SOFAStack

数据访问代理
技术白皮书

产品版本：AntStack Plus 1.11.0

文档版本：20220929

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.什么是数据访问代理

产品背景

数据访问代理是蚂蚁集团自主研发的金融级分布式数据库中间件，用于解决海量请求下数据访问的瓶颈及数据库的容灾问题，提供水平拆分、平滑扩缩容、读写分离的在线分布式数据库服务。十年来专注于为海量数据访问提供低消耗、高性能、高可用的轻量级解决方案，确保在高并发、数据库异常的情况下依然非常稳定与可靠。数据访问代理兼容 MySQL 协议和语法，支持分库分表、平滑扩容、服务升降配、透明读写分离和分布式事务等特性，具备分布式数据库全生命周期的运维管控能力。

发展现状

全面在蚂蚁集团站内、网商银行以及外部银行及公有云使用，功能成熟度较高。

面临的问题及关键挑战

数据访问代理主要应用场景在大规模在线数据操作上，通过贴合业务的拆分方式，将操作效率提升到极致，有效满足用户在线业务对关系性数据库要求。数据访问代理是一个分布式数据库系统，是一个实现了 MySQL 协议的服务器，前端用户可以把它看作是一个数据库代理，用 MySQL 客户端工具和命令行访问，而其后端可以用 MySQL 原生协议与多个 RDS 或者 OceanBase 服务器通信，解决了传统关系型数据库难以扩展，不可切分的现状，可以避免单机（单库）的性能缺陷，解决数据存储和业务规模迅速增长情况下的数据瓶颈问题。

