (IAM) page
- One IAM page per 4 GB of data needed
- More than 4 GB
  - Several IAM pages linked through an IAM chain together (linked list)
- IAM pages stores which extents belongs to an allocation unit
- IAM pages manages the same 4 GB interval as GAM/SGAM pages

IAM Page

- Each extent is represented with 1 bit
- Bit set
  - Extent belongs to this allocation unit
- Bit not set
  - Extent doesn‘t belong to this allocation unit

Demo

Table Metadata

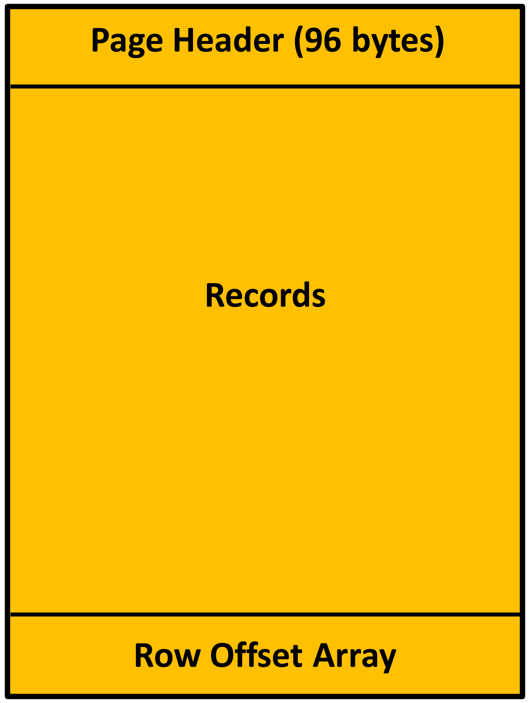
Agenda

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions

Data Page Structure

- Stores the records
- 8 KB (8.192 bytes)
- 3 parts
  - Page header (96 bytes)
  - Payload (the records)
  - Row Offset Array
- 8.096 bytes available for payload and Row Offset Array

Data Page Structure - Illustrated



Page Header Structure

- pageID
  - Page number
- nextPage
  - Page number of the next page
- prevPage
  - Page number of the previous page
- ObjectId
  - Id of the object to which this page belongs
- PartitionId
  - Id of the partition to which this page belongs

Page Header Structure

- AllocUnitId
  - Id of the allocation unit to which this page belongs
- LSN
  - Last log sequence number that changed data on this page
- slotCnt
  - Number of records stored on this page
- Level
  - Level of the page within an index

Page Header Structure

- indexId
  - Id of the index to which this page belongs – 0 with data pages
- freeData
  - Byte offset within the page where the unused space begins
- Pminlen
  - Bytes length of the fixed columns
- freeCnt
  - Number of unused bytes on this page

Page Header Structure

- reservedCnt
  - Number of bytes that are reserved by all transactions on this page
- Xactreserved
  - Number of bytes that are reserved by the last started transaction on this page
- tornBits
  - Bit-Mask used for torn page detection
- flagBits
  - 2 byte Bit-Mask with further information about the page

Payload

- Contains the records of the table
- Number of records per page depends on the chosen column definition of the table
- record_number = 8.096 / column_length
- Smaller records leads to more records on one page
  - Better performance
  - Less I/O required
  - More data stored internally in the buffer manager

