TiDB 7.4 发版：正式兼容 MySQL 8.0

← 查看全部博客

作者：PingCAP 公司动态 2023-10-12

目录

MySQL 是全球最受欢迎的开源数据库，长期位于 DB-Engines Ranking 排行榜第二名，在世界范围内拥有数量庞大的企业用户和开发者。然而，随着时间的推移，MySQL 用户正面临新挑战。Oracle 官宣将在 2023 年 10 月终止 MySQL 5.7 版本的官方技术支持。据第三方统计显示，目前仍有超过一半的 MySQL 服务器运行在 5.7 版本。在未来几个月，大量的 MySQL 实例必须升级至 8.0 及更高版本，否则将无法享受 Oracle 提供的技术支持和重要补丁更新，企业级用户将面临重大考验。

TiDB 作为新一代分布式关系型数据库，从诞生第一天起拥抱 MySQL 生态，不断地兼容 MySQL 5.7 和 MySQL 8.0，为用户带来更加顺畅的迁移和使用体验。本文将介绍 TiDB 7.4 DMR 在 MySQL 8.0 兼容方面的新进展，探讨 TiDB 如何从根本上解决 MySQL 用户面临的各种挑战。

MySQL 用户的五大挑战

- 升级影响业务连续性。单实例或 "主从模式" 运行的 MySQL 升级时会造成数据库服务的停机，可能会对业务运营造成冲击。运行着大量 MySQL 实例的企业级用户，为了应对升级存在的潜在风险，需要投入大量的人力、物力进行测试和演练。
- 业务规模扩展困难。随着业务规模的扩大和数据使用场景的增多，用户通常需要在单机容量限制和分片管理复杂度之间进行权衡，数据库扩展的难度制约了业务规模和发展速度。
- 缺乏极致高可用方案。对于支撑核心业务场景的 MySQL 数据库来说，如果遇到不可预测的宕机事件，恢复业务变得复杂，达成极低的恢复时间目标（RTO）成为数据库管理员的挑战。
- 实时分析能力不足。MySQL 在处理大规模数据实时分析时性能不如在 OLTP（联机事务处理）场景下出色。这对于需要进行复杂查询和数据分析的业务是一个挑战。
- 原厂托管服务受限。虽然云服务商都会提供 MySQL 托管服务，但大多缺乏 Oracle 原厂的官方支持。这意味着在处理深层次的产品问题和发现通用功能需求时，用户无法获得来自数据库原厂的快速反馈和支持。

因此，迁移到一个成熟的产品并一举解决上述难题，无疑是明智之举。TiDB 就是 MySQL 全面升级的理想之选。选择 TiDB，不仅可以摆脱 MySQL 升级和扩展性的困境，还能够享受 HTAP、数据库整合等多方面的额外收益。

高度兼容 MySQL 的分布式关系型数据库 TiDB

TiDB 是由 PingCAP 自主研发的企业级分布式关系型数据库，具备水平扩缩容、金融级高可用、实时 HTAP、云原生、兼容 MySQL 5.7 协议和生态等重要特性。TiDB 采用原生分布式架构设计，具备灵活的弹性伸缩能力，整个过程对业务透明，无需人工干预。TiDB 的多副本存储和 Multi-Raft 协议确保数据的强一致性和高可用性，在部分副本发生故障时不影响数据的可用性。TiDB 通过滚动升级的方式使得版本更新的影响降至最低，此外可采用增加临时节点的方式，确保 TiDB 在升级过程中的性能波动和连接闪断控制在 5% 以内，大幅降低升级对业务的影响。

另外，作为 TiDB 的缔造者，PingCAP 基于全球领先云服务商推出数据库托管服务 TiDB Cloud，服务支持涵盖复杂问题诊断、升级支持、紧急救援等，充分体现了原厂服务的优势。

自 TiDB 7.4 DMR 开始，TiDB 正式兼容 MySQL 8.0

从项目初期开始，TiDB 坚持拥抱 MySQL 生态的产品战略一直延续至今。TiDB 兼容 MySQL 的 wire protocol 和语法命令，这意味着 MySQL 客户端、MySQL 驱动程序以及部分 MySQL 工具可以直接在 TiDB 上运行。对于绝大多数在 MySQL 上运行的应用程序来说，几乎不需要修改任何代码。

随着 MySQL 8.0 的发布，TiDB 在兼容 MySQL 5.7 的基础之上，积极扩展了对 MySQL 8.0 的兼容。TiDB v7.4.0 版本发布了对 MySQL 8.0 常用功能的支持，这使得平滑迁移 MySQL 8.0 的应用变得轻而易举。本文列举了部分功能：

