



新手机 新应用 新娱乐

**PostgreSQL 9.1.3  
2Day DBA QuickGuide  
Ver: 0.11**

Author: Digoal.Zhou

Phone:

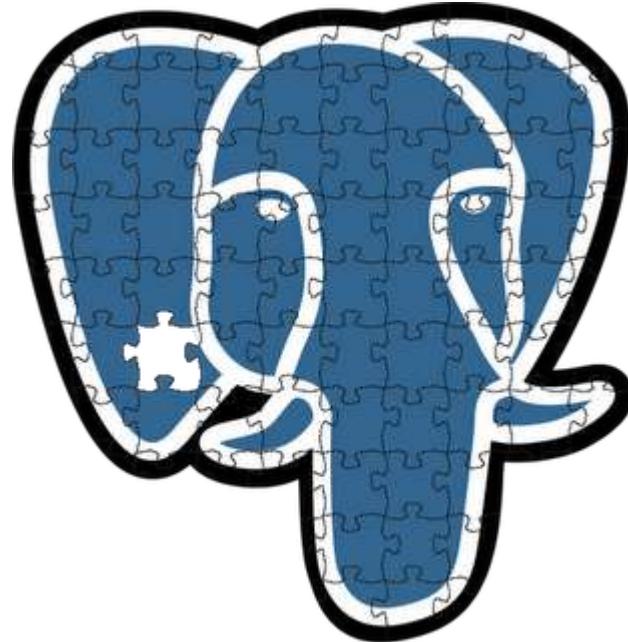
Mail: [digoal@126.com](mailto:digoal@126.com)

QQ: 276732431

Blog: <http://blog.163.com/digoal@126/>

# Day 1

## PostgreSQL 9.1.3 2Day DBA QuickGuide



# Day 1

## ■ About PostgreSQL

- 索引
- 并发控制
- SQL性能优化

## ■ Tutorial

- 安装
- 初始化集群
- 配置集群
- 启动/关闭集群
- 连接PostgreSQL

## ■ SQL Language

- SQL语法
- 数据定义
- 数据操作
- 权限
- 数据类型
- 函数与操作符
- 类型转换

# About PostgreSQL

- Maximum size for a database? unlimited
- Maximum size for a table? 32 TB
- Maximum size for a row? 400 GB
- Maximum size for a field? 1 GB
- Maximum number of rows in a table? unlimited
- Maximum number of columns in a table? 250-1600 depending on column types
- Maximum number of indexes on a table? unlimited

# Tutorial

- 安装
- 初始化集群
- 配置集群
- 启动/关闭集群
- 连接PostgreSQL

# Tutorial

## ■ 版本

- PostgreSQL 9.1.3
- CentOS 5.x 64

## ■ 安装前准备工作

- 下载源码
  - <http://www.postgresql.org/ftp/source/v9.1.3/>
- 配置存储
- 配置OS

# 配置存储

- 磁盘选择
  - 机械盘 VS SSD
- RAID选择
  - RAID5, RAID5/o, RAID1, RAID1/o
  - RAID5, RAID5/o 写性能差, 坏盘后性能下降严重, REBUILD耗时长. 可以使用n-1的容量.
  - RAID1, RAID1/o 读写性能好, 坏盘后基本没有性能下降, REBUILD耗时短. 可以使用n/2的容量.
  - 高端存储CACHE够大的情况下RAID5, RAID5/o写性能也可以接受.
- 存储CACHE
  - 有掉电保护的情况下, 建议开启存储或RAID卡的写CACHE.
  - 磁盘的CACHE一定要关闭.
- pg\_test\_fsync模块调用各种同步写函数测试存储处理IO的能力
- 测试包含write-back和write-through
- <http://blog.163.com/digoal@126/blog/static/163877040201141795025354/>
- <http://blog.163.com/digoal@126/blog/static/1638770402012449234965/>



# 配置存储

- 使用fdatasync函数测试同步写举例, 往同一个位置写.
- 【参考】Linux Programmer's Manual (open, write, lseek, fdatasync)

```
/*
 * Test fdatasync if available
 */
    printf<LABEL_FORMAT, "fdatasync">;
    fflush(stdout);

#ifndef HAVE_FDATASYNC
    if (<<tmpfile = open<filename, O_RDWR, 0>> == -1)
        die("could not open output file");
    gettimeofday(&start_t, NULL);
    for (ops = 0; ops < ops_per_test; ops++)
    {
        for (writes = 0; writes < writes_per_op; writes++)
            if (<<write<tmpfile, buf, XLOG_BLCKSZ> != XLOG_BLCKSZ>
                die("write failed");
        fdatasync(tmpfile);
        if (<<lseek(tmpfile, 0, SEEK_SET)> == -1>
            die("seek failed");
    }
    gettimeofday(&stop_t, NULL);
    close(tmpfile);
    print_elapse(start_t, stop_t);
#else
    printf<NA_FORMAT, "n/a\n";
#endif|
```

open一次,  
循环多次  
write  
fdatasync  
lseek

# 配置存储

## ■ 不调用同步写函数的写举例

```
test_non_sync(void)
{
    int                 tmpfile,
                      ops;

    /*
     * Test a simple write without fsync
     */
    printf("\nNon-Sync'ed %dkB writes:\n", XLOG_BLCKSZ_K);
    printf(LABEL_FORMAT, "write");
    fflush(stdout);

    gettimeofday(&start_t, NULL);
    for (ops = 0; ops < ops_per_test; ops++)
    {
        if ((tmpfile = open(filename, O_RDWR, 0)) == -1)
            die("could not open output file");
        if (write(tmpfile, buf, XLOG_BLCKSZ) != XLOG_BLCKSZ)
            die("write failed");
        close(tmpfile);
    }
    gettimeofday(&stop_t, NULL);
    print_elapse(start_t, stop_t);
}
```

每次  
open  
write  
close

# 配置OS

- 结合PostgreSQL编译安装时configure的选项, 有选择的安装OS的依赖包.
- OS配置
- /etc/sysctl.conf
- /etc/security/limits.conf
- /etc/sysconfig/iptables
- 时间调整
  - 自动配置ntpd服务 或者 使用crontab如下
  - 8 \* \* \* \* /usr/sbin/ntpdate asia.pool.ntp.org && /sbin/hwclock --systohc
- 设备管理
  - 逻辑卷, blockdev --setra
- 文件系统
  - XFS, ext3, ext4, ZFS
  - noatime
- 添加用户
- 配置环境
- 【参考】
- CentOS kernel-doc-x.x.xx-xxx

# 配置OS

- `#!/bin/bash`
- `# simple shmsetup script`
- `page_size=`getconf PAGE_SIZE``
- `phys_pages=`getconf _PHYS_PAGES``
- `shmall=`expr $phys_pages``
- `shmmax=`expr $shmall \* $page_size``
- `echo kernel.shmmax = $shmmax`
- `echo kernel.shmall = $shmall`
- `vi /etc/sysctl.conf`
- `kernel.shmmmax =`
- `kernel.shmall =`
- `kernel.shmmni = 4096`
- `kernel.sem = 50100 64128000 50100 1280`
- `fs.file-max = 7672460`
- `net.ipv4.ip_local_port_range = 9000 65000`
- `net.core.rmem_default = 1048576`
- `net.core.rmem_max = 4194304`
- `net.core.wmem_default = 262144`
- `net.core.wmem_max = 1048576`
- `fs.aio-max-nr = 1048576`
- `sysctl -p`

Table 17-2. PostgreSQL Shared Memory Usage

Usage	Approximate shared memory bytes required (as of 8.3)
Connections	<code>(1800 + 270 * max_locks_per_transaction) * max_connections</code>
Autovacuum workers	<code>(1800 + 270 * max_locks_per_transaction) * autovacuum_max_workers</code>
Prepared transactions	<code>(770 + 270 * max_locks_per_transaction) * max_prepared_transactions</code>
Shared disk buffers	<code>(block_size + 208) * shared_buffers</code>
WAL buffers	<code>(wal_block_size + 8) * wal_buffers</code>
Fixed space requirements	770 kB

# 配置OS

- vi /etc/security/limits.conf
  - \* soft nofile 131072
  - \* hard nofile 131072
  - \* soft nproc 131072
  - \* hard nproc 131072
  - \* soft core unlimited
  - \* hard core unlimited
  - \* soft memlock 50000000
  - \* hard memlock 50000000

- vi /etc/sysconfig/iptables
  - -A RH-Firewall-1-INPUT -i lo -j ACCEPT
  - # 允许源IP
  - -A RH-Firewall-1-INPUT -s 192.168.0.0/16 -j ACCEPT
  - # 允许源IP访问目标端口
  - # -A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -m tcp -p tcp --dport 1921 -j ACCEPT
  - # 允许任意IP访问目标端口
  - # -A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 5432 -j ACCEPT

# 配置OS

- useradd postgres
- vi ~/.bash\_profile
- export PGPORT=5432
- export PGDATA=/data01/pgdata/digoal/5432/pg\_root
- export PGHOME=/opt/pgsql
- export PGHOST=\$PGDATA
- export LANG=en\_US.utf8
- export LD\_LIBRARY\_PATH=\$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib
- export DATE=`date +"%Y%m%d%H%M"'
- export PATH=\$PGHOME/bin:\$PATH:
- export MANPATH=\$PGHOME/share/man:\$MANPATH
- alias rm='rm -i'
- alias ll='ls -lh'
- 【参考】
- <http://www.postgresql.org/docs/9.1/static/libpq-envvars.html>



# 安装

- su - root
  - tar -jxvf postgresql-9.1.3.tar.bz2
  - chown -R postgres:postgres postgresql-9.1.3
  - su - postgres
  - cd postgresql-9.1.3
  - ./configure --prefix=/opt/pgsql --with-pgport=5432 --with-perl --with-python --with-openssl --with-pam --without-ldap --with-libxml --with-libxslt --enable-thread-safety
  - gmake world
  - su - root
  - gmake install-world
- 
- 【参考】
  - postgresql-9.1.3/INSTALL

# 初始化集群

- initdb -A md5 -D \$PGDATA -E UTF8 --locale=C -W
  - -A, --auth=METHOD
    - default authentication method for local connections
  - [-D, --pgdata=]DATADIR
    - location for this database cluster
  - -E, --encoding=ENCODING
    - set default encoding for new databases
  - --locale=LOCALE
    - set default locale for new databases
  - -W, --pwprompt
    - prompt for a password for the new superuser

```
base
global
pg_clog
pg_hba.conf
pg_ident.conf
pg_multixact
pg_notify
pg_serial
pg_stat_tmp
pg_subtrans
pg_tblspc
pg_twophase
PG_VERSION
pg_xlog
postgresql.conf
postmaster.opts
```

# 配置集群

## ■ pg\_hba.conf

```
# TYPE DATABASE      USER      ADDRESS            METHOD
# "local" is for Unix domain socket connections only
# local  all          all                   md5
local  all          all                   trust
# IPv4 local connections:
# host   all          all      127.0.0.1/32      md5
host   all          all      127.0.0.1/32      trust
# IPv6 local connections:
host   all          all      ::1/128           md5
host test all 0.0.0.0/0 md5
host postgres all 0.0.0.0/0 reject
host all all 0.0.0.0/0 md5
```

# 配置集群(列出部分配置)

- postgresql.conf
  - listen\_addresses = '0.0.0.0'
  - unix\_socket\_directory = ':'
  - unix\_socket\_permissions = 0700
  - shared\_buffers = 512MB
  - maintenance\_work\_mem = 512MB
  - max\_stack\_depth = 8MB
  - shared\_preload\_libraries =  
'pg\_stat\_statements'
  - wal\_level = hot\_standby
  - wal\_buffers = 16384kB
  - synchronous\_commit = off
  - wal\_writer\_delay = 10ms
  - checkpoint\_segments = 128
  - archive\_mode = on
  - archive\_command = '/bin/date'
- max\_wal\_senders = 32
- hot\_standby = on
- random\_page\_cost = 2.0
- effective\_cache\_size = 12000MB
- log\_checkpoints = on
- log\_statement = 'ddl'
- track\_activity\_query\_size = 2048
- autovacuum = on
- log\_autovacuum\_min\_duration = 0
- custom\_variable\_classes =  
'pg\_stat\_statements'
- pg\_stat\_statements.max = 1000
- pg\_stat\_statements.track = all

# 配置集群

## ■ 可动态调整的配置修改后如何生效(包括pg\_hba.conf)

- pg\_ctl reload -D \$PGDATA
- 或者给postgres主进程发出SIGHUP信号

## ■ 静态配置修改后如何生效

- pg\_ctl stop -m fast -D \$PGDATA
- pg\_ctl start -D \$PGDATA

# 启动/关闭集群

- 启动
- su - postgres
- pg\_ctl start -D \$PGDATA

```
postgres@db-172-16-0-100:~$ pg_ctl start -D $PGDATA
server starting
postgres@db-172-16-0-100:~$ LOG:  loaded library "pg_stat_statements"
```

- 关闭
- su - postgres
- pg\_ctl stop -m fast -D \$PGDATA

```
postgres@db-172-16-0-100:~$ pg_ctl stop -m fast -D $PGDATA
waiting for server to shut down... done
server stopped
```

# 启动/关闭集群

- 关闭的几种模式
  - smart(默认)
    - 等待所有已经连接的客户端断开连接
    - 等待online 备份完成
  - fast
    - 不等待客户端断开连接
    - 所有在进行的事务全部回滚, 然后断开连接
    - 如果有的户, 终止online 备份
  - immediate
    - abort掉所有的进程, 最快的关集群方式, 但是重启集群时需要恢复.
    - 一般用在紧急维护, 如UPS的电不够了, 需要马上停库, 停主机.
- 一般可以先用smart模式关集群, 执行后数据库将不允许新的连接进来. 等已有的连接事务执行完成后用再fast关集群. 尽量减少事务回滚的可能.

# 连接PostgreSQL

## ■ 连接数据库

■ psql -h 127.0.0.1

```
postgres@db-172-16-0-100:~$ psql -h 127.0.0.1
psql (9.1.3)
Type "help" for help.

postgres=#
```

■ psql -h \$ip

```
postgres@db-172-16-0-100:~$ psql -h 172.16.0.100 -U postgres -d test
Password for user postgres:
psql (9.1.3)
Type "help" for help.

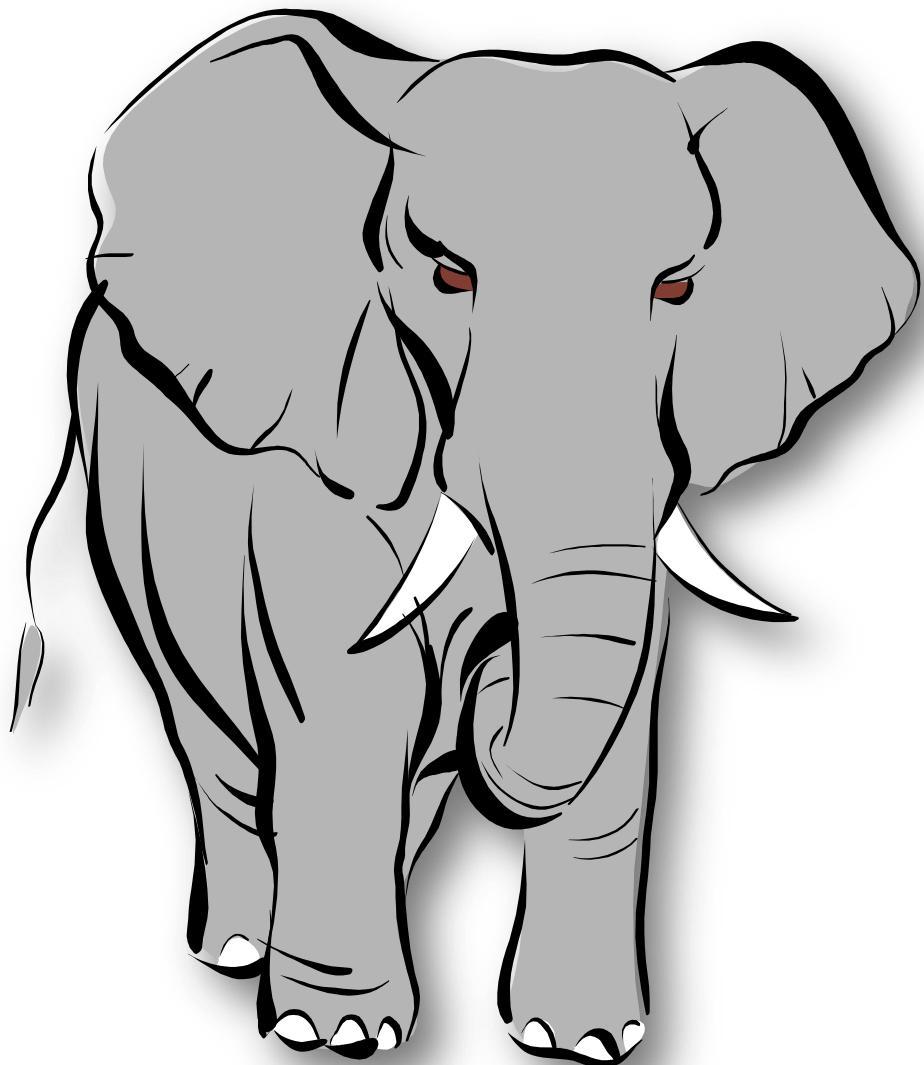
test=#
```

■ psql -h unix\_socket

```
postgres@db-172-16-0-100:~$ psql -h $PGDATA
psql (9.1.3)
Type "help" for help.

postgres=#
```

# SQL Language





# SQL Syntax

# SQL Syntax-Lexical Structure

- SELECT \* FROM pg\_class WHERE relname = 'pg\_statistic' LIMIT 1; -- is one comment
- TOKEN
  - keyword (SELECT FROM WHERE LIMIT)
  - identifier or quoted identifier (pg\_class relname, 表名, 列名, 对象名...)
    - 默认小写, 如需大写需使用双引号
  - literal or constant ('pg\_statistic' 1)
  - special character symbol (\*)
  - comment (-- is one-line comment) or /\* \*/ multi-line comment)
  - operator (=)

# keyword

- <http://www.postgresql.org/docs/9.1/static/sql-keywords-appendix.html>
- <http://www.postgresql.org/docs/9.1/static/sql-commands.html>



# identifier

```
■ define NAMEDATALEN 64 (长度限制, 截断到63)
■ truncate_identifier(char *ident, int len, bool warn)
{
    if (len >= NAMEDATALEN)
    {
        len = pg_mbcliplen(ident, len, NAMEDATALEN - 1);
        if (warn)
        {
            char      buf[NAMEDATALEN];
            memcpy(buf, ident, len);
            buf[len] = '\0';
            ereport(NOTICE,
                    (errcode(ERRCODE_NAME_TOO_LONG),
                     errmsg("identifier \"%s\" will be truncated to \"%s\"",
                           ident, buf)));
        }
        ident[len] = '\0';
    }
}
```

超过长度将截断  
到限制量-1, 同时  
报错.

# identifier

- 改变最大长度限制,需重新*initdb*
- src/include/pg\_config\_manual.h
- /\*
- \* Maximum length for identifiers (e.g. table names, column names,  
■ \* function names). Names actually are limited to one less byte than this,
- \* because the length must include a trailing zero byte.
- \*
- \* Changing this requires an initdb.
- \*/
- #define NAMEDATALEN 64

# literal or constant

- implicitly-typed literal or constant
  - string
    - E'digoal\\'
    - \$\$digoal\\\$\$
    - \$tag\$digoal\\\$tag\$
  - bit string
    - B'1010101'
  - number
    - 10 or +10
    - -23.4
    - +100.1 or 100.1
    - 10e-1
    - 98e+10 or 98e10
- explicitly-typed literal or constant
  - type 'string'
    - time '12:00:00'
  - 'string'::type
    - '1 hour'::interval
  - CAST ( 'string' AS type )
    - CAST('127.0.0.1' AS inet);

# operator

- + - \* / < > = ~ ! @ # % ^ & | ` ? ||
- postgres=# select count(\*) from pg\_operator;
- count
- -----
- 706
- SELECT 3 OPERATOR(pg\_catalog.+)

Column	Type	Modifiers
oprname	name	not null
oprnamespace	oid	not null
oprowner	oid	not null
oprkind	"char"	not null
oprcanmerge	boolean	not null
oprcanhash	boolean	not null
oprleft	oid	not null
oprright	oid	not null
oprresult	oid	not null
oprcom	oid	not null
oprnegate	oid	not null
oprcode	regproc	not null
oprrest	regproc	not null
oprjoin	regproc	not null

# special character

■ \$

- string quoted
- positional parameter in function or prepared statement

■ ()

- enforce precedence

■ []

- array selected elements

■ ,

- separate the elements of a list

■ ;

- terminate a SQL

■ :

- slice from array

■ \*

- all the fields of a table or composite value

■ .

