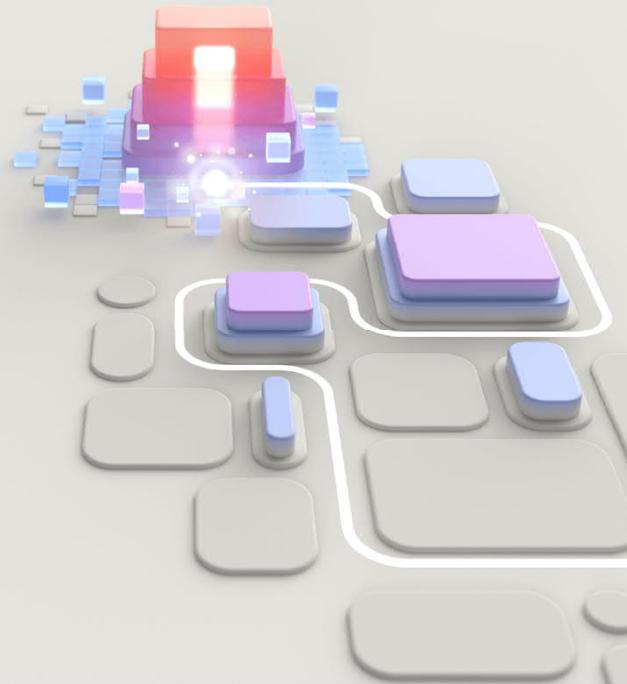




Pages & Extents SQL Index

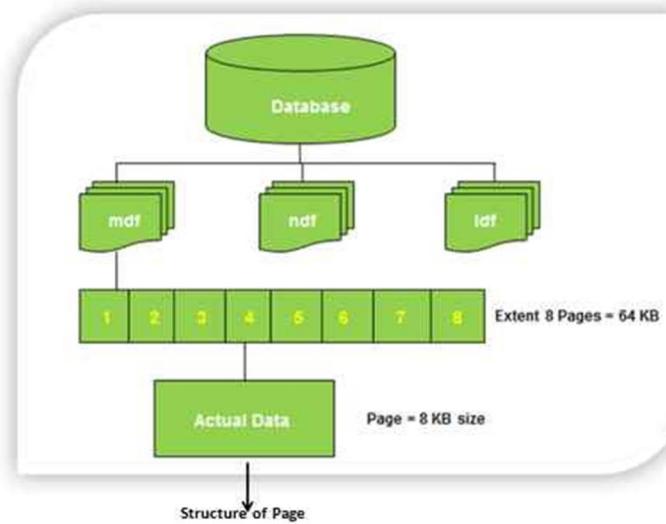


© Copyright Microsoft Corporation. All rights reserved.

Pages & Extents

The page is the fundamental unit of data storage in SQL Server.

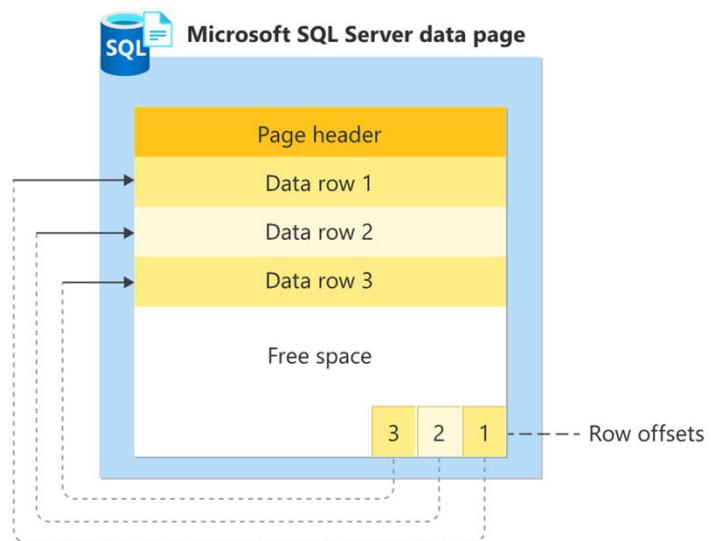
An extent is a collection of eight physically contiguous pages.



