

Your Logo

实现PostgreSQL逻辑复制实战

王青松

神州飞象(北京)数据科技有限公司

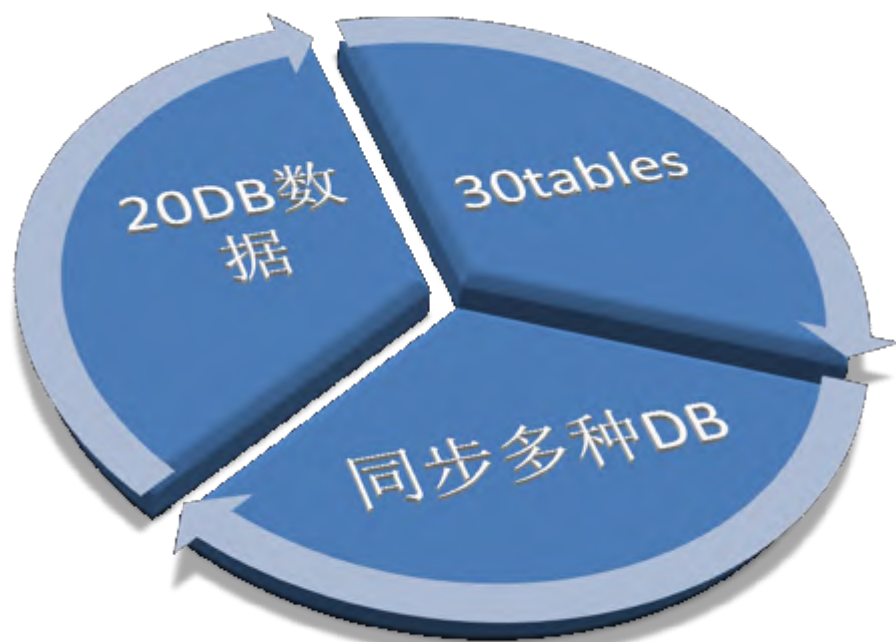
简介



逻辑复制的功能是从PG的WAL日志中，读取数据库更新信息，然后“翻译”（Decode）成逻辑的形式，可发送到远程从库做数据同步。



为什么要选择逻辑复制？



数据结构变化几乎没有变化

对数据库主要是增删改查等DML操作

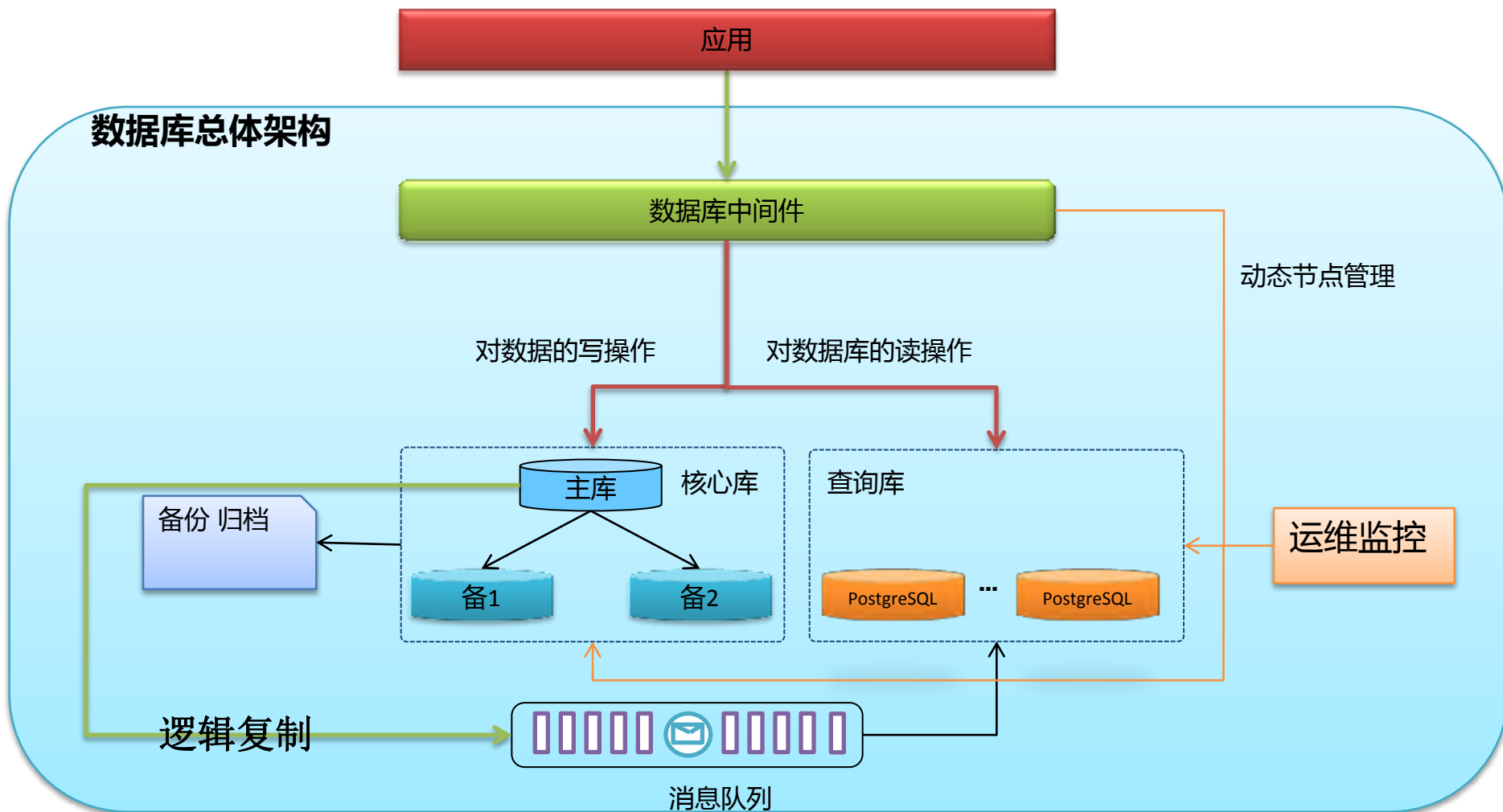
数据同步允许延迟

查询库为不同类型关系型数据库

TP系统，事务比较小



数据库总体架构

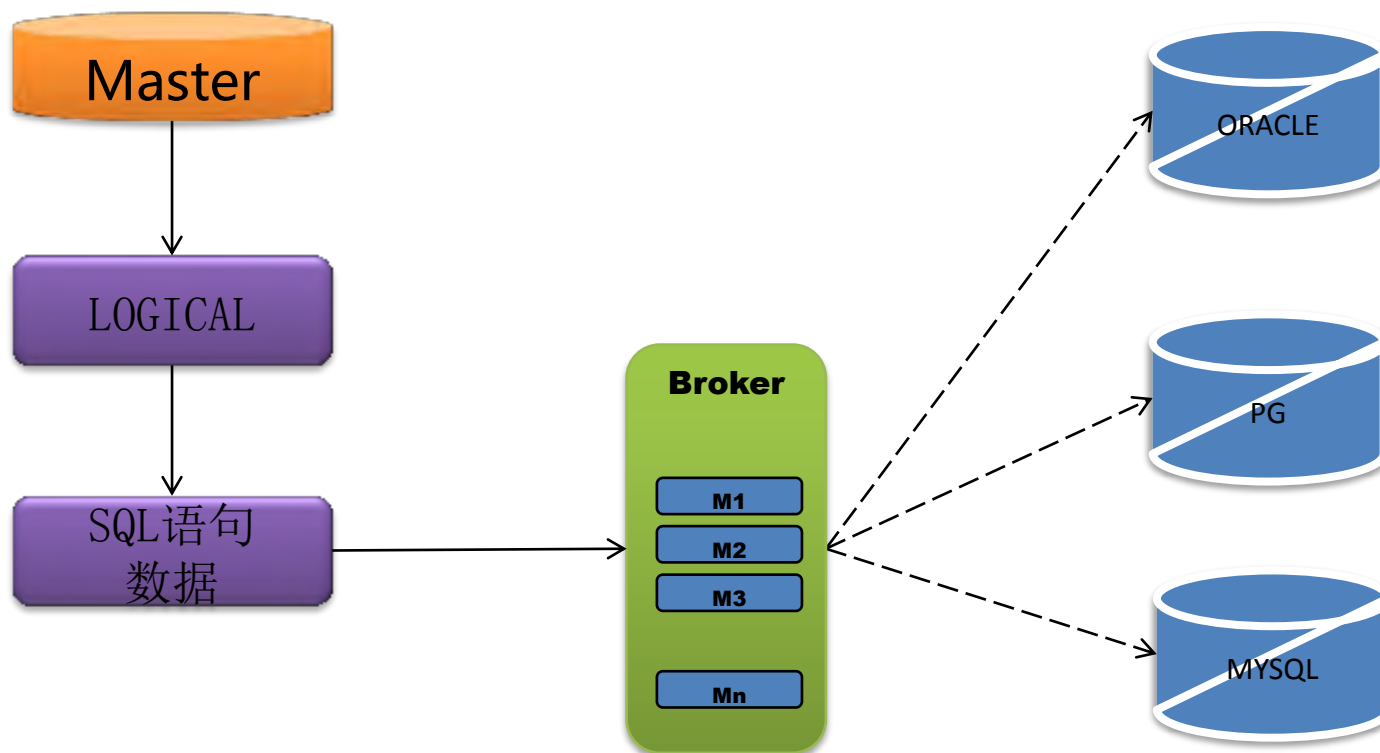