- numeric , separate schema, table, column names.

# SQL Syntax-Value Expressions

- constant
  - 前面已经讲到
- column reference
  - [schema\_name.]table.column\_name
  - alias.column\_name
- positional parameter
  - \$number, 比如用在function中或者 prepared sql中
  - 例如:
  - postgres=# CREATE TABLE digoal\_t1(id int,info text);
  - postgres=# INSERT INTO digoal\_t1 VALUES (1,'DIGOAL'),(2,'PostgreSQL'),(3,'Pg foundry'),(4,'pgxn');
- postgres=# PREPARE pre\_1(text)  
AS SELECT id FROM digoal\_t1  
WHERE lower(info) ~ lower(\$1);
- postgres=# EXECUTE pre\_1('post');
- id
- ----
- 2
- postgres=# DEALLOCATE pre\_1;

# SQL Syntax-Value Expressions

## ■ subscript

- 例如:
- `SELECT a[2:3][1:2] FROM (SELECT ARRAY[[1,2,3],[4,5,6],[7,8,9],[10,11,12]] AS a) t;`
- `a`
- `-----`
- `\{4,5\},\{7,8\}`

## ■ field selection

- 从行类型或composite类型中指定 field
- 例如:
- `CREATE TYPE inventory_item AS ( name text, supplier_id integer, price numeric`

- `);`
- `CREATE TABLE on_hand ( item inventory_item, count integer`
- `);`
- `INSERT INTO on_hand VALUES (ROW('fuzzy dice', 42, 1.99), 1000);`
- `postgres=# SELECT (item).name`  
`FROM on_hand WHERE`  
`(item).price > .99;`
- `name`
- `-----`
- `fuzzy dice`

# SQL Syntax-Value Expressions

- operator invocation
  - OPERATOR(schema.operatorname)
  - 例如:
  - postgres=# SELECT 3 OPERATOR(pg\_catalog.+) 4;
  - ?column?
  - -----
  - 7
- function call
  - postgres=# SELECT now();  
■ 2012-04-24 08:52:53.787523+08
- aggregate expression
  - aggregate\_name (expression [ , ... ] [ order\_by\_clause ] )
  - aggregate\_name (ALL expression [ , ... ] [ order\_by\_clause ] )
- aggregate\_name (DISTINCT expression [ , ... ] [ order\_by\_clause ] )
- aggregate\_name ( \* )
- 例如, 把多行聚集成一个数组的聚集函数
  - postgres=# SELECT array\_agg(id ORDER BY id desc) FROM (SELECT generate\_series(1,10) AS id) AS t;  
■ -----  
■ {10,9,8,7,6,5,4,3,2,1}
  - postgres=# SELECT array\_agg(id ORDER BY id) FROM (SELECT generate\_series(1,10) AS id) AS t;  
■ -----  
■ {1,2,3,4,5,6,7,8,9,10}

# SQL Syntax-Value Expressions

## ■ window function call

- 例如
- CREATE TABLE window\_test(id int, name text, subject text, score numeric);
- INSERT INTO window\_test VALUES(1,'digoal','数学',99.5),  
(2,'digoal','语文',89.5),  
(3,'digoal','英语',79.5),  
(4,'digoal','物理',99.5),  
(5,'digoal','化学',98.5),  
(6,'刘德华','数学',89.5),  
(7,'刘德华','语文',99.5),  
(8,'刘德华','英语',79.5),  
(9,'刘德华','物理',89.5),  
(10,'刘德华','化学',69.5),  
(11,'张学友','数学',89.5),  
(12,'张学友','语文',91.5),  
(13,'张学友','英语',92.5),  
(14,'张学友','物理',93.5),  
(15,'张学友','化学',94.5);
- -- 取出每门课程的第一名.
- SELECT id,name,subject,score FROM  
(SELECT row\_number() OVER  
(PARTITION BY subject ORDER BY  
score DESC) AS rn,\* FROM  
window\_test) AS t
- WHERE rn=1 ORDER BY SUBJECT;

# SQL Syntax-Value Expressions

- id | name | subject | score
- -----+-----+-----
- 5 | digoal | 化学 | 98.5
- 1 | digoal | 数学 | 99.5
- 4 | digoal | 物理 | 99.5
- 13 | 张学友 | 英语 | 92.5
- 7 | 刘德华 | 语文 | 99.5
- type cast
  - 前面有例子
- collation expression
  - SELECT a, b, c FROM tbl WHERE ...  
ORDER BY a COLLATE "C";
  - SELECT \* FROM tbl WHERE a > 'foo' COLLATE "C";
- SELECT \* FROM tbl WHERE a COLLATE "C" > 'foo';
- scalar subquery
- SELECT name, (SELECT max(pop)  
FROM cities WHERE cities.state =  
states.name)
- FROM states;
- array constructor
  - ARRAY[]
- row constructor
  - ROW()

# SQL Syntax-Function Call

- 创建
  - CREATE OR REPLACE FUNCTION f\_test(i\_left numeric, i\_right numeric) RETURNS numeric AS \$\$
  - DECLARE
  - BEGIN
  - RETURN i\_left \* i\_right;
  - END;
  - \$\$ LANGUAGE plpgsql;
  
- 调用函数的几种方法
  - Positional Notation
    - SELECT f\_test(10, 2.5);
  - Named Notation
    - SELECT f\_test(i\_left := 10, i\_right := 2.5);
  - Mixed Notation
    - SELECT f\_test(10, i\_right := 2.5);



# Data Definition

# Table and Default Value

- CREATE TABLE test (id serial PRIMARY KEY, name text, info text, crt\_time timestamp(0) default now());
- INSERT INTO test (name,info) VALUES ('digoal','DBA');
- SELECT \* FROM test;
- | id | name   | info | crt_time            |
|----|--------|------|---------------------|
| 1  | digoal | DBA  | 2012-04-24 09:52:19 |
- Table "public.test"
- | Column   | Type                           | Modifiers   |
|----------|--------------------------------|---|
| id       | integer                        | not null default nextval('test_id_seq'::regclass) |
| name     | text                           |   |
| info     | text                           |   |
| crt_time | timestamp(0) without time zone | default now()                                     |
- Indexes:
- "test\_pkey" PRIMARY KEY, btree (id)

# Constraint

## ■ check

- CREATE TABLE products (
  - product\_no integer,
  - name text,
  - price numeric,
    - CHECK (price > 0),
  - discounted\_price numeric,
    - CHECK (discounted\_price > 0),
    - CHECK (price > discounted\_price)
  - );

## ■ not null / unique / primary key / foreign key

- CREATE TABLE a(c1 text,c2 text,PRIMARY KEY (c1,c2));
- CREATE TABLE a(c1 text NOT NULL,c2 text NOT NULL,UNIQUE (c1,c2));

foreign key (columns) reference table(columns), 唯一约束或pk必须与  
foreign key的表一致.复合对复合,单个对单个.不能混用.

- foreign key
- CREATE TABLE a(c1 text,c2 text,UNIQUE (c1,c2));
- CREATE TABLE b(c1 text,c2 text,FOREIGN KEY(c1,c2) REFERENCES a(c1,c2));

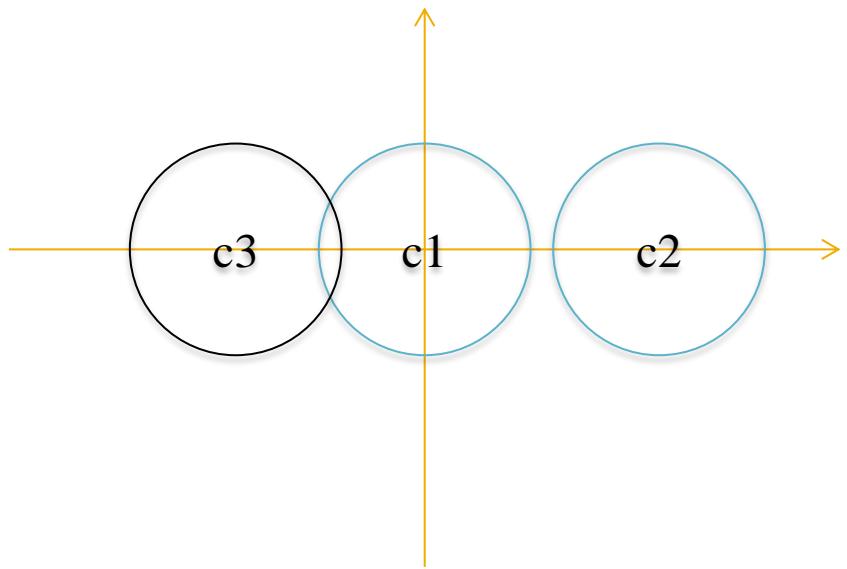
- CREATE TABLE a(c1 text UNIQUE,c2 text UNIQUE);
- CREATE TABLE b(c1 text,c2 text,FOREIGN KEY(c1,c2) REFERENCES a(c1,c2));
- ERROR: there is no unique constraint matching given keys for referenced table "a"

# Constraint

- foreign key
  - FOREIGN KEY ( column\_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]
  - [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] }
  - [ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
  - MATCH FULL will not allow one column of a multicolumn foreign key to be null unless all foreign key columns are null. MATCH SIMPLE allows some foreign key columns to be null while other parts of the foreign key are not null.
  - NO ACTION (default, deferrable enabled)
  - RESTRICT (like NO ACTION , deferrable disabled)
  - CASCADE
  - SET NULL
  - SET DEFAULT

# Constraint

- 更高级的约束用法, 例如exclusion约束
- CREATE TABLE test(id int,geo circle,EXCLUDE USING GIST (geo WITH pg\_catalog.&&));
- INSERT INTO test values(1,'<(0,0),2>'::circle);
- INSERT INTO test values(1,'<(4.1,0),2>'::circle);
- INSERT INTO test values(1,'<(-1.9,0),2>'::circle);
- ERROR: conflicting key value violates exclusion constraint "test\_geo\_excl"
- DETAIL: Key (geo)=(<(-1.9,0),2>) conflicts with existing key (geo)=(<(0,0),2>).



# System Column

## ■ oid (4 bytes)

```
postgres=# show default_with_oids;
default_with_oids
-----
off
```

```
postgres=# create table with_oids (id int) with oids;
CREATE TABLE
postgres=# create table without_oids (id int);
CREATE TABLE
postgres=# insert into with_oids values (1);
INSERT 26530 1
postgres=# insert into without_oids values (1);
INSERT 0 1
```

```
postgres=# select oid,id from with_oids;
      oid   | id
-----+-----
  26530  |  1
(1 row)
```

```
postgres=# select oid,id from without_oids;
ERROR:  column "oid" does not exist
LINE 1: select oid,id from without_oids;
```

```
postgres=# \d+ with_oids
           Table "public.with_oids"
  Column | Type   | Modifiers | Storage | Description
-----+-----+-----+-----+
    id   | integer |           | plain   |
Has OIDs: yes
```

```
postgres=# \d+ without_oids
           Table "public.without_oids"
  Column | Type   | Modifiers | Storage | Description
-----+-----+-----+-----+
    id   | integer |           | plain   |
Has OIDs: no
```

# System Column

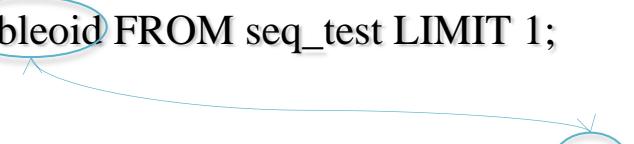
- oid主要被用于系统表中
  - Object identifiers (OIDs) are used internally by PostgreSQL as primary keys for various system tables.
- oid不能确保跨系统表的唯一性.
- obj\_description(123456,'pg\_class')
  - would retrieve the comment for the table with OID 123456.
  - 因此单个参数的obj\_description(oid)
  - 已经废弃

```

postgres=# \d+ pg_class
           Table "pg_catalog.pg_class"
  Column   |  Type   | Modifiers | Storage | Description
 -----------
  relname   | name   | not null | plain   |
  relnamespace | oid    | not null | plain   |
  reltype   | oid    | not null | plain   |
  reloftype  | oid    | not null | plain   |
  relowner   | oid    | not null | plain   |
  relam     | oid    | not null | plain   |
  relfilenode | oid   | not null | plain   |
  reltablespace | oid   | not null | plain   |
  relpages   | integer | not null | plain   |
  reltuples   | real   | not null | plain   |
  reltoastrelid | oid   | not null | plain   |
  reltoastidxid | oid   | not null | plain   |
  relhasindex | boolean | not null | plain   |
  relisshared  | boolean | not null | plain   |
  relpersistencelimit | "char" | not null | plain   |
  relkind    | "char" | not null | plain   |
  relnatts   | smallint | not null | plain   |
  relchecks   | smallint | not null | plain   |
  relhasoids  | boolean | not null | plain   |
  relhaspkey  | boolean | not null | plain   |
  relhasrules  | boolean | not null | plain   |
  relhastriggers | boolean | not null | plain   |
  relhassubclass | boolean | not null | plain   |
  relfrozenxid | xid    | not null | plain   |
  relacl     | aclitem[] |          | extended |
  reloptions  | text[]  |          | extended |
Indexes:
    "pg_class_oid_index" UNIQUE, btree (oid)
    "pg_class_relname_nsp_index" UNIQUE, btree (relname, relnamespace)
Has OIDs: yes
  
```

# System Column

## ■ tableoid (4 Bytes)

- postgres=# CREATE TABLE test (id int);
- postgres=# CREATE SEQUENCE seq\_test START WITH 1;
- postgres=# INSERT INTO test VALUES(1);
- postgres=# SELECT tableoid FROM test;
- 26534
- postgres=# SELECT tableoid FROM seq\_test LIMIT 1;
- 26537
- postgres=# SELECT relname FROM pg\_class WHERE oid IN (26534, 26537);
- relname
- -----
- test
- seq\_test

# System Column

- ctid (6 Bytes)
- xmin / xmax / cmin / cmax (4 Bytes)

```

CREATE TABLE test (id int PRIMARY KEY, name text);
INSERT INTO test(id,name) VALUES(0,'digoal'),(1,'刘德华');
BEGIN;
INSERT INTO test(id,name) VALUES(2,'张学友');
INSERT INTO test(id,name) VALUES(3,'黎明');
INSERT INTO test(id,name) VALUES(4,'郭富城');
END;

postgres=# SELECT ctid,xmin,xmax,cmin,cmax,* FROM test ORDER BY id;
 ctid | xmin   | xmax   | cmin   | cmax   | id | name
-----+-----+-----+-----+-----+-----+-----
 {0,1} | 78352588 | 0 | 0 | 0 | 0 | digoal
 {0,2} | 78352588 | 0 | 0 | 0 | 1 | 刘德华
 {0,3} | 78352589 | 0 | 0 | 0 | 2 | 张学友
 {0,4} | 78352589 | 0 | 1 | 1 | 3 | 黎明
 {0,5} | 78352589 | 0 | 2 | 2 | 4 | 郭富城

```

Heap Table's PAGE(0)

it1	it2	it3	it4	it5
it6				
				tup6
tup5	tup4	tup3	tup2	tup1

# System Column

老版本记录

```
SESSION A:  
BEGIN;  
UPDATE test SET name = 'DIGOAL' WHERE id = 1;  
  
SESSION B:  
postgres=# SELECT ctid,xmin,xmax,cmin,cmax,* FROM test WHERE id = 1;  
ctid | xmin | xmax | cmin | cmax | id | name  
-----+-----+-----+-----+-----+-----+  
(0,2) | 78352588 | 78352590 | 0 | 0 | 1 | 刘德华  
  
SESSION A:  
UPDATE test SET name = '刘德华1' WHERE id = 2;  
  
SESSION B:  
postgres=# SELECT ctid,xmin,xmax,cmin,cmax,* FROM test WHERE id = 2;  
ctid | xmin | xmax | cmin | cmax | id | name  
-----+-----+-----+-----+-----+-----+  
(0,3) | 78352589 | 78352590 | 1 | 1 | 2 | 张学友  
  
SESSION A:  
COMMIT;  
  
SESSION B:  
postgres=# SELECT ctid,xmin,xmax,cmin,cmax,* FROM test WHERE id in (1,2);  
ctid | xmin | xmax | cmin | cmax | id | name  
-----+-----+-----+-----+-----+-----+  
(0,6) | 78352590 | 0 | 0 | 0 | 1 | DIGOAL  
(0,7) | 78352590 | 0 | 0 | 1 | 2 | 刘德华1
```

老版本记录

新版本记录

# Modify Table

- add column
  - 有默认值, 将rewrite全表, 包括索引重建. 因此有排他锁, 谨慎操作.
  - 无默认值, 修改元数据, 速度快, 不会rewrite表和重建索引.
- drop column
  - pg\_attribute
  - Recover droped column from PostgreSQL
  - <http://blog.163.com/digoal@126/blog/static/163877040201112251058216/>
- Can session\_replication\_role used like MySQL's BlackHole Engine?
  - <http://blog.163.com/digoal@126/blog/static/163877040201119111234570/>
- ALTER TABLE的语法



```
Command:    ALTER TABLE
Description: change the definition of a table
Syntax:
ALTER TABLE [ ONLY ] name [ * ]
    action [, ... ]
ALTER TABLE [ ONLY ] name [ * ]
    RENAME [ COLUMN ] column TO new_column
ALTER TABLE name
    RENAME TO new_name
ALTER TABLE name
    SET SCHEMA new_schema
```

# Modify Table

## ■ ALTER TABLE的语法

```
ADD [ COLUMN ] column data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
DROP [ COLUMN ] [ IF EXISTS ] column [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] column [ SET DATA ] TYPE data_type [ COLLATE collation ] [ USING expression ]
ALTER [ COLUMN ] column SET DEFAULT expression
ALTER [ COLUMN ] column DROP DEFAULT
ALTER [ COLUMN ] column < SET | DROP > NOT NULL
ALTER [ COLUMN ] column SET STATISTICS integer
ALTER [ COLUMN ] column SET < attribute_option = value [, ... ] >
ALTER [ COLUMN ] column RESET < attribute_option [, ... ] >
ALTER [ COLUMN ] column SET STORAGE < PLAIN | EXTERNAL | EXTENDED | MAIN >
ADD table_constraint [ NOT VALID ]
ADD table_constraint_using_index
VALIDATE CONSTRAINT constraint_name
DROP CONSTRAINT [ IF EXISTS ] constraint_name [ RESTRICT | CASCADE ]
DISABLE TRIGGER [ trigger_name | ALL | USER ]
ENABLE TRIGGER [ trigger_name | ALL | USER ]
ENABLE REPLICA TRIGGER trigger_name
ENABLE ALWAYS TRIGGER trigger_name
DISABLE RULE rewrite_rule_name
ENABLE RULE rewrite_rule_name
ENABLE REPLICA RULE rewrite_rule_name
ENABLE ALWAYS RULE rewrite_rule_name
CLUSTER ON index_name
SET WITHOUT CLUSTER
SET WITH OIDS
SET WITHOUT OIDS
SET < storage_parameter = value [, ... ] >
RESET < storage_parameter [, ... ] >
INHERIT parent_table
NO INHERIT parent_table
OF type_name
NOT OF
OWNER TO new_owner
SET TABLESPACE new_tablespace
```

# Privilege

## ■ 创建角色

要我干活,  
给我权限  
ROLE



IN ROLE | GROUP, 把新增角色加盟到已有角色.  
USER | ROLE, 把已有角色加盟到新增角色.  
ADMIN, 已有角色加盟时带上WITH ADMIN OPTION选项.

