基于PostgreSQL构建灵活的表单系统

刘立兼　上海云贝网络科技有限公司

常见表单系统的数据结构

Excel

XML

伙伴云

MySQL

麦客CRM

MySQL

金数据

MongoDB

简道云

MongoDB

班牛

PostgreSQL

班牛数据现状

- 企业用户
  1. 用户数量：5W
  2. 数据量：20亿
  3. 平均每日持续使用：11.5小时

- 数据能力
  1. 表单组件在电商领域同类产品中最丰富
  2. 所有TP类操作的平均响应时间：< 50ms
  3. 所有AP类操作的平均响应时间：< 2s

表单系统对数据能力的要求

- ***丰富的组件：***

  需要不断跟据业务需求开发新组件，对扩展性要求极高

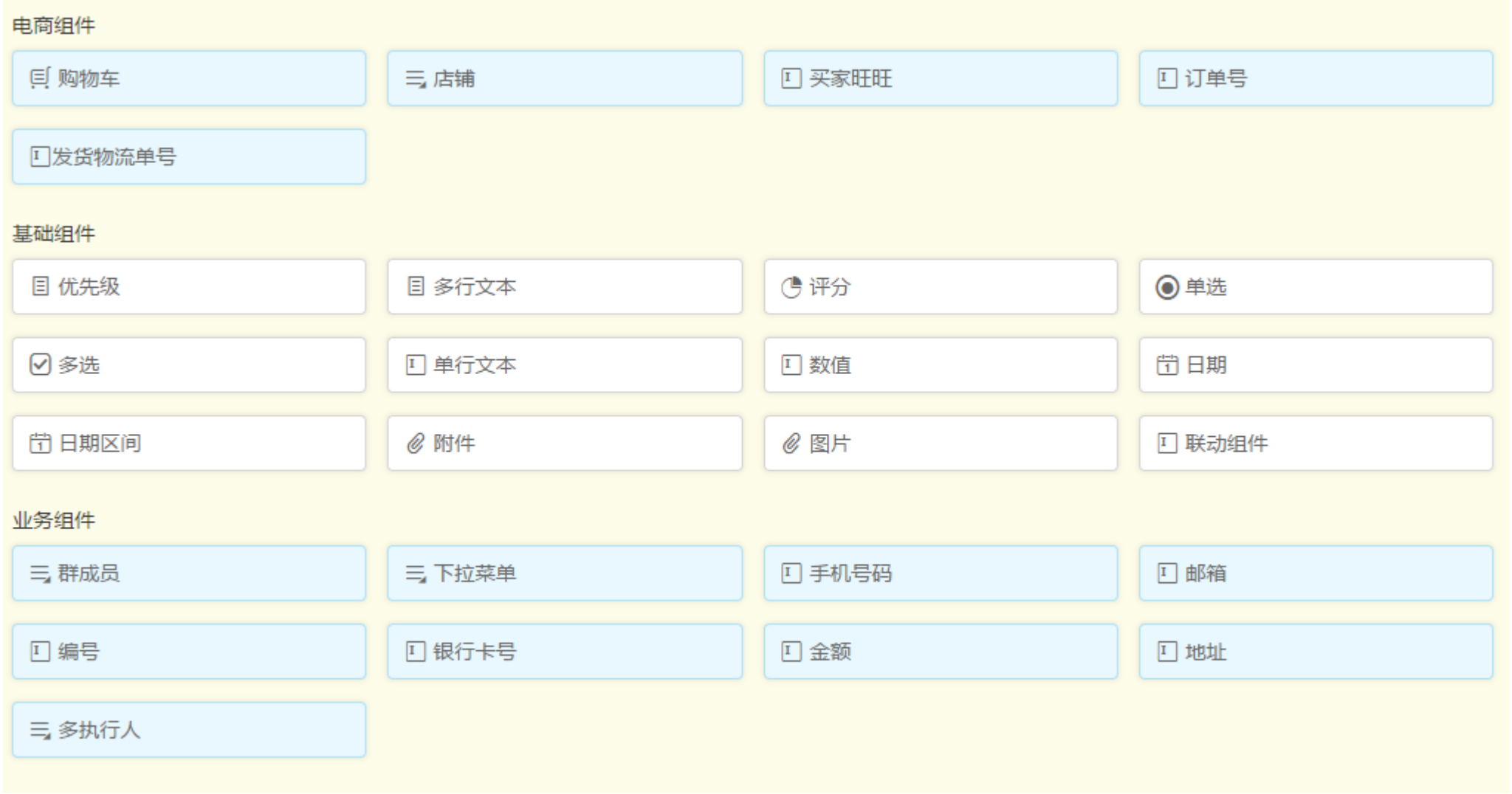
- ***高效的OLTP性能：***

  日常的高频操作对于一般的增删改查性能有很高的要求

- ***一定的OLAP能力：***

  表单中累计的登记信息具有业务指导性，所以AP的能力也不能少

丰富的组件


电商组件
购物车
店铺
买家旺旺
订单号
发货物流单号
基础组件
优先级
多行文本
评分
单选
多选
单行文本
数值
日期
日期区间
附件
图片
联动组件
业务组件
群成员
下拉菜单
手机号码
邮箱
编号
银行卡号
金额
地址
多执行人

组件即模型

- ***基础类型：与数据库类型差不多***

  单行文本，数值，日期，金额，编号，优先级……

- ***复合类型：由多种基础类型复合***

  订单号，物流单号，买家旺旺……

- ***有限选项型：从有限集合中选择一或多***

  单选，多选，下拉菜单，联动组件，购物车……

- ***无限递增型：在一条记录中可以无限增加***

  附件，图片……

高效的OLTP性能

- ***目前我们客户中最大的一张表单如下：***
  1. 25列（组件）
  2. 5千万行记录
  3. 数据表大小达到80GB

- ***我们系统提供如下功能：***
  1. 所有列（组件）不仅可以单独查询，还可以自由组合查询
  2. 复杂列（组件）不仅可以整体查询，还可以按具体子维度查询
  3. 大部分列（组件）可以支持排序查询

单独查询

- *基础数值类型的=<>><查询*

  使用not null btree索引

- *基础字符类型的模糊查询*

