

MySQL InnoDB Data Recovery

The Last Resort

Frédéric Descamps

Community Manager

Oracle MySQL

FOSDEM MySQL Devroom - February 2025





Who am I ?
about.me/lefred



Frédéric Descamps

-  @lefred
-  @lefredbe.bsky.social
-  @lefred@fosstodon.org
- **MySQL** Evangelist
- using **MySQL** since version 3.20
- devops believer
- living in 
- <https://lefred.be>





MySQL InnoDB Data Recovery

Last Resort - Why?



Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*

Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*
- *No binlogs (can't perform point-in-time recovery)*

Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*
- *No binlogs (can't perform point-in-time recovery)*
- *No replica*

Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*
- *No binlogs (can't perform point-in-time recovery)*
- *No replica*
- *Disaster Recovery Plan? HAHAHA!!*

Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*
- *No binlogs (can't perform point-in-time recovery)*
- *No replica*
- *Disaster Recovery Plan? HAHAHA!!*
- *Because we can!*

Reasons to perform Data Recovery

- *No backup (invalid, broken, untested)*
- *No binlogs (can't perform point-in-time recovery)*
- *No replica*
- *Disaster Recovery Plan? HAHAHA!!*
- *Because we can!*
 - *You have to live your life dangerously, don't you?*



MySQL InnoDB Data Recovery

Tools



MySQL DBA Toolset for InnoDB Data Recovery

There were some tools in C++ that required a lot of re-compilation:

- *Percona Data Recovery Tool for InnoDB (not updated since 2011)*
- *Undrop for InnoDB (TwinDB) (last commit in 2018)*
- *Undrop-InnoDB (Marco Tusa) (last commit in 2018)*

MySQL DBA Toolset for InnoDB Data Recovery

There were some tools in C++ that required a lot of re-compilation:

- *Percona Data Recovery Tool for InnoDB (not updated since 2011)*
- *Undrop for InnoDB (TwinDB) (last commit in 2018)*
- *Undrop-InnoDB (Marco Tusa) (last commit in 2018)*

But now there is a new tool made by NVIDIA engineers:

- **InnoDB_rs** written in Rust
 - *first commit July 31st, 2024*
 - *last commit August 10th, 2024*

MySQL DBA Toolset for InnoDB Data Recovery

- *ibd2sdi*
- *sdi2dll* (from Marcelo Atlmann)
- *innodb_sort*
- *MySQL Shell*
- *ibdNinja* (optional)
- *and of course a Linux box* 😂



MySQL InnoDB Data Recovery

Prerequisites



The preliminary tasks of a good DBA

- *Have a recent physical backup*

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗
- *Have a recent logical backup*

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗
- *Have a recent logical backup* ✗

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗
- *Have a recent logical backup* ✗
- *Have a ddl dump*

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗
- *Have a recent logical backup* ✗
- *Have a ddl dump* ✗

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✖
- *Have a recent logical backup* ✖
- *Have a ddl dump* ✖
- *Have the InnoDB sdi file for the tables*

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✖
- *Have a recent logical backup* ✖
- *Have a ddl dump* ✖
- *Have the InnoDB sdi file for the tables* ✖

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✖
- *Have a recent logical backup* ✖
- *Have a ddl dump* ✖
- *Have the InnoDB sdi file for the tables* ✖
- *Know your space IDs*

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✖
- *Have a recent logical backup* ✖
- *Have a ddl dump* ✖
- *Have the InnoDB sdi file for the tables* ✖
- *Know your space IDs* ✖

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✖
- *Have a recent logical backup* ✖
- *Have a ddl dump* ✖
- *Have the InnoDB sdi file for the tables* ✖
- *Know your space IDs* ✖ (*nobody does!*)

The preliminary tasks of a good DBA

- *Have a recent physical backup* ✗
- *Have a recent logical backup* ✗
- *Have a ddl dump* ✗
- *Have the InnoDB sdi file for the tables* ✗
- *Know your space IDs* ✗ (*nobody does!*)

All these elements facilitate recovery!

DDL dump

To be able to perform a successful recovery, we need to match the data with the table's definition. So it's important to have the definition of the table we want to recover.

One of the best way to save the definition of the tables (other than the backups) is to dump the DDL of the full instance.

*We use **MySQL Shell**'s `dumpInstance` utility with `{ddlOnly: 'True'}`*

DDL dump (2)

```
JS > util.dumpInstance("/home/fred/dump",{ddlOnly: 'True'})
Acquiring global read lock
Global read lock acquired
Initializing - done
1 out of 5 schemas will be dumped and within them 6 tables, 2 views.
2 out of 5 users will be dumped.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Writing users DDL
Running data dump using 4 threads.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Total duration: 00:00:00s
Schemas dumped: 1
Tables dumped: 6
```