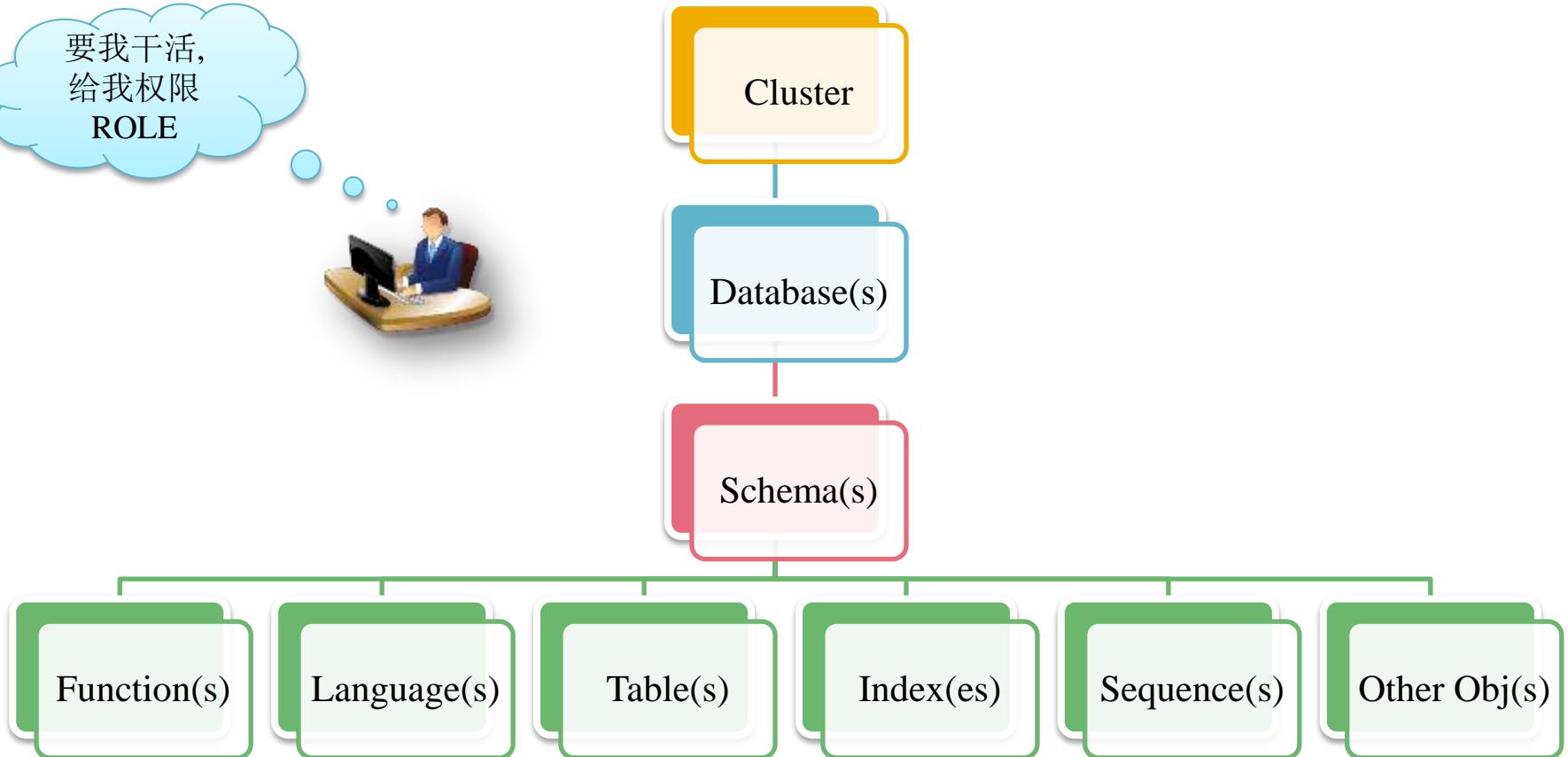
```
Command:      CREATE ROLE
Description:  define a new database role
Syntax:
CREATE ROLE name [ [ WITH ] option [ ... ] ]

where option can be:

  SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | CREATEUSER | NOCREATEUSER
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

# Privilege

要我干活，  
给我权限  
ROLE



# Privilege

- Database, 数据库的下一层是SCHEMA, 所以给数据库CREATE权限是有了在这个数据库创建SCHEMA的权限. TEMP指允许在该库创建临时表.
- GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [ , ... ] | ALL [ PRIVILEGES ] }
  - ON DATABASE database\_name [ , ... ]
  - TO { [ GROUP ] role\_name | PUBLIC } [ , ... ] [ WITH GRANT OPTION ]
- Schema
  - GRANT { { CREATE | USAGE } [ , ... ] | ALL [ PRIVILEGES ] }
  - ON SCHEMA schema\_name [ , ... ]
  - TO { [ GROUP ] role\_name | PUBLIC } [ , ... ] [ WITH GRANT OPTION ]
- Tablespace
  - GRANT { CREATE | ALL [ PRIVILEGES ] }
  - ON TABLESPACE tablespace\_name [ , ... ]
  - TO { [ GROUP ] role\_name | PUBLIC } [ , ... ] [ WITH GRANT OPTION ]

# Privilege

- Table
  - GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    - [, ...] | ALL [ PRIVILEGES ] }
    - ON { [ TABLE ] table\_name [, ...]
      - | ALL TABLES IN SCHEMA schema\_name [, ...] }
    - TO { [ GROUP ] role\_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
  - Column, PG比较强大的地方, 允许对列赋权.
    - GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column [, ...] )
      - [, ...] | ALL [ PRIVILEGES ] ( column [, ...] ) }
      - ON [ TABLE ] table\_name [, ...]
      - TO { [ GROUP ] role\_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
    - 列和表的权限比较容易混淆
    - <http://blog.163.com/digoal@126/blog/static/16387704020119193364585/>



# Privilege

- Language
  - GRANT { USAGE | ALL [ PRIVILEGES ] }
  - ON LANGUAGE lang\_name [, ...]
  - TO { [ GROUP ] role\_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
  
- Function
  - GRANT { EXECUTE | ALL [ PRIVILEGES ] }
  - ON { FUNCTION function\_name ( [ [ argmode ] [ arg\_name ] arg\_type [, ...] ] ) [, ...]  
| ALL FUNCTIONS IN SCHEMA schema\_name [, ...] }
  - TO { [ GROUP ] role\_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

# Privilege

- Sequence
  - GRANT { { USAGE | SELECT | UPDATE }  
[,...] | ALL [ PRIVILEGES ] }
  - ON { SEQUENCE sequence\_name [,...]  
| ALL SEQUENCES IN SCHEMA schema\_name [,...] }
  - TO { [ GROUP ] role\_name | PUBLIC } [,...] [ WITH GRANT OPTION ]
  
- Role
  - GRANT role\_name [,...] TO role\_name [,...] [ WITH ADMIN OPTION ]
  - (INHERIT | NOINHERIT 的区别)
  - PostgreSQL Role Membership
  - <http://blog.163.com/digoal@126/blog/static/1638770402011362564157/>
  
- 详见
  - <http://www.postgresql.org/docs/9.1/static/sql-grant.html>



# Schema

- Schema Search Path
- SHOW search\_path;
- search\_path
  - -----
- "\$user",public
  
- use qualified name:
  - schema.table
  - database.schema.table (如果用这种写法, 必须写当前连接的库名)
  
- The search path works in the same way for data type names, function names, and operator names as it does for table names. Data type and function names can be qualified in exactly the same way as table names. If you need to write a qualified operator name in an expression, there is a special provision: you must write :
- OPERATOR(schema.operator)
- SELECT 3 OPERATOR(pg\_catalog.+);

# Schema

- public Schema
- REVOKE CREATE ON SCHEMA public FROM PUBLIC;
  
- System Catalog Schema
- pg\_catalog is always effectively part of the search path. If it is not named explicitly in the path then it is implicitly searched before searching the path's schemas.
- However, you can explicitly place pg\_catalog at the end of your search path if you prefer to have user-defined names override built-in names.
- SET search\_path="\$user",public,pg\_catalog;
  
- 为每个用户创建与之同名的SCHEMA, 便于移植.
- 不建议使用public SCHEMA.

# Inheritance

- 一个表可以继承多个表
- 一个表可以被多个表继承
- 允许多级继承
- 不允许闭环继承
- 约束, FOREIGN KEY, UNIQUE KEY, PRIMARY KEY, CHECK, NOT NULL都只约束单

```
postgres=# create table p (id int primary key, name text unique, info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "p_pkey" for table "p"
NOTICE: CREATE TABLE / UNIQUE will create implicit index "p_name_key" for table "p"
postgres=# create table c1<like p including all> inherits(p);
NOTICE: merging column "id" with inherited definition
NOTICE: merging column "name" with inherited definition
NOTICE: merging column "info" with inherited definition
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "c1_pkey" for table "c1"
NOTICE: CREATE TABLE / UNIQUE will create implicit index "c1_name_key" for table "c1"
CREATE TABLE
postgres=# create table c2<like p including all> inherits(p);
NOTICE: merging column "id" with inherited definition
NOTICE: merging column "name" with inherited definition
NOTICE: merging column "info" with inherited definition
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "c2_pkey" for table "c2"
NOTICE: CREATE TABLE / UNIQUE will create implicit index "c2_name_key" for table "c2"
postgres=# INSERT INTO p VALUES(1,'DIGOAL','test');
INSERT 0 1
postgres=# INSERT INTO c1 VALUES(1,'DIGOAL','test');
INSERT 0 1
postgres=# INSERT INTO c2 VALUES(1,'DIGOAL','test');
INSERT 0 1
```

← 单表约束

# Inheritance

## ■ 查询

```
postgres=# SELECT tableoid,* FROM p;
tableoid | id | name | info
-----+---+-----+
 26588 | 1 | DIGOAL | test
 26614 | 1 | DIGOAL | test
 26624 | 1 | DIGOAL | test
(3 rows)

postgres=# SELECT tableoid,* FROM ONLY p;
tableoid | id | name | info
-----+---+-----+
 26588 | 1 | DIGOAL | test
(1 row)
```

- 主表以及所有子表全局约束举例1(不可行的做法, 并发时将无法确保唯一, 后面有例子)
- ALTER TABLE p ADD CONSTRAINT ck\_pk\_p CHECK(check\_pk\_p(id) IS true);

```
postgres=# CREATE FUNCTION check_pk_p (i_id int) RETURNS boolean AS $$  
DECLARE  
BEGIN  
PERFORM 1 FROM p WHERE id = i_id LIMIT 1;  
IF FOUND THEN  
RETURN false;  
END IF;  
RETURN true;  
EXCEPTION  
WHEN OTHERS THEN  
RETURN false;  
END;  
$$ LANGUAGE plpgsql;  
CREATE FUNCTION
```



"全局约束的函数"

# Inheritance

- 主表以及所有子表全局约束举例1(不可行)
- SESSION A:
  - postgres=# BEGIN;
  - postgres=# INSERT INTO p VALUES (1,'DIGOAL','test');
- SESSION B:
  - postgres=# INSERT INTO c1 VALUES (1,'DIGOAL','test');
- SESSION A:
  - postgres=# END;
  - postgres=# SELECT tableoid,\* FROM p;
  - tableoid | id | name | info
    - -----+----+-----+-----
    - 26588 | 1 | DIGOAL | test
    - 26614 | 1 | DIGOAL | test

# Inheritance

- 主表以及所有子表全局约束举例2(可行, 但仅仅适用于分区字段, 其他字段需要全局唯一怎么办? 可以考虑使用增加反向关系表, 需要全局唯一的字段作为分区字段, 或者使用触发器)
- 分段
- ```
create table p (id int primary key, name text unique, info text);
```
- ```
create table c0 (like p including all) inherits(p);
```
- ```
create table c1 (like p including all) inherits(p);
```
- ```
alter table p add constraint ck_p_1 check(false);
```
- ```
alter table c0 add constraint ck_c0_1 check(mod(id,2)=0);
```
- ```
alter table c0 add constraint ck_c0_2 check(mod(hashtext(name),2)=0);
```
- ```
alter table c1 add constraint ck_c1_1 check(mod(id,2)=1);
```
- ```
alter table c1 add constraint ck_c1_2 check(mod(hashtext(name),2)=1);
```
- 注意
  - DELETE, UPDATE, SELECT 父表时, 默认不加ONLY, 影响所有子表和目标表.
  - INSERT 没有ONLY选项, 也只会影响目标表. 除非有RULE或TRIGGER.

# Partition

- 可以用rule或trigger实现分区表(Range, Hash, List, Complex)
- rule不被COPY触发, 并且规则异常时插入的数据将插入主表. 无法简单处理此类问题.
- rule是每QUERY触发, 所以更适合bulk insert场景.
- trigger分区方法举例 :
- 主表:

```
CREATE TABLE p (
    city_id      int not null,
    logtime      timestamp(0) not null,
    peaktemp     int,
    unitsales    int
);
```
- 分区字段索引:

```
CREATE INDEX idx_p_logtime ON p (logtime);
```
- 子表:

```
CREATE TABLE p_201201 (LIKE p INCLUDING all) INHERITS (p);
CREATE TABLE p_201202 (LIKE p INCLUDING all) INHERITS (p);
CREATE TABLE p_201203 (LIKE p INCLUDING all) INHERITS (p);
CREATE TABLE p_201204 (LIKE p INCLUDING all) INHERITS (p);
```

# Partition

- CREATE TABLE p\_201205 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201206 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201207 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201208 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201209 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201210 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201211 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_201212 (LIKE p INCLUDING all) INHERITS (p);
- CREATE TABLE p\_default (LIKE p INCLUDING all) INHERITS (p);
- 子表分区字段约束
- ALTER TABLE p\_201201 ADD CONSTRAINT p\_201201\_ck1 CHECK (logtime>=date '2012-01-01' and logtime<date '2012-02-01');
- ALTER TABLE p\_201202 ADD CONSTRAINT p\_201202\_ck1 CHECK (logtime>=date '2012-02-01' and logtime<date '2012-03-01');
- ALTER TABLE p\_201203 ADD CONSTRAINT p\_201203\_ck1 CHECK (logtime>=date '2012-03-01' and logtime<date '2012-04-01');
- ALTER TABLE p\_201204 ADD CONSTRAINT p\_201204\_ck1 CHECK (logtime>=date '2012-04-01' and logtime<date '2012-05-01');

# Partition

- ALTER TABLE p\_201205 ADD CONSTRAINT p\_201205\_ck1 CHECK (logtime>=date '2012-05-01' and logtime<date '2012-06-01');
- ALTER TABLE p\_201206 ADD CONSTRAINT p\_201206\_ck1 CHECK (logtime>=date '2012-06-01' and logtime<date '2012-07-01');
- ALTER TABLE p\_201207 ADD CONSTRAINT p\_201207\_ck1 CHECK (logtime>=date '2012-07-01' and logtime<date '2012-08-01');
- ALTER TABLE p\_201208 ADD CONSTRAINT p\_201208\_ck1 CHECK (logtime>=date '2012-08-01' and logtime<date '2012-09-01');
- ALTER TABLE p\_201209 ADD CONSTRAINT p\_201209\_ck1 CHECK (logtime>=date '2012-09-01' and logtime<date '2012-10-01');
- ALTER TABLE p\_201210 ADD CONSTRAINT p\_201210\_ck1 CHECK (logtime>=date '2012-10-01' and logtime<date '2012-11-01');
- ALTER TABLE p\_201211 ADD CONSTRAINT p\_201211\_ck1 CHECK (logtime>=date '2012-11-01' and logtime<date '2012-12-01');
- ALTER TABLE p\_201212 ADD CONSTRAINT p\_201212\_ck1 CHECK (logtime>=date '2012-12-01' and logtime<date '2013-01-01');
- ALTER TABLE p\_default ADD CONSTRAINT p\_default\_ck1 CHECK (logtime<date '2012-01-01' or logtime>=date '2013-01-01');

# Partition

- 子表展示：
- postgres=# \d p\_201201
  - Table "public.p\_201201"
  - | Column    | Type                           | Modifiers |
|-----------|--------------------------------|-----------|
| city_id   | integer                        | not null  |
| logtime   | timestamp(0) without time zone | not null  |
| peaktemp  | integer                        |           |
| unitsales | integer                        |           |
  - Indexes:
    - "p\_201201\_logtime\_idx" btree (logtime)
  - Check constraints:
    - "p\_201201\_ck1" CHECK (logtime >= '2012-01-01'::date AND logtime < '2012-02-01'::date)
  - Inherits: p

# Partition

- 插入触发器函数
- CREATE OR REPLACE FUNCTION p\_insert\_trigger()  
RETURNS TRIGGER AS \$\$  
BEGIN  
    IF ( NEW.logtime >= DATE '2012-01-01' AND NEW.logtime < DATE '2012-02-01' ) THEN  
        INSERT INTO p\_201201 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-02-01' AND NEW.logtime < DATE '2012-03-01' ) THEN  
        INSERT INTO p\_201202 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-03-01' AND NEW.logtime < DATE '2012-04-01' ) THEN  
        INSERT INTO p\_201203 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-04-01' AND NEW.logtime < DATE '2012-05-01' ) THEN  
        INSERT INTO p\_201204 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-05-01' AND NEW.logtime < DATE '2012-06-01' ) THEN  
        INSERT INTO p\_201205 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-06-01' AND NEW.logtime < DATE '2012-07-01' ) THEN  
        INSERT INTO p\_201206 VALUES (NEW.\*);  
    ELSIF ( NEW.logtime >= DATE '2012-07-01' AND NEW.logtime < DATE '2012-08-01' ) THEN  
        INSERT INTO p\_201207 VALUES (NEW.\*);

# Partition

```
■ ELSIF ( NEW.logtime >= DATE '2012-08-01' AND NEW.logtime < DATE '2012-09-01' ) THEN
■     INSERT INTO p_201208 VALUES (NEW.*);
■ ELSIF ( NEW.logtime >= DATE '2012-09-01' AND NEW.logtime < DATE '2012-10-01' ) THEN
■     INSERT INTO p_201209 VALUES (NEW.*);
■ ELSIF ( NEW.logtime >= DATE '2012-10-01' AND NEW.logtime < DATE '2012-11-01' ) THEN
■     INSERT INTO p_201210 VALUES (NEW.*);
■ ELSIF ( NEW.logtime >= DATE '2012-11-01' AND NEW.logtime < DATE '2012-12-01' ) THEN
■     INSERT INTO p_201211 VALUES (NEW.*);
■ ELSIF ( NEW.logtime >= DATE '2012-12-01' AND NEW.logtime < DATE '2013-01-01' ) THEN
■     INSERT INTO p_201212 VALUES (NEW.*);
■ ELSIF ( NEW.logtime >= DATE '2013-01-01' OR NEW.logtime < DATE '2012-01-01' ) THEN
■     INSERT INTO p_default VALUES (NEW.*);
■ ELSE
■     RAISE EXCEPTION 'Date out of range. Fix the p_insert_trigger() function!';
■ END IF;
■ RETURN NULL;
■ END;
■ $$ LANGUAGE plpgsql;
```

# Partition

- 删除触发器函数
- CREATE OR REPLACE FUNCTION p\_delete\_trigger()  
RETURNS TRIGGER AS \$\$  
BEGIN  
    IF ( OLD.logtime >= DATE '2012-01-01' AND OLD.logtime < DATE '2012-02-01' ) THEN  
        DELETE FROM p\_201201 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-02-01' AND OLD.logtime < DATE '2012-03-01' ) THEN  
        DELETE FROM p\_201202 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-03-01' AND OLD.logtime < DATE '2012-04-01' ) THEN  
        DELETE FROM p\_201203 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-04-01' AND OLD.logtime < DATE '2012-05-01' ) THEN  
        DELETE FROM p\_201204 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-05-01' AND OLD.logtime < DATE '2012-06-01' ) THEN  
        DELETE FROM p\_201205 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-06-01' AND OLD.logtime < DATE '2012-07-01' ) THEN  
        DELETE FROM p\_201206 WHERE logtime=OLD.logtime;  
    ELSIF ( OLD.logtime >= DATE '2012-07-01' AND OLD.logtime < DATE '2012-08-01' ) THEN  
        DELETE FROM p\_201207 WHERE logtime=OLD.logtime;

# Partition