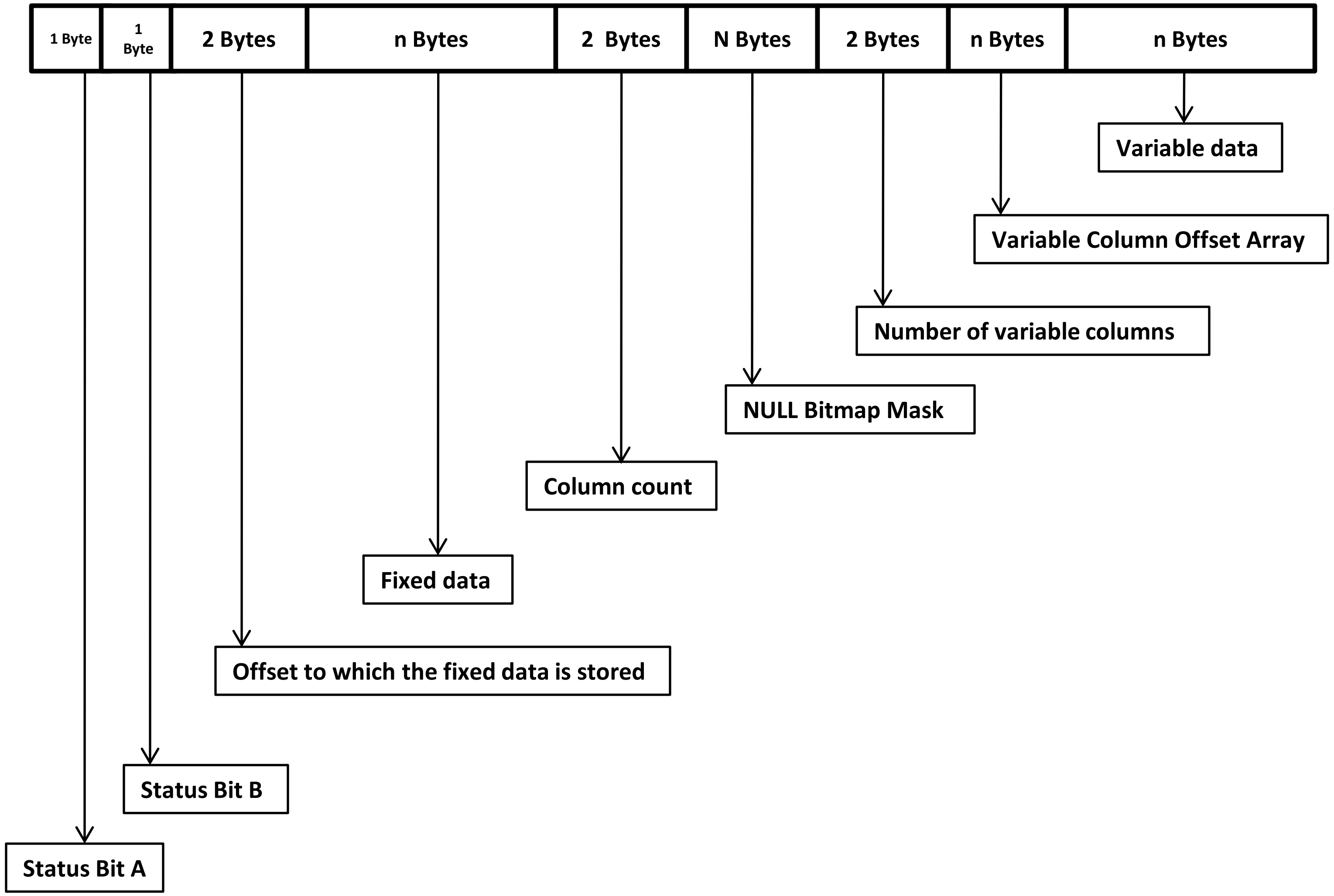
Row Offset Array

- Contains one 2 byte entry for each record on the page
- Stores the offset where the record data begins on the page
- Defines the physical order of the records on the page

