DTCC

数，造，未，来

第十二届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2021

2021 年 10 月 18 日 - 20 日 | 北京国际会议中心

Spark
Hadoop
OLAP
HANA
MPP
Oracle
SQL

IT168.com ChinaUnix.net ITPUB

纲举目张——金融业务PostgreSQL实践与体系规范

赵飞祥

Airwallex 空中云汇

赵飞祥

Airwallex空中云汇Senior DBA

曾就职于太极计算机、北京竞技世界网络技术有限公司、斗鱼等企业。Oracle 10g OCP，11g OCM，Oracle YEP年轻专家。喜爱技术总结和分享、多次行业会议和沙龙演讲嘉宾、IT Pub博客专家。2010年开始从事数据库相关运维、架构、开发工作，涉足postgresql、 mysql、Oracle、greenplum、MongoDB、redis等数据库，目前主要研究PostgreSQL数据库和DevOps方向

个人博客： http://blog.itpub.net/24638123/

目录

一、数据库规范的意义与内容

数据库规范的作用

- 规范是运维和开发经验的总结，一种统一的约定
- 在满足业务需求的前提下，更好的使用数据库，避免不必要的问题
- 对数据库的一致性理解和使用，避免黑盒和未知思维，保障系统高性能稳定运行，故障快速定位

数据库规范的内容

- 知道什么是PostgreSQL数据库，适合做什么?
- PostgreSQL数据库逻辑和物理架构
- PostgreSQL部署与业务访问架构
- PostgreSQL数据库的安装和部署
- Database、user、privileges设计
- Table、column、data type设计
- Index和jsonb设计
- SQL开发规范
- Data生命周期设计

数据库规范的内容

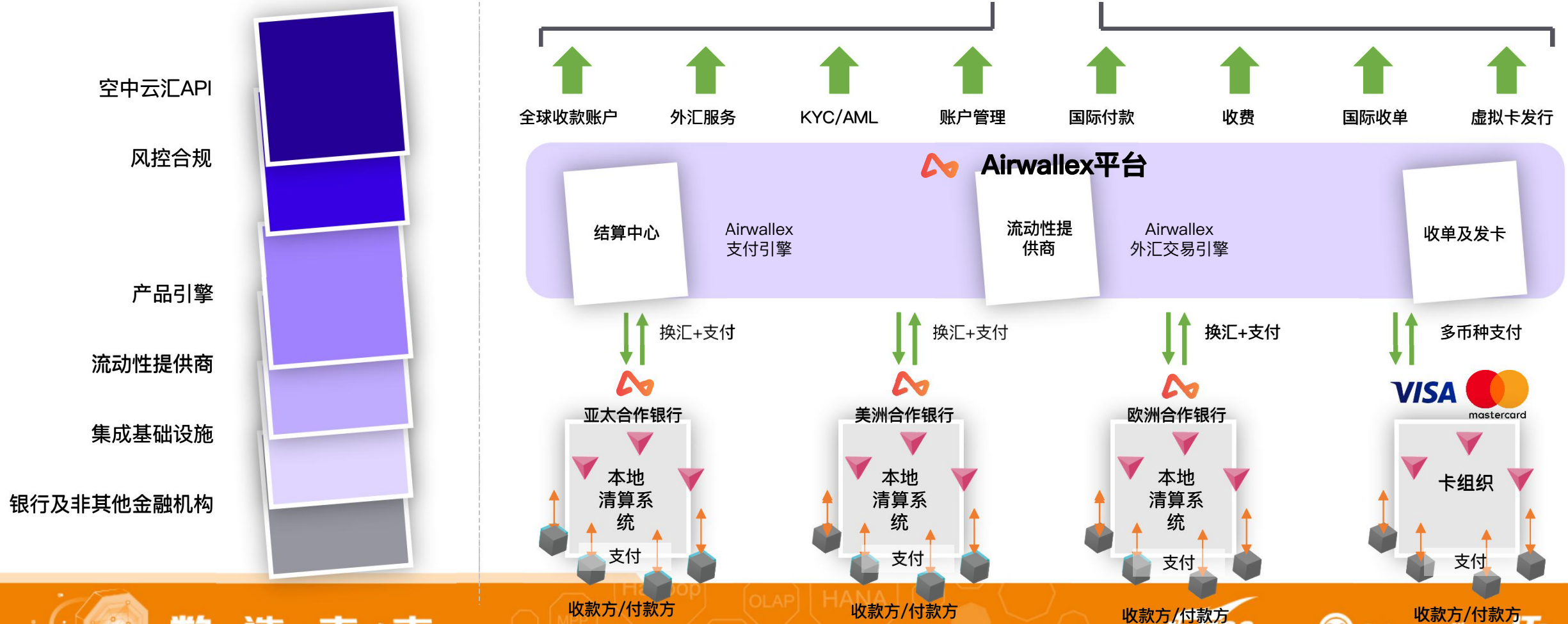
- PG架构规范
  - 部署和访问架构规范
  - PG实例部署规范
- PG开发规范
  - DB对象设计规范
  - SQL开发规范