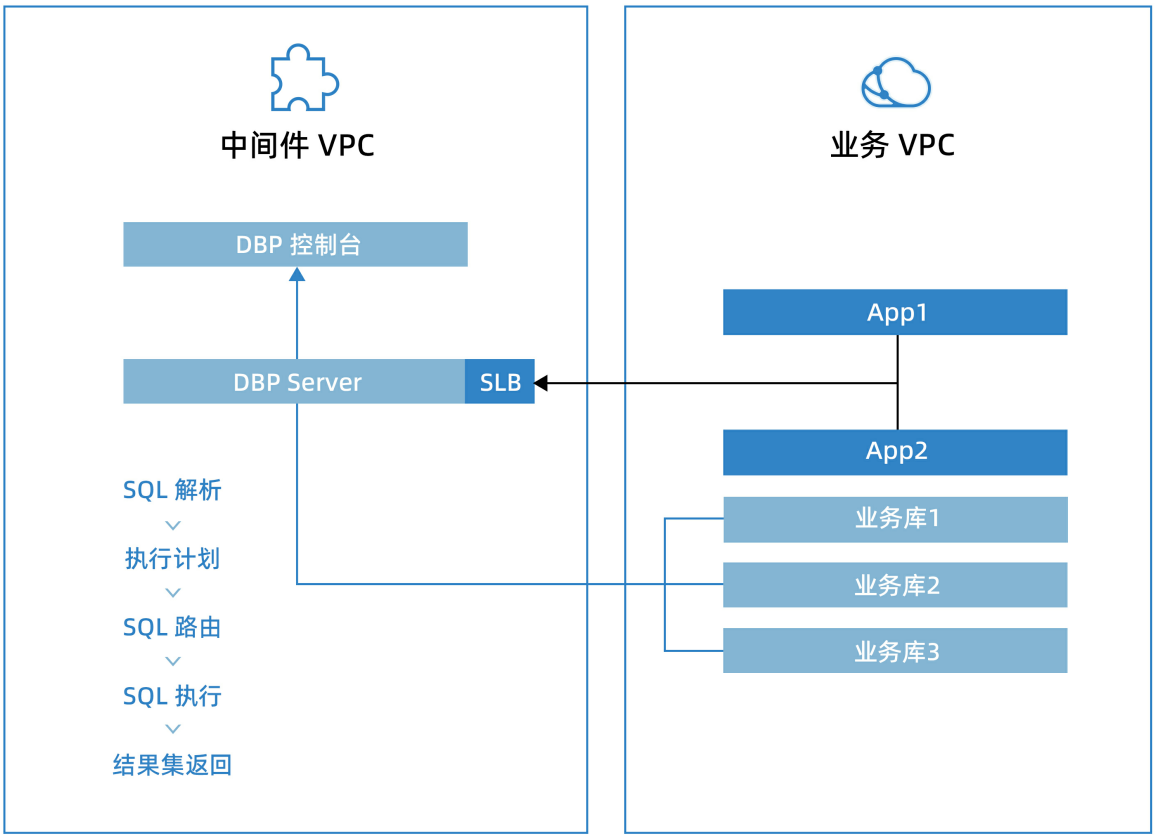
2.产品优势

数据访问代理主要有以下优势：

- **分布式**：数据读写存储集群化，不受单机限制，业务使用无连接数限制。
- **弹性**：数据服务可升降配，数据存储白屏化 scale-up 和 scale-out，读写分离线性提升读能力。
- **高性能**：分库分表经典方案让操作聚焦少量数据，多种拆分方式适应数据特点，并具备特定 SQL 并行执行能力，进一步提升执行效率。
- **安全**：完整的类单机 MySQL 账号体系，提供具备授权鉴权的 Open API 方便集成能力到业务管控中，产品服务支持体系化。
- **简单易用**：兼容 MySQL 协议和大部分 MySQL SQL 语法，无业务侵入式使用读写分离，全面的运维和监控能力。
- **成熟度高**：基于蚂蚁集团内部多年的金融级数据容灾场景，是阿里巴巴集团接入关系型数据库的标准。

3.产品架构

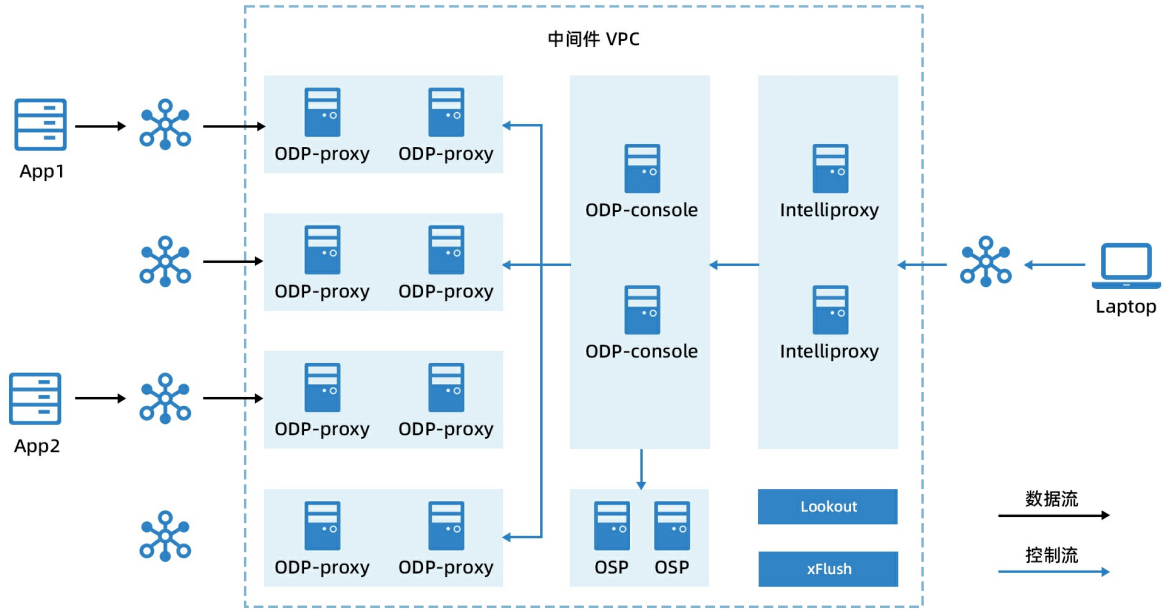
系统架构



数据访问代理主要有两个组件，即数据访问代理控制台、数据访问代理 Server:

- 数据访问代理控制台：主要用来管理用户的数据库节点、数据库分库分表配置及运维管控操作。
- 数据访问代理 Server：是一个支持标准 MySQL 协议的服务端，提供了分库分表、读写分离、分布式序列等功能。