```
■ ELSIF ( OLD.logtime >= DATE '2012-08-01' AND OLD.logtime < DATE '2012-09-01' ) THEN
  ■   DELETE FROM p_201208 WHERE logtime=OLD.logtime;
  ■ ELSIF ( OLD.logtime >= DATE '2012-09-01' AND OLD.logtime < DATE '2012-10-01' ) THEN
  ■   DELETE FROM p_201209 WHERE logtime=OLD.logtime;
  ■ ELSIF ( OLD.logtime >= DATE '2012-10-01' AND OLD.logtime < DATE '2012-11-01' ) THEN
  ■   DELETE FROM p_201210 WHERE logtime=OLD.logtime;
  ■ ELSIF ( OLD.logtime >= DATE '2012-11-01' AND OLD.logtime < DATE '2012-12-01' ) THEN
  ■   DELETE FROM p_201211 WHERE logtime=OLD.logtime;
  ■ ELSIF ( OLD.logtime >= DATE '2012-12-01' AND OLD.logtime < DATE '2013-01-01' ) THEN
  ■   DELETE FROM p_201212 WHERE logtime=OLD.logtime;
  ■ ELSIF ( OLD.logtime >= DATE '2013-01-01' OR OLD.logtime < DATE '2012-01-01' ) THEN
  ■   DELETE FROM p_default WHERE logtime=OLD.logtime;
  ■ ELSE
    ■   RAISE EXCEPTION 'Date out of range. Fix the p_insert_trigger() function!';
  ■ END IF;
  ■ RETURN NULL;
  ■ END;
  ■ $$ LANGUAGE plpgsql;
```

# Partition

- 创建插入触发器
- CREATE TRIGGER insert\_p\_trigger
  - BEFORE INSERT ON p
  - FOR EACH ROW EXECUTE PROCEDURE p\_insert\_trigger();
- 创建删除触发器
- CREATE TRIGGER delete\_p\_trigger
  - BEFORE DELETE ON p
  - FOR EACH ROW EXECUTE PROCEDURE p\_delete\_trigger();
- 插入测试数据, 定向到每个分区表
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (1, timestamp '2012-01-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (2, timestamp '2012-02-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (3, timestamp '2012-03-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (4, timestamp '2012-04-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (5, timestamp '2012-05-02 12:59:59', 20, 10);

# Partition

- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (6, timestamp '2012-06-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (7, timestamp '2012-07-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (8, timestamp '2012-08-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (9, timestamp '2012-09-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (10, timestamp '2012-10-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (11, timestamp '2012-11-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (12, timestamp '2012-12-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (13, timestamp '2013-01-02 12:59:59', 20, 10);
- INSERT INTO p (city\_id, logtime, peaktemp, unitsales) VALUES (14, timestamp '2011-12-02 12:59:59', 20, 10);

# Partition

- 查询是否正确定向到子表
- ```
SELECT t1.relname,t2.* FROM pg_class t1 WHERE t2.tableoid=t1.oid ORDER BY t2.logtime;
```
- | relname   | city_id | logtime             | peaktemp | unitsales |
|-----------|---------|---------------------|----------|-----------|
| p_default | 14      | 2011-12-02 12:59:59 | 20       | 10        |
| p_201201  | 1       | 2012-01-02 12:59:59 | 20       | 10        |
| p_201202  | 2       | 2012-02-02 12:59:59 | 20       | 10        |
| p_201203  | 3       | 2012-03-02 12:59:59 | 20       | 10        |
| p_201204  | 4       | 2012-04-02 12:59:59 | 20       | 10        |
| p_201205  | 5       | 2012-05-02 12:59:59 | 20       | 10        |
| p_201206  | 6       | 2012-06-02 12:59:59 | 20       | 10        |
| p_201207  | 7       | 2012-07-02 12:59:59 | 20       | 10        |
| p_201208  | 8       | 2012-08-02 12:59:59 | 20       | 10        |
| p_201209  | 9       | 2012-09-02 12:59:59 | 20       | 10        |
| p_201210  | 10      | 2012-10-02 12:59:59 | 20       | 10        |
| p_201211  | 11      | 2012-11-02 12:59:59 | 20       | 10        |
| p_201212  | 12      | 2012-12-02 12:59:59 | 20       | 10        |
| p_default | 13      | 2013-01-02 12:59:59 | 20       | 10        |

# Partition

- 分区表优化
- constraint\_exclusion = partition # on, off, or partition
- on
  - examine constraints for all tables
- off
  - never examine constraints
- partition
  - examine constraints only for inheritance child tables and UNION ALL subqueries

# Partition

- 更新操作执行计划
- postgres=# EXPLAIN UPDATE p SET unitsales=unitsales+1 WHERE logtime=timestamp '2011-12-02 12:59:59';
  - QUERY PLAN
  - 
  - Update on p (cost=0.00..9.79 rows=9 width=26)
    - -> Seq Scan on p (cost=0.00..0.00 rows=1 width=26)
    - Filter: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
    - -> Bitmap Heap Scan on p\_default p (cost=2.31..9.78 rows=8 width=26)
      - Recheck Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
      - -> Bitmap Index Scan on p\_default\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
      - Index Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
  - UPDATE p SET unitsales=unitsales+1 WHERE logtime=timestamp '2011-12-02 12:59:59';
    - relname | city\_id | logtime | peaktemp | unitsales
    - +-----+-----+-----+
    - p\_default | 14 | 2011-12-02 12:59:59 | 20 | 11

# Partition

- 删除操作执行计划
- postgres=# EXPLAIN DELETE FROM p WHERE logtime=timestamp '2011-12-02 12:59:59';  
                  QUERY PLAN

---
- Delete on p (cost=0.00..9.76 rows=9 width=6)
  - -> Seq Scan on p (cost=0.00..0.00 rows=1 width=6)
    - Filter: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
  - -> Bitmap Heap Scan on p\_default p (cost=2.31..9.76 rows=8 width=6)
    - Recheck Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
  - -> Bitmap Index Scan on p\_default\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
    - Index Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
- DELETE FROM p WHERE logtime=timestamp '2011-12-02 12:59:59';
- DELETE 1

# Partition

- 查询操作执行计划
- postgres=# EXPLAIN SELECT \* FROM p WHERE logtime=timestamp '2011-12-02 12:59:59';  
                  QUERY PLAN

---
- Result (cost=0.00..9.76 rows=9 width=20)
- -> Append (cost=0.00..9.76 rows=9 width=20)
  - -> Seq Scan on p (cost=0.00..0.00 rows=1 width=20)
  - Filter: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
  - -> Bitmap Heap Scan on p\_default p (cost=2.31..9.76 rows=8 width=20)
    - Recheck Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)
    - -> Bitmap Index Scan on p\_default\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
      - Index Cond: (logtime = '2011-12-02 12:59:59'::timestamp without time zone)

# Partition

- 分区字段作为WHERE条件使用时, 使用函数或常量作为过滤条件的执行计划区别
- postgres=# select proname,provolatile,proargtypes from pg\_proc where prorettype in (select oid from pg\_type where typname ~ 'timestam') order by proargtypes;
- | proname                              | provolatile | proargtypes |
|--------------------------------------|-------------|-------------|
| transaction_timestamp                | s           |             |
| statement_timestamp                  | s           |             |
| pg_stat_get_bgwriter_stat_reset_time | s           |             |
| pg_conf_load_time                    | s           |             |
| pg_postmaster_start_time             | s           |             |
| pg_last_xact_replay_timestamp        | v           |             |
| clock_timestamp                      | v           |             |
| now                                  | s           |             |
- postgres=# show constraint\_exclusion;
- constraint\_exclusion
- | partition |
|-----------|
|-----------|



# Partition

- ```
postgres=# EXPLAIN SELECT * FROM p WHERE logtime=now();
QUERY PLAN
-----
```

Result (cost=0.00..127.23 rows=105 width=20)

-> Append (cost=0.00..127.23 rows=105 width=20)

    -> Seq Scan on p (cost=0.00..0.00 rows=1 width=20)  
        Filter: (logtime = now())

    -> Bitmap Heap Scan on p\_201201 p (cost=2.31..9.79 rows=8 width=20)  
        Recheck Cond: (logtime = now())  
        -> Bitmap Index Scan on p\_201201\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)  
            Index Cond: (logtime = now())

    -> Bitmap Heap Scan on p\_201202 p (cost=2.31..9.79 rows=8 width=20)  
        Recheck Cond: (logtime = now())  
        -> Bitmap Index Scan on p\_201202\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)  
            Index Cond: (logtime = now())

    -> Bitmap Heap Scan on p\_201203 p (cost=2.31..9.79 rows=8 width=20)  
        Recheck Cond: (logtime = now())  
        -> Bitmap Index Scan on p\_201203\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)

# Partition

- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_201204 p (cost=2.31..9.79 rows=8 width=20)
- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201204\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_201205 p (cost=2.31..9.79 rows=8 width=20)
- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201205\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_201206 p (cost=2.31..9.79 rows=8 width=20)
- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201206\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_201207 p (cost=2.31..9.79 rows=8 width=20)
- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201207\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_201208 p (cost=2.31..9.79 rows=8 width=20)

# Partition

- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201208\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
  - Index Cond: (logtime = now())
- -> Bitmap Heap Scan on p\_201209 p (cost=2.31..9.79 rows=8 width=20)
  - Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201209\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
  - Index Cond: (logtime = now())
- -> Bitmap Heap Scan on p\_201210 p (cost=2.31..9.79 rows=8 width=20)
  - Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201210\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
  - Index Cond: (logtime = now())
- -> Bitmap Heap Scan on p\_201211 p (cost=2.31..9.79 rows=8 width=20)
  - Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201211\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
  - Index Cond: (logtime = now())
- -> Bitmap Heap Scan on p\_201212 p (cost=2.31..9.79 rows=8 width=20)
  - Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_201212\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)

# Partition

- Index Cond: (logtime = now())
  - > Bitmap Heap Scan on p\_default p (cost=2.31..9.79 rows=8 width=20)
- Recheck Cond: (logtime = now())
  - > Bitmap Index Scan on p\_default\_logtime\_idx (cost=0.00..2.31 rows=8 width=0)
- Index Cond: (logtime = now())
- 更改函数稳定性
  - postgres=# ALTER FUNCTION now() IMMUTABLE;
  - ALTER FUNCTION
  - postgres=# EXPLAIN SELECT \* FROM p WHERE logtime=now();
  - 同上
- postgres=# ALTER FUNCTION now() VOLATILE;
  - ALTER FUNCTION
  - postgres=# EXPLAIN SELECT \* FROM p WHERE logtime=now();
    - QUERY PLAN
- -----
- Result (cost=0.00..447.85 rows=105 width=20)

# Partition

- -> Append (cost=0.00..447.85 rows=105 width=20)
- -> Seq Scan on p (cost=0.00..0.00 rows=1 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201201 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201202 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201203 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201204 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201205 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201206 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201207 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- -> Seq Scan on p\_201208 p (cost=0.00..34.45 rows=8 width=20)

# Partition

- Filter: (logtime = now())
  - -> Seq Scan on p\_201209 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
  - -> Seq Scan on p\_201210 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
  - -> Seq Scan on p\_201211 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
  - -> Seq Scan on p\_201212 p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
  - -> Seq Scan on p\_default p (cost=0.00..34.45 rows=8 width=20)
  - Filter: (logtime = now())
- (30 rows)
  
- 函数稳定性
- Thinking PostgreSQL Function's Volatility Categories
- <http://blog.163.com/digoal@126/blog/static/163877040201151011105494/>



# Partition

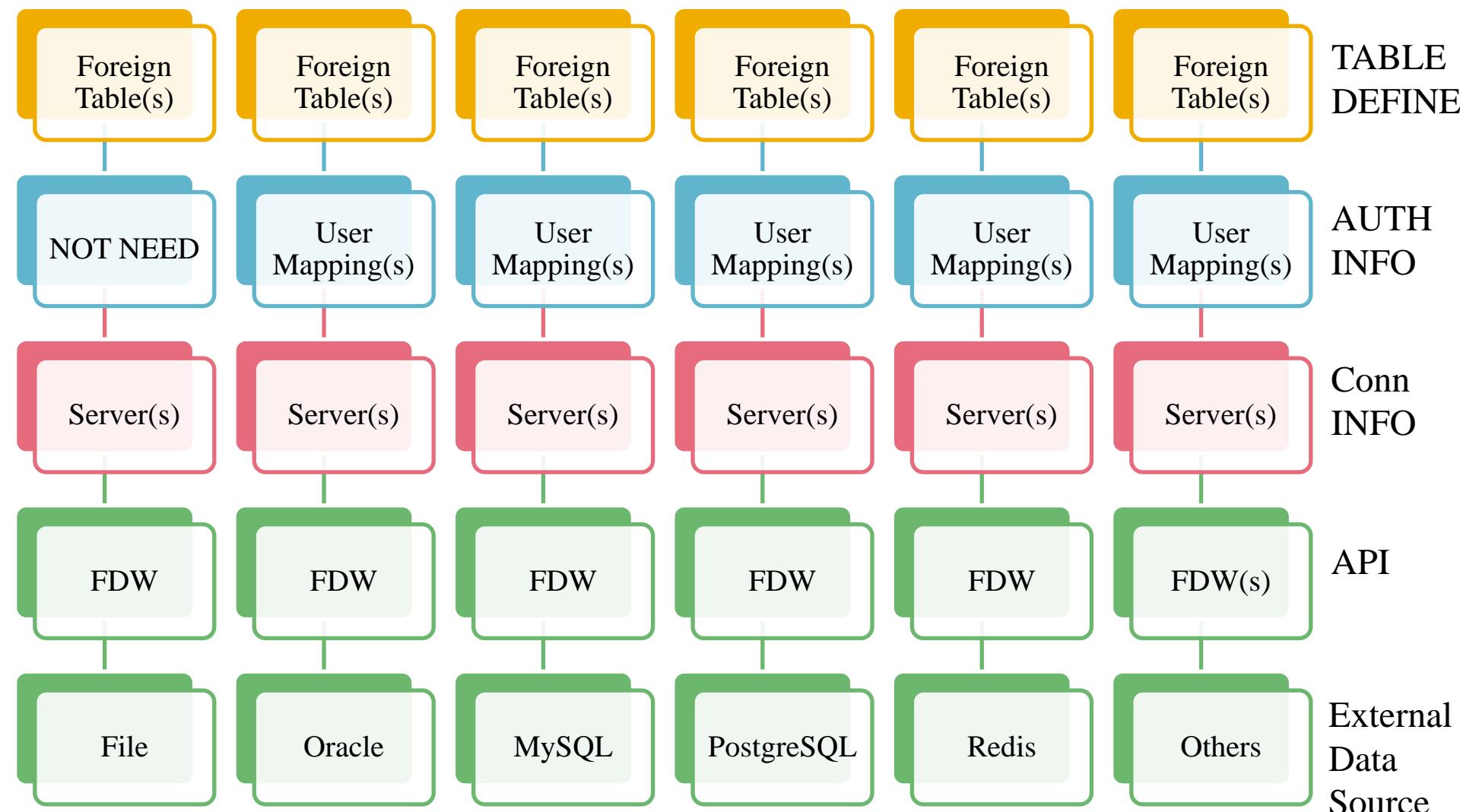
- 分区表使用注意事项
- 尽量将分区功能移至应用端代码中.
- constraint\_exclusion = partition
- WHERE条件带有分区字段作为过滤条件时, 传入的参数必须使用constant才能获得良好的执行计划
- 简化分区规则, 分区字段上使用简单的b-tree索引, 尽量避免函数索引.
- 使用数据库分区的潜在问题
  - CPU开销(触发器或rule, 硬解析)
- PostgreSQL partition table's arithmetic tuning example
- <http://blog.163.com/digoal@126/blog/static/1638770402011210114036419/>



# Foreign Data

- Foreign data wrapper
- A foreign data wrapper is a library that can communicate with an external data source, hiding the details of connecting to the data source and fetching data from it.
- There is a foreign data wrapper available as a contrib module, which can read plain data files residing on the server.
- Other kind of foreign data wrappers might be found as third party products.

# Foreign Data



# Foreign Data

- PostgreSQL Foreign Table - pgsql\_fdw
- <http://blog.163.com/digoal@126/blog/static/163877040201231514057303/>
- PostgreSQL Foreign Table - oracle\_fdw 1
- <http://blog.163.com/digoal@126/blog/static/163877040201181505331588/>
- PostgreSQL Foreign Table - oracle\_fdw 2
- <http://blog.163.com/digoal@126/blog/static/16387704020118151162340/>
- PostgreSQL Foreign Table - oracle\_fdw 3
- <http://blog.163.com/digoal@126/blog/static/16387704020118951953408/>
- PostgreSQL Foreign Table - file\_fdw
- <http://blog.163.com/digoal@126/blog/static/163877040201141641148311/>
- PostgreSQL Foreign Table - redis\_fdw
- <http://blog.163.com/digoal@126/blog/static/16387704020119181188247/>
- PostgreSQL Foreign Table - mysql\_fdw 1
- <http://blog.163.com/digoal@126/blog/static/1638770402011111233524987/>
- PostgreSQL Foreign Table - mysql\_fdw 2
- <http://blog.163.com/digoal@126/blog/static/16387704020121108551698/>



# DML

- INSERT
- UPDATE
- DELETE
  
- 一个事务最大 $2^{32}$ 条SQL(因为cmin,cmax的长度是4Bytes)
- PostgreSQL一个事务中可以包含DML, DDL, DCL.
  - 除了以下
  - `create tablespace`
  - `create database`
  - 使用`concurrently`并行创建索引
  - 其他未尽情况略
- (Oracle执行DDL前自动将前面的未提交的事务提交,所以Oracle不支持在事务中执行DDL语句)

# Query

- JOIN
- ALIAS
- Table as Function's Return data type
- GROUP BY [ HAVING ]
- DISTINCT
- COMBINING QUERY
- SORT
- LIMIT [ OFFSET ]
- WITH

# JOIN

- T1 CROSS JOIN T2 ( T1 INNER JOIN T2 ON TRUE )
- T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 ON boolean\_expression
- T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 USING ( join column list )
- T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2
  
- The words INNER and OUTER are optional in all forms. INNER is the default; LEFT, RIGHT, and FULL imply an outer join.

num	name
1	a
2	b
3	c

num	value
1	xxx
3	yyy
5	zzz

# JOIN

## ■ CROSS JOIN 产生笛卡尔积

```
=> SELECT * FROM t1 CROSS JOIN t2;
+-----+-----+-----+
| num | name | num | value |
+-----+-----+-----+
| 1   | a    | 1   | xxx  |
| 1   | a    | 3   | yyy  |
| 1   | a    | 5   | zzz  |
| 2   | b    | 1   | xxx  |
| 2   | b    | 3   | yyy  |
| 2   | b    | 5   | zzz  |
| 3   | c    | 1   | xxx  |
| 3   | c    | 3   | yyy  |
| 3   | c    | 5   | zzz  |
(9 rows)
```

## ■ 因为关联的两个表有两列同名同类型同长度, 所以下三种写法等同

```
=> SELECT * FROM t1 INNER JOIN t2 ON t1.num = t2.num;
+-----+-----+-----+
| num | name | num | value |
+-----+-----+-----+
| 1   | a    | 1   | xxx  |
| 3   | c    | 3   | yyy  |
(2 rows)

=> SELECT * FROM t1 INNER JOIN t2 USING (num);
+-----+-----+
| num | name | value |
+-----+-----+
| 1   | a    | xxx  |
| 3   | c    | yyy  |
(2 rows)

=> SELECT * FROM t1 NATURAL INNER JOIN t2;
+-----+-----+
| num | name | value |
+-----+-----+
| 1   | a    | xxx  |
| 3   | c    | yyy  |
(2 rows)
```

# JOIN

- 左或右连接, 不满足条件的右或左表的值
- 全关联, 不满足条件的值都置空

```
=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num;
+-----+
| num | name | num | value |
+-----+
| 1   | a    | 1   | xxx  |
| 2   | b    |     |       |
| 3   | c    | 3   | yyy  |
+-----+
(3 rows)

=> SELECT * FROM t1 LEFT JOIN t2 USING (num);
+-----+
| num | name | value |
+-----+
| 1   | a    | xxxx |
| 2   | b    |       |
| 3   | c    | yyyy |
+-----+
(3 rows)

=> SELECT * FROM t1 RIGHT JOIN t2 ON t1.num = t2.num;
+-----+
| num | name | num | value |
+-----+
| 1   | a    | 1   | xxx  |
| 3   | c    | 3   | yyy  |
|     |       | 5   | zzz  |
+-----+
(3 rows)
```