DDL dump - output

```
$ cat employees@employees.sql
-- MySQLShell dump 2.0.1  Distrib Ver 9.0.1 for Linux on x86_64 - for MySQL 9.0.1 (MySQL Community Server (GPL)), for Linux (x86_64)
--
-- Host: 127.0.0.1    Database: employees    Table: employees
-- -----
-- Server version    9.0.1
--
-- Table structure for table `employees`
--
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE IF NOT EXISTS `employees` (
  `emp_no` int NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

InnoDB SDI

SDI acronym stands for *Serialized Dictionary Information*.

MySQL 8.0 replaced the old way to store the metadata of tables (frm), their structure, into the new transactional Data Dictionary (in InnoDB).

That information is also part any InnoDB tablespaces, so the meta data and data are bundled together.

InnoDB SDI

SDI acronym stands for *Serialized Dictionary Information*.

MySQL 8.0 replaced the old way to store the metadata of tables (frm), their structure, into the new transactional Data Dictionary (in InnoDB).

That information is also part any InnoDB tablespaces, so the meta data and data are bundled together.

To extract the SDI from an InnoDB Tablespace, use `ibd2sdi`:

```
$ ibd2sdi employees.ibd > employees.sdi
```

Know your space IDs

This is something only few DBAs do, but I would recommend to save such information regularly as part of backup:

```
SQL > use INFORMATION_SCHEMA
```

```
SQL > select tblsp.name, space, index_id, idx.name, page_no  
       from INNODB_TABLESPACES tblsp JOIN INNODB_INDEXES idx using(space)  
       where tblsp.name like 'employees/%' and idx.name = 'PRIMARY';
```

name	space	index_id	name	page_no
employees/employees	2	154	PRIMARY	4
employees/departments	3	155	PRIMARY	4
employees/dept_manager	4	157	PRIMARY	4
employees/dept_emp	5	159	PRIMARY	4
employees/titles	6	161	PRIMARY	4
employees/salaries	7	162	PRIMARY	4

```
6 rows in set (0.0118 sec)
```




MySQL InnoDB Data Recovery

Let's Go !



Environment

This is the system:

- *MySQL Community Server - 9.1.0*
- *the MySQL datadir is a mount of a dedicated disk*
- *we use the **employees** test database*

```
$ ls -lh
total 179M
-rw-r----- 1 fred fred 128K Jan  8 19:15 departments.ibd
-rw-r----- 1 fred fred  25M Jan  8 19:16 dept_emp.ibd
-rw-r----- 1 fred fred 128K Jan  8 19:16 dept_manager.ibd
-rw-r----- 1 fred fred  22M Jan  8 19:15 employees.ibd
-rw-r----- 1 fred fred 104M Jan  8 19:16 salaries.ibd
-rw-r----- 1 fred fred  27M Jan  8 19:16 titles.ibd
```


Scenario

An innocent DBA will just remove the file `employees.ibd` directly from the filesystem:

```
$ sudo rm employees/employees.ibd
```



Control Measures

Before letting the Innocent DBA delete the file, let's take some information related to the employees table:

```
SQL > select count(*) from employees;
```

```
+-----+
```

```
| count(*) |
```

```
+-----+
```

```
| 300024 |
```

```
+-----+
```

```
1 row in set (0.0746 sec)
```

Control Measures (2)

First 10 records:

```
SQL > select * from employees limit 10;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24

10 rows in set (0.0009 sec)

Control Measures (2)

Last 10 records:

```
SQL > select * from (select * from employees order by emp_no desc limit 10) a order by emp_no;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
499990	1963-11-03	Khaled	Kohling	M	1985-10-10
499991	1962-02-26	Pohua	Sichman	F	1989-01-12
499992	1960-10-12	Siamak	Salverda	F	1987-05-10
499993	1963-06-04	DeForest	Mullainathan	M	1997-04-07
499994	1952-02-26	Navin	Argence	F	1990-04-24
499995	1958-09-24	Dekang	Lichtner	F	1993-01-12
499996	1953-03-07	Zito	Baaz	M	1990-09-27
499997	1961-08-03	Berhard	Lenart	M	1986-04-21
499998	1956-09-05	Patricia	Breugel	M	1993-10-13
499999	1958-05-01	Sachin	Tsukuda	M	1997-11-30

10 rows in set (0.0005 sec)

OUCH !!

After the delete of the file, MySQL could still see the data if the pages are still in the buffer pool, but after a while or after a restart...

```
SQL > select * from employees.employees;  
ERROR: 1812 (HY000): Tablespace is missing for table `employees`.`employees`.
```





Step 1: stop MySQL

If we want to recover the data, we need to stop MySQL as soon as we realize that something weird happened.

```
SQL > shutdown;
```

And umount the disk:

```
$ sudo umount /var/lib/mysql
```

Step 2: create an image of the disk

To avoid increasing the amount of problems (in case of a disk failure for example), it's recommended to perform a diskimage of the disk.

We can use **dd** for this:

```
$ sudo dd if=/dev/sdb of=datadisk.loop bs=1M status=progress
```


Step 3: extracting the pages

We need to extract all *InnoDB* pages we can find from the diskimage.

We use *InnoDB_rs* for that:

```
$ InnoDB_rs/target/debug/page_extractor --by-tablespace -o recovery datadisk.loop  
[0s] [=====>] (145.96 MiB/s) 4.99 GiB/5.00 GiB  
2025-01-08T21:19:02.440378Z INFO page_extractor: found 11431 pages that have valid  
checksum (0 index pages), 147 pages only failed checksum
```

Step 3: extracting the pages - result