公共表表达式（CTE）

作为 MySQL 8.0 引入的重要能力， TiDB 从 5.1 版本开始支持 ANSI SQL 99 标准的 CTE 及其递归的写法。在编写复杂查询的时候，利用公共表表达式 (CTE) 可以构建一个临时的中间结果集，在 SQL 语句中引用多次，提高 SQL 语句编写效率，可读性，执行效率。目前版本中，TiFlash 也同样支持 CTE。

比如表 `authers` 保存了作家的信息，`book_authors` 记录了作家 id 与其所编写书籍 id 的对应关系。

```
mysql> desc authors;
+------------+--------------+------+------+---------+-------+
| Field      | Type         | Null | Key  | Default | Extra |
+------------+--------------+------+------+---------+-------+
| id         | bigint(20)   | NO   | PRI  | NULL    |       |
| name       | varchar(100) | NO   |      | NULL    |       |
| gender     | tinyint(1)   | YES  |      | NULL    |       |
| birth_year | smallint(6)  | YES  |      | NULL    |       |
| death_year | smallint(6)  | YES  |      | NULL    |       |
+------------+--------------+------+------+---------+-------+
mysql> desc book_authors;
+-----------+------------+------+------+---------+-------+
| Field     | Type       | Null | Key  | Default | Extra |
+-----------+------------+------+------+---------+-------+
| book_id   | bigint(20) | NO   | PRI  | NULL    |       |
| author_id | bigint(20) | NO   | PRI  | NULL    |       |
+-----------+------------+------+------+---------+-------+
```

利用 CTE， 能够很容易编写出 SQL，列出最年长的 50 位作家分别编写过多少书籍。

```
mysql> WITH top_50_eldest_authors_cte AS (
    ->     SELECT a.id, a.name, (IFNULL(a.death_year, YEAR(NOW())) - a.birth_year) AS age
    ->     FROM authors a
    ->     ORDER BY age DESC
    ->     LIMIT 50
    -> )
    -> SELECT
    ->     ANY_VALUE(ta.id) AS author_id,
    ->     ANY_VALUE(ta.age) AS author_age,
    ->     ANY_VALUE(ta.name) AS author_name,
    ->     COUNT(*) AS books
    -> FROM top_50_eldest_authors_cte ta
    -> LEFT JOIN book_authors ba ON ta.id = ba.author_id
    -> GROUP BY ta.id;
+-----------+------------+--------------------+-------+
| author_id | author_age | author_name        | books |
+-----------+------------+--------------------+-------+
| 524470241 |         80 | Alexie Kirlin      |     7 |
|  67511645 |         80 | Bridgette Tromp    |     9 |
...
|  48355494 |         80 | Audrey Mayert      |     7 |
+-----------+------------+--------------------+-------+
50 rows in set (0.23 sec)
```

> *相关文档：https://docs.pingcap.com/zh/tidb/stable/dev-guide-use-common-table-expression*

窗口函数 (window function)

窗口函数(Window Function)，又被叫做分析函数， 在对数据进行分析、汇总、排序时会被用到。窗口函数能够以 SQL 形式的写法，来完成一些复杂的数据整理工作，协助用户发掘数据价值。例如，数据分组排序， 变化趋势分析等。

TiDB 目前已经完整支持了 MySQL 8.0 提供的窗口函数，大部分可以下推到 TiFlash 运行。

> *相关文档：https://docs.pingcap.com/zh/tidb/stable/window-functions*

资源管控

TiDB 在 7.1 版本引入了资源管控，目的是能够对集群资源做合理分配，提升数据库的稳定性，并降低数据库的使用成本。在多个应用共享一个 TiDB 集群的场景下， 资源隔离可以有效降低应用负载变化对其他应用产生的影响， 资源管理还能解决批量作业及后台任务对核心业务的影响，以及突发的 SQL 性能问题拖慢整个集群，是提升大集群稳定性的重要能力。

尽管和 MySQL 的实现方式有差别，TiDB 兼容了 MySQL 指定资源组的语法以及 hint，降低用户学习成本和迁移成本。另外，TiDB 的资源隔离能够更有效地对最重要的 I/O 资源进行管控，达到和 MySQL 同等甚至更好的效果。

下面展示了利用资源管控，将 `usr1` 使用的所有资源控制在每秒 500 RU 以内。

- 预估集群容量

```
mysql> CALIBRATE RESOURCE
```

- 创建 app1 资源组，限额是每秒 500 RU

```
mysql> CREATE RESOURCE GROUP IF NOT EXISTS app1 RU_PER_SEC = 500;
```