金融业务特殊规范

- 事务强一致性与持久性
- 多 region 业务系统
- 数据主权与隔离
- jsonb 类型使用
- 全球化数据一致性

技术架构

对接全球银行、卡组织和当地清算系统，打造全球支付的金融基础设施



全球布局

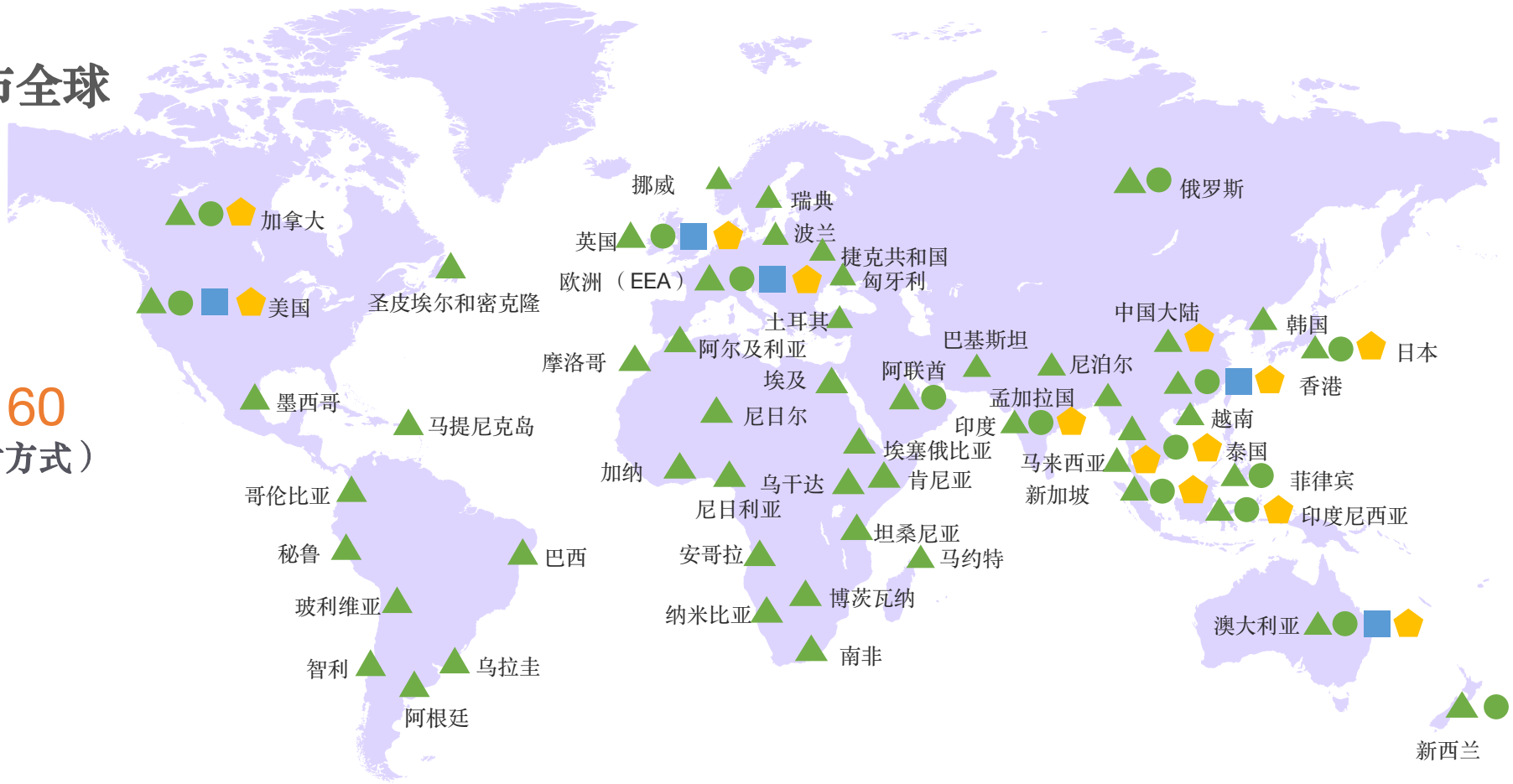
空中云汇的业务能力遍布全球

92 个国家：本地支付服务

54 个国家：本地收款服务

30 多个国家：发卡服务

40 多个国家：收单服务，支持160多种支付方式（卡组织和本地支付方式）



现有的本地支付

现有的本地收款

现有的发卡服务

现有的收单服务

注意：本地欧元支付与收款支持单一欧元支付区的40个国家与地区

二、PostgreSQL架构规范

架构类型演进

1. 单 region 单实例架构
2. 单 region 高可用架构
3. 单 region 高可用 + 缓存架构
4. 单 region 高可用 + 缓存 + OLAP架构
5. 单 region 高可用 + 缓存 + OLAP + 搜索系统架构
6. 多 region 一写多读高可用架构
7. 多 region 多写多读高可用架构
8. 多 region 分布式高可用架构

架构规范

1. 所有DB实例都必须是高可用架构，禁止生产系统单实例
2. 默认DB实例为“单region高可用架构”类型，满足单region业务和多region读写但每个region均独立业务
3. 一写多读业务，适用于全球公共数据与服务，建立“一region写实例+多region读实例”集群
4. 多写多读且多region数据关联业务，建立“多region读写”集群
5. PG实例架构只有在“单写单读”、“一写多读”、“多写多读”三者选择一种
6. 在PG之外的缓存、OLAP、搜索系统可根据业务需求搭配使用

PG实例部署规范

- 所有PG实例必须为统一版本（目前统一为PG12）
- 所有PG实例必须为加密实例
- 所有PG实例的字符集均统一为 UTF8
- 所有PG实例中与时间相关的数据时区统一为 UTC
- 所有PG实例的db name需要与 service name保持一致且全局唯一
- 所有PG实例的db user需要与权限、调用位置一致
- 所有PG实例的访问均需要设置对应client白名单
- 所有PG实例均需要定时备份并验证备份有效性
- 所有生产PG实例均需要开启SQL审计

