|  |  |
| --- | --- |
| **Page Type** | **Description** |
| Data | Stores all data except those data types stored by text and image pages. |
| Index | Stores index data. |
| Text and Image | Stores TEXT, NTEXT, IMAGE, VARCHAR(MAX), VARBINARY(MAX) and XML data types. In addition these pages also store variable length columns (VARCHAR, VARBINARY, SQL\_VARIANT data types) once they exceed 8KB in size. |
| Global Allocation Map (GAM) | Records information about allocation of mixed and uniform extents. GAM pages contain bits that help SQL Server determine whether the extent is used or is available for use. When a new page needs to be allocated, SQL Server reads GAM pages to find whether any extents are available. When an extent is de-allocated, the bit for this extent is reset in GAM to make it available for future allocations. |
| Shared Global Allocation Map (SGAM) | Records information about mixed extents. SGAM pages record whether each mixed extent has any unused pages. If no mixed extent has any unused pages SQL Server attempts to find an unused extent in GAM pages. Once an extent is found it will be allocated as a mixed extent. If GAM has no free extents and SGAM shows that all extent pages have been allocated the data file is full and must be expanded before you can store additional data. |
| Page Free Space (PFS) | Keeps track of how full each data page is. In addition PFS also tracks the allocation status of each individual page. GAM bits are set so that SQL Server knows that the extent is allocated; SQL Server uses PFS pages to see which pages can be used for new allocation requests. The free space percentage is only maintained for tables without a clustered index (also called "heaps") and for TEXT / IMAGE pages. If the page has any free space it can be used for newly inserted rows. |
| Index Allocation Map (IAM) | Stores information about extents used by tables without a clustered index (also called "heaps") or indexes within a single allocation unit. IAM pages have a large bitmap, each bit representing an extent. When you execute an [INSERT statement](http://sqlserverpedia.com/w/index.php?title=INSERT_Statement_Syntax&action=edit&redlink=1), SQL Server checks the extents currently allocated to the table which you wish to populate. SQL Server will use PFS pages to see if the row can be inserted into previously allocated page. It also uses IAM to determine which extents are allocated to the given partition (allocation unit). If no pages are found with enough room to store the row, SQL Server allocates a new extent. |
| Bulk Changed Map (BCM) | Stores information about extents altered by bulk operations such as Bulk Copy Program (BCP) or BULK INSERT statement, per allocation unit. BCM only tracks the extents modified since the last BACKUP LOG statement. BCM pages are only relevant for databases that use bulk-logged recovery model because with this model BACKUP LOG statement includes the modified extents in the log backup. Doing so allows the database to be recovered including the bulk operations. If the database uses the simple recovery model, then bulk operations aren't logged. The full recovery model treats bulk operations as fully logged commands. Bulk-logged recovery model logs only the fact that a bulk operation took place, but transaction log backup includes the copy of all extents that were modified through the bulk operation. The performance of bulk operations could be considerably faster if the database uses the bulk-logged model as compared to using the full recovery model, but the transaction log backup size can grow large.  So although BCM pages exist in every database they're only used with databases using the bulk-logged recovery model. |
| Differential Changed Map (DCM) | Stores information about all extents that have been modified since the last full database backup was taken, per allocation unit. Differential backups scan the DCM pages and copy only those extents that were altered since the last full backup. This makes differential database backups considerably faster than full database backups. |