**DATA** **DIRECTORY** **=** **'/external/directory';**

the server. For general information about setting system variables, see Section 5.1.9, “Using System

Variables” .

The following example demonstrates creating a table in an external directory using the DATA DIRECTORY clause. It is assumed that the innodb\_file\_per\_table variable is enabled and that the directory is known to InnoDB.

mysql> **USE** **test;**

Database changed

mysql> **CREATE** **TABLE** **t1** **(c1** **INT** **PRIMARY** **KEY)** **DATA** **DIRECTORY** **=** **'*/external/directory*';**

# MySQL creates the table's data file in a schema directory

# under the external directory

$> **cd** **/external/directory/test**

$> **ls**

t1.ibd

**Usage** **Notes:**

• MySQL initially holds the tablespace data file open, preventing you from dismounting the device, but might eventually close the file if the server is busy. Be careful not to accidentally dismount an external device while MySQL is running, or start MySQL while the device is disconnected. Attempting to access a table when the associated data file is missing causes a serious error that requires a server restart.

A server restart might fail if the data file is not found at the expected path. In this case, you can restore the tablespace data file from a backup or drop the table to remove the information about it from the data dictionary.

• Before placing a table on an NFS-mounted volume, review potential issues outlined in Using NFS with MySQL.

• If using an LVM snapshot, file copy, or other file-based mechanism to back up the table's data

file, always use the FLUSH TABLES ... FOR EXPORT statement first to ensure that all changes buffered in memory are flushed to disk before the backup occurs.

• Using the DATA DIRECTORY clause to create a table in an external directory is an alternative to using symbolic links, which InnoDB does not support.

• The DATA DIRECTORY clause is not supported in a replication environment where the source and replica reside on the same host. The DATA DIRECTORY clause requires a full directory path. Replicating the path in this case would cause the source and replica to create the table in same location.

• As of MySQL 8.0.21, tables created in file-per-table tablespaces can no longer be created in the undo tablespace directory (innodb\_undo\_directory) unless that directly is known to InnoDB. Known directories are those defined by the datadir, innodb\_data\_home\_dir, and innodb\_directories variables.

**Using** **CREATE** **TABLE** **...** **TABLESPACE** **Syntax**

CREATE TABLE ... TABLESPACE syntax can be used in combination with the DATA DIRECTORY clause to create a table in an external directory. To do so, specify innodb\_file\_per\_table as the tablespace name.

mysql> **CREATE** **TABLE** **t2** **(c1** **INT** **PRIMARY** **KEY)** **TABLESPACE** **=** **innodb\_file\_per\_table**

This method is supported only for tables created in file-per-table tablespaces, but does not require the innodb\_file\_per\_table variable to be enabled. In all other respects, this method is equivalent to the CREATE TABLE ... DATA DIRECTORY method described above. The same usage notes apply.

**Creating** **a** **Table** **in** **an** **External** **General** **Tablespace**

You can create a table in a general tablespace that resides in an external directory.

• For information about creating a general tablespace in an external directory, see Creating a General Tablespace.

• For information about creating a table in a general tablespace, see Adding Tables to a General Tablespace.

**15.6.1.3** **Importing** **InnoDB** **Tables**

This section describes how to import tables using the *Transportable* *Tablespaces* feature, which permits importing tables, partitioned tables, or individual table partitions that reside in file-per-table tablespaces. There are many reasons why you might want to import tables:

• To run reports on a non-production MySQL server instance to avoid placing extra load on a production server.

• To copy data to a new replica server.

• To restore a table from a backed-up tablespace file.

• As a faster way of moving data than importing a dump file, which requires reinserting data and rebuilding indexes.

• To move a data to a server with storage media that is better suited to your storage requirements. For example, you might move busy tables to an SSD device, or move large tables to a high-capacity

HDD device.

The *Transportable* *Tablespaces* feature is described under the following topics in this section:

• [Prerequisites](#_bookmark1)

• [Importing Tables](#_bookmark2)

• [Importing Partitioned Tables](#_bookmark3)

• [Importing Table Partitions](#_bookmark4)

• Limitations

• Usage Notes

• Internals

**Prerequisites**

• The innodb\_file\_per\_table variable must be enabled, which it is by default.