网络架构



该部署拓扑图中各组件的说明如下：

- ODP-console（原 dbp-console）和 ODP-proxy（原 DBP-proxy）构成了数据访问代理，这两个组件的作用如下：
  - ODP-console（控制台）：Web 应用系统，负责配置数据库、数据表规则等。
  - ODP-proxy（代理服务器）：数据访问代理服务器，提供 MySQL 服务。
- OSP 和 Intelliproxy 是中间件中台组件，为数据访问代理提供基本中台服务。
- Lookout 和 xflush 是监控组件，为数据访问代理提供监控服务。

4.性能指标

性能压测结果

不同的压测环境和硬件配置下，相应的性能各不相同。详细的性能数据如下表所示：

数据访问代理实例规格	并发数	每秒 read/write 数量
4C8G	50	7681
8C16G	50	15492
16C32G	80	32703

压测环境和硬件配置

本次性能压测中，使用的环境和硬件配置如下：

- **数据访问代理可用区：**上海金区
- **压力机：**4C8G，VPC 网络，CentOS
- **RDS：**8C16G MySQL 5.6 * 2

压测模型

在此压测环境下，使用 sysbench 对 2 库 4 表进行压测。

> 注意
>
> 数据访问代理暂不支持建表语句和跨库的多条 INSERT，请直连数据库单个表初始化数据。

```
CREATE TABLE `sbtest1` (
       `id` int(11) NOT NULL AUTO_INCREMENT,
       `k` int(11) NOT NULL DEFAULT '0',
       `c` char(120) NOT NULL DEFAULT '',
       `pad` char(60) NOT NULL DEFAULT '',
        KEY `xid` (`id`),
        KEY `k_1` (`k`)
) ENGINE=InnoDB
```

--test='/usr/local/share/sysbench/oltp_read_write.lua' 测试模型 oltp.lua

--oltp-table-size=10000000 准备1千万数据

--auto_inc=off 关闭自增主键

--secondary 将 id 设置为非主键防止主键冲突

--skip_trx=on 跳过事务

--range_selects=off 关闭范围查询

--db-ps-mode=disable 关闭 PS 模式

--threads=100 并发数

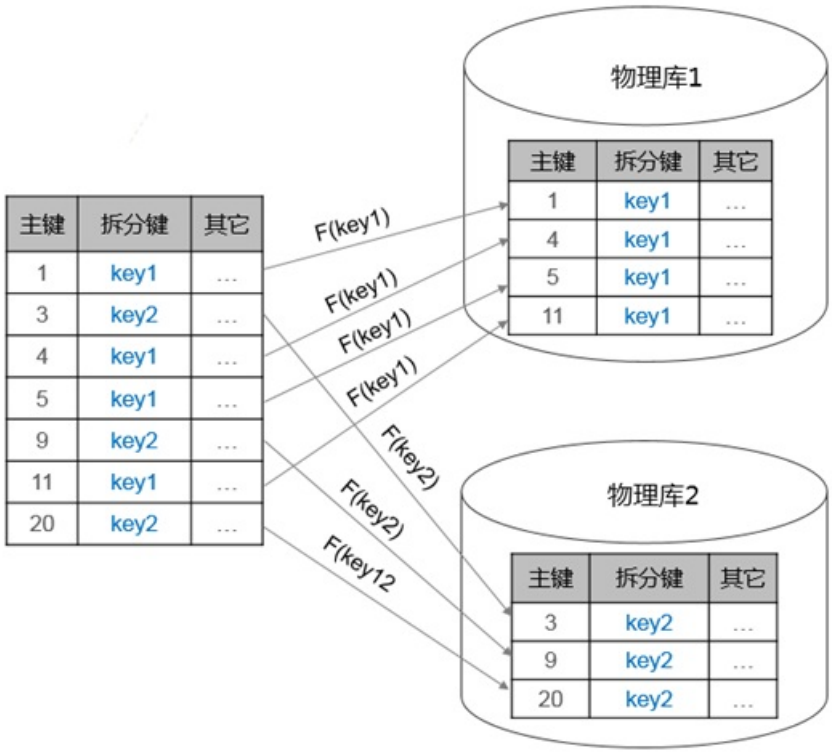
5.功能原理

5.1. 分库分表

数据访问代理将后端将数据量较大的数据表水平拆分到各个 RDS 数据库中，后端的这些 RDS 数据库被称为分库，分库中的表被称为分表。拆分后，每个分库负责一份数据的读写操作，从而有效的分散了整体访问压力。在系统扩容时，只需要水平增加分库的数量，并且迁移相关数据，就可以提高数据库访问代理系统的总体容量。