- 将用户与资源组关联， usr1 的会话自动关联到资源组 app1

```
mysql> ALTER USER usr1 RESOURCE GROUP app1;
```

- 也可以修改会话所属的资源组

```
mysql> SET RESOURCE GROUP `app1`;
```

- 或者利用 hint RESOURCE_GROUP() 指定语句所属的资源组

```
mysql> SELECT /*+ RESOURCE_GROUP(rg1) */ * FROM t limit 10;
```

> *相关文档：https://docs.pingcap.com/zh/tidb/stable/tidb-resource-control*

基于角色的权限管理

TiDB 支持 MySQL 兼容的角色管理。基于角色的授权，可以简化权限管理的工作，并降低了出错的风险。通过将权限与角色相关联，可以更好地控制数据库的访问。客户可以将不同场景的工作进行分类，创建对应角色，并把角色授予有权限的数据库用户， 数据库用户在实际操作时，根据场景不同，切换角色，降低误操作的可能。

这里举一个利用角色拆分权限场景的例子。用户 `dev_adm_usr` 作为应用管理员，要操作数据库 `app_db` 的数据，多数情况下只是查询，偶尔在需要做数据修正的时候才会做修改。为了防止 `dev_adm_usr` 的误操作，我们将两部分权限利用角色拆开，只有必要的时候，才给自己赋予读写的角色。

- 创建角色 app_read_role 和 app_write_role

```
mysql> CREATE ROLE 'app_read_role', 'app_write_role';
```

- 为角色授予对应的权限，这里为两个角色分别授予 `app_db` 的读和写的权限

```
mysql> GRANT SELECT ON app_db.* TO 'app_read_role'@'%';
mysql> GRANT INSERT, UPDATE, DELETE ON app_db.* TO 'app_write_role'@'%';
```

- 将两个角色授予用户 `dev_adm_usr`

```
mysql> GRANT 'app_read_role','app_write_role' TO 'dev_adm_usr'@'localhost';
```

- 把 `app_read_role` 设为 `dev_adm_usr` 的默认角色，这样用户 `dev_adm_usr` 登录时默认是只读权限

```
mysql> SET DEFAULT ROLE 'app_read_role' TO 'dev_adm_usr'@'localhost';
```

- 当 `dev_adm_usr` 需要修改数据时，启用角色 `app_write_role`

```
mysql> SET ROLE app_read_role,app_write_role;
```

或者启用所有角色

```
mysql> SET ROLE ALL;
```

> *相关文档：https://docs.pingcap.com/zh/tidb/stable/role-based-access-control*

增强 uft8mb4 字符集

MySQL 8.0 的一个重要变化是默认字符集变成了更通用的 `uft8mb4`，默认排序方式变为 `utf8mb4_0900_ai_ci`。TiDB 在新版本里也加入了 `utf8mb4_0900_ai_ci` 的排序方式，以便更轻松地进行系统迁移。

为了同时兼容 MySQL 5.7 和 MySQL 8.0，TiDB 支持了 MySQL 兼容的变量 `default_collation_for_utf8mb4`。允许用户调整 `utf8mb4` 字符集的默认排序方式。这个方式确保了 TiDB 在不同 MySQL 版本之间的平滑过渡，并能够适应不同应用程序的需求。

如果从 MySQL 8.0 迁移，设为 8.0 默认排序 `utf8mb4_0900_ai_ci`

```
set global default_collation_for_utf8mb4='utf8mb4_0900_ai_ci';
```