```
=> SELECT * FROM t1 FULL JOIN t2 ON t1.num = t2.num;
+-----+
| num | name | num | value |
+-----+
| 1   | a    | 1   | xxx  |
| 2   | b    |     |       |
| 3   | c    | 3   | yyy  |
|     |       | 5   | zzz  |
+-----+
(4 rows)
```

# ALIAS

- table alias:
  - FROM table\_reference AS alias
  - FROM table\_reference alias
  
- column alias:
  - SELECT expression [ [ AS ] output\_name ]
  
- subquery alias:
  - FROM (SELECT \* FROM table1) AS alias\_name

# Table as Function's Return data type

## ■ return table's row type

- create table t1 (id int, name text, crt\_time timestamp());
- create or replace function f\_t1 (i\_id int) returns setof t1 as \$\$
- declare
- begin
- return query select \* from t1 where id=i\_id;
- return;
- end;
- \$\$ language plpgsql;
- insert into t1 values(1,'digoal',now());
- insert into t1 values(1,'DIGOAL',now());
- select \* from f\_t1(1);
- id | name | crt\_time
- -----+-----+-----
- 1 | digoal | 2012-04-26 08:15:09
- 1 | DIGOAL | 2012-04-26 08:15:15

# Table as Function's Return data type

## ■ return composite type

- create type type1 as (id int, name text, crt\_time timestamp(0));
  - create or replace function f\_type1 (i\_id int) returns setof type1 as \$\$
  - declare
  - begin
  - return query select \* from t1 where id=i\_id;
  - return;
  - end;
  - \$\$ language plpgsql;
- 
- select \* from f\_type1(1);
  - id | name | crt\_time
  - -----+-----+-----
  - 1 | digoal | 2012-04-26 08:15:09
  - 1 | DIGOAL | 2012-04-26 08:15:15

# Table as Function's Return data type

## ■ return record

- create or replace function f\_record1 (i\_id int) returns setof record as \$\$
- declare
- begin
- return query select \* from t1 where id=i\_id;
- return;
- end;
- \$\$ language plpgsql;
  
- select \* from f\_record1(1) as (id int,name text,crt\_time timestamp(o));
- id | name | crt\_time
- -----+-----+-----
- 1 | digoal | 2012-04-26 08:15:09
- 1 | DIGOAL | 2012-04-26 08:15:15

# DISTINCT

- SELECT DISTINCT select\_list ... (NULL在DISTINCT [ON] 中视为相等)
  - postgres=# select \* from t1 ;
  - id | name | crt\_time
  - 1 | digoal | 2012-04-26 08:15:09
  - 1 | DIGOAL | 2012-04-26 08:15:15
  
  - postgres=# select distinct id from t1;
  - id
  - 1
  
- SELECT DISTINCT ON (expression [, expression ...]) select\_list ...
  - Here expression is an arbitrary value expression that is evaluated for all rows. A set of rows for which all the expressions are equal are considered duplicates, and only the first row of the set is kept in the output. Note that the "first row" of a set is unpredictable unless the query is sorted on enough columns to guarantee a unique ordering of the rows arriving at the DISTINCT filter. (DISTINCT ON processing occurs after ORDER BY sorting.)

# DISTINCT

- postgres=# select distinct on (id) id,name,crt\_time from t1 ;  
■ id | name | crt\_time  
■ -----+-----+-----  
■ 1 | digoal | 2012-04-26 08:15:09
  
- postgres=# select distinct on (id) id,name,crt\_time from t1 order by crt\_time;  
■ ERROR: SELECT DISTINCT ON expressions must match initial ORDER BY expressions  
■ LINE 1: select distinct on (id) id,name,crt\_time from t1 order by cr...  
■ ^
  
- postgres=# select distinct on (id) id,name,crt\_time from t1 order by id;  
■ id | name | crt\_time  
■ -----+-----+-----  
■ 1 | digoal | 2012-04-26 08:15:09

# DISTINCT

- postgres=# select distinct on (id) id,name,crt\_time from t1 order by id,crt\_time;
- id | name | crt\_time
- -----+-----+-----
- 1 | digoal | 2012-04-26 08:15:09
  
- postgres=# select distinct on (id) id,name,crt\_time from t1 order by id,crt\_time desc;
- id | name | crt\_time
- -----+-----+-----
- 1 | DIGOAL | 2012-04-26 08:15:15

# DISTINCT

- postgres=# select distinct on (id,name) id,name,crt\_time from t1 order by id,crt\_time desc;
- ERROR: SELECT DISTINCT ON expressions must match initial ORDER BY expressions
- LINE 1: select distinct on (id,name) id,name,crt\_time from t1 order ...
  - ^
- postgres=# select distinct on (id,name) id,name,crt\_time from t1 order by id,name,crt\_time desc;
- id | name | crt\_time
- -----+-----+-----
- 1 | DIGOAL | 2012-04-26 08:15:15
- 1 | digoal | 2012-04-26 08:15:09

# DISTINCT

- 使用DISTINCT ON实现前面章节用窗口函数实现的取第一名的功能
- postgres=# CREATE TABLE window\_test(id int, name text, subject text, score numeric);
- postgres=# INSERT INTO window\_test VALUES (1,'digoal','数学',99.5), (2,'digoal','语文',89.5),  
(3,'digoal','英语',79.5), (4,'digoal','物理',99.5), (5,'digoal','化学',98.5),  
(6,'刘德华','数学',89.5), (7,'刘德华','语文',99.5), (8,'刘德华','英语',79.5),  
(9,'刘德华','物理',89.5), (10,'刘德华','化学',69.5),  
(11,'张学友','数学',89.5), (12,'张学友','语文',91.5), (13,'张学友','英语',92.5),  
(14,'张学友','物理',93.5), (15,'张学友','化学',94.5);
- -- 取出每门课程的第一名.
- postgres=# select distinct on (subject) id,name,subject,score from window\_test order by subject, score desc;

# DISTINCT

- id | name | subject | score
- -----+-----+-----
- 5 | digoal | 化学 | 98.5
- 1 | digoal | 数学 | 99.5
- 4 | digoal | 物理 | 99.5
- 13 | 张学友 | 英语 | 92.5
- 7 | 刘德华 | 语文 | 99.5
- (5 rows)
  
- 与使用窗口函数得到的结果一致，并且写法更简洁.

# NULL IN ORDER BY and DISTINCT

- order by和distinct处理NULLs时视为相等
  - postgres=# select 1 where null is null;
  - ?column?
  - -----
  - 1
  - (1 row)
  
- postgres=# select 1 where null <> null;
  - ?column?
  - -----
  - (0 rows)
  
- postgres=# select distinct name from t1;
  - name
  - -----
  - (1 row)

# NULL IN ORDER BY and DISTINCT

- NULL视为相等
- postgres=# select \* from t1 order by name,id;
- id | name | crt\_time
- -----+-----+-----
- 1 | | 2012-04-26 09:29:23
- 2 | | 2012-04-26 09:29:26
- 3 | | 2012-04-26 09:29:28
- 4 | | 2012-04-26 09:29:32
- 5 | | 2012-04-26 08:30:04
- 如果NULL视为不相等, 结果应该是无序的.
- postgres=# select \* from t1 order by crt\_time,id;
- id | name | crt\_time
- -----+-----+-----
- 5 | | 2012-04-26 08:30:04
- 1 | | 2012-04-26 09:29:23
- 2 | | 2012-04-26 09:29:26
- 3 | | 2012-04-26 09:29:28
- 4 | | 2012-04-26 09:29:32

# COMBINING QUERY

- query1 UNION [ALL] query2
- query1 INTERSECT [ALL] query2
- query1 EXCEPT [ALL] query2
  
- UNION effectively appends the result of query2 to the result of query1 (although there is no guarantee that this is the order in which the rows are actually returned)
  
- INTERSECT returns all rows that are both in the result of query1 and in the result of query2.
  
- EXCEPT returns all rows that are in the result of query1 but not in the result of query2.
  
- Combining Query eliminates duplicate rows from its result, in the same way as DISTINCT,  
unless ALL is used
  
- query1 and query2 must return the same number of columns and the corresponding columns  
have compatible data types.

# SORT

- SELECT select\_list
- FROM table\_expression
- ORDER BY sort\_expression1 [ASC | DESC] [NULLS { FIRST | LAST }]
- [ , sort\_expression2 [ASC | DESC] [NULLS { FIRST | LAST }] ... ]

# LIMIT [ OFFSET ]

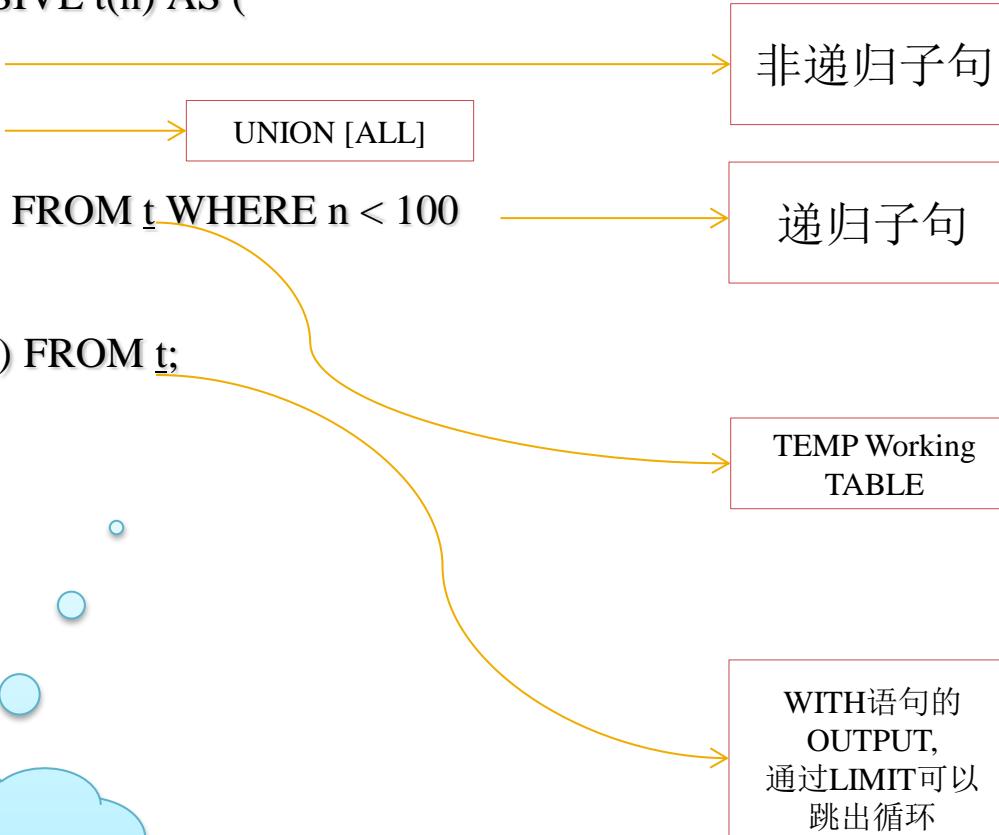
- `SELECT select_list`
- `FROM table_expression`
- `[ ORDER BY ... ]`
- `[ LIMIT { number | ALL } ] [ OFFSET number ]`

# WITH(Common Table Expressions)

- WITH regional\_sales AS (
- SELECT region, SUM(amount) AS total\_sales
- FROM orders
- GROUP BY region
- ), top\_regions AS (
- SELECT region
- FROM regional\_sales
- WHERE total\_sales > (SELECT SUM(total\_sales)/10 FROM regional\_sales)
- )
- SELECT region,
- product,
- SUM(quantity) AS product\_units,
- SUM(amount) AS product\_sales
- FROM orders
- WHERE region IN (SELECT region FROM top\_regions)
- GROUP BY region, product;

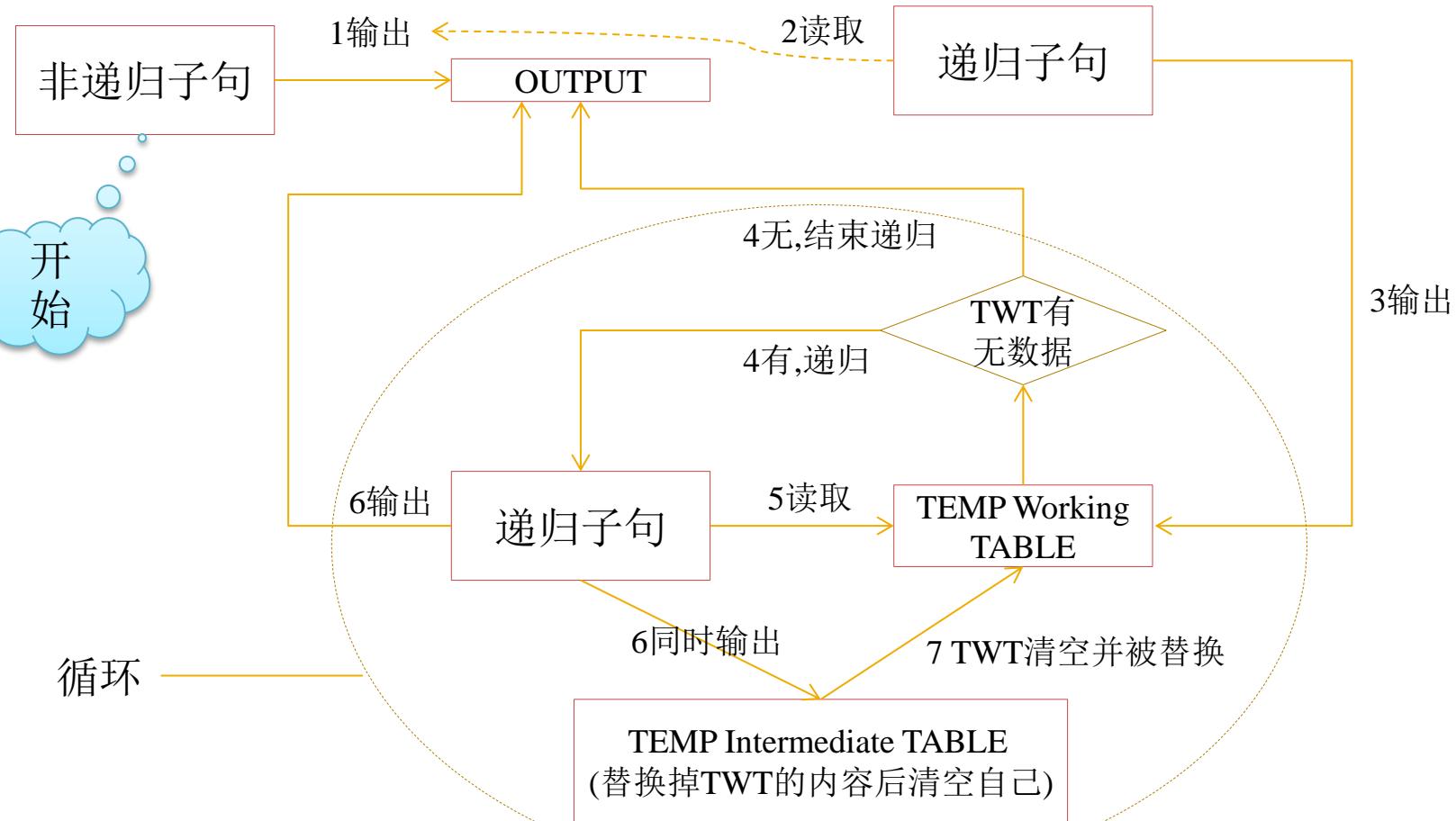
# WITH(Common Table Expressions)

- WITH RECURSIVE t(n) AS (
- VALUES (1)
- UNION ALL
- SELECT n+1 FROM t WHERE n < 100
- )
- SELECT sum(n) FROM t;



# WITH(Common Table Expressions)

- UNION ALL 去重复(去重复时NULL 视为等同)
- 图中所有输出都涉及UNION [ALL]的操作, 包含以往返回的记录和当前返回的记录



# WITH(Common Table Expressions)

- TEMP Working Table 没有ctid, cmin, cmax, xmin, xmax, tableoid字段
  
- postgres=# create table test (id int,name text);
- postgres=# insert into test values (1,'digoal1'),(2,'digoal2');
- postgres=# begin;
- postgres=# with t1 as (update test set name='DIGOAL2' where id=2 returning \*)
- select ctid from t1;
- ERROR: column "ctid" does not exist
- LINE 2: select ctid from t1;
  - ^
- postgres=# rollback;
  
- 其他字段(cmin,cmax,xmin,xmax,tableoid)同样错误

# WITH(Common Table Expressions)

- 递归查询输出产品部以及该部门的所有子部门信息.
- 然后输出各个子部门以及各个子部门的人数
  
- WITH RECURSIVE included\_parts(sub\_part, part, quantity) AS (
  - SELECT sub\_part, part, quantity FROM parts WHERE part = '产品部' .
  - UNION ALL
  - SELECT p.sub\_part, p.part, p.quantity
  - FROM included\_parts pr, parts p
  - WHERE p.part = pr.sub\_part
  - )
- SELECT sub\_part, SUM(quantity) as total\_quantity
- FROM included\_parts
- GROUP BY sub\_part



第一步时读取的是初始输出,  
后面都是TEMP Working  
TABLE

# WITH(Common Table Expressions)

- 死循环
- WITH RECURSIVE search\_graph(id, link, data, depth) AS (
  - SELECT g.id, g.link, g.data, 1
  - FROM graph g
  - UNION ALL
  - SELECT g.id, g.link, g.data, sg.depth + 1
  - FROM graph g, search\_graph sg
  - WHERE g.id = sg.link
  - )
  - SELECT \* FROM search\_graph;

每次递归输出的记录与以往的记录都不一样，  
TEMP Working Table 永远都有记录，  
因此无限循环。

# WITH(Common Table Expressions)

- 规避上一个死循环的方法
- 让递归SQL有机会没记录输出
- WITH RECURSIVE search\_graph(id, link, data, depth, path, cycle) AS (
  - SELECT g.id, g.link, g.data, 1,
  - ARRAY[g.id],
  - false)
- FROM graph g
- UNION ALL
- SELECT g.id, g.link, g.data, sg.depth + 1,
  - path || g.id,
  - g.id = ANY(path)
- FROM graph g, search\_graph sg
- WHERE g.id = sg.link AND NOT cycle
- )
- SELECT \* FROM search\_graph;

# WITH(Common Table Expressions)

- 多值比较需使用ROW类型的ARRAY.
- WITH RECURSIVE search\_graph(id, link, data, depth, path, cycle) AS (
  - SELECT g.id, g.link, g.data, 1,
  - ARRAY[ROW(g.f1, g.f2)],
  - false
  - FROM graph g
  - UNION ALL
  - SELECT g.id, g.link, g.data, sg.depth + 1,
  - path || ROW(g.f1, g.f2),
  - ROW(g.f1, g.f2) = ANY(path)
  - FROM graph g, search\_graph sg
  - WHERE g.id = sg.link AND NOT cycle
  - )
  - SELECT \* FROM search\_graph;

# WITH(Common Table Expressions)

- 还有什么情况可以跳出循环
- WITH RECURSIVE t(n) AS (
  - SELECT 1
  - UNION ALL
  - SELECT n+1 FROM t
  - )
- SELECT n FROM t LIMIT 100;
- 注意如果t表在外围被join了然后再limit的. 还死循环
- 使用递归查询注意防止死循环

# WITH(Common Table Expressions)

- 把属于产品部以及它的子部门的记录删除.
- WITH RECURSIVE included\_parts(sub\_part, part) AS (
  - SELECT sub\_part, part FROM parts WHERE part = '产品部'
  - UNION ALL
  - SELECT p.sub\_part, p.part
  - FROM included\_parts pr, parts p
  - WHERE p.part = pr.sub\_part
  - )
- DELETE FROM parts
- WHERE part IN (SELECT part FROM included\_parts);

# WITH(Common Table Expressions)

- WITH的所有子句包括MAIN子句查看到的是一个SNAPSHOT.
- 各个子句对记录的变更相互看不到, 如果要看到变更的数据需使用RETURNING子句.
  
- WITH t AS (
  - UPDATE products SET price = price \* 1.05 WHERE id = 10
  - RETURNING \*
  - )
  - SELECT \* FROM products WHERE id = 10;
  
- WITH t AS (
  - UPDATE products SET price = price \* 1.05 WHERE id = 10
  - RETURNING \*
  - )
  - SELECT \* FROM t;

# WITH(Common Table Expressions)

- 测试表
- postgres=# create table test (id int,name text);

■ CREATE TABLE

- postgres=# insert into test values(1,'digoal1'),(2,'digoal2');

这样会看到老数据

- postgres=# with t1 as (update test set name='NEW' where id=2 returning \*)
- postgres-# select \* from test where name='NEW';
- id | name
- (0 rows)

这样才能看到新数据

- postgres=# with t1 as (update test set name='NEWNEW' where id=2 returning \*)
- postgres-# select \* from t1 where name='NEWNEW';
- id | name
- 2 | NEWNEW
- (1 row)

# WITH(Common Table Expressions)

- 避免WITH子句包括MAIN子句修改同一条记录. 因为先执行哪条是不可预知的.
- 避免类似SQL:
- postgres=# create table test (id int,name text);
- postgres=# insert into test values (1,'digoal1'),(2,'digoal2');
- postgres=# with t1 as (delete from test where id=1)
- postgres-# update test set name='DIGOAL1' where id=1;
- UPDATE 1
  
- postgres=# select \* from test where id=1;
- id | name
- -----+-----
- 1 | DIGOAL1

# WITH(Common Table Expressions)

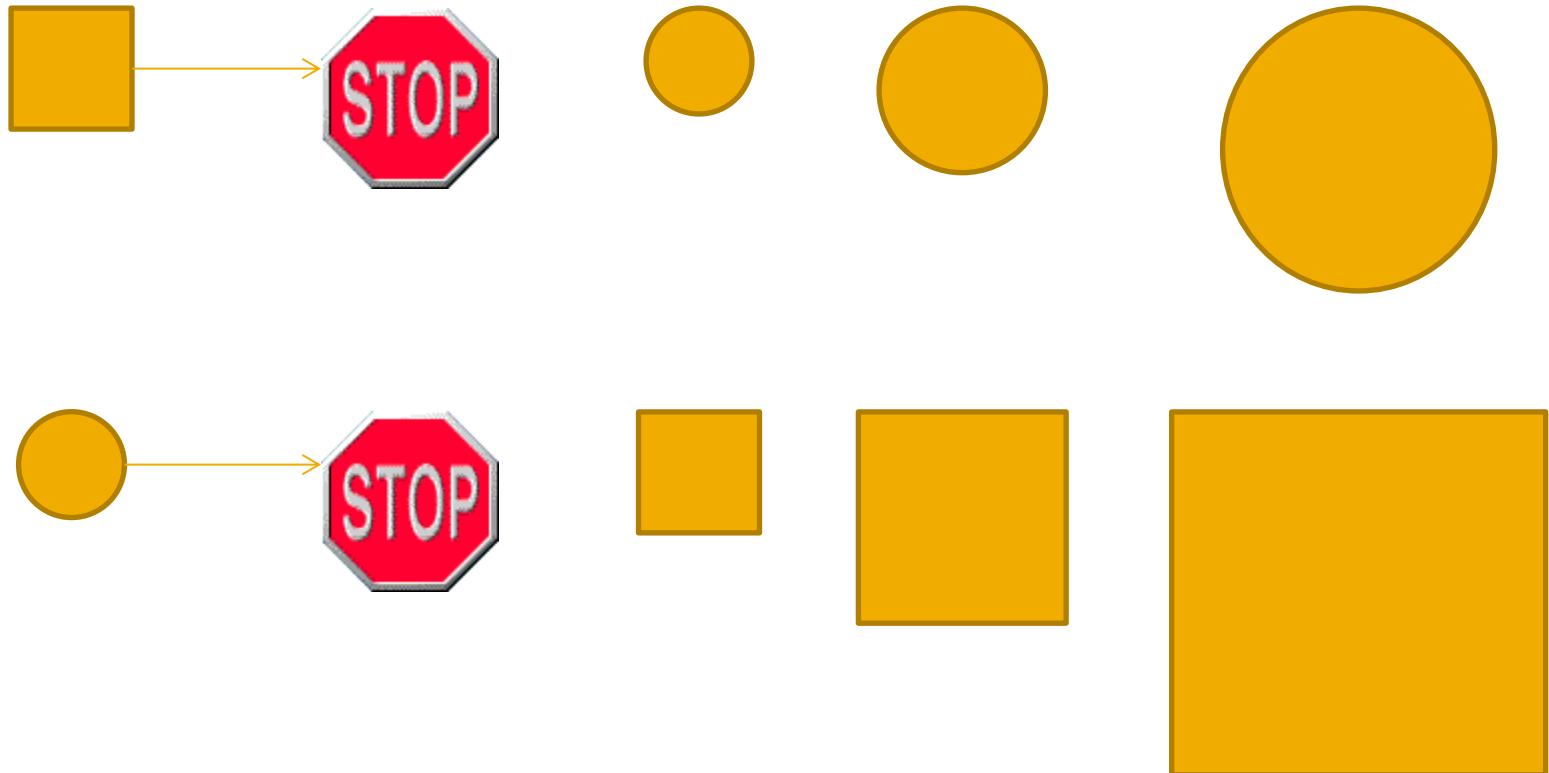
- 避免类似SQL:
  - postgres=# with t1 as (update test set name='digoal1' where id=1)
  - postgres=# delete from test where id=1;
  - DELETE 1
  - postgres=# select \* from test where id=1;
    - id | name
    - (0 rows)

这段测试和手册不符,  
手册上表示update和delete子句同时针对一条记录操作时, delete子句不会执行.

- 避免类似SQL:
  - postgres=# with t1 as (update test set name='DIGOAL2' where id=2)
  - postgres=# update test set name='NEW' WHERE id=2;
  - UPDATE 1
  - postgres=# select \* from test where id=2;
    - id | name
    - 2 | NEW

# Data Type

- 强类型



# Data Type

Table 45-49. typcategory Codes

Code	Category
A	Array types
B	Boolean types
C	Composite types
D	Date/time types
E	Enum types
G	Geometric types
I	Network address types
N	Numeric types
P	Pseudo-types
S	String types
T	Timespan types
U	User-defined types
V	Bit-string types
X	unknown type

Column	Type	Modifiers
typname	name	not null
typnamespace	oid	not null
typowner	oid	not null
typlen	smallint	not null
typbyval	boolean	not null
typtype	"char"	not null
typcategory	"char"	not null
typispreferred	boolean	not null
typisdefined	boolean	not null
typdelim	"char"	not null
typrelid	oid	not null
typelem	oid	not null
typarray	oid	not null
typinput	regproc	not null
typoutput	regproc	not null
typreceive	regproc	not null
typsend	regproc	not null
typmodin	regproc	not null
typmodout	regproc	not null
typanalyze	regproc	not null
typalign	"char"	not null
typstorage	"char"	not null
typnotnull	boolean	not null
typbasetype	oid	not null
typtypmod	integer	not null