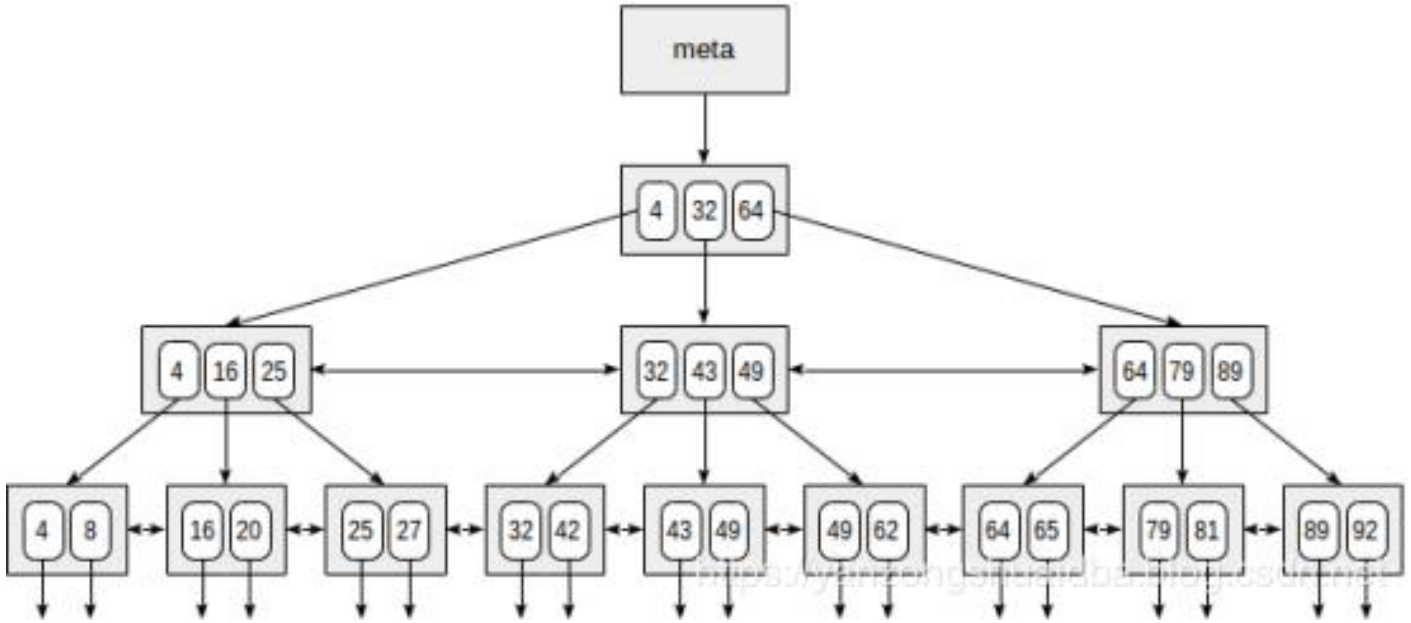
PG实例user规范

- PG中所有的user均有明确的命名规范、权限与使用场景
- 从user级别上分为实例级user和DB级user

- 实例级user
- DBA管理账号
- DBA只读账号
- Replication同步账号

- DB级user
- DB DDL账号
- DB DML账号
- DB Readonly 账号

三、PostgreSQL开发规范



PG对象名称规范

- DB Name与service name保持一致，是所有基础设施的名称
- DB name与table name需要注意长度限制，不能超过63位
- DB 内的对象名只能使用小写字母、数字、下划线，不能使用其他字符
- query中的别名只能使用小写字母、数字、下划线，不能使用其他字符
- 主键索引应以 pk_开头，唯一索引应以 uniq_开头，普通索引以 idx_ 开头
- 不同业务的db以database进行区分，默认都使用 public schema，以方便开发人员与其他数据库使用习惯兼容
- 建议所以可以添加comment的地方均添加comment，且以英文描述

PG对象设计规范

- PG中最常用的是数字、字符、时间类型。设计时应尽可能选择合适的数据类型，能用数字，就不用字符串。能用限定长度的varchar类型，就不用大对象类型。使用好的数据类型，可以使用数据库的索引，操作符和函数，可以提高数据的查询效率。
- 在考虑表结构时，需要考虑创建对应的索引，避免全表扫描。
- 多个table中相同的列，或者进行join的列，需要保证列名一致，数据类型一致。
- Btree索引的字段不建议超过2000字符，如果超过，建议使用函数索引或分词索引。
- 表结构中定义的数据类型，必须与应用程序中的定义一致，表之间的校对规则一致，避免报错，或无法使用索引的情况发生。
- 考虑全球化需求，所有字符存储和表示，均以UTF-8编码。所有数据内与时间相关的数据，时区均为UTC时间，最好使用int或bigint存储秒或毫秒。业务程序可以根据需求，进行前端显示的时区转换。

PG对象设计规范

- PG中应尽量避免触发器的使用，这会使数据处理和数据库迁移逻辑复杂，不便于调试。
- 有定时海量数据需要归档和删除的表，应考虑表按时间列分区，归档后清理时，不要使用delete，而是用drop或truncate清理对应表。
- 未使用的大对象，一定要定时删除部分数据，否则大对象就会一直在数据库中，与内存泄露类似。
- 对于大型文本类数据进行查询时，要尽量避免 like %xxx% 的模糊匹配。如果实在有类似需求，可以考虑在PG中建立gin索引，所以使用专门的ES等搜索系统。
- 对于频繁使用的大表（大小超过10GB，或者记录数超过1000万）应考虑进行分区，保证单表比较小，可以提升查询效率、更新的效率、创建索引的效率、备份恢复的效率等。

