DTCC

2017第八届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2017

PostgreSQL 10.0 新特性介绍

彭煜玮
武汉大学
PostgreSQL中国用户会

个人简介

yuwei.peng@postgres.cn
http://www.pengyuwei.net

- 2008年毕业于武汉大学
- 毕业后留武汉大学计算机学院工作

- 2004年开始接触PostgreSQL
- 《PostgreSQL数据库内核分析》合著者
- 兴趣：
  - PostgreSQL
  - 数字水印、时空数据库

PostgreSQL 10.0

经过多年的发展，PostgreSQL 已经来到了 9.6.X

从下一版本开始，版本号规则变为 major.minor

10.0 将是更改版本号规则后的第一个发行版，社区在这个新版本中增加了丰富的新特性和加强：

- 功能增强
- 性能增强
- 安全性和可靠性增强
- 应用开发
- 迁移的注意事项

1.功能增强

逻辑订阅

PostgreSQL 早在2010年就支持了物理流式复制，可以用来支持容灾、读写分离、HA等业务场景。为什么还需要逻辑订阅的功能呢？

逻辑订阅 vs. 物理流复制

① 物理流复制目前只能做到整个集群的复制，逻辑订阅可以做到表级。

物理流式复制是基于REDO的块级别复制，复制出来的数据库与上游数据库一模一样，每个块都是一样的，就好像克隆的一样。

物理流式复制，目前只能做到整个集群的复制，虽然技术上来讲也可以做到按表空间、按数据库级别的复制，目前PG社区还没有这么做

② 物理流复制的备库为只读，不能写入。逻辑订阅支持读写。

逻辑订阅

③ 物理流复制不需要等待事务提交，即可将REDO日志发往备库，备库也可以直接应用它；目前逻辑订阅需要等待事务提交后，发布端才会使用 wal_sender 进程将解码后的数据发送给订阅端，订阅端流式接收并且应用。

将来逻辑订阅也可能不需要等事务结束就应用，因为在CLOG中已有记录事务状态的标志位。

对于大事务，物理流复制的延迟比逻辑订阅更低。

④ 物理流复制需要复制所有的REDO（包括回滚的）日志；逻辑订阅不需要复制所有的REDO日志，仅仅需要复制“订阅表产生的REDO解析后的数据”（回滚的事务不会被复制）。

逻辑订阅

⑤ 逻辑订阅需要产生额外的REDO信息（主键或者整行）；物理流复制不需要在REDO中记录这些信息。逻辑订阅对主库的性能影响比物理流复制更大。

⑥ 逻辑订阅需要用到发布端的表定义，将REDO翻译为可供订阅者使用的行格式。PostgreSQL允许表定义变更，但是旧版本被保留到订阅端不需要它为止，可能会导致发布端的CATALOG膨胀；而物理流复制不存在这样的问题。

⑦ 物理流复制的备库，如果要被用来只读，为了避免备库上的长查询与清理REDO发生冲突，有两种解决方案：主库延迟清理（导致主库膨胀）；备库应用避让查询（导致备库应用延迟）。逻辑订阅不存在此类冲突。

逻辑订阅

两者适用的场景

- 物理流复制：
  - 适合于单向同步。
  - 适合于任意事务，任意密度写（重度写）的同步。
  - 适合于HA、容灾、读写分离。
  - 适合于备库只读的场景。