typndims	integer	not null
typcollation	oid	not null
typdefaultbin	pg_node_tree	
typdefault	text	

- p:
- e:
- m:
- x:

存储方法

# Data Type

## ■ 常用数据类型, 数字

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
decimal / numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

# Data Type

## ■ 常用数据类型, 字符

Name	Storage Size	Description
character varying(n), varchar(n)	variable(can store n chars)	variable-length with limit
character(n), char(n)	n chars	fixed-length, blank padded
text	variable	variable unlimited length
"char"	1 byte	single-byte internal type
name	64 bytes	internal type for object names

- postgres=# create table char\_test (c1 char(1),c2 "char");
- postgres=# insert into char\_test values('a','a'), ('数','数');
- postgres=# select \*,octet\_length(c1),octet\_length(c2) from char\_test ;

■ c1 | c2 | octet\_length | octet\_length

■ -----+-----+-----+

a   a	1	1
数	3	1

# Data Type

## ■ 常用数据类型, 时间

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [ (p) ] [ without time zone ]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond / 14 digits
timestamp [ (p) ] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond / 14 digits
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [ (p) ] [ without time zone ]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond / 14 digits
time [ (p) ] with time zone	12 bytes	times of day only, with time zone	00:00:00+14:59	24:00:00-1459	1 microsecond / 14 digits
interval [ fields ] [ (p) ]	12 bytes	time interval	-178000000 years	178000000 years	1 microsecond / 14 digits

# Data Type

- 常用数据类型, 时间
- 特殊日期/时间输入

<b>Input String</b>	<b>Valid Types</b>	<b>Description</b>
epoch	date, timestamp	1970-01-01 00:00:00+00 (Unix system time zero)
infinity	date, timestamp	later than all other time stamps
-infinity	date, timestamp	earlier than all other time stamps
now	date, time, timestamp	current transaction's start time
today	date, timestamp	midnight today
tomorrow	date, timestamp	midnight tomorrow
yesterday	date, timestamp	midnight yesterday
allballs	time	00:00:00.00 UTC

- postgres=# select timestamp 'epoch', date 'infinity', time 'now', date 'today', time 'allballs';
- timestamp    |   date    |    time    |   date    |   time
- -----+-----+-----+-----+-----
- 1970-01-01 00:00:00 | infinity | 15:14:13.461166 | 2012-04-27 | 00:00:00

# Data Type

- 常用数据类型, 时间
- 时间输入输出格式

Style Specification	Description	Example
ISO	ISO 8601/SQL standard	1997-12-17 07:37:16-08
SQL	traditional style	12/17/1997 07:37:16.00 PST
POSTGRES	original style	Wed Dec 17 07:37:16 1997 PST
German	regional style	17.12.1997 07:37:16.00 PST

datestyle Setting	Input Ordering	Example Output
SQL, DMY	day/month/year	17/12/1997 15:37:16.00 CET
SQL, MDY	month/day/year	12/17/1997 07:37:16.00 PST
Postgres, DMY	day/month/year	Wed 17 Dec 07:37:16 1997 PST

- postgres=# set datestyle='SQL,DMY';
- postgres=# select now();
- 27/04/2012 15:49:51.373789 CST
- postgres=# set datestyle='SQL,MDY';
- postgres=# select now();
- 04/27/2012 15:50:07.882063 CST

# Data Type

- 常用数据类型, 时间
- 时间间隔interval 格式
- [ @ ] quantity unit [ quantity unit... ] [ direction ]
- P quantity unit [ quantity unit ... ] [ T [ quantity unit ... ] ]
- P [ years-months-days ] [ T hours:minutes:seconds ]
- IntervalStyle样式

Abbreviation	Meaning
Y	Years
M	Months (in the date part)
W	Weeks
D	Days
H	Hours
M	Minutes (in the time part)
S	Seconds

Style Specification	Year-Month Interval	Day-Time Interval	Mixed Interval
sql_standard	1-2	3 4:05:06	-1-2 +3 -4:05:06
postgres	1 year 2 mons	3 days 04:05:06	-1 year -2 mons +3 days -04:05:06
postgres_verbose	@ 1 year 2 mons	@ 3 days 4 hours 5 mins 6 secs	@ 1 year 2 mons -3 days 4 hours 5 mins 6 secs ago
iso_8601	P1Y2M	P3DT4H5M6S	P-1Y-2M3DT-4H-5M-6S

- postgres=# show IntervalStyle ;
- postgres
- postgres=# select interval 'P-1Y-2M3DT-4H-5M-6S';
- -1 years -2 mons +3 days -04:05:06
- postgres=# select interval '1 day ago';
- -1 days
- postgres=# set IntervalStyle ='sql\_standard';
- postgres=# select interval 'P-1Y-2M3DT-4H-5M-6S';
- -1-2 +3 -4:05:06

# Data Type

## ■ 常用数据类型, 布尔

Name	Storage Size	Description
boolean	1 byte	state of true or false

- 真
- TRUE 't' 'true' 'y' 'yes' 'on' '1'
  
- 假
- FALSE 'f' 'false' 'n' 'no' 'off' '0'
  
- unknown
- NULL

# Data Type

- 常用数据类型, 枚举
- CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
- CREATE TABLE person (name text, current\_mood mood);
- INSERT INTO person VALUES ('Moe', 'happy');
- SELECT \* FROM person WHERE current\_mood = 'happy';
- name | current\_mood
- Moe | happy
- (1 row)
- -- 输入一个不存在的枚举值, 将报错
- postgres=# SELECT \* FROM person WHERE current\_mood = 'happ';
- ERROR: invalid input value for enum mood: "happ"
- -- 避免报错的方法, 把枚举转换成text
- postgres=# SELECT \* FROM person WHERE current\_mood::text = 'happ';
- name | current\_mood
- -----+-----
- (0 rows)

# Data Type

- 枚举值每一个在行中占用4 bytes :
- postgres=# select current\_mood,pg\_column\_size(current\_mood) from person;
- current\_mood | pg\_column\_size
- -----+-----
- happy | 4
  
- 枚举的标签在定义中最大限制由NAMEDATALEN决定, 默认是64-1. 前面已经讲过.
- 查找枚举的数据结构 :
- postgres=# select oid,typename from pg\_type where typename='mood';
- oid | typename
- -----+-----
- 3952969 | mood
- postgres=# select \* from pg\_enum where enumtypid=3952969;
- enumtypid | enumsortorder | enumlabel
- -----+-----+-----
- 3952969 | 1 | sad
- 3952969 | 2 | ok
- 3952969 | 3 | happy

# Data Type

- 枚举类型变更
- ALTER TYPE name ADD VALUE new\_enum\_value [ { BEFORE | AFTER } existing\_enum\_value ]
  - This form adds a new value to an enum type. If the new value's place in the enum's ordering is not specified using BEFORE or AFTER, then the new item is placed at the end of the list of values.
- 注意事项, 添加枚举元素时尽量不要改动原来的元素的位置, 即尽量新增值插到最后.
- 否则可能会带来性能问题.
- ALTER TYPE ... ADD VALUE (the form that adds a new value to an enum type) cannot be executed inside a transaction block.
- Comparisons involving an added enum value will sometimes be slower than comparisons involving only original members of the enum type. This will usually only occur if BEFORE or AFTER is used to set the new value's sort position somewhere other than at the end of the list. However, sometimes it will happen even though the new value is added at the end (this occurs if the OID counter "wrapped around" since the original creation of the enum type). The slowdown is usually insignificant; but if it matters, optimal performance can be regained by dropping and recreating the enum type, or by dumping and reloading the database.

# Data Type

- money类型
- 显示和客户端参数lc\_monetary有关

Name	Storage Size	Description	Range
money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

- postgres=# show lc\_monetary;
- C
- postgres=# SELECT '12.345'::money;
- \$12.35
- postgres=# set lc\_monetary='zh\_CN';
- postgres=# SELECT '12.345'::money;
- ¥ 12.35

# Data Type

## ■ bytea类型

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

- The bytea data type allows storage of binary strings
- A binary string is a sequence of octets (or bytes)
- bytea与字符类型的区别
  - binary strings specifically allow storing octets of value zero and other "non-printable" octets.
  - Character strings disallow zero octets, and also disallow any other octet values and sequences of octet values that are invalid according to the database's selected character set encoding.
  - Second, operations on binary strings process the actual bytes, whereas the processing of character strings depends on locale settings. In short, binary strings are appropriate for storing data that the programmer thinks of as "raw bytes", whereas character strings are appropriate for storing text.

# Data Type

- bytea类型
- 同时支持两种格式输入
  - escape
    - select E'\\336\\255\\276\\357'::bytea;
  - hex, 每两个16进制数字为一组, 表示一个"raw byte"
    - SELECT E'\\x DE AD BE EF'::bytea;
- 支持两种格式输出, 需配置
  - 9.0引入hex输出(通过配置bytea\_output)
  - 9.0以前为escape输出
  - 如果有从老版本数据库迁移到9.0及以后版本的情况, 需要注意, 可能再次与程序不兼容, 只需要将默认值调整为escape即可.
- 推荐使用hex格式输入输出

# Data Type

## ■ 几何类型

Name	Storage Size	Representation	Description
point	16 bytes	Point on a plane	$(x,y)$
line	32 bytes	Infinite line (not fully implemented)	$((x_1,y_1),(x_2,y_2))$
lseg	32 bytes	Finite line segment	$((x_1,y_1),(x_2,y_2))$
box	32 bytes	Rectangular box	$((x_1,y_1),(x_2,y_2))$
path	$16+16n$ bytes	Closed path (similar to polygon)	$((x_1,y_1),\dots)$
path	$16+16n$ bytes	Open path	$[(x_1,y_1),\dots]$
polygon	$40+16n$ bytes	Polygon (similar to closed path)	$((x_1,y_1),\dots)$
circle	24 bytes	Circle	$\langle(x,y),r\rangle$ (center point and radius)

# Data Type

## ■ Network Address Types

Name	Storage Size	Description
cidr	7 or 19 bytes	IPv4 and IPv6 networks
inet	7 or 19 bytes	IPv4 and IPv6 hosts and networks
macaddr	6 bytes	MAC addresses

cidr Input	cidr Output	abbrev(cidr)
192.168.100.128/25	192.168.100.128/25	192.168.100.128/25
192.168/24	192.168.0.0/24	192.168.0/24
192.168/25	192.168.0.0/25	192.168.0.0/25
192.168.1	192.168.1.0/24	192.168.1/24
192.168	192.168.0.0/24	192.168.0/24
128.1	128.1.0.0/16	128.1/16
128	128.0.0.0/16	128.0/16
128.1.2	128.1.2.0/24	128.1.2/24
10.1.2	10.1.2.0/24	10.1.2/24
10.1	10.1.0.0/16	10.1/16
10	10.0.0.0/8	10/8
10.1.2.3/32	10.1.2.3/32	10.1.2.3/32
2001:4f8:3:ba::/64	2001:4f8:3:ba::/64	2001:4f8:3:ba::/64
2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128	2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128	2001:4f8:3:ba:2e0:81ff:fe22:d1f1
::ffff:1.2.3.0/120	::ffff:1.2.3.0/120	::ffff:1.2.3/120
::ffff:1.2.3.0/128	::ffff:1.2.3.0/128	::ffff:1.2.3.0/128

# Data Type

- 网段填充：
- Table "digoal.tbl\_ip\_info"
- | Column   | Type                  | Modifiers |
|----------|-----------------------|-----------|
| id       | integer               |           |
| province | character varying(10) | 省份        |
| start_ip | inet                  | 开始IP      |
| end_ip   | inet                  | 结束IP      |
- ```
digoal=> insert into tbl_ip_info values (1,'浙江','192.168.1.254','192.168.2.5');
```
- ```
digoal=> insert into tbl_ip_info values (2,'广东','192.168.2.254','192.168.3.5');
```
- ```
digoal=> insert into tbl_ip_info values (3,'湖南','192.168.3.254','192.168.4.5');
```

# Data Type

- digoal=> select id,generate\_series(0,end\_ip-start\_ip)+start\_ip from tbl\_ip\_info ;
  - 2 | 192.168.3.0
  - 2 | 192.168.3.1
  - 2 | 192.168.3.2
  - 2 | 192.168.3.3
  - 2 | 192.168.3.4
  - 2 | 192.168.3.5
  - 3 | 192.168.3.254
  - 3 | 192.168.3.255
  - 3 | 192.168.4.0
  - 3 | 192.168.4.1
  - 3 | 192.168.4.2
  - 3 | 192.168.4.3
  - 3 | 192.168.4.4
  - 3 | 192.168.4.5
  - (24 rows)
- id | ?column?
- -----+-----
- 1 | 192.168.1.254
- 1 | 192.168.1.255
- 1 | 192.168.2.0
- 1 | 192.168.2.1
- 1 | 192.168.2.2
- 1 | 192.168.2.3
- 1 | 192.168.2.4
- 1 | 192.168.2.5
- 2 | 192.168.2.254
- 2 | 192.168.2.255

# Data Type

- Bit String Type
- Bit strings are strings of 1's and 0's. They can be used to store or visualize bit masks. There are two SQL bit types: bit(n) and bit varying(n), where n is a positive integer.
  
- CREATE TABLE test (a BIT(3), b BIT VARYING(5));
- INSERT INTO test VALUES (B'101', B'00');
- INSERT INTO test VALUES (B'10', B'101');
- ERROR: bit string length 2 does not match type bit(3)
  
- INSERT INTO test VALUES (B'10'::bit(3), B'101');
- SELECT \* FROM test;
- a | b
- -----+-----
- 101 | 00
- 100 | 101

# Data Type

- 全文检索类型
- tsvector
  - 去除重复分词后按分词顺序存储
  - 可以存储位置信息和权重信息
- tsquery
  - 存储查询的分词，可存储权重信息

| dictname        | dictnamespace | dictowner | dicttemplate | dictinitoption                                    |
|-----------------|---------------|-----------|--------------|---------------------------------------------------|
| simple          | 11            | 10        | 3727         |                                                   |
| danish_stem     | 11            | 10        | 12144        | language = 'danish', stopwords = 'danish'         |
| dutch_stem      | 11            | 10        | 12144        | language = 'dutch', stopwords = 'dutch'           |
| english_stem    | 11            | 10        | 12144        | language = 'english', stopwords = 'english'       |
| finnish_stem    | 11            | 10        | 12144        | language = 'finnish', stopwords = 'finnish'       |
| french_stem     | 11            | 10        | 12144        | language = 'french', stopwords = 'french'         |
| german_stem     | 11            | 10        | 12144        | language = 'german', stopwords = 'german'         |
| hungarian_stem  | 11            | 10        | 12144        | language = 'hungarian', stopwords = 'hungarian'   |
| italian_stem    | 11            | 10        | 12144        | language = 'italian', stopwords = 'italian'       |
| norwegian_stem  | 11            | 10        | 12144        | language = 'norwegian', stopwords = 'norwegian'   |
| portuguese_stem | 11            | 10        | 12144        | language = 'portuguese', stopwords = 'portuguese' |
| romanian_stem   | 11            | 10        | 12144        | language = 'romanian'                             |
| russian_stem    | 11            | 10        | 12144        | language = 'russian', stopwords = 'russian'       |
| spanish_stem    | 11            | 10        | 12144        | language = 'spanish', stopwords = 'spanish'       |
| swedish_stem    | 11            | 10        | 12144        | language = 'swedish', stopwords = 'swedish'       |
| turkish_stem    | 11            | 10        | 12144        | language = 'turkish', stopwords = 'turkish'       |

# Data Type

## 全文检索类型