  使用not null trgm_gin索引，或使用结巴分词 + 全文索引

- *集合元素的包含查询*

  使用intarray类型的not null gin索引，或仅仅使用btree的=操作符

- *有限集合元素的数值比较查询*

  行列转换 + not null btree索引（DDL依赖），或使用jsonb类型 + 函数索引（部分场景）

- *无限集合元素的数值比较查询*

  外置索引，pg内部无解

组合查询（带limit）

- *所有查询列的选择性均很高*

执行计划会挑选择性最高的几列（大致1~3）分别bitmap index scan后做bitmap and，去获得一个很小的中间结果集，然后在bitmap heap scan的时候去filter其他条件。很小的中间结果集保证了filter其他条件的行数不会很多，所以整体性能是有保障的。

- *所有查询列的选择性均很低*

由于选择性均很低，很容易扫描到符合条件记录，所以列执行计划会倾向于执行顺序扫描。顺序扫描对磁盘非常有利，性能是有保障的。

- *查询列的选择性有高有低*

如果有多列有较高的选择性，执行计划会对他们bitmap index scan后做bitmap and，来得到一个更小的中间结果集，然后在bitmap heap scan的时候去filter选择性低的列的条件。由于选择性低的列很容易filter到，所以整体性能是不错的。

如果有一列选择性特别高，执行计划就会直接对他进行index scan，然后在tuple中filter其他条件。这个策略一般性能更高。

子维度查询

- *方法1：*

通过行列转换将复杂组件的每一个子维度都存入列中，就可以实现子维度的查询。但是这种方法存在对DDL的依赖，存在DDL可能产生的性能风险。

比如：

在行数很多的表中插入一个带默认值的列，造成长时间锁表

插入过多带默认值的列，造成表的尺寸快速增大

- *方法2：*

利用json类型 + 函数索引，可以实现子维度的单独查询。但是这种方法会因为json的统计信息不准，从而造成执行计划的潜在风险。

排序查询

- ***排序列与其他查询列的选择性均很高***

执行计划就会直接对排序列进行index scan，然后在tuple中filter其他条件。由于利用了索引的天然有序性，所以无需再排序。虽然其他条件的选择性很高会造成filter的行数比较多，但是由于中间结果集很小，所以总体上filter的行数也不会多。整体性能是有保证的。