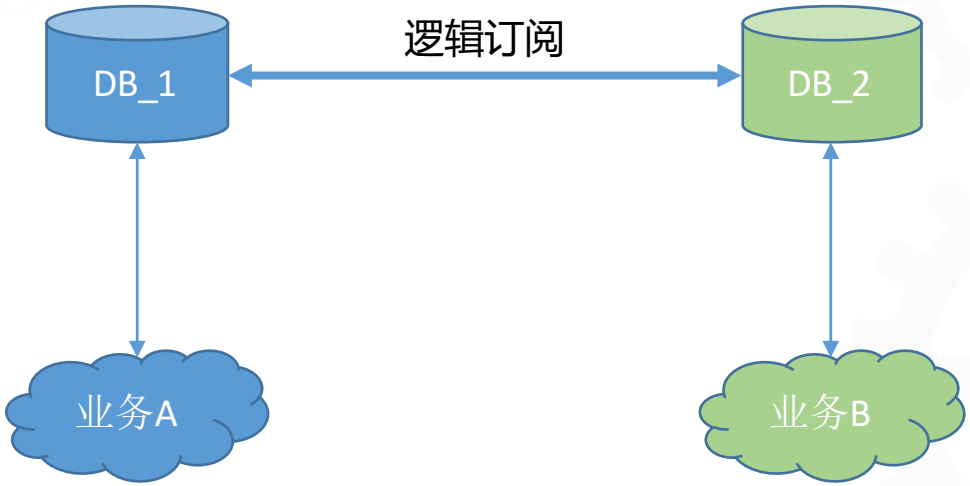
- 逻辑订阅：
  - 适合于发布端与订阅端都有读写的情况。
  - 更适合于小事务，或者低密度写（轻度写）的同步。大事务、高密度写场景中，逻辑订阅的延迟相比物理复制更高。
  - 适合于双向，多向同步。

逻辑订阅

逻辑订阅的应用

① 多个业务之间有少量的数据需要同步

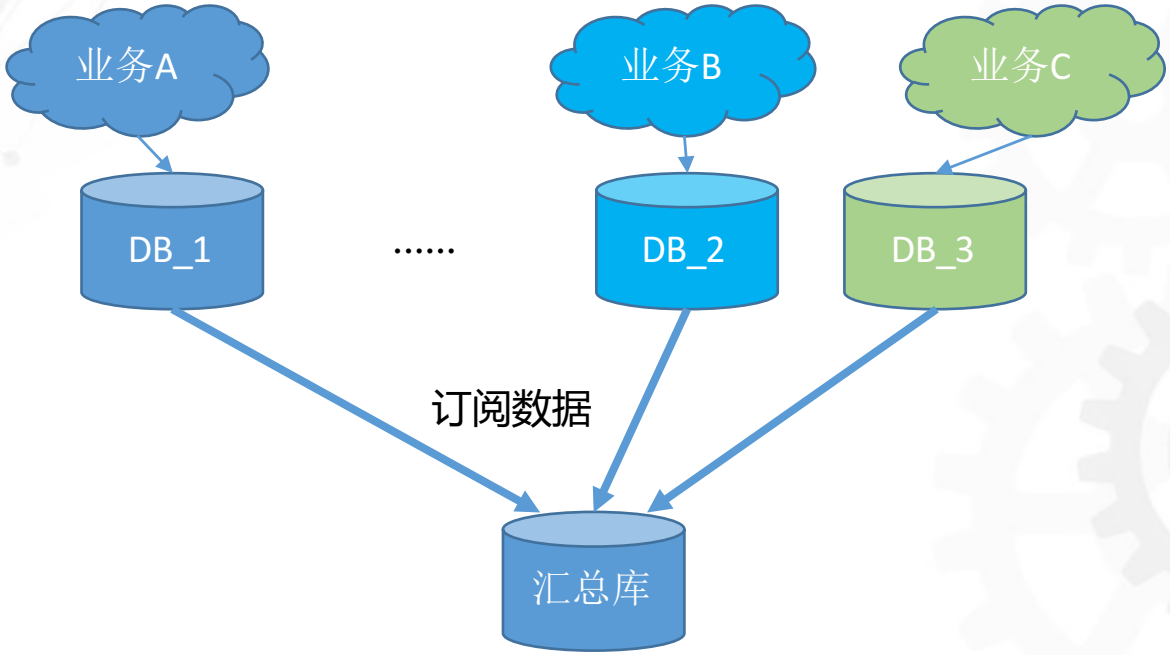
例如：A业务和B业务，分别使用两个数据库，但是它们有少量的数据需要共享且都要对这部分共享数据进行读写。