数据访问代理根据指定拆分键的值，采用特定的算法进行计算，然后根据计算结果将数据存储到对应的分库/分表中。拆分键即分库/分表字段，因此分为分库键和分表键。

- 分库键：数据库访问代理根据分库键的值将数据水平拆分到后端的各个 RDS 分库里。键值相同的数据一定位于同一个 RDS 数据库。
- 分表键：每一张逻辑表都可以定义自己的分表键，键值相同的数据一定位于同一个 RDS 数据表。



> 注意
>
> 在执行带有 WHERE 条件的 UPDATE、DELETE 语句时，如果 SQL 语句中没有使用拆分键，或者虽然指定了拆分键但是范围太广，会导致 SQL 语句被分发到所有分库上执行（即全表扫描），且执行结果会在数据库访问代理中进行合并。全表扫描响应较慢，在高并发业务场景中应尽量避免使用。

5.2. SQL 路由

在分库分表模式下，数据访问代理会根据拆分键（即拆分字段）以及 SQL 语义把 SQL 语句分发到底层中各个存储数据的分表进行执行。执行结束后，数据访问代理会将从各个分表获取的数据合并，然后返回给用户。本文介绍在分库分表场景中数据访问代理执行 SQL 语句时的路由原理。

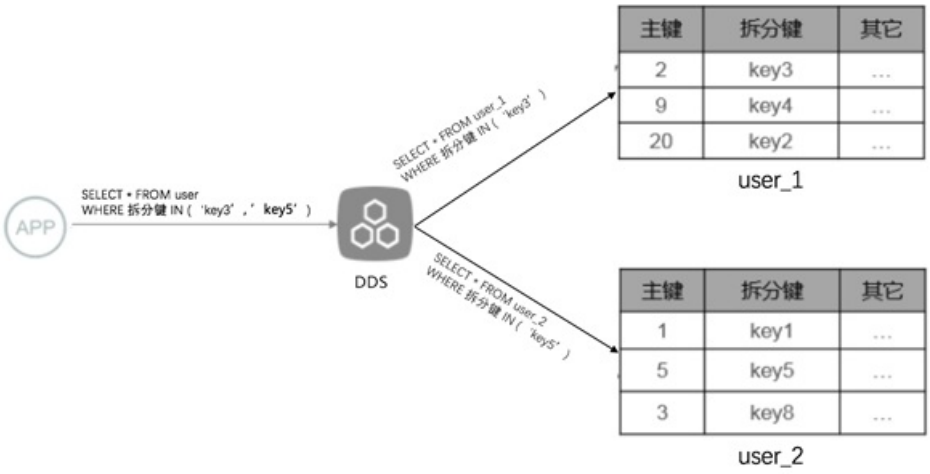
拆分键

分库分表过程中，数据访问代理按照指定的拆分键，采用特定的算法进行计算，然后根据计算结果将数据存储到对应的分表中。拆分键是数据访问代理中数据分布和 SQL 路由的凭证。

主键	拆分键	其它
1	key1	...
3	key2	...
4	key1	...
5	key1	...
9	key2	...
11	key1	...
20	key2	...