```
$ ls -lh recovery/BY_TABLESPACE/
total 179M
-rw-r--r-- 1 fred fred 208K Jan  8 22:19 00000000.pages
-rw-r--r-- 1 fred fred  80K Jan  8 22:18 00000001.pages
-rw-r--r-- 1 fred fred 14M Jan  8 22:18 00000002.pages
-rw-r--r-- 1 fred fred  96K Jan  8 22:18 00000003.pages
-rw-r--r-- 1 fred fred  96K Jan  8 22:18 00000004.pages
-rw-r--r-- 1 fred fred 17M Jan  8 22:18 00000005.pages
-rw-r--r-- 1 fred fred 19M Jan  8 22:19 00000006.pages
-rw-r--r-- 1 fred fred 100M Jan  8 22:19 00000007.pages
-rw-r--r-- 1 fred fred 6.5M Jan  8 22:19 4294967278.pages
-rw-r--r-- 1 fred fred 6.0M Jan  8 22:19 4294967279.pages
-rw-r--r-- 1 fred fred 18M Jan  8 22:19 4294967294.pages
```

Step 4: table definition

*We need to provide the **SQL** CREATE statement of the table in a `.sql` file to explore correctly the page and extract the data.*

Step 4: table definition

*We need to provide the **SQL** CREATE statement of the table in a `.sql` file to explore correctly the page and extract the data.*

If we have the create statement, we just copy it into a dedicate file.

Step 4: table definition

*We need to provide the **SQL** CREATE statement of the table in a **.sql** file to explore correctly the page and extract the data.*

If we have the create statement, we just copy it into a dedicate file.

*If we have the SDI, we use **sdi2ddl** to generate the **SQL** CREATE statement.*

Step 4: table definition

*We need to provide the **SQL** CREATE statement of the table in a **.sql** file to explore correctly the page and extract the data.*

If we have the create statement, we just copy it into a dedicate file.

*If we have the SDI, we use **sd2ddl** to generate the **SQL** CREATE statement.*

If we don't have that information, we need to generate it!

Step 4: generating the table definition

We need to find which extracted pages file is the one we want to recover.

Step 4: generating the table definition

We need to find which extracted pages file is the one we want to recover.

We use `ibd2sdi` on all files to find the needed one:

```
$ for i in `ls 00000*.pages`  
do  
    echo $i  
    echo "=====  
    ibd2sdi $i | grep filename  
    echo  
done
```


00000000.pages

\=====

```
[ERROR] ibd2sdi: Page 0 corruption detected. Page size is either zero or out of bound.
[ERROR] ibd2sdi: Minimum valid page size is [page size: physical=1024, logical=4096, compressed=1].
[ERROR] ibd2sdi: Maximum valid page size is [page size: physical=65536, logical=65536, compressed=0].
[ERROR] ibd2sdi: Reading multiple pages to determine the page_size.
[INFO] ibd2sdi: Page size determined is : [page size: physical=16384, logical=16384, compressed=0].
[WARNING] ibd2sdi: Unexpected SDI version. Expected: 1 Got: 0.
[ERROR] ibd2sdi: Couldn't find valid root page number.
[ERROR] ibd2sdi: SDI doesn't exist for this tablespace or the SDI root page numbers couldn't be determined.
```

00000001.pages

\=====

```
    "filename": "../sys/sys_config.ibd",
```

00000002.pages

\=====

```
[ERROR] ibd2sdi: Page 0 corruption detected. Page size is either zero or out of bound.
[ERROR] ibd2sdi: Minimum valid page size is [page size: physical=1024, logical=4096, compressed=1].
[ERROR] ibd2sdi: Maximum valid page size is [page size: physical=65536, logical=65536, compressed=0].
[ERROR] ibd2sdi: Reading multiple pages to determine the page_size.
[INFO] ibd2sdi: Page size determined is : [page size: physical=16384, logical=16384, compressed=0].
[WARNING] ibd2sdi: Unexpected SDI version. Expected: 1 Got: 0.
[ERROR] ibd2sdi: Couldn't find valid root page number.
[ERROR] ibd2sdi: SDI doesn't exist for this tablespace or the SDI root page numbers couldn't be determined.
```

00000003.pages

\=====

"filename": "../employees/departments.ibd",

00000004.pages

\=====

"filename": "../employees/dept_manager.ibd",

00000005.pages

\=====

[ERROR] ibd2sdi: Page 0 corruption detected. Page size is either zero or out of bound.

[ERROR] ibd2sdi: Minimum valid page size is [page size: physical=1024, logical=4096, compressed=1].

[ERROR] ibd2sdi: Maximum valid page size is [page size: physical=65536, logical=65536, compressed=0].

[ERROR] ibd2sdi: Reading multiple pages to determine the page_size.

[INFO] ibd2sdi: Page size determined is : [page size: physical=16384, logical=16384, compressed=0].

[WARNING] ibd2sdi: Unexpected SDI version. Expected: 1 Got: 0.

[ERROR] ibd2sdi: Couldn't find valid root page number.

[ERROR] ibd2sdi: SDI doesn't exist for this tablespace or the SDI root page numbers couldn't be determined.

00000006.pages

\=====