<https://learn.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide>

Page Structure

- In a regular book, all content is written on pages.
- Similarly, SQL Server writes all data rows on pages, and all data pages are the same size: 8 KB.
- In a book, most pages contain the data - the main content of the book - and some pages contain metadata about the content (for example, the table of contents and index)



<https://learn.microsoft.com/en-us/sql relational-databases/pages-and-extents-architecture-guide>

Pages (cont...)

- Most pages contain actual rows of data that were stored by users
 - Data Pages: Data rows with all data, except `text`, `ntext`, `image`, `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, and `xml` data
 - Index Pages: Contain index references about where the data is.
 - System pages: Store various metadata about the organization of the data.
- Log files don't contain pages. They contain a series of log records which don't have a fixed size.

Variable length columns when the data row exceeds 8

KB: `varchar`, `nvarchar`, `varbinary`, and `sql_variant`.

Data rows are stored on the page serially, starting immediately after the header. A row offset table starts at the end of the page, and each row offset table contains one entry for each row on the page. Each row offset entry stores how far the first byte of the row is from the start of the page. Thus, the function of the row offset table is to help SQL Server locate rows on a page quickly.

The entries in the row offset table are in reverse sequence from the sequence of the rows on the page.

Large row support

Rows can't span pages; however, portions of the row can be moved off the row's page, so the row can be very large. The maximum amount of data and overhead that is contained in a single row on a page is 8,060 bytes. This doesn't include the data stored in the `text`/`image` page type.

This restriction is relaxed for tables that contain **varchar**, **nvarchar**, **varbinary**, or **sql_variant** columns. When the total row size of all fixed and variable columns in a table exceeds the 8,060-byte limitation, SQL Server dynamically moves one or more variable length columns to pages in the **ROW_OVERFLOW_DATA** allocation unit, starting with the column with the largest width.

This is done whenever an insert or update operation increases the total size of the row beyond the 8,060-byte limit. When a column is moved to a page in the **ROW_OVERFLOW_DATA** allocation unit, a 24-byte pointer on the original page in the **IN_ROW_DATA** allocation unit is maintained. If a subsequent operation reduces the row size, SQL Server dynamically moves the columns back to the original data page.

Extents

- Extents are the basic unit in which space is managed.
 - An extent is eight physically contiguous pages, or 64 KB.
 - SQL Server databases have 16 extents per megabyte ($16 \times 64 = 1024$ bytes)
- SQL Server has two types of extents:
 - Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owning object.
 - Mixed extents are shared by up to eight objects. Each of the eight pages in the extent can be owned by a different object.

Extents in SQL Server

In SQL Server, an **extent** is a fundamental unit of storage that consists of **eight physically contiguous pages** (each page is 8 KB in size). This means an extent is **64 KB** in total. Extents are used to manage how data is stored and allocated in the database, ensuring efficient use of space.

Allocation Behavior

For SQL Server versions up to 2014, small tables or indexes initially allocate pages from mixed extents. Once they grow to eight pages, they switch to uniform extents.

Starting with SQL Server 2016, the default behavior is to allocate uniform extents for most objects, improving performance and reducing fragmentation.

Allocations for master, msdb, and model databases still retain the previous behavior.

Key Characteristics

Alignment: Extents are aligned on **64 KB boundaries** in the data file.