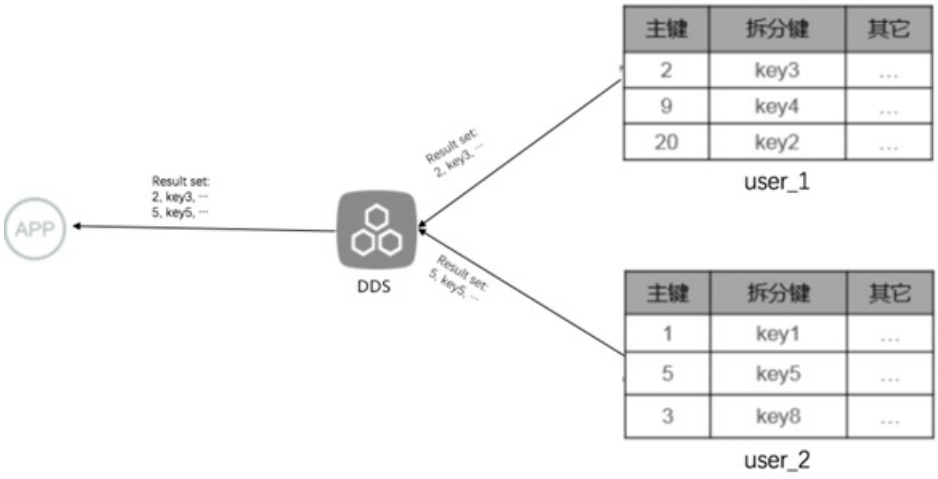
SQL 路由

当用户发起执行 SQL 语句的请求时，数据访问代理会理解 SQL 语句的含义，然后按照拆分键的值和执行策略将 SQL 语句路由到对应分区进行执行，如下图所示：



数据合并

如果一个语句被路由到多个分表执行，数据访问代理会将各个分表返回的数据按照原始 SQL 语义进行合并，并将最终结果返回给用户。

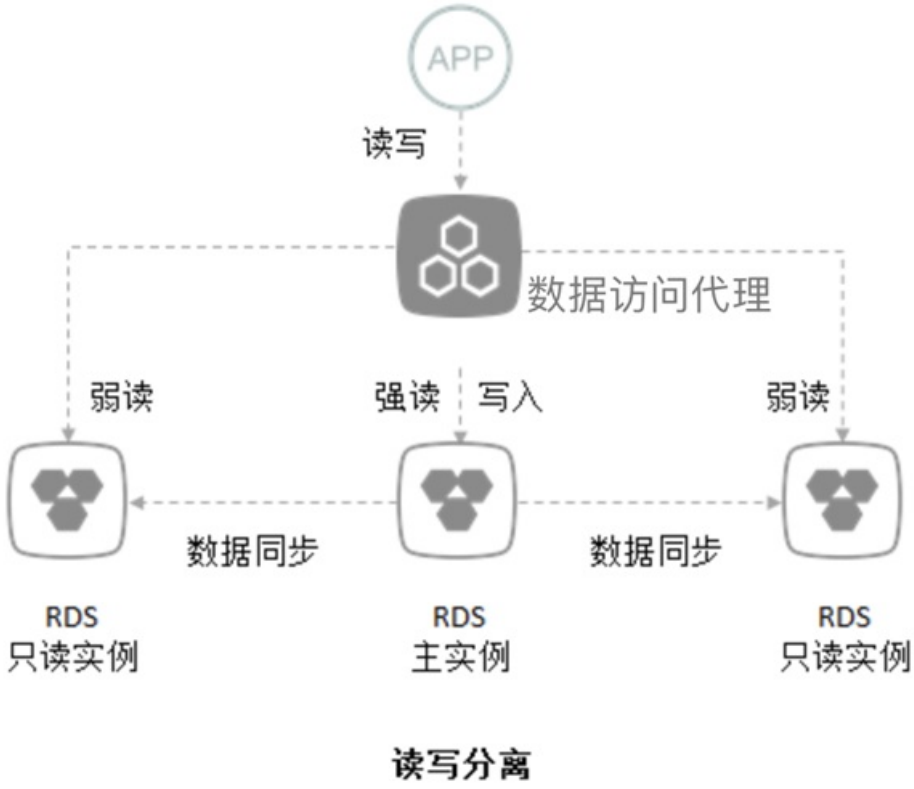


5.3. 读写分离

在主实例的读请求较多、读压力比较大的时候，可以通过数据访问代理读写分离功能对读流量进行分流，减轻 RDS 主实例的读压力。

数据访问代理的读写分离功能是对应用透明的设计。应用在不修改任何代码的情况下，只需要在数据访问代理控制台中调整读权重，即可将读流量按配置的比例在主 RDS 实例与多个 RDS 只读实例之间进行分流；写流量则全部到主实例，不做分流。

设置读写分离后，从主 RDS 实例读取的是强读，既实时强一致读。而只读实例上的数据是从主实例上异步复制的，存在毫秒级的延迟，因此从只读 RDS 实例读取的是弱读，属于非强一致性读。在金融级业务场景下，当需要实时性、强一致性读时，可以通过数据访问代理的控制台的数据库设置页面来关闭读写分离功能，这时数据访问代理会保证访问该库的 SQL 语句只在主 RDS 实例上执行。