• The page size of the tablespace must match the page size of the destination MySQL server instance. InnoDB page size is defined by the innodb\_page\_size variable, which is configured when initializing a MySQL server instance.

• If the table has a foreign key relationship, foreign\_key\_checks must be disabled before

executing DISCARD TABLESPACE. Also, you should export all foreign key related tables at the same logical point in time, as ALTER TABLE ... IMPORT TABLESPACE does not enforce foreign key constraints on imported data. To do so, stop updating the related tables, commit all transactions, acquire shared locks on the tables, and perform the export operations.

• When importing a table from another MySQL server instance, both MySQL server instances must have General Availability (GA) status and must be the same version. Otherwise, the table must be created on the same MySQL server instance into which it is being imported.



• If the table was created in an external directory by specifying the DATA DIRECTORY clause in the CREATE TABLE statement, the table that you replace on the destination instance must be defined with the same DATA DIRECTORY clause. A schema mismatch error is reported if the clauses do not match. To determine if the source table was defined with a DATA DIRECTORY clause, use SHOW CREATE TABLE to view the table definition. For information about using the DATA DIRECTORY clause, see Section 15.6.1.2, “Creating Tables Externally” .

• If a ROW\_FORMAT option is not defined explicitly in the table definition or ROW\_FORMAT=DEFAULT is used, the innodb\_default\_row\_format setting must be the same on the source and destination instances. Otherwise, a schema mismatch error is reported when you attempt the import operation. Use SHOW CREATE TABLE to check the table definition. Use SHOW VARIABLES to check the innodb\_default\_row\_format setting. For related information, see Defining the Row Format of a Table.

**Importing** **Tables**

This example demonstrates how to import a regular non-partitioned table that resides in a file-per-table tablespace.

1. On the destination instance, create a table with the same definition as the table you intend to import. (You can obtain the table definition using SHOW CREATE TABLE syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> USE test;

mysql> CREATE TABLE t1 (c1 INT) ENGINE=INNODB;

2. On the destination instance, discard the tablespace of the table that you just created. (Before importing, you must discard the tablespace of the receiving table.)

mysql> ALTER TABLE t1 DISCARD TABLESPACE;

3. On the source instance, run FLUSH TABLES ... FOR EXPORT to quiesce the table you intend to import. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> USE test;

mysql> FLUSH TABLES t1 FOR EXPORT;

FLUSH TABLES ... FOR EXPORT ensures that changes to the named table are flushed to disk so that a binary table copy can be made while the server is running. When FLUSH TABLES ... FOR EXPORT is run, InnoDB generates a .cfg metadata file in the schema directory of the table. The .cfg file contains metadata that is used for schema verification during the import operation.

**Note**

The connection executing FLUSH TABLES ... FOR EXPORT must remain open while the operation is running; otherwise, the .cfg file is removed as locks are released upon connection closure.

4. Copy the .ibd file and .cfg metadata file from the source instance to the destination instance. For example:

$> scp */path/to/datadir*/test/t1.{ibd,cfg} destination-server:*/path/to/datadir*/test

The .ibd file and .cfg file must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing a table from an encrypted tablespace, InnoDB generates a .cfp file in addition to a .cfg metadata file. The .cfp file must be copied to the destination instance together with the .cfg file. The

.cfp file contains a transfer key and an encrypted tablespace key. On import, InnoDB uses the transfer key to decrypt the tablespace key. For related information, see Section 15.13, “InnoDB Data-at-Rest Encryption” .

5. On the source instance, use UNLOCK TABLES to release the locks acquired by the FLUSH TABLES ... FOR EXPORT statement:

mysql> USE test;

mysql> UNLOCK TABLES;

The UNLOCK TABLES operation also removes the .cfg file.

6. On the destination instance, import the tablespace:

mysql> USE test;

mysql> ALTER TABLE t1 IMPORT TABLESPACE;

**Importing** **Partitioned** **Tables**

This example demonstrates how to import a partitioned table, where each table partition resides in a file-per-table tablespace.