```
[ERROR] ibd2sdi: Page 0 corruption detected. Page size is either zero or out of bound.
[ERROR] ibd2sdi: Minimum valid page size is [page size: physical=1024, logical=4096, compressed=1].
[ERROR] ibd2sdi: Maximum valid page size is [page size: physical=65536, logical=65536, compressed=0].
[ERROR] ibd2sdi: Reading multiple pages to determine the page_size.
[INFO] ibd2sdi: Page size determined is : [page size: physical=16384, logical=16384, compressed=0].
[WARNING] ibd2sdi: Unexpected SDI version. Expected: 1 Got: 0.
[ERROR] ibd2sdi: Couldn't find valid root page number.
[ERROR] ibd2sdi: SDI doesn't exist for this tablespace or the SDI root page numbers couldn't be determined.
```

00000007.pages

\=====

```
[ERROR] ibd2sdi: Page 0 corruption detected. Page size is either zero or out of bound.
[ERROR] ibd2sdi: Minimum valid page size is [page size: physical=1024, logical=4096, compressed=1].
[ERROR] ibd2sdi: Maximum valid page size is [page size: physical=65536, logical=65536, compressed=0].
[ERROR] ibd2sdi: Reading multiple pages to determine the page_size.
[INFO] ibd2sdi: Page size determined is : [page size: physical=16384, logical=16384, compressed=0].
[WARNING] ibd2sdi: Unexpected SDI version. Expected: 1 Got: 0.
[ERROR] ibd2sdi: Couldn't find valid root page number.
[ERROR] ibd2sdi: SDI doesn't exist for this tablespace or the SDI root page numbers couldn't be determined.
```

Step 4: generating the table definition (2)

So far, we only know that:

```
00000000.pages ==> ./sys/sys_config.ibd
```

```
00000003.pages ==> ./employees/departments.ibd
```

```
00000004.pages ==> ./employees/dept_manager.ibd
```


Step 4: generating the table definition (2)

So far, we only know that:

00000000.pages ==> ./sys/sys_config.ibd

00000003.pages ==> ./employees/departments.ibd

00000004.pages ==> ./employees/dept_manager.ibd

Do you remember this?

name	space	index_id	name	page_no
employees/employees	2	154	PRIMARY	4
employees/departments	3	155	PRIMARY	4
employees/dept_manager	4	157	PRIMARY	4
employees/dept_emp	5	159	PRIMARY	4
employees/titles	6	161	PRIMARY	4
employees/salaries	7	162	PRIMARY	4

Step 4: generating the table definition (3)

*We need to fix the page files we couldn't get the SDI. We will start with the next one, **00000002.pages***

(which matches the id of the tablespace you want to retrieve but we don't know)

*As the SDI cannot be extracted, we need to sort the pages to let **idb2sdi** find the metadata where it is supposed to be. We use **innodb_sort**:*

```
$ innodb_sort 00000002.pages
```

```
$ ls -lh 00000002.pages*
```

```
-rw-r--r-- 1 fred fred 14M Aug 22 08:42 00000002.pages
```

```
-rw-r--r-- 1 fred fred 15M Aug 23 10:07 00000002.pages.sorted
```

Step 4: generating the table definition (4)

We can now verify if this is the information we are looking for:

```
$ ibd2sdi 00000002.pages.sorted | grep filename  
      "filename": "./employees/employees.ibd",
```

Step 4: generating the table definition (4)

We can now verify if this is the information we are looking for:

```
$ ibd2sdi 00000002.pages.sorted | grep filename  
    "filename": "./employees/employees.ibd",
```

Wooohooo \o/ Half a victory!

Step 4: generating the table definition (5)

If we want, we can also verify using a **NEW** tool: `ibdNinja`, but it works only on the sorted pages file too.

Note: I've modified `ibdNinja.cc` to support 9.1.0 format:

```
@@ -1680,7 +1680,7 @@ bool Table::ContainFulltext() {
#define UNSUPP_TABLE_MASK_VERSION 0x10
void Table::PreCheck() {
    // TODO(Zhao): Support more MySQL versions
-   if (dd_mysql_version_id_ < 80016 || dd_mysql_version_id_ > 80040) {
+   if (dd_mysql_version_id_ < 80016 || dd_mysql_version_id_ > 90100) {
        unsupported_reason_ |= UNSUPP_TABLE_MASK_VERSION;
    }

@@ -1732,7 +1732,7 @@ std::string Table::UnsupportedReason() {
if (unsupported_reason_ & UNSUPP_TABLE_MASK_VERSION) {
    reason += ("[Table was created in unsupported version " +
              std::to_string(dd_mysql_version_id_) +
-              ", expected in [80016, 80040] ]");
+              ", expected in [80016, 80040, 90100] ]");
    }
return reason;
}
```

```
$ ibdNinja -f 00000002.pages.sorted
```

```
\=====
|  FILE INFORMATION                                     |
|-----|
File name:          00000002.pages.sorted
File size:          15040512 B
Space id:           2
Page logical size:  16384 B
Page physical size: 16384 B
Total number of pages: 918
Is compressed page? 0
First page number:  0
SDI root page number: 3
Post antelop:       1
Atomic blobs:       1
Has data dir:       0
Shared:             0
Temporary:          0
Encryption:         0
```