Record Storage

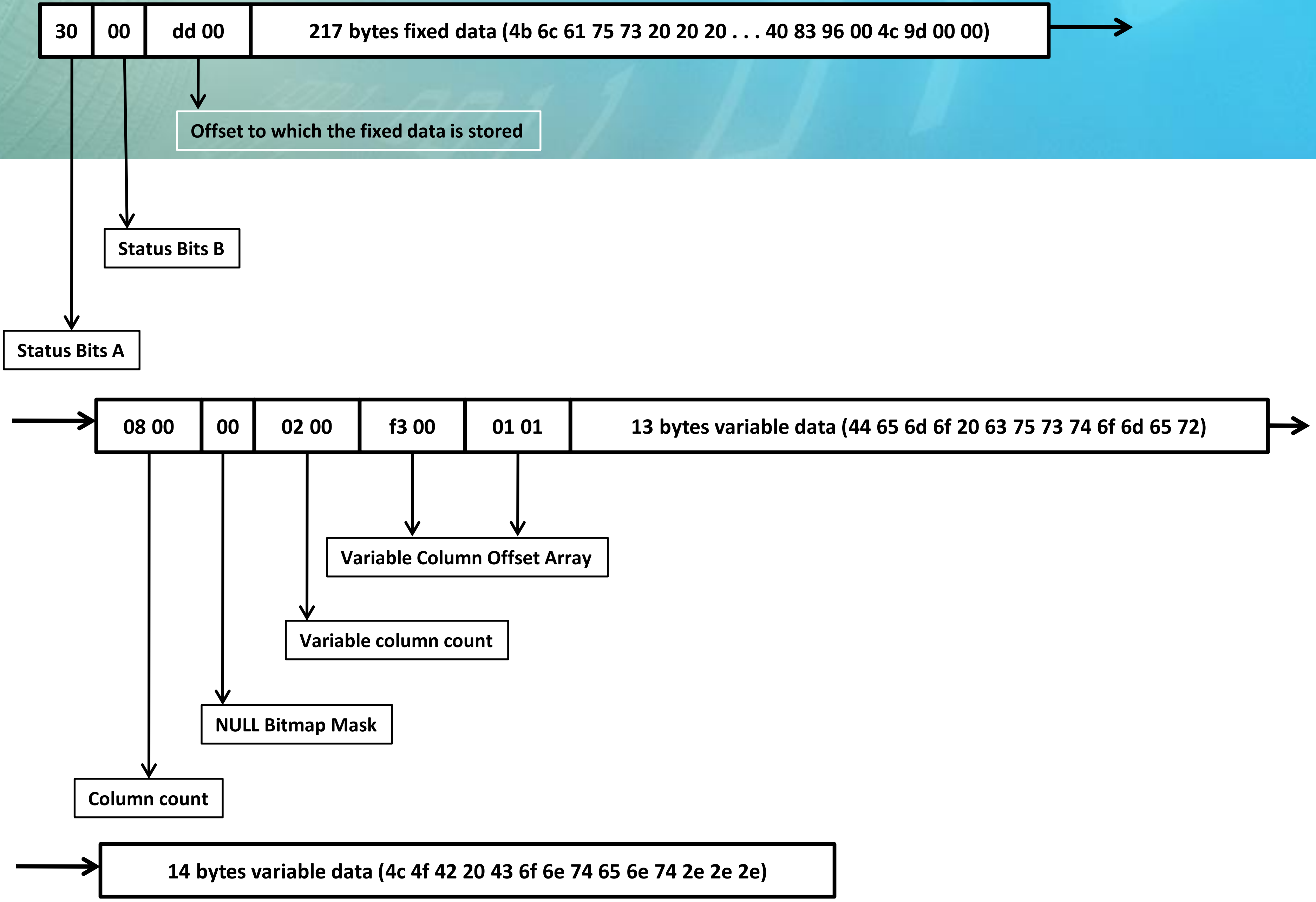
- FixedVar format
  - First fixed columns (CHAR(50))
  - Then variable columns (VARCHAR(50))
- No logical column order

Record Storage



Record analysis

- DBCC PAGE command
- Hexadecimal encoding
- Byte swapping needed
- ASCII code storage


30
00
dd 00
217 bytes fixed data (4b 6c 61 75 73 20 20 20 . . . 40 83 96 00 4c 9d 00 00)
Offset to which the fixed data is stored
Status Bits B
Status Bits A
08 00
00
02 00
f3 00
01 01
13 bytes variable data (44 65 6d 6f 20 63 75 73 74 6f 6d 65 72)
Variable Column Offset Array
Variable column count
NULL Bitmap Mask
Column count
14 bytes variable data (4c 4f 42 20 43 6f 6e 74 65 6e 74 2e 2e 2e)

DATETIME Storage

- 8 bytes storage needed
- From 01.01.1753 to 31.12.9999
- Precision: 0,00333 seconds
- 2 internal components
  - DATE part (4 bytes)
    - Days from 01.01.1900
  - TIME part (4 bytes)
    - Ticks from midnight
    - Tick: 1/300 second