如果从 MySQL 5.7 迁移，设为 5.7 为 `utf8mb4` 的默认排序 `utf8mb4_general_ci`

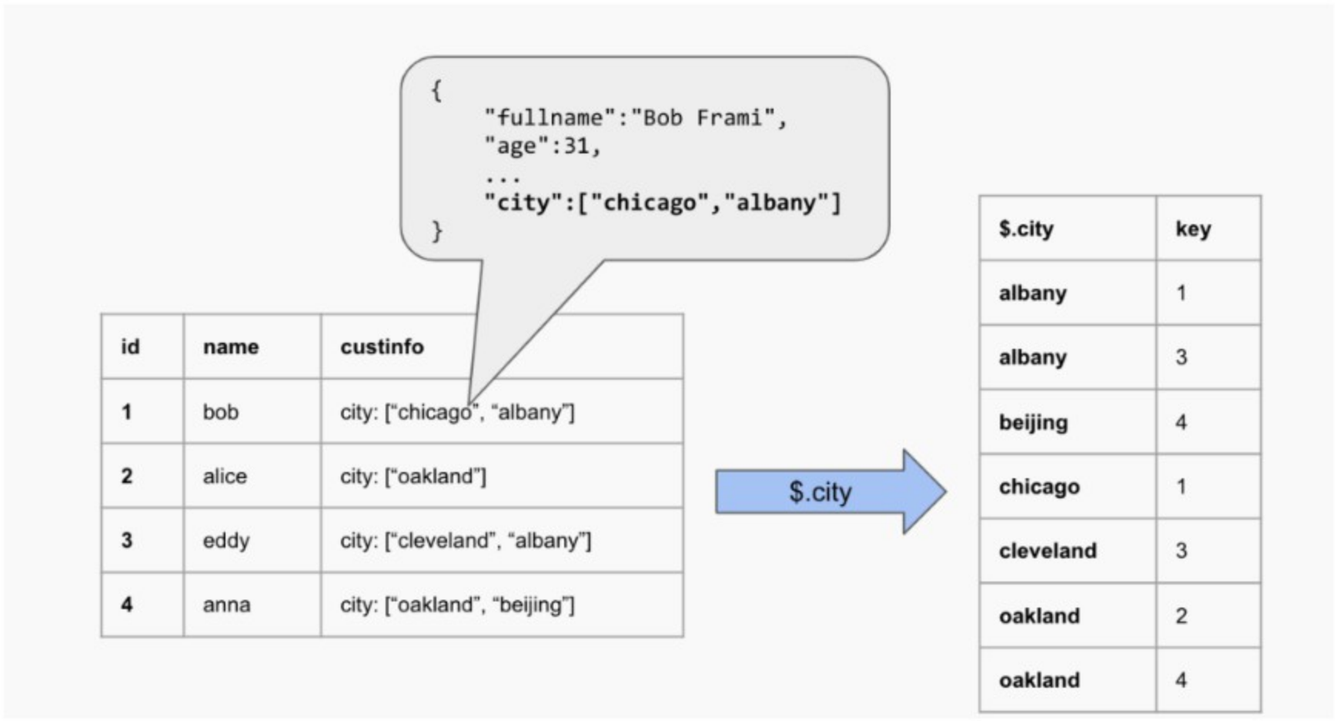
```
set global default_collation_for_utf8mb4='utf8mb4_general_ci';
```

JSON 多值索引 (Multi-valued Index)

在支持了 MySQL 5.7 的完整函数之后，TiDB 在不断添加对 MySQL 8.0 新发布功能的支持。最近的版本支持了"多值索引"，允许对 JSON 类型中的某个"数组"进行索引，从而提高了对 JSON 数据的检索效率。与 MySQL 用法完全相同，这意味着在迁移过程中，无需修改数据建模或应用程序，用户可以继续按照熟悉的方式操作 JSON 数据。

多值索引是对普通索引结构的延伸。不同于普通索引与表 1:1 的对应关系， 多值索引与表的对应是 N:1。与 MySQL 相同，条件中利用 `MEMBER OF()`，`JSON_CONTAINS()`，`JSON_OVERLAPS()` 这几个函数检索时，都可能会选择到多值索引。

比如，我们假定有一张客户信息表，所有详细信息以 JSON 格式编入一个 JSON 类型的列中， 其中有一个数组结构保存客户所在的几个城市。



当我们需要检索哪些客户在北京时，如果没有多值索引，这个查询需要扫描整张表。

```
SELECT name FROM customer
WHERE 'beijing' MEMBER OF $.city;
```

这时我们可以针对 `city` 这个数组创建多值索引，上述查询就可以利用索引检索符合的记录，大幅提升查询性能。

```
ALTER TABLE customers add index idx_city (name, (CAST(custinfo->'$.city' AS char(20) ARRAY)));
```

和普通索引一样， 当优化器没有选择到多值索引时，可以利用优化器提示 `USE_INDEX()`或 `USE_INDEX_MERGE()` 强制优化器做选择。

> *相关文档：https://docs.pingcap.com/zh/tidb/stable/choose-index#%E4%BD%BF%E7%94%A8%E5%A4%9A%E5%80%BC%E7%B4%A2%E5%BC%95*

修改会话变量的 hint（SET_VAR()）

MySQL 8.0 引入了一个特殊的 hint `SET_VAR()`。利用这个 hint，可以在语句运行期间修改某个会话级系统变量。TiDB 在 v7.4.0 也支持了这个 hint，提升了系统变量设置的灵活度， 能够针对 SQL 语句做“定制”。包括优化器相关的，执行时相关的多个变量都支持用 hint 修改。