```
-----
[ibdNinja]: Successfully loaded      1 tables with      1 indexes.
```

```
=====
Listing all *supported* tables and indexes in the specified ibd file:
```

```
-----
[Table] id: 1064   name: employees.employees
[Index] id: 154    , root page no: 4      , name: PRIMARY
```

Step 4: generating the table definition (6)

We can now generate the **SQL** CREATE statement we need to explore the data and perform the data recovery:

```
$ ibd2sdi 00000002.pages.sorted | sdi2ddl | tee employees.sql
CREATE TABLE `employees` (
  `emp_no` int NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Step 5: test exploring the pages

Now that we have the table's definition, we can check if the data matches the expected layout:

```
$ page_explorer -t employees.sql --limit 1 00000002.pages
INFO page_explorer: Loaded Table:
TableDefinition {
  name: "employees",
  cluster_columns: [
    Field {
      name: "emp_no",
      field_type: Int(
        true,
      ),
      nullable: false,
    },
  ],
  data_columns: [
    Field {
      name: "birth_date",
      field_type: Date,
    },
  ],
}
INFO page_explorer: Found (0 data + 886 node pointer)/886 records on index page 4
INFO page_explorer: Exiting early due to --limit argument
INFO page_explorer: Processed 5 pages, total records: 0, potentially missing: 0, Incomplete: 0
```


Step 5: test exploring the pages (2)

We can also run it on the sorted pages file, doesn't matter. If the table definition is not correct, we would end up with a similar message:

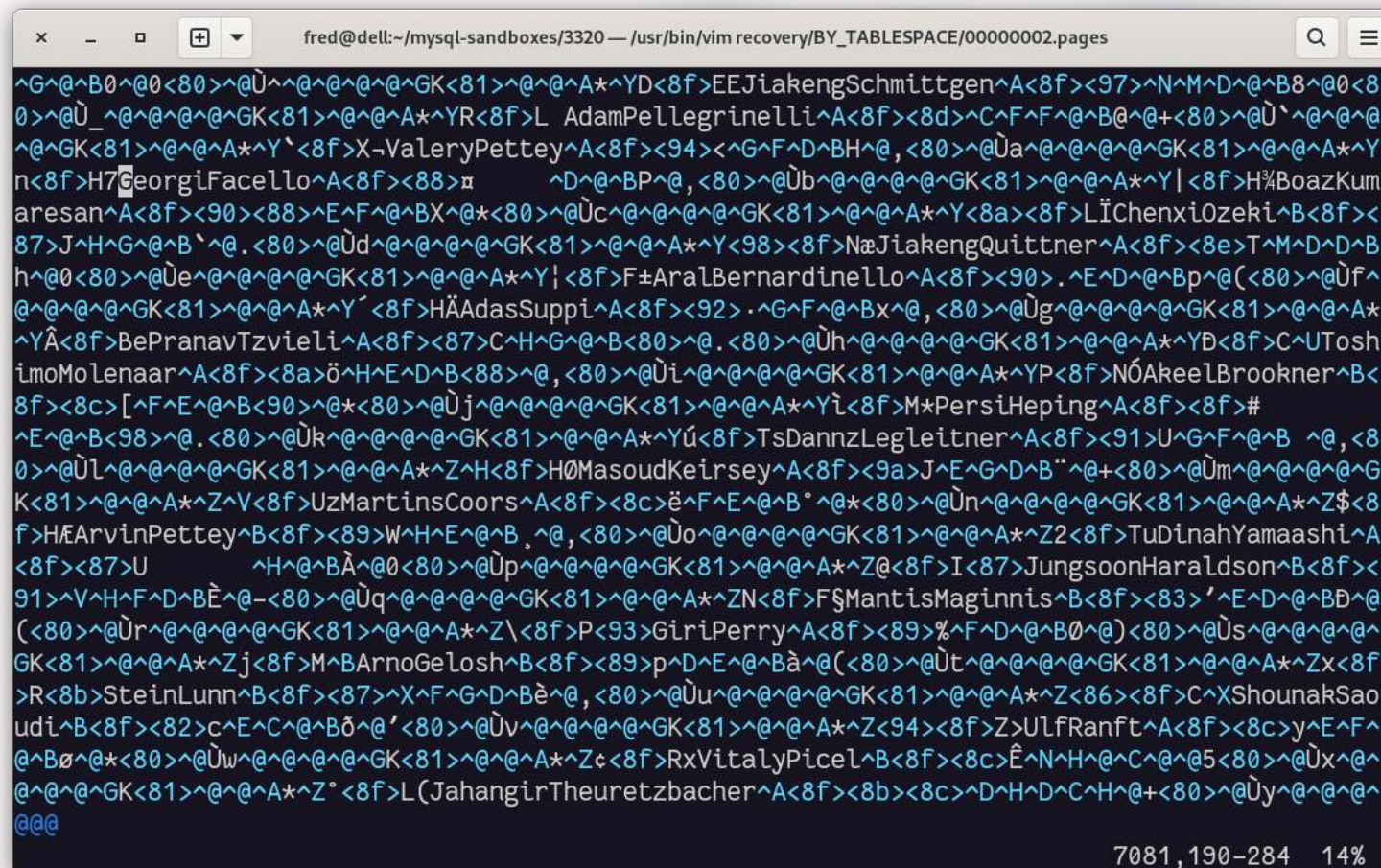
```
thread 'main' panicked at src/innodb/table/field.rs:206:21:  
Enum Value is larger than expected? 99 vs 2
```