- ***排序列与其他查询列的选择性均很低***

如果带limit，那么执行计划会对排序列进行index scan，然后在tuple中filter其他条件。limit可以使得index scan无需扫描所有满足条件的记录，外加其他列的低选择性使得filter的命中率很高，所以整体性能是不错的。

如果不带limit，由于选择性均很低，所以列执行计划会倾向于执行顺序扫描。获得所有满足条件的记录后进行sort。结果集大的情况下无论是顺序扫描还是在磁盘上sort性能均不会很好。如果结果集不大，那性能还行。在我们的系统中不存在不带limit（或游标）的查询，所以这种情况不存在我们的系统中。

- ***排序列的选择性很高，其他查询列的选择性很低***

执行计划就会直接对排序列进行index scan，然后在tuple中filter其他条件。由于利用了索引的天然有序性，所以无需再排序。并且高选择性意味着中间结果集很小，所以filter的行数也不会多。整体性能是很高的。

- ***排序列的选择性很低，其他查询列的选择性很高***

执行计划会对选择性高的列做bitmap index scan，然后做bitmap and，来得到一个更小的中间结果集，然后在bitmap heap scan的时候去filter选择性低的排序列，最后再通过top-N sort（带limit）或 quick sort（结果集不大)在内存中进行排序。由于中间结果集的大小有限，无论是filter还是sort都不会消耗太多时间与资源，所以整体性能是不错的。

槽点

- *btree_gin索引不支持范围操作符*

  阻碍了想要利用gin的posting list/tree特性来缩小btree的体积的愿望

- *ltree类型的统计信息是hardcoding*

  ltree类型理论上完全可以提供类似数组类型的统计信息，但是却没有做

- *json类型的统计信息也是hardcoding*

  虽然json理论上没办法自动统计，但至少应该提供指定key的统计设置

- *pg_trgm插件没有很好的处理热词问题*

  应该提供自动选择低频分词+heap scan过滤高频词的策略

- *pg_hint插件没有提供指定预测行数的方法*

  目前只提供join场景下的指定预测行数方法，没有提供一般场景下的

- *bitmap index scan无法支持limit*

  只有在bitmap heap scan的时候才能支持limit，这对于低选择性的gin索引列很不利

- *join场景下CBO有时会失效*

  在有些join场景下，明明index scan的成本更低，执行计划却会选择seq scan，只能手动控制

一定的OLAP能力

- ***背景：***
  1. 大部分客户的表单大小都在2GB以下
  2. 不存在特别复杂的计算类统计功能
  3. 没有跨表单的关联统计功能

- ***解法：***
  1. Parallel scan可以提高QD，充分发挥磁盘性能
  2. 对于任何有限集合信息，使用id代替而不是内容，减少表体积
  3. 物化视图可以从原表中抽取统计相关列，减少扫描数据量
  4. 多写数据，一份用于查询，一份用于统计

槽点

- 目前9.6版本仅支持顺序扫描的并行，这对于写入数据有更高的要求。
- 目前的并行顺序扫描的worker数预计非常不准，还不如基于表的大小手动指定worker数
- 未来10版本中，虽然Bitmap heap scan支持并行，但Bitmap index scan却不支持并行
- 未来10版本中，虽然fdw可以支持下推，但不支持分区表的下推

感谢

PostgreSQL帮我们实现了

- 不亚于Nosql的灵活性和扩展性
- 极其强大的组合查询能力
- 压榨磁盘般的并发统计能力
- 高效的关联查询能力
- 齐全的外部数据源连接与导入能力

……

以及与各位齐聚一堂

最后

祝贺PostgreSQL分会的成立，大象一定会越来越好。