MySQL与PG 对比学习

面向开发者

阿里云

digoal

目录

数据类型

	MySQL	PG
数值	https://dev.mysql.com/doc/refman/8.0/en/numeric-types.html	https://www.postgresql.org/docs/12/datatype-numeric.html
整型	tinyint , smallint , mediumint , int , bigint signed , unsigned zerofill	int2 , int4 , int8 domain: int1,uint1,uint2,int3,uint3,uint4,uint8 func: lpad https://github.com/digoal/blog/blob/master/202001/20200105_04.md
numeric	decimal , numeric (m,n) 最大 65 digits	decimal , numeric (m,n) 最大 [左131072 , 右16383] digits
浮点	4byte(prec: 0~23) real , float 8byte(prec: 24~53) double	real , float , float4 float8 , double precision float(0~53)
BIT	bit(1~64)	bit(1~83886080) func https://github.com/digoal/blog/blob/master/202001/20200105_03.md
序列	auto_increment 1,1	serial2 , serial4 , serial8 序列， 可修改start, increment by, maxvalue, cache, cycle

	MySQL	PG
时间	https://dev.mysql.com/doc/refman/8.0/en/date-and-time-types.html	https://www.postgresql.org/docs/12/datatype-datetime.html
date	date	date
time	time	time , timetz
datetime	datetime	timestamp , timestamptz
timestamp	timestamp	timestamp , timestamptz
year	year 1901~2155	- domain: int2 check (1901~2155) https://github.com/digoal/blog/blob/master/202001/20200106_01.md

	MySQL	PG
字符串	https://dev.mysql.com/doc/refman/8.0/en/string-types.html	https://www.postgresql.org/docs/12/datatype-character.html
char , varchar	char(n) 最大255个字符 varchar(n) 最大n字符 (小于64K字节)	char(n) 最大10485760字符 varchar(n) (小于1G 字节) name 固定64字节 "char" 固定1字节
binary , varbinary	binary(n) 最大255个字符 varbinary(n) 最大n字符 (小于64K字节) 使用字节流存储字符串	bytea 最大1GB
blob , text	tinyblob , tinytext < 2^8字节 blob , text < 2^16字节 mediumblob , mediumtext < 2^24字节 longblob , longtext < 2^32字节	bytea 最大1GB text 最大1GB
enum	enum 最大 64K 个值	https://www.postgresql.org/docs/12/datatype-enum.html 最大1GB
set	set 最大 64 个值	intarray uniq(sort(arr)) 最大1GB

	MySQL	PG
空间	https://dev.mysql.com/doc/refman/8.0/en/spatial-types.html	http://postgis.org/
geometry		geometry
geography	-	geography
pointcloud	-	pgpointcloud
raster	-	raster
top	-	top

	MySQL	PG
JSON	https://dev.mysql.com/doc/refman/8.0/en/json.html	https://www.postgresql.org/docs/12/datatype-json.html
JSON	json	json jsonb
JSON path	JSON path	JSON path

PG 额外类型

varbit
货币
interval
平面几何
网络
全文检索
uuid
xml
数组
复合
范围
域

树类型
多维类型
高维向量类型
图像特征值
化学分子
DNA
roaringbitmap
PostGIS(轨迹、234D、栅格、路径规划、TOP)
。。。

	MySQL	PG
函数和操作符	https://dev.mysql.com/doc/refman/8.0/en/func-op-summary-ref.html	https://www.postgresql.org/docs/12/functions.html
空间		http://postgis.net/docs/manual-3.0/

分区	https://dev.mysql.com/doc/refman/8.0/en/partitioning.html	https://www.postgresql.org/docs/12/ddl-partitioning.html https://www.postgresql.org/docs/12/sql-createtable.html
hash	PARTITION BY hash	PARTITION BY hash partition of attach
range	PARTITION BY RANGE	PARTITION BY RANGE partition of attach
list	PARTITION BY list	PARTITION BY list partition of attach
默认分区	无	PARTITION OF ***parent_table*** { FOR VALUES ***partition_bound_spec*** | DEFAULT }
分级分区	2级	不限级数、不限组合
表达式分区	-	immutable表达式
PK|UK	pk,uk必须包含分区键，可以包含其他键	pk,uk必须包含分区键，可以包含其他键

语法	https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html	https://www.postgresql.org/docs/12/sql-syntax.html https://www.postgresql.org/docs/12/sql-commands.html
offset limit		
类型转换	CAST(expression AS TYPE);	CAST (expression AS target_type); value::newtype
事务隔离级别		
2PC		
锁		
upsert , replace		
load data		
隐式commit	https://dev.mysql.com/doc/refman/8.0/en/implicit-commit.html	DDL支持事务
use	use dbname	\c newdb 新建连接
desc	desc tablename	\d[?] object name