逻辑订阅

② 数据汇总

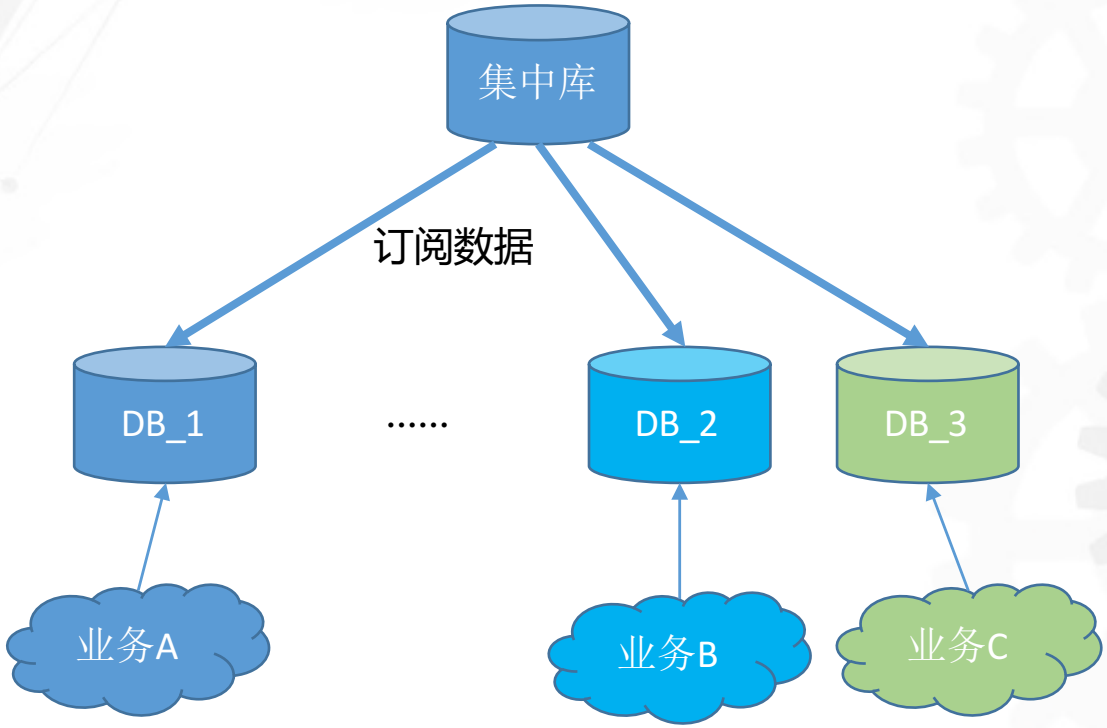
例如：多个业务库的数据要汇总到一个分析库。以往可能要构建庞大的ETL和调度系统，并且很难做到实时的同步。现在有了逻辑订阅，可以方便的应对这样的场景。



逻辑订阅

③ 数据拆分

例如：在垂直拆分时，使用逻辑订阅，可以将一个集中式的数据库拆分成多个数据库。由于逻辑订阅是增量的，可以节约拆分过程的停机时间。



逻辑订阅

④ 跨云线上线下同步

有些场景中有多份数据放置在线上多个云服务上，线下也有多个副本，使用物理流复制很难同步：

- 云厂商的数据库内核可能修改过，物理流复制不一定兼容
- 不同厂商的版本可能不兼容
- 数据块大小可能不一样导致不兼容
- 背后使用的插件可能不一样，导致不兼容
- 云厂商不一定会开放物理流复制的接口