逻辑复制架构图

逻辑解码
产生SQL+数据

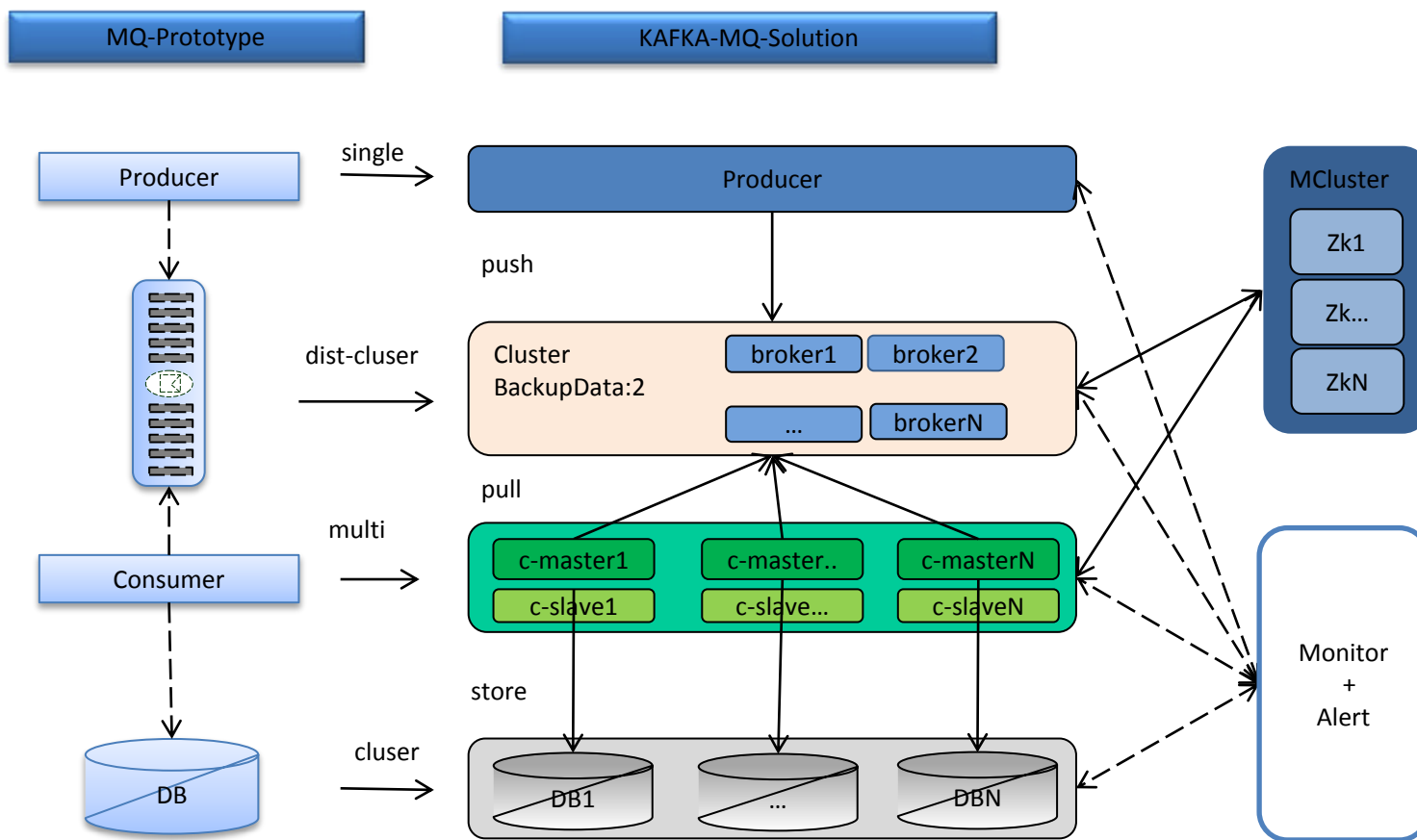
MQ消息队列

DB重放语句



如何保证数据的安全性

Kafka利用冗余、持久化、偏移量和校验、消息反馈



逻辑复制特点

A diagram illustrating the features of logical replication. On the left, there are three concentric blue semi-circles of increasing size. To the right of these circles is a vertical stack of five colored rectangular boxes. The top box is orange and contains the text '它兼顾有基于触发器复制技术的灵活性'. The second box is purple and contains '同时又有基于日志复制技术的高效性'. The third box is green and contains '它使用发布/订阅模型对选择性的数据复制相对物理复制来说是非常方便'. The bottom two boxes are thin, with the fourth being purple and the fifth being orange, and they are empty.