or

```
thread 'main' panicked at src/innodb/table/field.rs:148:26:  
Failed parsing UTF-8: FromUtf8Error { bytes: [16, 143, 130, 33], error:  
    Utf8Error { valid_up_to: 1, error_len: Some(1) } }  
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

Manually check the content

We can also manually open the file and verify what kind of data is stored:



```
fred@dell:~/mysql-sandboxes/3320 — /usr/bin/vim recovery/BY_TABLESPACE/00000002.pages
^G^@^B0^@0<80>^@Ù^@^@^@^@GK<81>^@^@^A^YD<8f>EEJiakengSchmittgen^A<8f><97>^N^M^D^@^B8^@0<8
0>^@Ù_ ^@^@^@^@GK<81>^@^@^A^YR<8f>L AdamPellegrinelli^A<8f><8d>^C^F^F^@^B^@+<80>^@Ù`^@^@^@
^@GK<81>^@^@^A^Y`<8f>X-ValeryPetтей^A<8f><94><^G^F^D^B^H^@,<80>^@Ùa^@^@^@^@GK<81>^@^@^A^Y
n<8f>H7GeorgiFacello^A<8f><88>^D^@^BP^@,<80>^@Ùb^@^@^@^@GK<81>^@^@^A^Y|<8f>H%BoazKum
aresan^A<8f><90><88>^E^F^@^BX^@*^<80>^@Ùc^@^@^@^@GK<81>^@^@^A^Y<8a><8f>LİChenxiOzeki^B<8f><
87>J^H^G^@^B`^@.<80>^@Ùd^@^@^@^@GK<81>^@^@^A^Y<98><8f>NæJiakengQuittner^A<8f><8e>T^M^D^D^B
h^@0<80>^@Ùe^@^@^@^@GK<81>^@^@^A^Y|<8f>F±AralBernardinello^A<8f><90>.^E^D^@^Bp^@(<80>^@Ùf^
^@^@^@^@GK<81>^@^@^A^Y`<8f>HÅAdasSuppi^A<8f><92>.^G^F^@^Bx^@,<80>^@Ùg^@^@^@^@GK<81>^@^@^A^
YÅ<8f>BePranavTzvieli^A<8f><87>C^H^G^@^B<80>^@.<80>^@Ùh^@^@^@^@GK<81>^@^@^A^YD<8f>C^U^Tosh
imoMolenaar^A<8f><8a>ö^H^E^D^B<88>^@,<80>^@Ùi^@^@^@^@GK<81>^@^@^A^YP<8f>NÓAkeelBrookner^B<
8f><8c>[^F^E^@^B<90>^@*^<80>^@Ùj^@^@^@^@GK<81>^@^@^A^Yİ<8f>M*PersiHeping^A<8f><8f>#
^E^@^B<98>^@.<80>^@Ùk^@^@^@^@GK<81>^@^@^A^YÚ<8f>TsDannzLegleitner^A<8f><91>U^G^F^@^B ^@,<8
0>^@Ùl^@^@^@^@GK<81>^@^@^A^YZ^H<8f>H0MasoudKeirseY^A<8f><9a>J^E^G^D^B`^@+<80>^@Ùm^@^@^@^@G
K<81>^@^@^A^YZ^V<8f>UzMartinsCoors^A<8f><8c>ë^F^E^@^B`^@*^<80>^@Ùn^@^@^@^@GK<81>^@^@^A^YZ$<8
f>HÆArvinPetтей^B<8f><89>W^H^E^@^B _^@,<80>^@Ùo^@^@^@^@GK<81>^@^@^A^YZ2<8f>TuDinahYamaashi^A
<8f><87>U ^H^@^BÅ^@0<80>^@Ùp^@^@^@^@GK<81>^@^@^A^YZ<8f>I<87>JungsoonHaraldson^B<8f><
91>^V^H^F^D^BÈ^@-<80>^@Ùq^@^@^@^@GK<81>^@^@^A^YZN<8f>F§MantisMaginnis^B<8f><83>`^E^D^@^BĐ^@
(<80>^@Ùr^@^@^@^@GK<81>^@^@^A^YZ\<8f>P<93>GiriPerry^A<8f><89>%^F^D^@^B0^@)<80>^@Ùs^@^@^@^@G
K<81>^@^@^A^YZj<8f>M^B^ArnoGelosh^B<8f><89>p^D^E^@^Bà^@(<80>^@Ùt^@^@^@^@GK<81>^@^@^A^YZx<8f
>R<8b>SteinLunn^B<8f><87>^X^F^G^D^Bè^@,<80>^@Ùu^@^@^@^@GK<81>^@^@^A^YZ<86><8f>C^XShounakSao
udi^B<8f><82>c^E^C^@^Bð^@`<80>^@Ùv^@^@^@^@GK<81>^@^@^A^YZ<94><8f>Z>UlfRanft^A<8f><8c>y^E^F^
^@^B0^@*^<80>^@Ùw^@^@^@^@GK<81>^@^@^A^YZ<8f>RxVitalyPicel^B<8f><8c>Ê^N^H^@^C^@^5<80>^@Ùx^@^
^@^@^@GK<81>^@^@^A^YZ°<8f>L(JahangirTheuretzbacher^A<8f><8b><8c>^D^H^D^C^H^@+<80>^@Ùy^@^@^@
^@^@
```


Manually check the content

We can also manually open the file and verify what kind of data is stored:

```
fred@dell:~/mysql-sandboxes/3320 — /usr/bin/vim recovery/BY_TABLESPACE/00000002.pages
0096c020: 0000 0000 0002 002c 3b45 8153 1dc4 1d88 .....;E.S....
0096c030: 0000 0005 0000 00a8 0000 0000 0000 0000 .....
0096c040: 0000 0000 0000 0000 009a 0000 0000 0000 .....
0096c050: 0000 0000 0000 0000 0000 0000 0000 0100 .....
0096c060: 0200 1c69 6e66 696d 756d 0001 000b 0000 ...infimum.....
0096c070: 7375 7072 656d 756d 0706 0000 1000 2c80 supremum.....,
0096c080: 0027 1100 0000 0007 4482 0000 008d 0110 .'.....D.....
0096c090: 8f43 2247 656f 7267 6946 6163 656c 6c6f .C"GeorgiFacello
0096c0a0: 018f 84da 0607 0000 1800 2c80 0027 1200 .....',..
0096c0b0: 0000 0007 4482 0000 008d 011d 8f58 c242 ....D.....X.B
0096c0c0: 657a 616c 656c 5369 6d6d 656c 028f 8375 ezalelSimmel...u
0096c0d0: 0705 0000 2000 2b80 0027 1300 0000 0007 ....+. '.....
0096c0e0: 4482 0000 008d 012a 8f4f 8350 6172 746f D.....*.O.Parto
0096c0f0: 4261 6d66 6f72 6401 8f85 1c07 0904 0028 Bamford.....(
0096c100: 002f 8000 2714 0000 0000 0744 8200 0000 ./...'.....D....
0096c110: 8d01 378f 44a1 4368 6972 7374 6961 6e4b ..7.D.ChirstianK
0096c120: 6f62 6c69 636b 018f 8581 0807 0000 3000 oblick.....0.
0096c130: 2e80 0027 1500 0000 0007 4482 0000 008d ...'.....D.....
0096c140: 0144 8f46 354b 796f 6963 6869 4d61 6c69 .D.F5KyoichiMali
0096c150: 6e69 616b 018f 8b2c 0706 0000 3800 2c80 niak...,...8.,.
0096c160: 0027 1600 0000 0007 4482 0000 008d 0151 .'.....D.....Q
0096c170: 8f42 9441 6e6e 656b 6550 7265 7573 6967 .B.AnnekePreusig
0096c180: 028f 8ac2 0907 0000 4000 2f80 0027 1700 .....@./...'..
0096c190: 0000 0007 4482 0000 008d 015e 8f4a b754 ....D.....^J.T
617482,19 67%
```

Step 6: extract all the data

We can now process all the pages and store the records in a JSON file:

```
$ page_explorer -t employees.sql -o employees.json 00000002.pages
...
INFO page_explorer: Found (286 data + 0 node pointer)/286 records on index page 917
INFO page_explorer: Processed 891 pages, total records: 300024, potentially missing: 0,
                                                                Incomplete: 0
```


Step 6: extract all the data

We can now process all the pages and store the records in a JSON file:

```
$ page_explorer -t employees.sql -o employees.json 00000002.pages
...
INFO page_explorer: Found (286 data + 0 node pointer)/286 records on index page 917
INFO page_explorer: Processed 891 pages, total records: 300024, potentially missing: 0,
                                                                Incomplete: 0
```

```
$ ls -lh employees.json
-rw-r--r-- 1 fred fred 41M Jan  8 23:06 employees.json
```

Step 7: generate a CSV file

The easiest way to import the data back into MySQL is to generate a CSV file from the JSON file:

```
$ cat employees.json | jq -r '.[ ] | [.emp_no,  
  .birth_date, .first_name, .last_name, .gender,  
  .hire_date] | @csv' > employees.csv  
$ ls -lh employees.csv  
-rw-r--r-- 1 fred fred 17M Jan  8 23:07 employees.csv
```

Step 7: generate a CSV file

The easiest way to import the data back into MySQL is to generate a CSV file from the JSON file:

```
$ cat employees.json | jq -r '.[ ] | [.emp_no,  
  .birth_date, .first_name, .last_name, .gender,  
  .hire_date] | @csv' > employees.csv  
$ ls -lh employees.csv  
-rw-r--r-- 1 fred fred 17M Jan  8 23:07 employees.csv
```

```
$ wc -l employees.csv  
300024 employees.csv
```

Step 7: generate a CSV file - verify