逻辑订阅规避了以上问题：

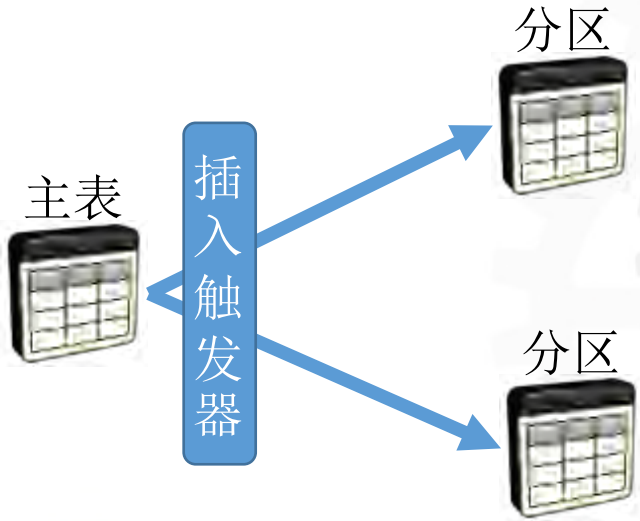
- 逻辑订阅可以跨版本
- 逻辑订阅不受数据块大小影响

内置分区表

PostgreSQL 10.0中将加入内置分区表的功能

没有内置分区表之前是怎么做的？

1.建主表
2.从主表继承若干子表作为分区表
3.在主表上建立触发器，用来把数据路由到各个子表（分区规则实际体现在触发器的主体中）



内置分区表

PostgreSQL 10.0 分区表的设计

- 依旧使用了继承的特性，但不需要手工写规则了。
- 目前支持range、list分区（10.0 发布时应该会支持更多的方法）。
- 父表被称为被分区表且总是为空，它可以没有索引。
- 子女表被称为分区并且包含所有的实际数据，每个分区有一个隐式的分区约束。不允许多继承，并且分区和继承不能混合。
- 被插入到父表中的元组会被自动路由到正确的分区中，因此不需要用来路由元组的 ON INSERT 触发器。目前不支持路由元组到外部表分区，且不支持跨越分区边界的更新。

内置分区表

10.0中分区表的创建只能分成两步执行

首先创建父表：

```
CREATE TABLE table_name ( ... )[ PARTITION BY { RANGE | LIST }
( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [, ... ] ) ]
```

- 主表不会有任何数据，数据会根据分区规则进入对应的分区表
- 如果插入数据时，分区键的值没有匹配的分区，会报错
- 不支持全局的唯一、主键、排除、外键约束，只能在对应的分区建立这些约束

内置分区表

然后创建分区：

CREATE TABLE table_name
PARTITION OF parent_table [({ column_name [column_constraint [...]] | table_constraint } [, ...])] FOR VALUES partition_bound_spec

partition_bound_spec的写法
{ IN (expression [, ...]) | FROM ({ expression | UNBOUNDED } [, ...]) TO ({ expression | UNBOUNDED } [, ...]) }

- 创建分区表需要指定它的主表
- 范围、列表不能有重叠
- 分区表和主表的 列数量，定义必须完全一致（包括OID也必须一致，要么都有，要么都没有）
- 可以为分区表的增加列级和表级约束，如果新增的分区表检查约束名称与主表的约束名一致，则约束内容必须与主表一致

内置分区表

绑定分区、解绑分区：

ALTER TABLE可以将表绑定到分区表的某个分区，也可以将某个已有分区从分区表剔除。

- 支持伪表作为分区，例如外部表、物化视图
- 外键约束也只能逐个分区定义
- 绑定时，需要指定分区键的边界（范围或LIST值）
- 列定义与主表一致（类型、顺序、约束）
- 加入的分区需要进行全表扫描，确认所有的记录都在指定的分区键边界内
- 如果要避免全表扫描，建议在执行绑定前，对这个表加入CHECK约束，确保约束可以保证记录都在边界内。但是这种方法不适用于分区键包含表达式并且分区不允许NULL值的情况。

内置分区表

PostgreSQL 10.0 分区表的特性：

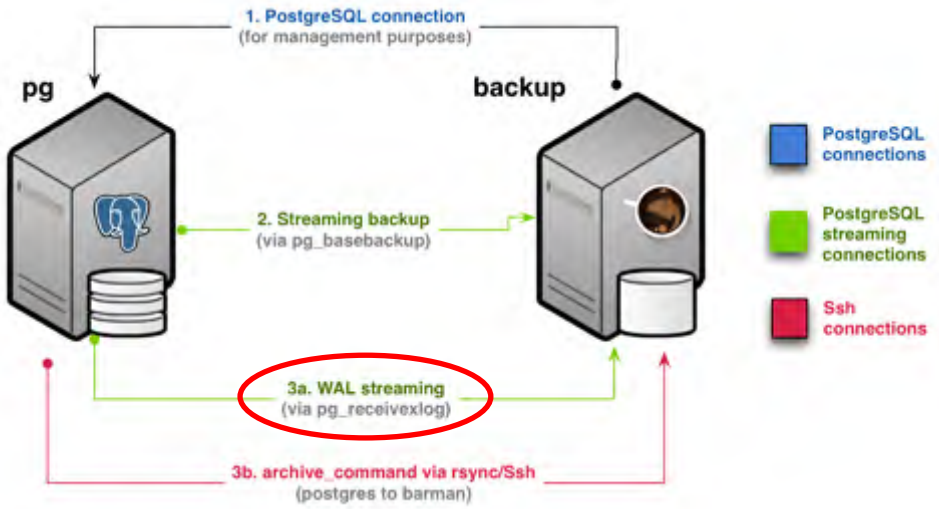
- 不支持全局索引，因此无法实现全局的唯一约束
- 更新数据时不能导致数据跨区移动，否则会报错
- 修改主表的字段名、字段类型时，会自动同时修改所有的分区
- TRUNCATE 主表时，会清除所有继承表分区的记录（如果有多级分区，也会一直级联下去）
- 目前支持分区表的ON CONFLICT .. DO NOTHING ，暂时还不支持ON CONFLICT .. DO UPDATE

