**Storage Engine Features and their Definitions**

* <https://dev.mysql.com/doc/refman/8.0/en/glossary.html>

|  |
| --- |
| **B-tree indexes**   * A tree data structure that is popular for use in database indexes. The structure is kept sorted at all times, enabling fast lookup for exact matches (equals operator) and ranges (for example, greater than, less than, and BETWEEN operators). * Because B-tree nodes can have many children, a B-tree is not the same as a binary tree, which is limited to 2 children per node. * The use of the term B-tree is intended as a reference to the general class of index design. B-tree structures used by MySQL storage engines may be regarded as variants due to sophistications not present in a classic B-tree design. |
| **Backup/point-in-time recovery**   * Back-up: The process of copying some or all table data and metadata from a MySQL instance, for safekeeping. Can also refer to the set of copied files. This is a crucial task for DBAs. The reverse of this process is the **restore** operation. * PITR: The process of restoring a **backup** to recreate the state of the database at a specific date and time. Commonly abbreviated “PITR”. Because it is unlikely that the specified time corresponds exactly to the time of a backup, this technique usually requires a combination of a **physical backup** and a **logical backup**. * PITR: For example, with the **MySQL Enterprise Backup** product, you restore the last backup that you took before the specified point in time, then replay changes from the **binary log** between the time of the backup and the PITR time. |
| **Cluster database support**   * Support to run several computers with MySQL servers and other software in a cluster (a distributed computing environment). |
| **Clustered indexes**   * The InnoDB term for a **primary key** index. * InnoDB table storage is organized based on the values of the primary key columns, to speed up queries and sorts involving the primary key columns. * For best performance, choose the primary key columns carefully based on the most performance-critical queries. * Because modifying the columns of the clustered index is an expensive operation, choose primary columns that are rarely or never updated. * In the Oracle Database product, this type of table is known as an **index-organized table**. |
| **Compressed data**   * A feature with wide-ranging benefits from using less disk space, performing less I/O, and using less memory for caching. * InnoDB supports both table-level and page-level compression. * InnoDB page compression is also referred to as **transparent page compression**. |
| **Data caches**   * The general term for any memory area that stores copies of data for frequent or high-speed retrieval. * In InnoDB, the primary kind of cache structure is the **buffer pool**. |

|  |
| --- |
| **Encrypted data** |
| **Foreign key support**   * A type of pointer relationship, between rows in separate InnoDB tables. The foreign key relationship is defined on one column in both the **parent table** and the **child table**. * In addition to enabling fast lookup of related information, foreign keys help to enforce **referential integrity**, by preventing any of these pointers from becoming invalid as data is inserted, updated, and deleted. |
| **Full-text search indexes**   * search index: In MySQL, **full-text search** queries use a special kind of index, the **FULLTEXT index**. * full-text search: The MySQL feature for finding words, phrases, Boolean combinations of words, and so on within table data, in a faster, more convenient, and more flexible way than using the SQL LIKE operator or writing your own application-level search algorithm. It uses the SQL function [MATCH()](https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html#function_match) and **FULLTEXT indexes** * FULLTEXT index: The special kind of **index** that holds the **search index** in the MySQL **full-text search** mechanism. Represents the words from values of a column, omitting any that are specified as **stopwords**. |
| **Geospatial data type support** |
| **Geospatial indexing support** |
| **Hash indexes**   * A type of **index** intended for queries that use equality operators, rather than range operators such as greater-than or BETWEEN. |
| **Index caches**   * A memory area that holds the token data for InnoDB **full-text search**. It buffers the data to minimize disk I/O when data is inserted or updated in columns that are part of a **FULLTEXT index**. * The token data is written to disk when the index cache becomes full. * Each InnoDB FULLTEXT index has its own separate index cache, whose size is controlled by the configuration option [innodb\_ft\_cache\_size](https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html#sysvar_innodb_ft_cache_size). |
| **Locking granularity** |
| **MVCC**   * Acronym for “multiversion concurrency control”. * This technique lets InnoDB **transactions** with certain **isolation levels** perform **consistent read** operations; that is, to query rows that are being updated by other transactions, and see the values from before those updates occurred. * This is a powerful technique to increase **concurrency**, by allowing queries to proceed without waiting due to **locks** held by the other transactions. |
| **Replication support**   * replica: A database **server** machine in a **replication** topology that receives changes from another server (the **source**) and applies those same changes. * The practice of sending changes from a **source**, to one or more **replicas**, so that all databases have the same data. * This technique has a wide range of uses, such as load-balancing for better scalability, disaster recovery, and testing software upgrades and configuration changes. * The changes can be sent between the databases by methods called **row-based replication** and **statement-based replication**. |
| **Storage limits**   * The maximum storage capacity. |
| **T-tree indexes** |
| **Transactions**   * An acronym standing for atomicity, consistency, isolation, and durability. * Closely tied to the notion of a **transaction**. * Transactions are **atomic** units of work that can be **committed** or **rolled back**. * When a transaction makes multiple changes to the database, either all the changes succeed when the transaction is committed, or all the changes are undone when the transaction is rolled back. * The database remains in a consistent state at all times — after each commit or rollback, and while transactions are in progress. * If related data is being updated across multiple tables, queries see either all old values or all new values, not a mix of old and new values. |
| **Update statistics for data dictionary**   * Data Dictionary: Metadata that keeps track of database objects such as **tables**, **indexes**, and table **columns**. |