```

SELECT $$the lexeme 'Joe''s' contains a quote$$::tsvector;
      tsvector
-----
'Joe''s' 'a' 'contains' 'lexeme' 'quote' 'the'

SELECT 'a:1 fat:2 cat:3 sat:4 on:5 a:6 mat:7 and:8 ate:9 a:10 fat:11 rat:12'::tsvector;
      tsvector
-----
'a':1,6,10 'and':8 'ate':9 'cat':3 'fat':2,11 'mat':7 'on':5 'rat':12 'sat':4

SELECT 'a:1A fat:2B,4C cat:5D'::tsvector;
      tsvector
-----
'a':1A 'cat':5 'fat':2B,4C

SELECT to_tsvector('english', 'The Fat Rats');
      to_tsvector
-----
'fat':2 'rat':3
  
```

```

SELECT 'fat & rat'::tsquery;
      tsquery
-----
'fat' & 'rat'

SELECT 'fat & (rat | cat)'::tsquery;
      tsquery
-----
'fat' & ('rat' | 'cat')

SELECT 'fat & rat & ! cat'::tsquery;
      tsquery
-----
'fat' & 'rat' & !'cat'

SELECT 'fat:ab & cat'::tsquery;
      tsquery
-----
'fat':AB & 'cat'

SELECT to_tsquery('postgres:*');
      to_tsquery
-----
'postgres'*  

(1 row)
  
```

label \* specify prefix matching

```

SELECT to_tsvector('postgraduate') @@ to_tsquery('postgres:*');
      ?column?
-----
t
(1 row)
  
```

# Data Type

- uuid
- UUIDs could be generated by client applications or other libraries invoked through a server-side function.
- specifically a group of 8 digits followed by three groups of 4 digits followed by a group of 12 digits, for a total of 32 digits representing the 128 bits.

输出格式:

```
a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11
```

输入格式:

```
A0EEBC99-9C0B-4EF8-BB6D-6BB9BD380A11
{a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11}
a0eebc999c0b4ef8bb6d6bb9bd380a11
a0ee-bc99-9c0b-4ef8-bb6d-6bb9-bd38-0a11
{a0eebc99-9c0b4ef8-bb6d6bb9-bd380a11}
```

# Data Type

- xml
- Use of this data type requires the installation to have been built with configure --with-libxml
- 构造xml类型的语法
- SQL标准写法
- XMLPARSE ( { DOCUMENT | CONTENT } value)
- 例如
- XMLPARSE (DOCUMENT '<?xml  
version="1.0"?><book><title>Manual</title><chapter>...</chapter></book>')
- XMLPARSE (CONTENT 'abc<foo>bar</foo><bar>foo</bar>')
- PostgreSQL写法
- xml '<foo>bar</foo>'
- '<foo>bar</foo>'::xml
- 从xml到字符串的转换
- XMLSERIALIZE ( { DOCUMENT | CONTENT } value AS type )
- 例如
- XMLSERIALIZE ( CONTENT '<foo>bar</foo>'::xml AS text )

# Data Type

- Array
  - 不限长度
    - 目前PostgreSQL未对长度强限定,如int[]和int[10]都不会限定元素个数.
    - array\_length(ARRAY[[1,2,3,4,5],[6,7,8,9,10]], 1)
  - 不限维度
    - 目前PostgreSQL未对维度强限定,如int[]和int[][]，效果是一样的,都可以存储任意维度的数组.
  - 矩阵强制
    - 多维数组中,同一个维度的元素个数必须相同.
    - 正确
    - array[[1,2,3,4],[5,6,7,8]]
    - 不正确
    - array[[1,2,3,4],[5,6,7]]
  - 元素强制
    - 元素类型必须一致
    - 正确
    - array[1,2,3]
    - 不正确
    - array[1,2,'abc']

# Data Type

- Array
  - 一维数组支持prepend, append, cat操作
    - array\_append(ARRAY['digoal','francs'],'david')
    - array\_prepend('david',ARRAY['digoal','francs'])
  - 二维数组仅支持cat操作
    - array\_cat(ARRAY[['digoal','zhou'],['francs','tan']], ARRAY['david','guo'])
- subscript
  - 元素脚本默认从1开始, 也可以指定.
  - array\_lower(ARRAY[[1,2,3,4,5],[6,7,8,9,10]], 2)
  - array\_lower('[-3:-2]={1,2}':int[], 1)
  - select array\_upper('[-3:-2]={1,2}':int[], 1)

# Data Type

- Array
- slice
  - `array_dims(ARRAY[[1,2,3,4,5],[6,7,8,9,10]])`
- $a[1:2][1:1] = \{\{1\},\{3\}\}$
- 第一个[]中的1表示低位subscript, 2表示高位subscript值.
- 第二个[]中左边的1表示低位subscript, 右边的1表示高位subscript值.
- $a[2:3][1:2] = \{\{3,4\},\{5,6\}\}$
- 分片的另一种写法, 只要其中的一个维度用了分片写法, 其他的维度如果没有使用分片写法, 默认视为高位
- 如 $a[2:3][2]$  等同于  $a[2:3][1:2]$
  
- PostgreSQL ARRAY datatype introduce
- <http://blog.163.com/digoal@126/blog/static/163877040201201275922529/>



# Data Type

- Array
- function 与 操作符

| oprname | oprleft | oprright | oprresult | opropcode      | oprrest      | oprjoin         |
|---------|---------|----------|-----------|----------------|--------------|-----------------|
|         | 2277    | 2283     | 2277      | array_append   | -            | -               |
|         | 2283    | 2277     | 2277      | array_prepend  | -            | -               |
|         | 2277    | 2277     | 2277      | array_cat      | -            | -               |
| =       | 2277    | 2277     | 16        | array_eq       | eqsel        | eqjoinsel       |
| ◊       | 2277    | 2277     | 16        | array_ne       | neqsel       | neqjoinsel      |
| <       | 2277    | 2277     | 16        | array_lt       | scalarltsel  | scalarltjoinsel |
| >       | 2277    | 2277     | 16        | array_gt       | scalargttsel | scalargtjoinsel |
| ≤       | 2277    | 2277     | 16        | array_le       | scalarltsel  | scalarltjoinsel |
| ≥       | 2277    | 2277     | 16        | array_ge       | scalargttsel | scalargtjoinsel |
| @@      | 2277    | 2277     | 16        | arrayoverlap   | areasel      | areajoinsel     |
| @>      | 2277    | 2277     | 16        | arraycontains  | contsel      | contjoinsel     |
| @@      | 2277    | 2277     | 16        | arraycontained | contsel      | contjoinsel     |

# Data Type

- Composite Type
- 自定义
- create type test as (info text,id int,crt\_time timestamp(0));
- 创建表时默认创建一个同名composite type, 因此表名和自定义类名不能重复
- create table test (id int primary key,info text);
- ERROR: relation "test" already exists
- 举例
- CREATE TYPE inventory\_item AS (
  - name text,
  - supplier\_id integer,
  - price numeric)
- CREATE TABLE on\_hand (
  - item inventory\_item,
  - count integer)

# Data Type

- Composite Type
- INSERT INTO on\_hand VALUES (ROW('fuzzy dice', 42, 1.99), 1000);
- SELECT (on\_hand.item).name FROM on\_hand WHERE (on\_hand.item).price < 10;

■ name

-----

■ fuzzy dice

SET和INTO子句复合类型不能加括号引用,其他子句中的复合类型可以加括号引用

- UPDATE on\_hand SET item = ROW('fuzzy dice', 10, 100) WHERE count=1000;
- UPDATE on\_hand SET item.price = (on\_hand.item).price + 100 WHERE (on\_hand.item).name='fuzzy dice';
- INSERT INTO on\_hand (item.name, item.supplier\_id) VALUES('test', 2.2);
- postgres=# select \* from on\_hand;

■ item | count

-----+-----

■ ("fuzzy dice",10,200) | 1000  
■ (test,2,) |

# Data Type

- oid (object identifier) 4 bytes
- xid (transaction identifier) 4 bytes xmin,xmax
- cid (command identifier) 4 bytes cmin,cmax
- tid (tuple identifier) 6 bytes ctid
  
- 以下为各系统表对应的oid列的alias, 类型都是oid
- 可使用namespace, 或者默认的search\_path先后顺序检索

| Name          | References   | Description                  | Value Example                            |
|---------------|--------------|------------------------------|------------------------------------------|
| oid           | any          | numeric object identifier    | 564182                                   |
| regproc       | pg_proc      | function name                | sum                                      |
| regprocedure  | pg_proc      | function with argument types | sum(int4)                                |
| regoper       | pg_operator  | operator name                | +                                        |
| regoperator   | pg_operator  | operator with argument types | *(integer, integer) OR - (NONE, integer) |
| regclass      | pg_class     | relation name                | pg_type                                  |
| regtype       | pg_type      | data type name               | integer                                  |
| regconfig     | pg_ts_config | text search configuration    | english                                  |
| regdictionary | pg_ts_dict   | text search dictionary       | simple                                   |

# Data Type

■ test=# create sequence seq\_test start with 1;

■ CREATE SEQUENCE

■ test=# select 'seq\_test'::regclass;

■ regclass

■ seq\_test

■ test=# select 'seq\_test'::regclass::oid;

■ oid

■ 49247

■ test=# select 'sum(int4)'::regprocedure;

■ regprocedure

■ sum(integer)

■ test=# select 'sum(int4)'::regprocedure::oid;

■ oid

■ 2108

# Data Type

## ■ Pseudo-Types 伪类型

| Name             | Description                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------|
| any              | Indicates that a function accepts any input data type.                                                                      |
| anyarray         | Indicates that a function accepts any array data type (see <a href="#">Section 35.2.5</a> ).                                |
| anyelement       | Indicates that a function accepts any data type (see <a href="#">Section 35.2.5</a> ).                                      |
| anyenum          | Indicates that a function accepts any enum data type (see <a href="#">Section 35.2.5</a> and <a href="#">Section 8.7</a> ). |
| anynonarray      | Indicates that a function accepts any non-array data type (see <a href="#">Section 35.2.5</a> ).                            |
| cstring          | Indicates that a function accepts or returns a null-terminated C string.                                                    |
| internal         | Indicates that a function accepts or returns a server-internal data type.                                                   |
| language_handler | A procedural language call handler is declared to return <code>language_handler</code> .                                    |
| fdw_handler      | A foreign-data wrapper handler is declared to return <code>fdw_handler</code> .                                             |
| record           | Identifies a function returning an unspecified row type.                                                                    |
| trigger          | A trigger function is declared to return <code>trigger</code> .                                                             |
| void             | Indicates that a function returns no value.                                                                                 |
| opaque           | An obsolete type name that formerly served all the above purposes.                                                          |

# Functions and Operators

- 摘录部分
- 详见
- <http://www.postgresql.org/docs/9.1/static/functions.html>



# Functions and Operators

- 逻辑
- 比较
- 算数
- 字符
- bytea
- bit
- 规则表达式
- 格式化输出
- 时间
- 枚举
- 几何
- 网络地址
- 全文检索
- XML
- 序列
- 条件表达式
- 数组
- 集合
- 窗口
- 子查询表达式
- 行与数组比较
- 返回集合的函数
- 触发器函数

# Functions and Operators

- 逻辑操作符
- AND
- OR
- NOT

| $a$   | $b$   | $a \text{ AND } b$ | $a \text{ OR } b$ |
|-------|-------|--------------------|-------------------|
| TRUE  | TRUE  | TRUE               | TRUE              |
| TRUE  | FALSE | FALSE              | TRUE              |
| TRUE  | NULL  | NULL               | TRUE              |
| FALSE | FALSE | FALSE              | FALSE             |
| FALSE | NULL  | FALSE              | NULL              |
| NULL  | NULL  | NULL               | NULL              |

| $a$   | $\text{NOT } a$ |
|-------|-----------------|
| TRUE  | FALSE           |
| FALSE | TRUE            |
| NULL  | NULL            |

# Functions and Operators

- 比较
- a BETWEEN x AND y
- a >= x AND a <= y
- a NOT BETWEEN x AND y
- a < x OR a > y
- IS [NOT] NULL
  - test=# select 1 where null = null;  
(0 rows)
  - test=# select 1 where null <> null;  
(0 rows)
- test=# select 1 where null is distinct from null;  
(0 rows)
- test=# select 1 where null is not distinct from null;
- 1
- expression IS TRUE
- expression IS NOT TRUE
- expression IS FALSE
- expression IS NOT FALSE
- expression IS UNKNOWN
- expression IS NOT UNKNOWN

| Operator | Description              |
|----------|--------------------------|
| <        | less than                |
| >        | greater than             |
| <=       | less than or equal to    |
| >=       | greater than or equal to |
| =        | equal                    |
| ◊ or !=  | not equal                |

# Functions and Operators

## ■ 数学函数、操作符

- 略
- 数学函数
- 三角函数

| Operator | Description                                      | Example   | Result |
|----------|--------------------------------------------------|-----------|--------|
| +        | addition                                         | 2 + 3     | 5      |
| -        | subtraction                                      | 2 - 3     | -1     |
| *        | multiplication                                   | 2 * 3     | 6      |
| /        | division (integer division truncates the result) | 4 / 2     | 2      |
| %        | modulo (remainder)                               | 5 % 4     | 1      |
| ^        | exponentiation                                   | 2.0 ^ 3.0 | 8      |
| /        | square root                                      | / 25.0    | 5      |
| /        | cube root                                        | / 27.0    | 3      |
| !        | factorial                                        | 5 !       | 120    |
| !!       | factorial (prefix operator)                      | !! 5      | 120    |
| @        | absolute value                                   | @ -5.0    | 5      |
| &        | bitwise AND                                      | 91 & 15   | 11     |
|          | bitwise OR                                       | 32   3    | 35     |
| #        | bitwise XOR                                      | 17 # 5    | 20     |
| ~        | bitwise NOT                                      | ~1        | -2     |
| <<       | bitwise shift left                               | 1 << 4    | 16     |
| >>       | bitwise shift right                              | 8 >> 2    | 2      |

# Functions and Operators

## ■ 字符函数、操作符

| Function                                                                | Return Type | Description                                                                                                                    | Example                                                    | Result     |
|-------------------------------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|------------|
| <code>string    string</code>                                           | text        | String concatenation                                                                                                           | <code>'Post'    'greSQL'</code>                            | PostgreSQL |
| <code>string    non-string OR non-string    string</code>               | text        | String concatenation with one non-string input                                                                                 | <code>'Value: '    42</code>                               | Value: 42  |
| <code>bit_length(string)</code>                                         | int         | Number of bits in string                                                                                                       | <code>bit_length('jose')</code>                            | 32         |
| <code>char_length(string) OR character_length(string)</code>            | int         | Number of characters in string                                                                                                 | <code>char_length('jose')</code>                           | 4          |
| <code>lower(string)</code>                                              | text        | Convert string to lower case                                                                                                   | <code>lower('TOM')</code>                                  | tom        |
| <code>octet_length(string)</code>                                       | int         | Number of bytes in string                                                                                                      | <code>octet_length('jose')</code>                          | 4          |
| <code>overlay(string placing string from int [for int])</code>          | text        | Replace substring                                                                                                              | <code>overlay('Txxxxas' placing 'hom' from 2 for 4)</code> | Thomas     |
| <code>position(substring in string)</code>                              | int         | Location of specified substring                                                                                                | <code>position('om' in 'Thomas')</code>                    | 3          |
| <code>substring(string [from int] [for int])</code>                     | text        | Extract substring                                                                                                              | <code>substring('Thomas' from 2 for 3)</code>              | hom        |
| <code>substring(string from pattern)</code>                             | text        | Extract substring matching POSIX regular expression. See <a href="#">Section 9.7</a> for more information on pattern matching. | <code>substring('Thomas' from '...\$')</code>              | mas        |
| <code>substring(string from pattern for escape)</code>                  | text        | Extract substring matching SQL regular expression. See <a href="#">Section 9.7</a> for more information on pattern matching.   | <code>substring('Thomas' from '%#_o_a#_%' for '#')</code>  | oma        |
| <code>trim([leading   trailing   both] [characters] from string)</code> | text        | Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string       | <code>trim(both 'x' from 'xTomxx')</code>                  | Tom        |
| <code>upper(string)</code>                                              | text        | Convert string to upper case                                                                                                   | <code>upper('tom')</code>                                  | TOM        |

# Functions and Operators

## bytea函数、操作符

| Function                                          | Return Type | Description                                                                                   |
|---------------------------------------------------|-------------|-----------------------------------------------------------------------------------------------|
| string    string                                  | bytea       | String concatenation                                                                          |
| octet_length(string)                              | int         | Number of bytes in binary string                                                              |
| overlay(string placing string from int [for int]) | bytea       | Replace substring                                                                             |
| position(substring in string)                     | int         | Location of specified substring                                                               |
| substring(string [from int] [for int])            | bytea       | Extract substring                                                                             |
| trim([both] bytes from string)                    | bytea       | Remove the longest string containing only the bytes in bytes from the start and end of string |

| Example                                                                  | Result           |
|--------------------------------------------------------------------------|------------------|
| E'\\Post'::bytea    E'\\047gres\\000'::bytea                             | \\Post'gres\\000 |
| octet_length(E'jo\\000se'::bytea)                                        | 5                |
| overlay(E'Th\\000omas'::bytea placing E'\\002\\003'::bytea from 2 for 3) | T\\002\\003mas   |
| position(E'\\000om'::bytea in E'Th\\000omas'::bytea)                     | 3                |
| substring(E'Th\\000omas'::bytea from 2 for 3)                            | h\\000o          |
| trim(E'\\000'::bytea from E'\\000Tom\\000'::bytea)                       | Tom              |

# Functions and Operators

## ■ bit函数、操作符

| Operator | Description         | Example               | Result   |
|----------|---------------------|-----------------------|----------|
|          | concatenation       | B' 10001'    B' 011'  | 10001011 |
| &        | bitwise AND         | B' 10001' & B' 01101' | 00001    |
|          | bitwise OR          | B' 10001'   B' 01101' | 11101    |
| #        | bitwise XOR         | B' 10001' # B' 01101' | 11100    |
| ~        | bitwise NOT         | ~ B' 10001'           | 01110    |
| <<       | bitwise shift left  | B' 10001' << 3        | 01000    |
| >>       | bitwise shift right | B' 10001' >> 2        | 00100    |

# Functions and Operators

- 样式匹配、规则表达式
- LIKE
- SIMILAR TO 规则表达式
- POSIX 规则表达式

| Operator         | Description                                         | Example                                |
|------------------|-----------------------------------------------------|----------------------------------------|
| <code>~</code>   | Matches regular expression, case sensitive          | <code>' thomas' ~ '.*thomas.*'</code>  |
| <code>~*</code>  | Matches regular expression, case insensitive        | <code>' thomas' ~* '.*Thomas.*'</code> |
| <code>!~</code>  | Does not match regular expression, case sensitive   | <code>' thomas' !~ '.*Thomas.*'</code> |
| <code>!~*</code> | Does not match regular expression, case insensitive | <code>' thomas' !~* '.*vadim.*'</code> |

# Functions and Operators

- 格式化输出函数
- 略
- 日期、时间样式
- 数字样式

| Function                        | Return Type              | Description                             | Example                                      |
|---------------------------------|--------------------------|-----------------------------------------|----------------------------------------------|
| to_char(timestamp, text)        | text                     | convert time stamp to string            | to_char(current_timestamp, 'HH12:MI:SS')     |
| to_char(interval, text)         | text                     | convert interval to string              | to_char(interval '15h 2m 12s', 'HH24:MI:SS') |
| to_char(int, text)              | text                     | convert integer to string               | to_char(125, '999')                          |
| to_char(double precision, text) | text                     | convert real/double precision to string | to_char(125.8::real, '999D9')                |
| to_char(numeric, text)          | text                     | convert numeric to string               | to_char(-125.8, '999D99S')                   |
| to_date(text, text)             | date                     | convert string to date                  | to_date('05 Dec 2000', 'DD Mon YYYY')        |
| to_number(text, text)           | numeric                  | convert string to numeric               | to_number('12,454.8', '99G999D9S')           |
| to_timestamp(text, text)        | timestamp with time zone | convert string to time stamp            | to_timestamp('05 Dec 2000', 'DD Mon YYYY')   |
| to_timestamp(double precision)  | timestamp with time zone | convert Unix epoch to time stamp        | to_timestamp(1284352323)                     |

# Functions and Operators

- 日期、时间函数或操作符
- 操作符
- 函数
  - extract , date\_part
  - date\_trunc
  - pg\_sleep
    - Make sure that your session does not hold more locks than necessary when calling pg\_sleep. Otherwise other sessions might have to wait for your sleeping process, slowing down the entire system.

# Functions and Operators

## 日期、时间函数或操作符

| Operator | Example                                                     | Result                          |
|----------|-------------------------------------------------------------|---------------------------------|
| +        | date '2001-09-28' + integer '7'                             | date '2001-10-05'               |
| +        | date '2001-09-28' + interval '1 hour'                       | timestamp '2001-09-28 01:00:00' |
| +        | date '2001-09-28' + time '03:00'                            | timestamp '2001-09-28 03:00:00' |
| +        | interval '1 day' + interval '1 hour'                        | interval '1 day 01:00:00'       |
| +        | timestamp '2001-09-28 01:00' + interval '23 hours'          | timestamp '2001-09-29 00:00:00' |
| +        | time '01:00' + interval '3 hours'                           | time '04:00:00'                 |
| -        | - interval '23 hours'                                       | interval '-23:00:00'            |
| -        | date '2001-10-01' - date '2001-09-28'                       | integer '3' (days)              |
| -        | date '2001-10-01' - integer '7'                             | date '2001-09-24'               |
| -        | date '2001-09-28' - interval '1 hour'                       | timestamp '2001-09-27 23:00:00' |
| -        | time '05:00' - time '03:00'                                 | interval '02:00:00'             |
| -        | time '05:00' - interval '2 hours'                           | time '03:00:00'                 |
| -        | timestamp '2001-09-28 23:00' - interval '23 hours'          | timestamp '2001-09-28 00:00:00' |
| -        | interval '1 day' - interval '1 hour'                        | interval '1 day -01:00:00'      |
| -        | timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00' | interval '1 day 15:00:00'       |
| *        | 900 * interval '1 second'                                   | interval '00:15:00'             |
| *        | 21 * interval '1 day'                                       | interval '21 days'              |
| *        | double precision '3.5' * interval '1 hour'                  | interval '03:30:00'             |
| /        | interval '1 hour' / double precision '1.5'                  | interval '00:40:00'             |

# Functions and Operators

## 日期、时间函数或操作符