流式接收端在线压缩WAL

虽然现在磁盘已经很廉价，大多数时候不需要压缩了。在一些数据变化大的场景中，压缩还是非常有必要的，例如物联网（IoT），每天单库可能新增TB级的增量。

这样的场景，数据库的归档日志也会很大，这个时候就体现压缩的重要性了。压缩势必会增加CPU的开销，我们可以选择是在数据库的服务端压缩，还是在客户端压缩。

PostgreSQL很早就提供了pg_receivexlog命令，可以通过的流式协议实时接收来自数据库的WAL日志。



流式接收端在线压缩WAL

PostgreSQL 10.0 中pg_receivexlog 支持对WAL日志在线压缩

- pg_receivexlog支持通过开关控制是否需要开启压缩、以及选择压缩级别。
- pg_receivexlog启动时，自动扫描存放归档文件的目标目录，选择断点续传的位置，然后向PostgreSQL数据库请求相应位置为起点的REDO。

```
$ pg_receivexlog --compress=1 -D /path/to/logs/ --verbose
pg_receivexlog: starting log streaming at 0/[illegible] (timeline 1)
pg_receivexlog: finished segment at 0/2000000 (timeline 1)
pg_receivexlog: finished segment at 0/3000000 (timeline 1) [...]
```

0表示无压缩，
9表示最高压缩

会产生很多压缩文件

```
$ ls /path/to/logs/ 000000010000000000000001.gz
000000010000000000000002.gz  [...]
000000010000000000000027.gz.partial
```

查看清理进度

PostgreSQL 中依靠Vacuum来清理过时数据，Vacuum的过程可能会很长，以前无法了解Vacuum工作的进度

10.0增加了对Vacuum的可视化监控

- 增加了动态视图pg_stat_progress_vacuum
- 显示每个vacuum worker进程扫描了多少页面、回收了多少页面，结合清理对象的总页面数，可以估计进度

以此为模板，将来还会加入更多的progress动态视图，方便用户了解长任务的执行情况

后台运行

对于一些长时间运行的查询，有可能会因为网络问题或者其他问题导致终端断开，查询执行情况不明

10.0增加了对后台运行的支持，提供了三个SQL函数

- pg_background_launch：开启后台work进程与会话，执行用户提供的SQL，返回后台会话的PID
- pg_background_result：根据提供的PID，返回这个后台会话执行SQL的结果
- pg_background_detach：根据提供的PID，返回这个后台会话执行SQL的结果，同时关闭这个后台进程。