PG jsonb对象设计规范

- PG在设计表结构时，建议尽量规划好，避免后续经常添加新列，或者修改数据类型。某些操作可能会触发表的重写，例如添加新列并有默认值，修改字段的类型。
- 如果用户不好规划结构，可以考虑使用jsonb类型存储用户数据。
- Jsonb类型的存储内容需要合理规划，对于经常需要查询的value，也可以创建单个字段或多个字段的btree索引，提高查询性能。
- 使用jsonb修改数据时，建议使用jsonb_set函数，同时区分好如果修改对象上一级是否存在，修改对象不存在时，是否需要新增。
- 对于jsonb中的对象，不要存放过多数据。如果数据过多，会影响查询和更改性能。可以考虑规划存放不同数据在多个列，或多个表中。
- 如果jsonb中查询对象不是一个value，而是一个集合，则需要将集合根据范式设计，拆分到新表中，避免查询时扫描所有的jsonb数据。
- 对于jsonb中一些需要进行检索的内容，可以考虑创建gin索引。

PG query规范

- PG在查询数据时，要在select后写明需要查询的所有列名，不要返回不使用的任何字段，不要使用 select *，这样会查询过多内容，也可能出现程序匹配错误。
- PG在查询时，一定要考虑查询返回数据量，避免一次SQL返回过多数据，影响查询性能和网络。
- 在统计数量时，应使用 count(*)，而不用count(col_name)，或者 count(1)。
- 在count多列列名是，必须使用括号 count(col1,col2,col3)。
- 在count(distinct col)中，只计算非NULL列的不重复结果，NULL列不会被计算。
- NULL是UNKOWN的意思，也就是不知道是什么，因为NULL与任意值逻辑判断都返回NULL。
- 除非是ETL程序，否则应该尽量避免向客户端返回大量的数据。若数据量过大，应考虑需求是否合理。

PG 数据操作规范

- PG数据订正时，删除和修改数据时，要先select，避免误删除，要确认无误后才能提交执行。
- 大批量删除和更新数据时，不要再一个事物中完成，建议分批次操作，避免一次产生较多垃圾和日志，对系统资源和相关系统产生不好的影响。
- 大批量数据入库，可以使用 copy语法，或者insert into table value(),(),(); 的方式，提高写入速度。
- DDL操作与其他可能获取大锁的操作（如vacuum full，create index）可以设置锁等待，防止堵塞与DDL锁相关的所有query。
- 可以使用 explain 查询SQL的执行计划，使用 explain analyze 就会实际执行SQL并显示对应的执行计划。
- 创建index时，为了并行创建，不阻塞其他DML，可以添加 create index concurrently 关键字。
- 如果PG实例配置了standby，并且使用了slot，则必须监控stadnby实例的延时和slot的状态，否则可能会造成主库XLOG不断堆积，占满空间而产生问题。