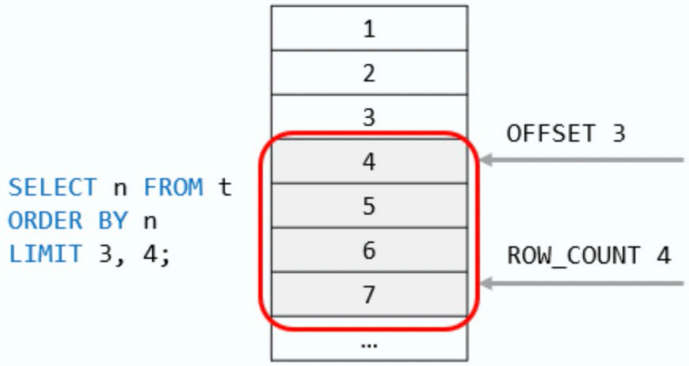
limit offset

```
SELECT
    select_list
FROM
    table_name
LIMIT [offset,] row_count;
```

In this syntax:

- The `offset` specifies the offset of the first row to return. The `offset` of the first
- The `row_count` specifies the maximum number of rows to return.

The following picture illustrates the `LIMIT` clause:



In addition to the above syntax, MySQL provides the following alternative `LIMIT` clause for compatibility with PostgreSQL.

```
LIMIT row_count OFFSET offset
```

```
SELECT
   *
FROM
   table
LIMIT n OFFSET m;
```

The statement first skips `m` rows before returning `n` rows generated by the query. If `m` is zero, the statement will work like without the `OFFSET` clause.

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [, transaction_characteristic] ...

transaction_characteristic: {
    ISOLATION LEVEL level
  | access_mode
}

level: {
     REPEATABLE READ
   | READ COMMITTED
   | READ UNCOMMITTED
   | SERIALIZABLE
}

access_mode: {
     READ WRITE
   | READ ONLY
}
```

BEGIN

BEGIN — start a transaction block

Synopsis

```
BEGIN [ WORK | TRANSACTION ] [ transaction_mode [, ...] ]

where transaction_mode is one of:

    ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
    READ WRITE | READ ONLY
    [ NOT ] DEFERRABLE
```

13.3.8.1 XA Transaction SQL Statements

To perform XA transactions in MySQL, use the following statements:

```
XA {START|BEGIN} xid [JOIN|RESUME]

XA END xid [SUSPEND [FOR MIGRATE]]

XA PREPARE xid

XA COMMIT xid [ONE PHASE]

XA ROLLBACK xid

XA RECOVER [CONVERT XID]
```

PREPARE TRANSACTION

PREPARE TRANSACTION — prepare the current transaction for two-phase commit

Synopsis

```
PREPARE TRANSACTION transaction_id
```

COMMIT PREPARED, ROLLBACK PREPARED

13.3.6 LOCK TABLES and UNLOCK TABLES Statements

```
LOCK TABLES
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...

lock_type: {
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE
}

UNLOCK TABLES
```

LOCK

LOCK — lock a table

Synopsis

```
LOCK [ TABLE ] [ ONLY ] name [ * ] [, ...] [ IN lockmode MODE ] [ NOWAIT ]

where lockmode is one of:

    ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE
    | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE
```

```
REPLACE [INTO] table_name(column_list)
VALUES(value_list);
```

```
INSERT INTO table_name(column_list) VALUES(value_list)
ON CONFLICT target action;
```

PostgreSQL added the `ON CONFLICT target action` clause to the INSERT statement to support the upsert feature

The target can be:

- `(column_name)` — a column name.
- `ON CONSTRAINT constraint_name` — where the constraint name could be a name of the UNIQUE constraint.
- `WHERE predicate` — a WHERE clause with a predicate

The action can be:

- `DO NOTHING` — means do nothing if the row already exists in the table.
- `DO UPDATE SET column_1 = value_1, .. WHERE condition` — update some fields in the table.

```
INSERT INTO customers (name, email)
VALUES
   (
      'Microsoft',
      'hotline@microsoft.com'
   )
ON CONFLICT (name)
DO
      UPDATE
     SET email = EXCLUDED.email || ';' || customers.email;
```

13.2.7 LOAD DATA Statement

```
LOAD DATA
    [LOW_PRIORITY | CONCURRENT] [LOCAL]
    INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var
        [, col_name_or_user_var] ...)]
    [SET col_name={expr | DEFAULT},
        [, col_name={expr | DEFAULT}] ...]
```

COPY

COPY — copy data between a file and a table

Synopsis

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | PROGRAM 'command' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]
    [ WHERE condition ]

COPY { table_name [ ( column_name [, ...] ) ] | ( query ) }
    TO { 'filename' | PROGRAM 'command' | STDOUT }
    [ [ WITH ] ( option [, ...] ) ]

where option can be one of:

    FORMAT format_name
    FREEZE [ boolean ]
    DELIMITER 'delimiter_character'
    NULL 'null_string'
    HEADER [ boolean ]
    QUOTE 'quote_character'
    ESCAPE 'escape_character'
    FORCE_QUOTE { ( column_name [, ...] ) | * }
    FORCE_NOT_NULL ( column_name [, ...] )
    FORCE_NULL ( column_name [, ...] )
    ENCODING 'encoding_name'
```