1. On the destination instance, create a partitioned table with the same definition as the partitioned table that you want to import. (You can obtain the table definition using SHOW CREATE TABLE syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> **USE** **test;**

mysql> **CREATE** **TABLE** **t1** **(i** **int)** **ENGINE** **=** **InnoDB** **PARTITION** **BY** **KEY** **(i)** **PARTITIONS** **3;**

In the /*datadir*/test directory, there is a tablespace .ibd file for each of the three partitions.

mysql> **\!** **ls** ***/path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd

2. On the destination instance, discard the tablespace for the partitioned table. (Before the import operation, you must discard the tablespace of the receiving table.)

mysql> **ALTER** **TABLE** **t1** **DISCARD** **TABLESPACE;**

The three tablespace .ibd files of the partitioned table are discarded from the /*datadir*/test directory.

3. On the source instance, run FLUSH TABLES ... FOR EXPORT to quiesce the partitioned table that you intend to import. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> **USE** **test;**

mysql> **FLUSH** **TABLES** **t1** **FOR** **EXPORT;**

FLUSH TABLES ... FOR EXPORT ensures that changes to the named table are flushed to disk so that binary table copy can be made while the server is running. When FLUSH TABLES ... FOR EXPORT is run, InnoDB generates .cfg metadata files in the schema directory of the table for each of the table's tablespace files.

mysql> **\!** **ls** ***/path/to/datadir*/test/**

t1#p#p0 .ibd t1#p#p1 .ibd t1#p#p2 .ibd

t1#p#p0.cfg t1#p#p1.cfg t1#p#p2.cfg

The .cfg files contain metadata that is used for schema verification when importing the tablespace. FLUSH TABLES ... FOR EXPORT can only be run on the table, not on individual table partitions.

4. Copy the .ibd and .cfg files from the source instance schema directory to the destination instance schema directory. For example:



$>**scp** ***/path/to/datadir*/test/t1\*.{ibd,cfg}** **destination-server:*/path/to/datadir*/test**

The .ibd and .cfg files must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing a table from an encrypted tablespace, InnoDB generates a .cfp files in addition to a .cfg metadata files. The .cfp files must be copied to the destination instance together with the .cfg files. The .cfp files contain a transfer key and an encrypted tablespace key. On import, InnoDB uses the transfer key to decrypt the tablespace key. For related information, see Section 15.13, “InnoDB Data-at-Rest Encryption” .

5. On the source instance, use UNLOCK TABLES to release the locks acquired by FLUSH TABLES ... FOR EXPORT:

mysql> **USE** **test;**

mysql> **UNLOCK** **TABLES;**

6. On the destination instance, import the tablespace of the partitioned table:

mysql> **USE** **test;**

mysql> **ALTER** **TABLE** **t1** **IMPORT** **TABLESPACE;**

**Importing** **Table** **Partitions**

This example demonstrates how to import individual table partitions, where each partition resides in a file-per-table tablespace file.

In the following example, two partitions (p2 and p3) of a four-partition table are imported.

1. On the destination instance, create a partitioned table with the same definition as the partitioned table that you want to import partitions from. (You can obtain the table definition using SHOW CREATE TABLE syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> **USE** **test;**

mysql> **CREATE** **TABLE** **t1** **(i** **int)** **ENGINE** **=** **InnoDB** **PARTITION** **BY** **KEY** **(i)** **PARTITIONS** **4;**

In the /*datadir*/test directory, there is a tablespace .ibd file for each of the four partitions.

mysql> **\!** **ls** ***/path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd t1#p#p3.ibd

2. On the destination instance, discard the partitions that you intend to import from the source instance. (Before importing partitions, you must discard the corresponding partitions from the receiving partitioned table.)

mysql> **ALTER** **TABLE** **t1** **DISCARD** **PARTITION** **p2,** **p3** **TABLESPACE;**

The tablespace .ibd files for the two discarded partitions are removed from the /*datadir*/test directory on the destination instance, leaving the following files:

mysql> **\!** **ls** ***/path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd

**Note**

When ALTER TABLE ... DISCARD PARTITION ... TABLESPACE is run on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.