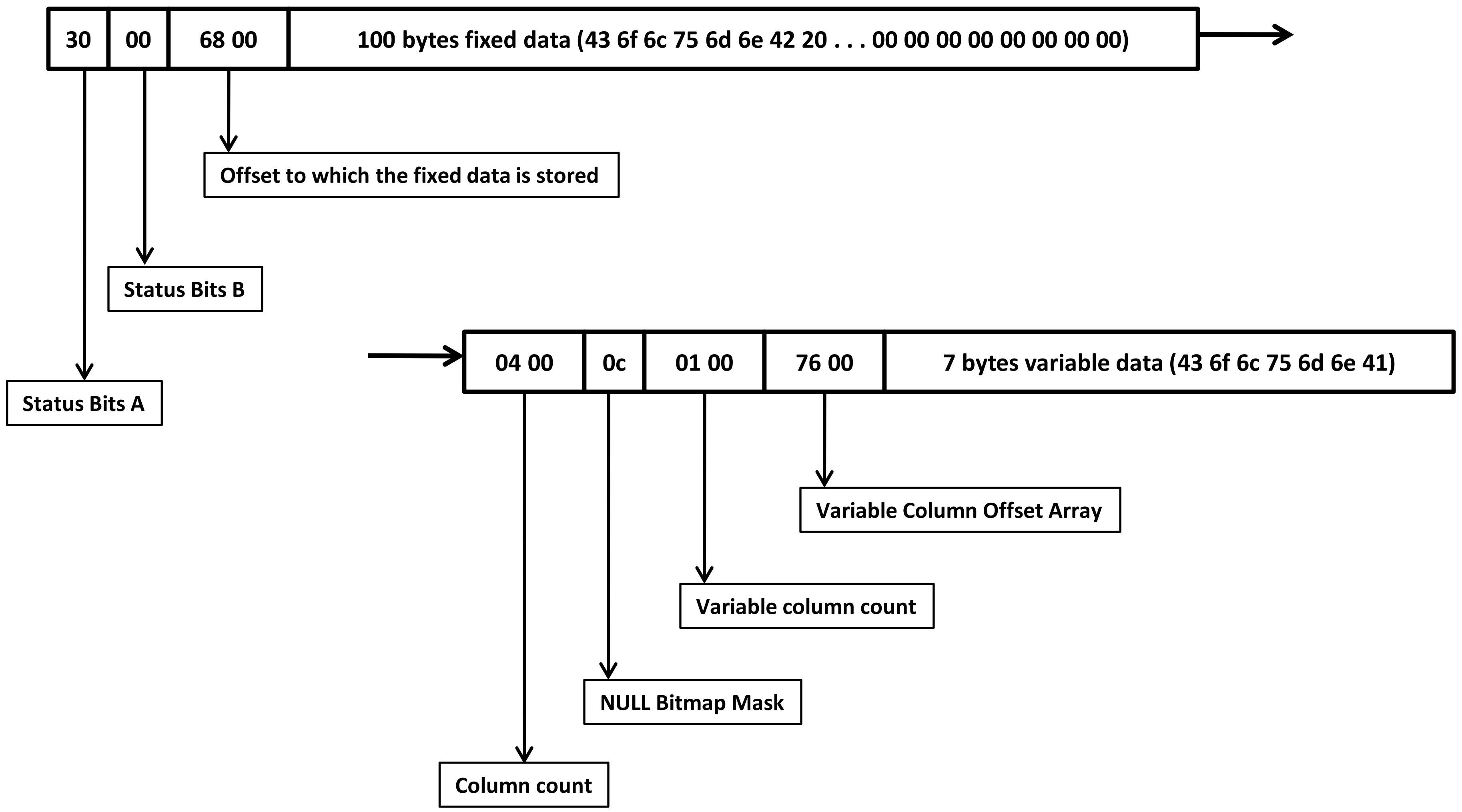
Demo

Record Analysis

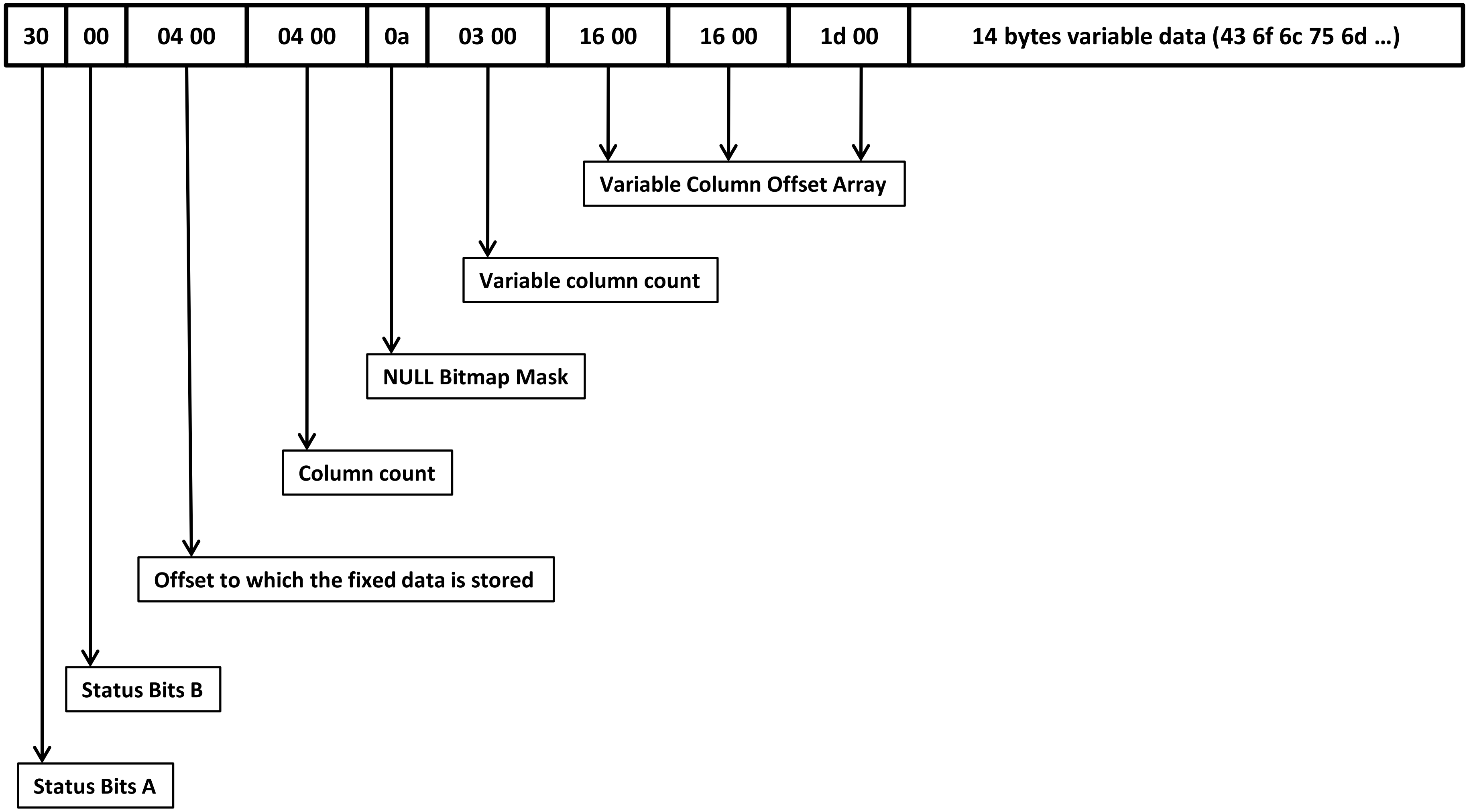
NULL Values

- Encoded in NULL bitmap mask
- NULL bitmap mask is interpreted from right to left
- Fixed column length
  - Space is also used
- Variable column length
  - Space is not used

NULL Values - Illustriated


30
00
68 00
100 bytes fixed data (43 6f 6c 75 6d 6e 42 20 . . . 00 00 00 00 00 00 00 00)
Offset to which the fixed data is stored
Status Bits B
Status Bits A
04 00
0c
01 00
76 00
7 bytes variable data (43 6f 6c 75 6d 6e 41)
Variable Column Offset Array
Variable column count
NULL Bitmap Mask
Column count

Records without fixed data

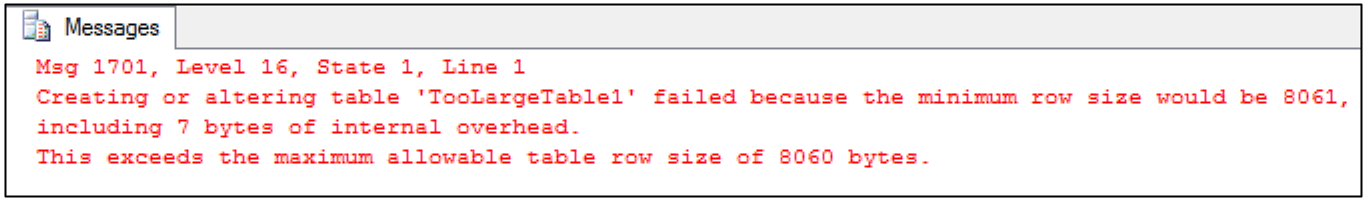


Agenda

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions

Data Page Restrictions

- Fixed column length (IN_ROW_DATA)
  - Maximum: 8.060 bytes
  - 7 bytes overhead (minimum)
    - 2 Bytes: Status Bits A & B
    - 2 Byte: Offset to which the fixed data is stored
    - 2 Bytes: Column count
    - 1 Byte (minimum): NULL bitmap mask
  - Payload: 8.053 bytes



```
Msg 1701, Level 16, State 1, Line 1
Creating or altering table 'TooLargeTable1' failed because the minimum row size would be 8061,
including 7 bytes of internal overhead.
This exceeds the maximum allowable table row size of 8060 bytes.
```

Demo

Data Page Restrictions

Summary

- Database Structure
- Table Metadata
- Anatomy of a Data Page
- Data Page Restrictions