pg_background_result还会返回执行所用的时间

2.性能增强

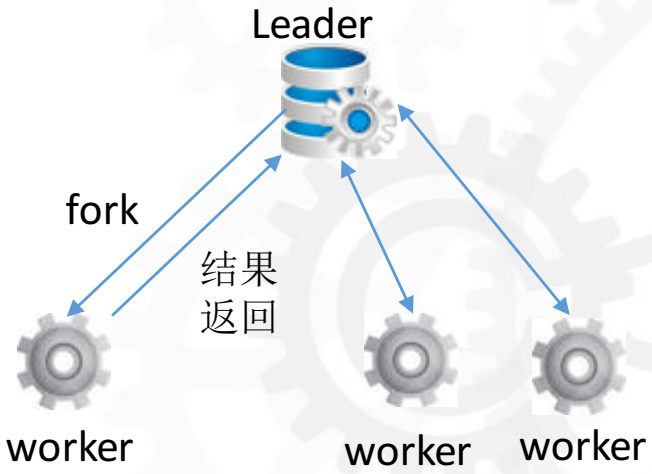
并行增强

控制集群并行度

从 9.6 开始，PostgreSQL就一直在增强其并行执行的功能：

- 有一个Leader进程，其下会创建很多Worker进程来并行执行任务，单个任务的并行度可以控制
- 通过max_worker_processes参数控制整个集群的并行度，同时运行的查询和Worker总数不能超过max_worker_processes

10.0 新增了一个参数max_parallel_workers，用于控制整个集群允许开启的用于多核计算的Worker进程



并行增强

并行排序

在 10.0 中，PostgreSQL增加了一个对元组进行并行排序的模块：

- 原则上，任何现有需要调用tuplesort的功能都可以使用这个并行排序模块
- 基于并行排序模块，将能够支持B-Tree的并行创建（代价模型比较直接）

贡献者对两种情况作了简单的性能对比：

```
-- 非并行创建
CREATE INDEX serial_idx ON parallel_sort_test (randint) WITH (parallel_workers = 0);
Total time: 00:15:42.15

-- 8个工作者并行创建
CREATE INDEX patch_8_idx ON parallel_sort_test (randint) WITH (parallel_workers = 7);
Total time: 00:06:03.86
```

并行增强

Merge Sort

查询执行时，一种常见的情况是计划中的Sort节点下有多个Append节点，常规的执行方法是先执行所有Append合并得到结果集，再进行排序。例如分区表的排序。

在 10.0 中，PostgreSQL增加了Merge Sort，计划中体现为Gather Merge节点：

- 可以对多个Append节点并行地执行排序
- 多个有序集合进行Merge Sort

Merge Sort的原理和Merge Join类似

Query 4: With GM 7901.480 -> Without GM 9064.776
Query 5: With GM 53452.126 -> Without GM 55059.511
Query 9: With GM 52613.132 -> Without GM 98206.793
Query 15: With GM 68051.058 -> Without GM 68918.378
Query 17: With GM 129236.075 -> Without GM 160451.094
Query 20: With GM 259144.232 -> Without GM 306256.322
Query 21: With GM 153483.497 -> Without GM 168169.916

并行增强

其他并行增强

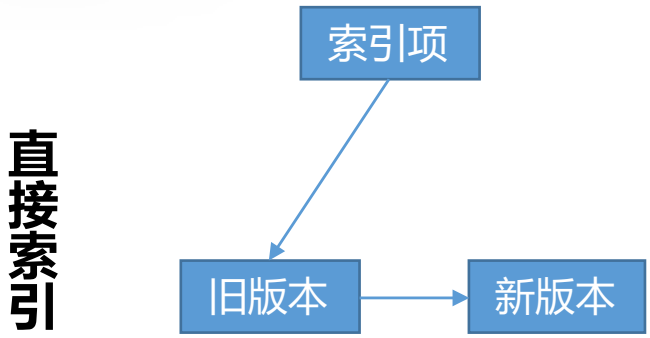
- Bitmap heap scan
- Index scan
- Merge Join
- Hash Join
- 块级并行 Vacuum

间接索引

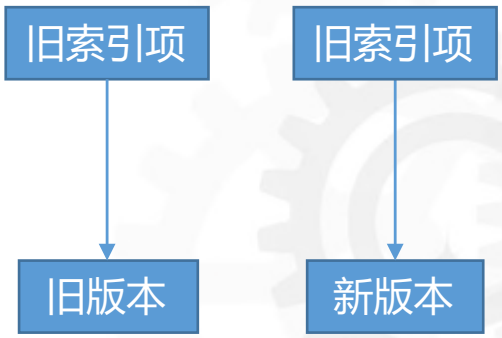
直接索引

目前 PG 中的索引都是直接索引，索引项中有堆元组的TID：

- MVCC导致堆元组更新会产生新版本
- 如果新旧版本在同一个块中，HOT机制保证索引键值不变的索引不变
- 如果新旧版本不在同一个块中，即使索引键值不变也需要更新索引