读写分离

5.4. 分布式序列

数据访问代理提供了生成分布式环境下的分布式唯一序列的能力，该序列有全局唯一、全局递增的特性，常用于分库分表下的主键、业务主键生成的场景。

> 注意
>
> 数据访问代理分布式序列功能是基于数据库实现，如果需要使用该功能，需要在业务数据库中创建 odp_sequence 表。

普通序列

数据访问代理的分布式序列功能提供了类 Oracle 语法的 SQL 语句，`seq_name.nextval` 其中 seq_name 是任意字符串，一般是一张逻辑表使用同一个 seq_name。使用如下：

基于单库单表的 odp_sequence 表实现为 `SELECTorder_seq.nextval FROM dual` 。

业务序列

如前面提到的，分布式序列功能基于数据库表实现，odp_sequence 可以部署成单库单表模式，同样也可以部署成分库分表模式，分库分表模式下有如下优点：

- 提升 Sequence 表的读写能力。
- 提升 Sequence 表的可用性，无单点故障。
- 通过将 Sequence 表和业务数据表部署在一起，保持数据拆分规则一致，方便生成业务主键。

数据访问代理的分布式序列在 nextVal 语法基础下，扩展了更多业务型的字段，在获得 Sequence 值之外，还可以获得更多分库分表相关的信息，开发者可以根据实际场景灵活组装业务的序列号。使用如下：

```
SELECT
order_seq.timestamp, order_seq.dbtimestamp,order_seq.groupid,
order_seq.tableid, order_seq.nextval
FROMdual
WHERE sharding_col = ?
```

其中各项字段的含义：

- timestamp：获取 Sequence 的时间戳，该时间是机器时间。
- dbtimestamp：获取 Sequence 数据库的时间，使用该命令的话，数据访问代理会实时向物理数据库请求当前时间。
- groupid：获取 Sequence 所在分片的 ID。
- tableid：获取 Sequence 所在分表的 ID。
- sharding_col：获取 Sequence 的分库分表字段，该字段是虚拟并不真实存在，用以方便数据访问代理计算分库分表规则。

配置分表规则

在数据库中添加 odp_sequence 表的分表规则，以下是以 100 个分片 100 个分表，以 sharding_col 字段 hash 规则为示例：

注意事项

- 分库分表场景下，nextval 是每个分片/分表独立递增，不同分片/分表的 nextval 会有重复的情况。所以请拼上其他信息用以区分，如 groupid。
- 不保证严格递增，且也有上限，默认 nextval 在 1 ~ 99999999 中重复循环。
- 单条 SQL 中不支持多个 Sequence 名字。

细节说明

分布式序列功能是基于数据库的表来实现，主要是基于 odp_sequence 表，参考附录。在当前的机制中，表中存放当前某个 Key 的最新 value odp_sequence 表申请一个步长（step 字段）的 Sequence 段。拿到 Sequence 段之后，value 值会在数据访问代理的内存中做原子递增。如果发现当前值大于 Sequence 段的最大值，开始获取新的 Sequence 段，关键 SQL 实现如下：

- `select * from sequence where name = ?`
- `update sequence set value = $currentValue + step * (1 + random(1)) where value =$currentValue and name = ?`

附录

建表语句如下：

```
CREATE TABLE odp_sequence (
      `id` INT AUTO_INCREMENT,
      `name` VARCHAR(255),
      `value` INT,
      `min_value` BIGINT,
      `max_value` BIGINT,
     `step` BIGINT,
     `gmt_create` DATETIME,
     `gmt_modified` DATETIME,
     PRIMARY KEY (`id`),
     UNIQUE KEY (`name`)
);
```