比如，针对大表的分析处理，适当增加 SQL 的执行并行度。

```
SELECT /*+ set_var(tidb_executor_concurrency=20) */
    l_orderkey,
    SUM(
        l_extendedprice * (1 - l_discount)
    ) AS revenue,
    o_orderdate,
    o_shippriority
FROM
    customer,
    orders,
    lineitem
WHERE
    c_mktsegment = 'BUILDING'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < DATE '1996-01-01'
AND l_shipdate > DATE '1996-02-01'
GROUP BY
    l_orderkey,
    o_orderdate,
    o_shippriority
ORDER BY
    revenue DESC,
    o_orderdate
limit 10;
```

你也可以利用这个 hint 强制刚才的查询选择 TiFlash，而其他查询保持不变。

```
SELECT /*+ set_var(tidb_isolation_read_engines='tidb,tiflash') */
    l_orderkey,
    SUM(
        l_extendedprice * (1 - l_discount)
    ) AS revenue,
    o_orderdate,
    o_shippriority
FROM
    customer,
    orders,
    lineitem
WHERE
    c_mktsegment = 'BUILDING'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < DATE '1996-01-01'
AND l_shipdate > DATE '1996-02-01'
GROUP BY
    l_orderkey,
    o_orderdate,
    o_shippriority
ORDER BY
    revenue DESC,
    o_orderdate
limit 10;
```

> *相关文档：https://docs.pingcap.com/zh/tidb/v7.4/optimizer-hints#set_varvar_namevar_value*

CHECK 约束

CHECK 约束 是一致性约束检查的一种，用来维护数据的准确性。CHECK 约束可以用于限制表中某个字段的值必须满足指定条件。当为表添加 CHECK 约束后，在插入或者更新数据时，TiDB 会检查约束条件是否满足，如果不满足，则会报错。

MySQL 在 8.0 之前只支持 CHECK 约束的语法，在实际运行中并不会真正去检查， 在 8.0 之后才全面支持。TiDB 在新版本中也添加了这个功能， 为了防止客户的 DDL 中有残存的 CHECK 条件，可能会因为这个特性产生问题，TiDB 默认并不会开启 CHECK 约束的检查，而是通过变量 `tidb_enable_check_constraint` 手工开启， 这充分体现了 TiDB 同时兼容 MySQL 5.7 和 8.0 的产品策略。

```
mysql> set global tidb_enable_check_constraint=on;
mysql> CREATE TABLE t
    -> ( a INT CHECK(a > 10) NOT ENFORCED, -- 不生效 check
    ->   b INT,
    ->   c INT,
    ->   CONSTRAINT c1 CHECK (b > c)
    -> );
mysql> insert into t values (20,20,20);
ERROR 3819 (HY000): Check constraint 'c1' is violated.
```

> *相关文档：https://docs.pingcap.com/zh/tidb/dev/constraints#check-%E7%BA%A6%E6%9D%9F*

TiDB 工具生态提供平滑迁移体验

为了降低用户数据迁移的复杂度，TiDB 推出了一款工具 TiDB Data Migration (DM) 。它能够协助用户从与 MySQL 协议兼容的数据库（MySQL、MariaDB、Aurora MySQL）到 TiDB 的全量数据迁移和增量数据同步。DM 支持 DDL 同步，分库分表合并，并内置多种过滤器以灵活适应不同场景，切实地提升了数据迁移的效率。

写在最后

TiDB 7.4 将是 TiDB 7 系列最后一个 DMR 版本，针对 MySQL 8.0 做出了诸多优化。作为 MySQL 的全面升级，TiDB 的技术领先性帮助用户应对不断变化的业务数据挑战，实现业务的持续增长和创新。TiDB 在高度兼容 MySQL 5.7 和 MySQL 8.0 特性的同时，也将持续提供技术支持，确保用户能够平滑地迁移各类业务应用程序，从而减少迁移过程中的工作量和风险。

浏览 TiDB 7.4 Release Notes，了解更多新增和优化特性。

Release　TiDB

隐私政策
Cookie 政策
安全合规
版本支持策略
漏洞管理策略
网站使用条款

GitHub
PingCAP Education
商标下载及使用

010-53326356

媒体合作
pr@pingcap.com

联系我们

友情链接

OSS Insight
TiKV
Chaos Mesh
亚马逊云科技