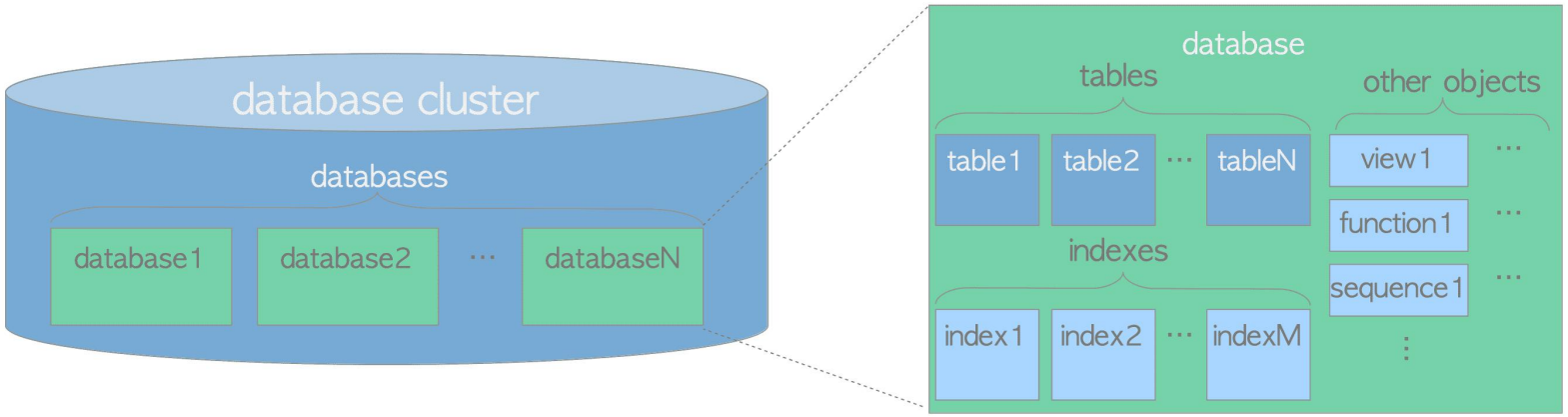
PG 稳定性规范

- PG中应尽量避免长事务，长事务会造成垃圾膨胀。
- PG在代码中写分页逻辑时，如果count为0应直接返回，避免执行后续的分页语句。
- 两阶段提交的事务，要及时提交或回滚，否则可能导致数据库膨胀。
- 在高并发场景下，务必使用程序的连接池，否则性能会很低下。如果程序没有连接池，可以考虑使用pgpool-II或pgbouncer中间件。
- 程序务必要有重连机制，如果没有重连机制，一个长期空闲的链接可能会被强制断开，数据库高可用切换后，程序也可能有问题。
- 必须使用合理的隔离级别，不要越级使用隔离级别，以满足业务需求为准。
- 高峰期对大表添加新列时，建议先不加默认值，避免rewrite，后面再用业务逻辑添加默认值。
- 自增字段建议使用序列，根据情况选择2字节、4字节或8字节。禁止使用触发器产生序列。
- 线上表结构的变更，包括添加字段、索引等，应尽量在业务低峰期执行。
- OLTP系统在业务高峰期或高并发期间，应拒绝长SQL、大事务、大批量。
- 冷热数据要进行分离，尽量保证线上实例只存在有限的经常查询的数据

PG 索引优化规范

- 访问PG的SQL应进行查询优化，尽量避免全表扫描，首先要考虑在 where, order by,group by 的列上，建立索引。查询特别多的SQL要考虑满足覆盖索引。
- 应尽量避免在 where子句中使用 != 或 <> 操作符，这种不等于会让PG放弃索引，使用全表扫描。
- 索引应该建在选择性高的字段上，应该建立在小字段上，选择性低的字段，或者大的文本字段甚至超长字段，一般不要建索引。
- 复合索引要符合最左原则，将选择性最好的字段作为第一个列，其他非第一个字段的列，如果经常会有查询用到，也需要创建单独的索引。
- 频繁进行数据操作的表，不要建立太多的索引，因为索引会降低数据操作的性能，增大数据操作的成本。
- 对于btree索引、hash索引、gin索引、gist索引、BRIN索引要根据不同的索引特点和适用场景进行合理选择和使用。