存储过程、函数、触发器

语法	https://dev.mysql.com/doc/refman/8.0/en/stored-objects.html	https://www.postgresql.org/docs/12/plpgsql.html
procedure		
function		
debugger		
trigger		
event trigger		

约束

	MySQL	PG
pk	primary key	primary key
uk	unique	unique
not null	not null	not null
check	不约束	强制约束
exclude	-	排他，例如 新插入数据与已有数据有空间、范围相交

- MySQL
  - btree,invert(fulltext, multi-value, json array) ,
  - 表达式索引 （SPATIAL and FULLTEXT indexes cannot have functional key parts.）
- PG
  - btree,hash,gin,gist,spgist,brin,bloom,rum,
  - exclude索引、表达式索引、partial 索引（分区索引）

PG full text search

- 操作
  - 包含、相交、filter(rank,。。。)
- 存储
  - 分类（1,2,3,4 大标题，副标题，摘要，正文）
  - 位置
- 自定义分词
- 索引
  - gin (倒排)
  - 分词+其他 (cover index)
    - rum

特征化过程(to_tsvector)

- parse
- token 归类，位置
- 配置：类别-字典 映射
- 扫描字典：转换为别名，位置

全文查询

- tsquery
- 包含
- 不包含
- 与
- 或
- 位置，分类(4大类)。。。
- rank

倒排索引内核优化技术

- 如何加速写入
  - 倒排索引 fastupdate
- 如何做到实时查询
  - list+tree 合并查询
- 为什么有时候慢
  - pending list
  - page inspect
- 倒排索引用途
  - 模糊查询、正则匹配、全文检索、json、数组、任意字段组合查询、

空间索引

- mysql -> geohash 编码，btree索引
  - 只支持包含 (geohash prefix search)

- pg -> base on r-tree
  - postgis, ganos
  - 支持搜索种类多，支持专业GIS处理（平面、立体、栅格、点云、轨迹、路径规划），
  - 234D，点、线、面、立体，
  - 支持包含、相交、距离排序、范围、srid、空间+非空间+多值列 组合过滤

	MySQL	PG

kill session
 	
show process
	show processlist;
	kill pid;
		select pid,query,state,wait_event from pg_stat_activity;
		select pg_terminate_backend(pid);
		select pg_cancel_backend(pid);

show database	show databases;	\l select * from pg_database;
show tables	show tables	\dt n.* select * from pg_tables;
show columns	show columns from table desc table	desc table \d table
show users	show users	\du select * from pg_roles
explain	explain query	explain query explain (analzye,verbose,timing,costs,buffers) query
analyze	analyze	analyze table
job		pg_cron
lock、wait		pg_locks pg_stat_activity
top sql		pg_stat_statements calls, total_time, iotime, stddev
history		csvlog auto_explain
perf insight		pg_stat_activity pg_stat_sample

参考资料

- MySQL手册
  - https://www.mysqltutorial.org/
  - https://dev.mysql.com/doc/refman/8.0/en/
- PG 管理、开发规范
  - https://github.com/digoal/blog/blob/master/201609/20160926_01.md
- PG “十万个为什么”
- PG手册
  - https://www.postgresql.org/docs/current/index.html
  - https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-vs-mysql/
- GIS手册
  - http://postgis.net/docs/manual-3.0/

一期开课计划(PG+MySQL联合方案)

- - 2019.12.30 19:30 RDS PG产品概览，如何与MySQL结合使用
- - 2019.12.31 19:30 如何连接PG，GUI，CLI的使用
- - 2020.1.3 19:30 如何压测PG数据库、如何瞬间构造海量测试数据
- - 2020.1.6 19:30 MySQL与PG对比学习(面向开发者)
- - 2020.1.7 19:30 如何将MySQL数据同步到PG（DTS）
- - 2020.1.8 19:30 PG外部表妙用 - mysql_fdw, oss_fdw（直接读写MySQL数据、冷热分离）
- - 2020.1.9 19:30 PG应用场景介绍 - 并行计算，实时分析
- - 2020.1.10 19:30 PG应用场景介绍 - GIS
- - 2020.1.13 19:30 PG应用场景介绍 - 用户画像、实时营销系统
- - 2020.1.14 19:30 PG应用场景介绍 - 多维搜索
- - 2020.1.15 19:30 PG应用场景介绍 - 向量计算、图像搜索
- - 2020.1.16 19:30 PG应用场景介绍 - 全文检索、模糊查询
- - 2020.1.17 19:30 PG 数据分析语法介绍
- - 2020.1.18 19:30 PG 更多功能了解：扩展语法、索引、类型、存储过程与函数。如何加入PG技术社群

技术社群

PG技术交流钉钉群(3500+人)