HOT更新



非HOT更新

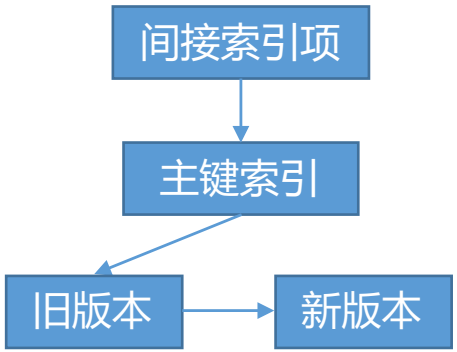
间接索引

间接索引

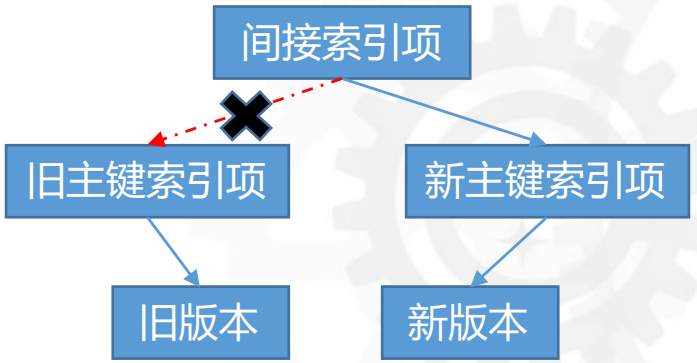
10.0 引入了间接索引的概念，索引项中有堆元组的主键值：

- 只要不更新堆元组的主键，索引键值不变的间接索引都不需要更新
- 但间接索引不能单独服务于查询，必须经过主键索引中转：从间接索引得到主键，再由主键索引得到TID，最后得到堆元组
- 主键只能是小于等于 6 字节的类型： 281,474,976,710,656行

间接索引



HOT更新



非HOT更新

用不完全索引支持复合排序

用索引支持排序是很好的手段：

- ORDER BY a,b,c 可以利用到索引(a,b,c,*)
- ORDER BY a,b,c 无法利用索引(a,b)或者(a)

10.0 将使得第二种情况能够用上(a,b)或者(a)这样不完全包含排序列的索引：

- 第一个阶段，利用不完全的索引来做基于前几个排序列的排序
- 第二个阶段，对前几个排序列上值相等的数据，取出后面的列值进行单独排序

当前几个排序列的值分布得比较散时这种利用索引的方法效果比较好

自动预热共享缓存

数据库的共享缓存用来保存经常被使用的数据块，以减少反复Io提高效率，共享缓存中的数据一般都是热数据。

如果数据库重启或者主备切换，热数据需要重新从磁盘中载入到共享缓存，这期间用户响应会变慢。

10.0 中增加了自动预热共享缓存的技术：

- 建立一个预热器后台进程，它在系统关闭时把缓冲池中的数据块信息转储到文件中
- 重启时，预热器自动把转储出来的数据块重新载入到缓存中

用法：
在shared_preload_libraries参数中加上pg_autoprewarm

JIT支持

JIT是OLAP数据库应具备的能力

- OLAP分析可能涉及很多聚集、条件过滤等，这些操作在数据库内核中是一些对应的函数，可能每一行都会流经这些函数，当分析的数据量很庞大时，需要大量的函数调用、栈的进出交换、内存拷贝等。
- 动态编译，可以减少函数调用，解决以上瓶颈。

PostgreSQL 10.0 已经开始为 JIT 做铺垫

- 把SQL执行的框架从递归调用方式改成了非递归的opcode驱动模式
  - 非递归式降低了栈使用和开销
  - 可以在不同的子表达式之间共享一些状态
  - 简单函数以简单跳转实现而不需函数调用
  - ……