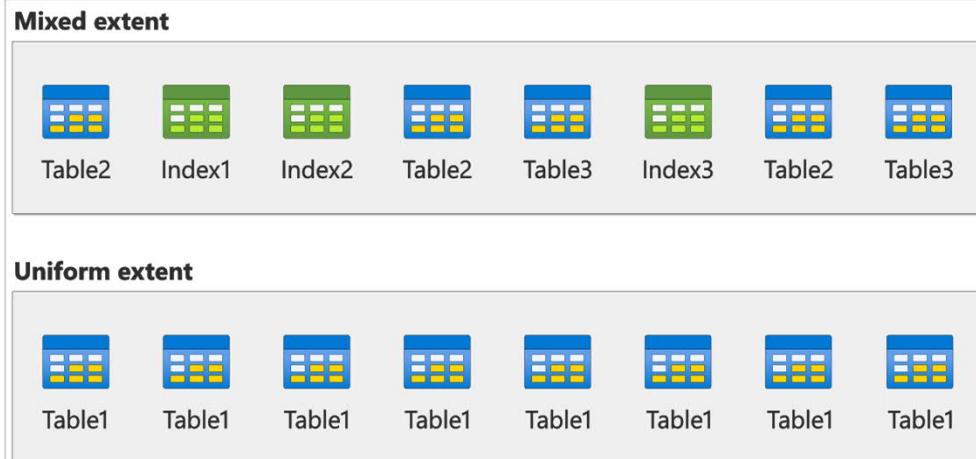
Efficient Management: By grouping pages into extents, SQL Server minimizes fragmentation and improves performance.

Extent Management

SQL Server uses **allocation maps** to track and manage extents:

- **Global Allocation Map (GAM):** Tracks whether extents are free or allocated. A bit value of 1 indicates a free extent, while 0 indicates an allocated extent.
- **Shared Global Allocation Map (SGAM):** Tracks mixed extents with at least one free page. A bit value of 1 indicates a mixed extent with free pages, while 0 indicates either a fully allocated mixed extent or a uniform extent.

Extents (cont...)



Types of Extents

Mixed Extents:

Contain pages that can belong to **different objects** (e.g., tables, indexes). Typically used for **small objects** or when an object is just starting to grow. The first eight pages of a new object are allocated from mixed extents.

Uniform Extents:

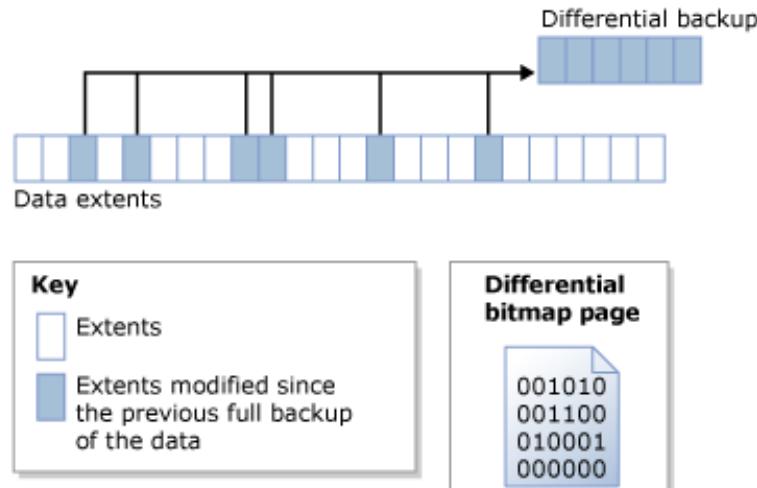
Contain pages that belong to **a single object**. Used for **larger objects** or when an object grows beyond eight pages.

Starting with SQL Server 2012 (11.x), the **sys.dm_db_database_page_allocations** system function can report page allocation information for a database, table, index, and partition.

Important

The **sys.dm_db_database_page_allocations** system function isn't documented and is subject to change. Compatibility isn't guaranteed.

Extents (Usage by Backup Service)



The illustration shows how a differential backup works.

The figure shows 24 data extents, 6 of which have changed. The differential backup (contains only the data that has been modified from the previous backup) contains only these six data extents.

The differential backup operation relies on a bitmap page that contains a bit for every extent. For each extent updated since the base, the bit is set to 1 in the bitmap.