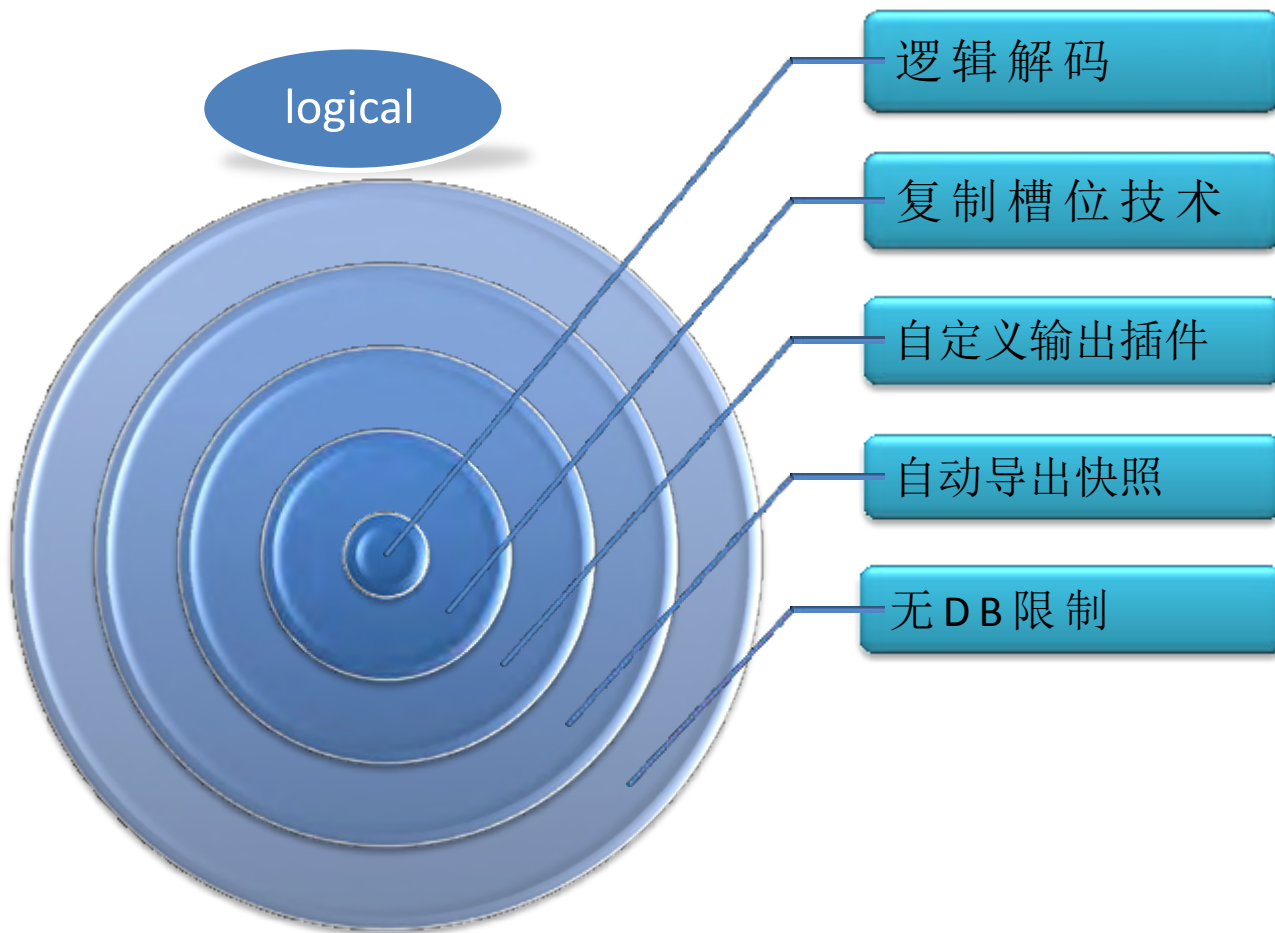
它兼顾有基于触发器复制技术的灵活性

同时又有基于日志复制技术的高效性

它使用发布/订阅模型对选择性的数据复制相对物理复制来说是非常方便



特性



逻辑解码

一个槽表示一个更改流
这些流可以根据需要更改成用户需要的状态

Slot

- WAL
- DISK

LOGICAL

- understandably
- application

- APPLY
- Other

SQL/tuple

持久化WAL解析成便于理解的格式
不必知道内部细节

用户可以根据需要的状态
进行需要的操作



复制槽技术

复制槽技术

非复制槽

是否能保证备库使用WAL段之前不会过早的移除它们



可以保证



不能保证

主库不会过早的清除掉从库正在使用的记录



可以保证



不能保证

是否可以控制日志大小



可能会造成膨胀



不会造成膨胀



自定义输出插件

用户可以根据自己的需要自定义输出插件，根据不同的场景定义不同的输出规则。也可以在相同的场景定义不同的规则



自动导出快照

当使用流复制接口创建一个新的复制槽时，将会自动产生一个快照。

无DB限制

Pg的逻辑复制可以根据需求把WAL日志直接翻译成可以直接解析的SQL语句，并通过槽技术向不同DB进行传递，从而实现不同DB数据库的复制。



如何自定义输出插件

我们所要实现以下几个回调函数：

```
LogicalDecodeStartupCB startup_cb;
```

```
LogicalDecodeBeginCB begin_cb;
```

```
LogicalDecodeChangeCB change_cb;
```

```
LogicalDecodeCommitCB commit_cb;
```

```
LogicalDecodeShutdownCB shutdown_cb;
```



函数意义