字段说明：

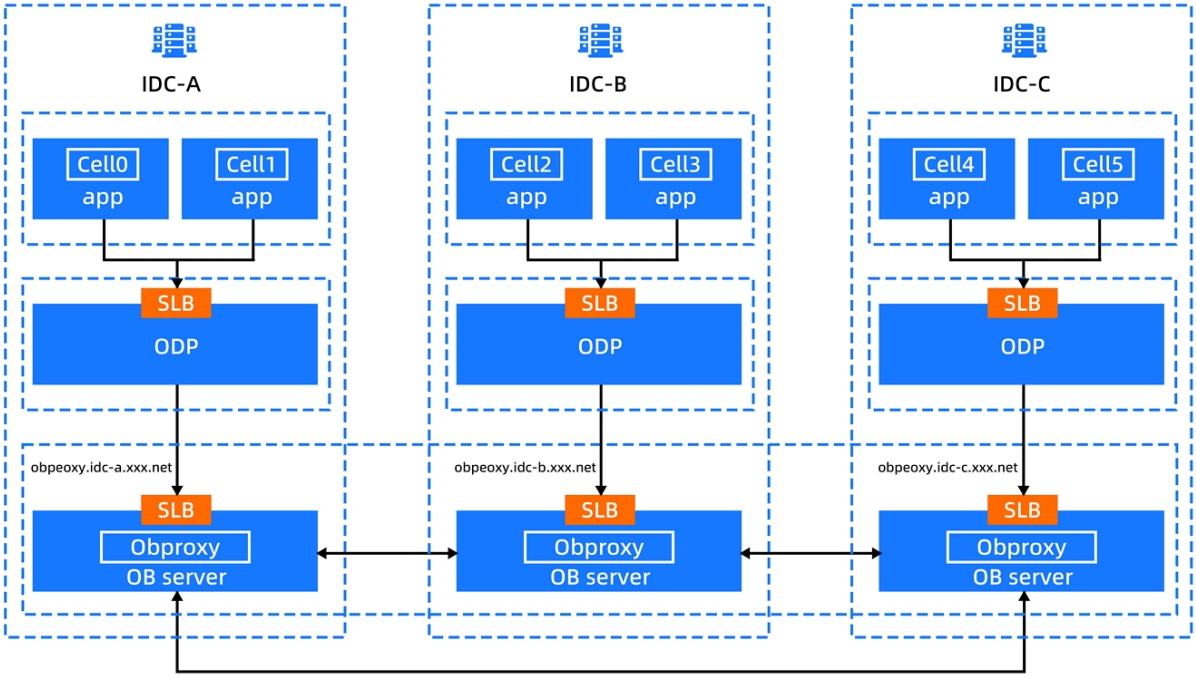
字段名称	类型	说明
name	varchar	记录对应的业务表名，例如 trade_order。
min_value	int	Sequence 最小值, 用于校验 Seqeunce 不能低于该数值，否则报错，默认 1。
max_value	int	Sequence 最大值，当到达最大值以后，将从 min_value 开始重新增加，默认 99999999。
step	int	一次获取的 Sequence 区间，默认 10000。
value	bigint	当前 Sequence 值。
gmt_create	DATETIME	创建时间
gmt_modified	DATETIME	修改时间

6.单元化能力

数据访问代理 1.5.0 开始支持单元化能力。单元化部署架构下，数据访问代理实例跨机房部署，每个机房（IDC）部署一套数据访问代理集群，业务应用在访问数据访问代理时需寻址到当前机房的数据访问代理集群访问，数据访问代理内部根据业务配置的数据库地址访问当前机房的数据库连接，完成数据的读写。

> **说明**
>
> 仅适用于开通了专有云单元化功能的用户。

单元化下，数据访问代理部署架构如下图所示：



7.附录：基础术语

实例

实例是数据访问代理的集群，相当于一个逻辑的物理数据库实例。实例基于 MySQL 标准接口，作为统一的 SQL 请求入口，所有 SQL 都需要请求给实例进行处理，实例会负责将访问请求发到正确的物理数据库上。

数据库

表示数据访问代理的逻辑数据库。以这个作为访问的入口，通过分库分表、读写分离等规则可以请求到后端真实的物理数据库。

数据表

表示数据访问代理的逻辑表，应用访问数据访问代理时中的表即数据表。一个逻辑表会对应多个物理表，数据访问代理在路由时，会将逻辑表名替换成物理表名。

逻辑表

同数据表。

物理数据库

真实的物理数据库。可能是 RDS（MySQL）、OceanBase 等。目前支持 RDS（MySQL） 和 OceanBase 两种物理数据库。

物理数据源

同物理数据库。

物理表

物理库内存放的真实表。

数据库分片

表示分库分表中的数据库分库。分库是一个逻辑上的概念，物理上可能是一个物理数据库代表一个分库，也可能是多个物理数据库组成一个分库，在数据访问代理里面统一概念称分片。

分片

同数据库分片。

数据节点

代理层后端的用户物理数据节点实例，类型可以是 MySQL（RDS）或者 OceanBase。