3.安全性和可靠性

安全性增强

SCRAM认证机制

PostgreSQL SCRAM机制基于RFC文档 5802 、7677实现。

- 类似于PG原有的GSS和SSPI认证，在HBA规则中约定客户端所使用的认证机制
- SCRAM机制基于SASL方法，其下可支撑和扩充多种认证技术
- 如果客户端被告知使用SCRAM认证，在认证过程中会通过AuthenticationSASLcontinue、PasswordMessage两个过程进行SASL消息交换

目前只支持SCRAM-SHA-256算法，但是基于SASL认证方法，未来可以支持更多的更强的算法

安全性增强

SCRAM vs. md5

① md5 认证易被攻破
客户端可以直接递交 md5（服务端存储的秘钥+SALT），所以用md5存储的秘钥，也不能泄露。

② SCRAM更难被攻破
服务端存储了多次加密后的秘钥，加密方法不可逆转。仅仅泄露这个多次加密后的秘钥，无法攻破数据库。

SCRAM相比md5，可以避免因为数据库存储的加密秘钥泄露导致客户端可以篡改认证协议连接数据库的危险。

SCRAM和md5不兼容，二者选一。

新增内置角色

PostgreSQL 10.0 开始植入了一些内置的角色：

- pg_backend_pid：可用来取消、中止任何进程，不包含其他超级用户权限
- pg_monitor：可以查看统计信息，便于DBA等检查数据库健康状态
- pg_read_all_gucs：可以查看所有的 GUC 配置

未来PostgreSQL还会对植入更多的内置角色，让数据库的权限分组管理更加便捷。逐步形成像Oracle这样内部有许多角色可选的状况。

防止执行不带条件的更新/删除

不带条件的更新/删除很危险

正常情况下，这样的SQL不应该在业务逻辑中出现。通常出现在SQL注入或者误操作中。

10.0 提供了一个参数来防止此类SQL

```
bool allow_empty_deletes = true;
bool allow_empty_updates = true;
```

- 分别控制是否能执行不带条件的Update或Delete
- 可以设置为全局、会话级、用户级、库级、或者事务级别。

WAL日志支持的Hash索引

很长一段时间，PG中的Hash索引是不做WAL日志的，因此数据库崩溃可能会导致Hash索引不可用。

10.0 中为Hash索引的操作加上了WAL支持，现在对Hash索引的创建、插入、分裂等操作都会记录WAL日志，数据库崩溃后可以用这些信息来恢复索引

4.应用开发

Libpq增强

支持pipeline batch模式

- 客户端可以将多个查询塞入pipeline，作为一个batch提交给服务端，从而减少客户端和服务端的网络交互次数。
- 在网络环境不太好的环境中，特别是云环境，能大幅提升性能。。

增加多连接功能

- 支持连接串中配置多个主机名、端口，libpq自动连接到第一个正常响应的数据库
- 允许配置每一对主机名端口的target_session_attrs，是否支持读写。适合物理流复制，一主多备的读写分离场景

标准/兼容性支持

- 增加类似serial的identify column：虽然已经可以使用serial来达到同样效果，不过实现这一标准，可以兼容更多的数据库。
- 支持SQL:2016的SQL/JSON标准
- 支持ICU
  - ICU是一个成熟的，被广泛使用的跨平台一致性全球化支持库
  - ICU的好处是与UNICODE标准最为贴近，而且可以使用ICU，软件可以做到跨平台保持一致性

5.迁移的注意事项

数据目录更名

10.0中把pg_xlog、pg_clog、pg_log更名为pg_wal、pg_xact、log。

- 理由：之前的名称带有log，容易被误解为操作日志而误删
- 问题：
  - 如果有对应的程序用到了这几个目录名，需要修改

迁移注意

- 使用pg_upgrade升级时，Hash索引需要重建
- XLOG相关的系统管理函数重命名，xlog改为wal
- 不再支持浮点 datetimes/timestamps类型，编译项--disable-integer-datetimes去除
- 不再支持client/server protocol version 1.0
- 不再支持contrib/tsearch2
- 不再支持version-0版本的C语言函数

敬请关注，2017-09-XX，

PostgreSQL 10.0

全球官网：http://www.postgresql.org
中国社区：http://www.postgres.cn

微信公众号：

THANKS

SequeMedia 盛拓传媒 IT168.com ITPUB ChinaUnix