复制槽初始化会调用 `startup_cb`

之前活跃的复制槽不再使用，就会调用可选的 `shutdown_cb`

开始动作被解码，就会调用 `begin_cb` 回调。被中止的事务及其内容不会被解码。

已提交事务的提交动作被解码，就会调用必须提供的 `commit_cb` 回调。在此之前，如果有任何被修改的行，将为所有被修改的行调用 `change_cb` 回调。

一个事务中的每一个行修改，都将调用必须提供的 `change_cb` 回调，这种修改可能是一个 `INSERT`、`UPDATE` 或者 `DELETE`。即使原始命令一次修改了多行，该回调也会为其中的每一行调用一次。



如何使用逻辑复制

- 创建:
- `pg_create_logical_replication_slot(slot_name name, plugin name)`
- 使用输出插件`plugin`创建一个名为 `slot_name`的新逻辑（解码）复制槽。
- 删除:
- `pg_replication_origin_drop(node_name text)`
- 删除之前创建的复制源，包含任何相关的回放进程。
- 测试:
- `SELECT * FROM pg_replication_slots;`
- 查看是否创建成功。
- `pg_logical_slot_get_changes`
- 返回槽`slot_name`中的改变，从上一次已经被消费的点开始返回。
- `pg_logical_slot_peek_changes`
- 参数和行为就像`pg_logical_slot_get_changes()`函数，不过改变不会被消费，即在未来的调用中还会返回这些改变。



使用限制

目前不支持DDL的解析, 只有DML的解析
如: create, drop

TEMPORARY表和UNLOGGED表不会被复制

表必须有主键或者唯一约束, 否则像update或者delete这样的操作无法被复制



Thanks!

Q & A