```
$ head -n 2 employees.csv  
11523,"1953-01-25","Yuguang","Pezzoli","F","1992-12-23"  
11524,"1955-10-26","Xiadong","Standera","M","1988-09-20"
```


Step 7: generate a CSV file - verify

```
$ head -n 2 employees.csv
11523,"1953-01-25","Yuguang","Pezzoli","F","1992-12-23"
11524,"1955-10-26","Xiadong","Standera","M","1988-09-20"
```

```
SQL > select * from employees limit 2;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21

Step 7: generate a CSV file - verify

```
$ head -n 2 employees.csv
11523,"1953-01-25","Yuguang","Pezzoli","F","1992-12-23"
11524,"1955-10-26","Xiadong","Standera","M","1988-09-20"
```

```
SQL > select * from employees limit 2;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21

Remember we used an unsorted tablespace file.

Step 7: generate a CSV file - verify

```
$ grep Facello employees.csv | grep Georgi
55649,"1956-01-23","Georgi","Facello","M","1988-05-04"
10001,"1953-09-02","Georgi","Facello","M","1986-06-26"
```

```
SQL > select * from employees limit 2;
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21

Remember we used an unsorted tablespace file.

Step 8: import the data

We can now restart **MySQL** (don't forget to eventually mount the disk).

We then need to delete the table even if the tablespace has been already deleted:

```
SQL > set foreign_key_checks=0;  
Query OK, 0 rows affected (0.0006 sec)
```

```
SQL > drop table employees;  
Query OK, 0 rows affected (0.0189 sec)
```


Step 8: import the data

We can now restart **MySQL** (don't forget to eventually mount the disk).

We then need to delete the table even if the tablespace has been already deleted:

```
SQL > set foreign_key_checks=0;  
Query OK, 0 rows affected (0.0006 sec)
```

```
SQL > drop table employees;  
Query OK, 0 rows affected (0.0189 sec)
```

```
2025-01-08T22:12:32.080078Z 8 [Warning] [MY-012111] [InnoDB] Trying to access missing tablespace 2  
2025-01-08T22:12:47.120167Z 8 [ERROR] [MY-012592] [InnoDB] Operating system error number 2 in a file operation.  
2025-01-08T22:12:47.120202Z 8 [ERROR] [MY-012593] [InnoDB] The error means the system cannot find the path specified.  
2025-01-08T22:12:47.120210Z 8 [ERROR] [MY-012216] [InnoDB] Cannot open datafile for read-only: './employees/employees.ibd' OS error: 71
```

MySQL

Step 8: import the data (2)

We recreate the table:

```
SQL > CREATE TABLE `employees` (  
  `emp_no` int NOT NULL,  
  `birth_date` date NOT NULL,  
  `first_name` varchar(14) NOT NULL,  
  `last_name` varchar(16) NOT NULL,  
  `gender` enum('M','F') NOT NULL,  
  `hire_date` date NOT NULL,  
  PRIMARY KEY (`emp_no`)  
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
Query OK, 0 rows affected (0.0229 sec)
```

Step 8: import the data (3)

We load back the data using **MySQL Shell**:

```
MySQL 127.0.0.1:9100 employees 2025-01-08 23:15:25
JS \sql set global local infile=1;
Query OK, 0 rows affected (0.0002 sec)
MySQL 127.0.0.1:9100 employees 2025-01-08 23:15:31
JS util.importTable('employees.csv', {"dialect": "csv-unix"})
Importing from file '/home/fred/recovery/recovery/BY_TABLESPACE/employees.csv' to table `employees`.`employees` in
MySQL Server at 127.0.0.1:9100 using 1 thread
[Worker000]: employees.csv: Records: 300024 Deleted: 0 Skipped: 0 Warnings: 0
100% (16.82 MB / 16.82 MB), 16.82 MB/s
File '/home/fred/recovery/recovery/BY_TABLESPACE/employees.csv' (16.82 MB) was imported in 1.4291 sec at 11.77 MB/s
Total rows affected in employees.employees: Records: 300024 Deleted: 0 Skipped: 0 Warnings: 0
```

Let's verify

We can count the records to compare:

```
SQL > select count(*) from employees;
```

```
+-----+
```

```
| count(*) |
```

```
+-----+
```

```
| 300024 |
```

```
+-----+
```

```
1 row in set (0.0124 sec)
```


Let's verify (2)

And compare the first and last record:

```
SQL > select * from employees limit 1;
```

```
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi     | Facello   | M       | 1986-06-26 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.0013 sec)
```

```
SQL > select * from employees order by emp_no desc limit 1;
```

```
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 499999 | 1958-05-01 | Sachin     | Tsukuda   | M       | 1997-11-30 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.0009 sec)
```




MySQL InnoDB Data Recovery

Resources



Resources

- https://github.com/Codetector1374/InnoDB_rs
- https://github.com/YukiHinana/innodb_sort
- <https://github.com/altmannmarcelo/sdi2ddl>
- <https://dev.mysql.com/doc/refman/9.0/en/ibd2sdi.html>
- <https://github.com/KernelMaker/ibdNinja>



Share your ❤️ to **MySQL**

#mysql #MySQLCommunity



Join our slack channel!

bit.ly/mysql-slack

Questions ?