- 对于无用的索引要及时删除，无用的索引不仅会导致更新数据的代价变大，还可能产生错误的执行计划。
- 所有的新程序的表结构和SQL在上线前最好与DBA确认是否都有索引再上线到生产环境。

四、规范落地与实践经验


database cluster
databases
database1
database2
...
databaseN
database
tables
table1
table2
...
tableN
other objects
view1
...
function1
...
sequence1
...
indexes
index1
index2
...
indexM

规范落地步骤

- DBA根据业务实践和经验整理出规范
- DBA与开发人员沟通规范，达成一致
- 按照达成一致的规范来进行数据库部署、初始化和上线
- 对于与实际业务有出入的地方调整规范
- 对于新出现的业务场景加入规范
- 对完善后的规范作为数据库使用标准，保证DB的稳定和高性能

规范常见问题

- 规范不是一方要求另一方，而是为了共同的系统稳定、高效运行目标而需要遵循的标准

- 对于架构规范不清楚，服务数据读写错误
- 不同的系统、不同的region，时间数据的格式和时区不一致
- 新业务缺乏合理的表结构和索引设计，上线产生性能问题
- 不合理的大批量操作，影响线上DB性能
- 线上缺少历史数据归档和冷热分离，表数据量过大

规范落地经验

- 有了规范可以让开发人员更好的了解数据、使用数据库
- 规范可以有效解决DBA的重复解释工作，由于没有共识造成开发与运维之间的问题
- 规范并不是单方面的制约，而是开发和运维共同参与和完善的、为了达到共同目的的重要合作方式
- 规范中具有一些场景的最佳实践，可以为大家在有多种选择处理同一个问题时，提供一个标准
- 规范有通用的规范和具体业务场景的规范，适合的才是最好的

Q & A


THANKS
时序
开源
自研
分布式
文档
Spark
HANA
SQL
OLAP
Oracle
Hadoop
MPP
DB2