```
age(timestamp, timestamp)
age(timestamp)
clock_timestamp()
current_date
current_time
current_timestamp
date_part(text, timestamp)
date_part(text, interval)
date_trunc(text, timestamp)
extract(field from timestamp)
extract(field from interval)
isfinite(date)
isfinite(timestamp)
isfinite(interval)
justify_days(interval)
justify_hours(interval)
justify_interval(interval)
localtime
localtimestamp
now()
statement_timestamp()
timeofday()
transaction_timestamp()
```

# Functions and Operators

## ■ 枚举函数或操作符

| Function                     | Description                                                                                                                                                                                                                                                                                                   | Example                                         | Example Result                             |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|--------------------------------------------|
| enum_first(anyenum)          | Returns the first value of the input enum type                                                                                                                                                                                                                                                                | enum_first(null::rainbow)                       | red                                        |
| enum_last(anyenum)           | Returns the last value of the input enum type                                                                                                                                                                                                                                                                 | enum_last(null::rainbow)                        | purple                                     |
| enum_range(anyenum)          | Returns all values of the input enum type in an ordered array                                                                                                                                                                                                                                                 | enum_range(null::rainbow)                       | {red, orange, yellow, green, blue, purple} |
| enum_range(anyenum, anyenum) | Returns the range between the two given enum values, as an ordered array. The values must be from the same enum type. If the first parameter is null, the result will start with the first value of the enum type. If the second parameter is null, the result will end with the last value of the enum type. | enum_range('orange'::rainbow, 'green'::rainbow) | {orange, yellow, green}                    |
|                              |                                                                                                                                                                                                                                                                                                               | enum_range(NULL, 'green'::rainbow)              | {red, orange, yellow, green}               |
|                              |                                                                                                                                                                                                                                                                                                               | enum_range('orange'::rainbow, NULL)             | {orange, yellow, green, blue, purple}      |

# Functions and Operators

## ■ 集合函数或操作符

| Function                  | Return Type      | Description            | Example                               |
|---------------------------|------------------|------------------------|---------------------------------------|
| area( <i>object</i> )     | double precision | area                   | area(box'((0,0), (1,1))')             |
| center( <i>object</i> )   | point            | center                 | center(box'((0,0), (1,2))')           |
| diameter( <i>circle</i> ) | double precision | diameter of circle     | diameter(circle'((0,0), 2.0))'        |
| height( <i>box</i> )      | double precision | vertical size of box   | height(box'((0,0), (1,1))')           |
| isclosed( <i>path</i> )   | boolean          | a closed path?         | isclosed(path'((0,0), (1,1), (2,0))') |
| isopen( <i>path</i> )     | boolean          | an open path?          | isopen(path'[(0,0), (1,1), (2,0)])'   |
| length( <i>object</i> )   | double precision | length                 | length(path'((-1,0), (1,0))')         |
| npoints( <i>path</i> )    | int              | number of points       | npoints(path'[(0,0), (1,1), (2,0)])'  |
| npoints( <i>polygon</i> ) | int              | number of points       | npoints(polygon'((1,1), (0,0))')      |
| pclose( <i>path</i> )     | path             | convert path to closed | pclose(path'[(0,0), (1,1), (2,0)])'   |
| popen( <i>path</i> )      | path             | convert path to open   | popen(path'((0,0), (1,1), (2,0))')    |
| radius( <i>circle</i> )   | double precision | radius of circle       | radius(circle'((0,0), 2.0))'          |
| width( <i>box</i> )       | double precision | horizontal size of box | width(box'((0,0), (1,1))')            |

# Functions and Operators

- 集合函数或操作符
- 类型转换函数

| Function                                  | Return Type | Description                          | Example                                 |
|-------------------------------------------|-------------|--------------------------------------|-----------------------------------------|
| box(circle)                               | box         | circle to box                        | box(circle '((0,0), 2.0)')              |
| box(point, point)                         | box         | points to box                        | box(point '(0,0)', point '(1,1)')       |
| box(polygon)                              | box         | polygon to box                       | box(polygon '((0,0), (1,1), (2,0))')    |
| circle(box)                               | circle      | box to circle                        | circle(box '((0,0), (1,1))')            |
| circle(point, double precision)           | circle      | center and radius to circle          | circle(point '(0,0)', 2.0)              |
| circle(polygon)                           | circle      | polygon to circle                    | circle(polygon '((0,0), (1,1), (2,0))') |
| lseg(box)                                 | lseg        | box diagonal to line segment         | lseg(box '((-1,0), (1,0))')             |
| lseg(point, point)                        | lseg        | points to line segment               | lseg(point '(-1,0)', point '(1,0)')     |
| path(polygon)                             | point       | polygon to path                      | path(polygon '((0,0), (1,1), (2,0))')   |
| point(double precision, double precision) | point       | construct point                      | point(23.4, -44.5)                      |
| point(box)                                | point       | center of box                        | point(box '((-1,0), (1,0))')            |
| point(circle)                             | point       | center of circle                     | point(circle '((0,0), 2.0)')            |
| point(lseg)                               | point       | center of line segment               | point(lseg '((-1,0), (1,0))')           |
| point(polygon)                            | point       | center of polygon                    | point(polygon '((0,0), (1,1), (2,0))')  |
| polygon(box)                              | polygon     | box to 4-point polygon               | polygon(box '((0,0), (1,1))')           |
| polygon(circle)                           | polygon     | circle to 12-point polygon           | polygon(circle '((0,0), 2.0)')          |
| polygon(npts, circle)                     | polygon     | circle to <i>npts</i> -point polygon | polygon(12, circle '((0,0), 2.0)')      |
| polygon(path)                             | polygon     | path to polygon                      | polygon(path '((0,0), (1,1), (2,0))')   |

# Functions and Operators

## 全文检索函数或操作符

| Operator                | Description                            | Example                                                                       | Result                                   |
|-------------------------|----------------------------------------|-------------------------------------------------------------------------------|------------------------------------------|
| <code>@@</code>         | tsvector matches tsquery ?             | <code>to_tsvector('fat cats ate rats') @@ to_tsquery('cat &amp; rat')</code>  | <code>t</code>                           |
| <code>@@@</code>        | deprecated synonym for <code>@@</code> | <code>to_tsvector('fat cats ate rats') @@@ to_tsquery('cat &amp; rat')</code> | <code>t</code>                           |
| <code>  </code>         | concatenate tsvectorS                  | <code>'a':1 b':2'::tsvector    'c':1 d':2 b':3'::tsvector</code>              | <code>'a':1 'b':2,5 'c':3 'd':4</code>   |
| <code>&amp;&amp;</code> | AND tsqueryS together                  | <code>'fat'   'rat'::tsquery &amp;&amp; 'cat'::tsquery</code>                 | <code>('fat'   'rat') &amp; 'cat'</code> |
| <code>  </code>         | OR tsqueryS together                   | <code>'fat'   'rat'::tsquery    'cat'::tsquery</code>                         | <code>('fat'   'rat')   'cat'</code>     |
| <code>!!</code>         | negate a tsquery                       | <code>!! 'cat'::tsquery</code>                                                | <code>!cat'</code>                       |
| <code>@&gt;</code>      | tsquery contains another ?             | <code>'cat'::tsquery @&gt; 'cat &amp; rat'::tsquery</code>                    | <code>f</code>                           |
| <code>&lt;@</code>      | tsquery is contained in ?              | <code>'cat'::tsquery &lt;@ 'cat &amp; rat'::tsquery</code>                    | <code>t</code>                           |

# Functions and Operators

## ■ 序列函数或操作符

| Function                                       | Return Type         | Description                                                                          |
|------------------------------------------------|---------------------|--------------------------------------------------------------------------------------|
| <code>currval(regclass)</code>                 | <code>bigint</code> | Return value most recently obtained with <code>nextval</code> for specified sequence |
| <code>lastval()</code>                         | <code>bigint</code> | Return value most recently obtained with <code>nextval</code> for any sequence       |
| <code>nextval(regclass)</code>                 | <code>bigint</code> | Advance sequence and return new value                                                |
| <code>setval(regclass, bigint)</code>          | <code>bigint</code> | Set sequence's current value                                                         |
| <code>setval(regclass, bigint, boolean)</code> | <code>bigint</code> | Set sequence's current value and <code>is_called</code> flag                         |

# Functions and Operators

## ■ 条件函数或操作符

### ■ CASE

```
CASE WHEN condition THEN result
      [WHEN ...]
      [ELSE result]
END
```

```
CASE expression
      WHEN value THEN result
      [WHEN ...]
      [ELSE result]
END
```

### ■ COALESCE

- The COALESCE function returns the first of its arguments that is not null. Null is returned only if all arguments are null. It is often used to substitute a default value for null values when data is retrieved for display, for example:

### ■ NULLIF

- The NULLIF function returns a null value if value1 equals value2; otherwise it returns value1.

### ■ GREATEST and LEAST

# Functions and Operators

## ■ 数组函数或操作符

## ■ 操作符

| Operator | Description                       | Example                                         | Result                              |
|----------|-----------------------------------|-------------------------------------------------|-------------------------------------|
| =        | equal                             | ARRAY[1, 1, 2, 1, 3, 1]::int[] = ARRAY[1, 2, 3] | t                                   |
| ◊        | not equal                         | ARRAY[1, 2, 3] ◊ ARRAY[1, 2, 4]                 | t                                   |
| <        | less than                         | ARRAY[1, 2, 3] < ARRAY[1, 2, 4]                 | t                                   |
| >        | greater than                      | ARRAY[1, 4, 3] > ARRAY[1, 2, 4]                 | t                                   |
| ≤        | less than or equal                | ARRAY[1, 2, 3] ≤ ARRAY[1, 2, 3]                 | t                                   |
| ≥        | greater than or equal             | ARRAY[1, 4, 3] ≥ ARRAY[1, 4, 3]                 | t                                   |
| @>       | contains                          | ARRAY[1, 4, 3] @> ARRAY[3, 1]                   | t                                   |
| @@       | is contained by                   | ARRAY[2, 7] @@ ARRAY[1, 7, 4, 2, 6]             | t                                   |
| &&       | overlap (have elements in common) | ARRAY[1, 4, 3] && ARRAY[2, 1]                   | t                                   |
|          | array-to-array concatenation      | ARRAY[1, 2, 3]    ARRAY[4, 5, 6]                | {1, 2, 3, 4, 5, 6}                  |
|          | array-to-array concatenation      | ARRAY[1, 2, 3]    ARRAY[[4, 5, 6], [7, 8, 9]]   | { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} } |
|          | element-to-array concatenation    | 3    ARRAY[4, 5, 6]                             | {3, 4, 5, 6}                        |
|          | array-to-element concatenation    | ARRAY[4, 5, 6]    t                             | {4, 5, 6, 7}                        |

# Functions and Operators

## ■ 数组函数或操作符

## ■ 函数

| Function                                 | Return Type      | Description                                                                                                | Example                                            | Result             |
|------------------------------------------|------------------|------------------------------------------------------------------------------------------------------------|----------------------------------------------------|--------------------|
| array_append(anyarray, anyelement)       | anyarray         | append an element to the end of an array                                                                   | array_append(ARRAY[1, 2], 3)                       | {1, 2, 3}          |
| array_cat(anyarray, anyarray)            | anyarray         | concatenate two arrays                                                                                     | array_cat(ARRAY[1, 2, 3], ARRAY[4, 5])             | {1, 2, 3, 4, 5}    |
| array_ndims(anyarray)                    | int              | returns the number of dimensions of the array                                                              | array_ndims(ARRAY[[1, 2, 3], [4, 5, 6]])           | 2                  |
| array_dims(anyarray)                     | text             | returns a text representation of array's dimensions                                                        | array_dims(ARRAY[[1, 2, 3], [4, 5, 6]])            | [1:2][1:3]         |
| array_fill(anyelement, int[], [, int[]]) | anyarray         | returns an array initialized with supplied value and dimensions, optionally with lower bounds other than 1 | array_fill(7, ARRAY[3], ARRAY[2])                  | [2:4]=[7, 7, 7]    |
| array_length(anyarray, int)              | int              | returns the length of the requested array dimension                                                        | array_length(array[1, 2, 3], 1)                    | 3                  |
| array_lower(anyarray, int)               | int              | returns lower bound of the requested array dimension                                                       | array_lower('0:2]={1, 2, 3}': int[], 1)            | 0                  |
| arrayprepend(anyelement, anyarray)       | anyarray         | append an element to the beginning of an array                                                             | arrayprepend(1, ARRAY[2, 3])                       | {1, 2, 3}          |
| array_to_string(anyarray, text [, text]) | text             | concatenates array elements using supplied delimiter and optional null string                              | array_to_string(ARRAY[1, 2, 3, NULL, 5], ',', '*') | 1, 2, 3, *, 5      |
| array_upper(anyarray, int)               | int              | returns upper bound of the requested array dimension                                                       | array_upper(ARRAY[1, 8, 3, 7], 1)                  | 4                  |
| string_to_array(text, text [, text])     | text[]           | splits string into array elements using supplied delimiter and optional null string                        | string_to_array('xx~~~yy~~~zz', '~~~', 'yy')       | {xx, NULL, zz}     |
| unnest(anyarray)                         | setof anyelement | expand an array to a set of rows                                                                           | unnest(ARRAY[1, 2])                                | 1<br>2<br>(2 rows) |

# Functions and Operators

- 集合函数
- 多值输入单值输出

| Function                                       | Argument Type(s)                                                    | Return Type                                                                                                                                                     | Description                                                                     |
|------------------------------------------------|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| <code>array_agg(expression)</code>             | any                                                                 | array of the argument type                                                                                                                                      | input values, including nulls, concatenated into an array                       |
| <code>avg(expression)</code>                   | smallint, int, bigint, real, double precision, numeric, OR interval | numeric for any integer-type argument, double precision for a floating-point argument, otherwise the same as the argument data type                             | the average (arithmetic mean) of all input values                               |
| <code>bit_and(expression)</code>               | smallint, int, bigint, OR bit                                       | same as argument data type                                                                                                                                      | the bitwise AND of all non-null input values, or null if none                   |
| <code>bit_or(expression)</code>                | smallint, int, bigint, OR bit                                       | same as argument data type                                                                                                                                      | the bitwise OR of all non-null input values, or null if none                    |
| <code>bool_and(expression)</code>              | bool                                                                | bool                                                                                                                                                            | true if all input values are true, otherwise false                              |
| <code>bool_or(expression)</code>               | bool                                                                | bool                                                                                                                                                            | true if at least one input value is true, otherwise false                       |
| <code>count(*)</code>                          |                                                                     | bigint                                                                                                                                                          | number of input rows                                                            |
| <code>count(expression)</code>                 | any                                                                 | bigint                                                                                                                                                          | number of input rows for which the value of <code>expression</code> is not null |
| <code>every(expression)</code>                 | bool                                                                | bool                                                                                                                                                            | equivalent to <code>bool_and</code>                                             |
| <code>max(expression)</code>                   | any array, numeric, string, or date/time type                       | same as argument type                                                                                                                                           | maximum value of <code>expression</code> across all input values                |
| <code>min(expression)</code>                   | any array, numeric, string, or date/time type                       | same as argument type                                                                                                                                           | minimum value of <code>expression</code> across all input values                |
| <code>string_agg(expression, delimiter)</code> | text, text                                                          | text                                                                                                                                                            | input values concatenated into a string, separated by delimiter                 |
| <code>sum(expression)</code>                   | smallint, int, bigint, real, double precision, numeric, OR interval | bigint for smallint or int arguments, numeric for bigint arguments, double precision for floating-point arguments, otherwise the same as the argument data type | sum of <code>expression</code> across all input values                          |
| <code>xmllagg(expression)</code>               | xml                                                                 | xml                                                                                                                                                             | concatenation of XML values (see also <a href="#">Section 9.14.1.7</a> )        |

# Functions and Operators

- 窗口函数
- 前面的章节有例子

| Function                                                                              | Return Type                            | Description                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>row_number()</code>                                                             | <code>bigint</code>                    | number of the current row within its partition, counting from 1                                                                                                                                                                                                                                                                                                              |
| <code>rank()</code>                                                                   | <code>bigint</code>                    | rank of the current row with gaps; same as <code>row_number</code> of its first peer                                                                                                                                                                                                                                                                                         |
| <code>dense_rank()</code>                                                             | <code>bigint</code>                    | rank of the current row without gaps; this function counts peer groups                                                                                                                                                                                                                                                                                                       |
| <code>percent_rank()</code>                                                           | <code>double precision</code>          | relative rank of the current row: $(\text{rank} - 1) / (\text{total rows} - 1)$                                                                                                                                                                                                                                                                                              |
| <code>cume_dist()</code>                                                              | <code>double precision</code>          | relative rank of the current row: $(\text{number of rows preceding or peer with current row}) / (\text{total rows})$                                                                                                                                                                                                                                                         |
| <code>ntile(<i>num_buckets</i> integer)</code>                                        | <code>integer</code>                   | integer ranging from 1 to the argument value, dividing the partition as equally as possible                                                                                                                                                                                                                                                                                  |
| <code>lag(<i>value</i> any [, <i>offset</i> integer [, <i>default</i> any ]])</code>  | <code>same type as <i>value</i></code> | returns <code>value</code> evaluated at the row that is <code>offset</code> rows before the current row within the partition; if there is no such row, instead return <code>default</code> . Both <code>offset</code> and <code>default</code> are evaluated with respect to the current row. If omitted, <code>offset</code> defaults to 1 and <code>default</code> to null |
| <code>lead(<i>value</i> any [, <i>offset</i> integer [, <i>default</i> any ]])</code> | <code>same type as <i>value</i></code> | returns <code>value</code> evaluated at the row that is <code>offset</code> rows after the current row within the partition; if there is no such row, instead return <code>default</code> . Both <code>offset</code> and <code>default</code> are evaluated with respect to the current row. If omitted, <code>offset</code> defaults to 1 and <code>default</code> to null  |
| <code>first_value(<i>value</i> any)</code>                                            | <code>same type as <i>value</i></code> | returns <code>value</code> evaluated at the row that is the first row of the window frame                                                                                                                                                                                                                                                                                    |
| <code>last_value(<i>value</i> any)</code>                                             | <code>same type as <i>value</i></code> | returns <code>value</code> evaluated at the row that is the last row of the window frame                                                                                                                                                                                                                                                                                     |
| <code>nth_value(<i>value</i> any, <i>nth</i> integer)</code>                          | <code>same type as <i>value</i></code> | returns <code>value</code> evaluated at the row that is the <code>nth</code> row of the window frame (counting from 1); null if no such row                                                                                                                                                                                                                                  |

# Functions and Operators

- 子查询表达式
- row\_constructor operator (subquery)
- EXISTS
  - EXISTS (subquery)
- [NOT] IN
  - expression [NOT] IN (subquery)
  - row\_constructor [NOT] IN (subquery)
- ANY / SOME
  - expression operator ANY | SOME (subquery)
  - row\_constructor operator ANY | SOME (subquery)
  - IN is equivalent to = ANY
- ALL
  - expression operator ALL (subquery)
  - row\_constructor operator ALL (subquery)
  - NOT IN is equivalent to <> ALL

# Functions and Operators

- ARRAY与表达式比较
- expression operator ANY | SOME (array expression)
- expression operator ALL (array expression)

# Functions and Operators

## ■ 返回多行的函数

| Function                                    | Argument Type                         | Return Type                                                               |
|---------------------------------------------|---------------------------------------|---------------------------------------------------------------------------|
| generate_series(start, stop)                | int OR bigint                         | setof int OR setof bigint (same as argument type)                         |
| generate_series(start, stop, step)          | int OR bigint                         | setof int OR setof bigint (same as argument type)                         |
| generate_series(start, stop, step interval) | timestamp OR timestamp with time zone | setof timestamp OR setof timestamp with time zone (Same as argument type) |

| Function                                                      | Return Type |
|---------------------------------------------------------------|-------------|
| generate_subscripts(array anyarray, dim int)                  | setof int   |
| generate_subscripts(array anyarray, dim int, reverse boolean) | setof int   |

# Functions and Operators

- 系统信息函数
- 会话信息

| Name                         | Return Type              | Description                                                                                               |
|------------------------------|--------------------------|-----------------------------------------------------------------------------------------------------------|
| current_catalog              | name                     | name of current database (called "catalog" in the SQL standard)                                           |
| current_database()           | name                     | name of current database                                                                                  |
| current_query()              | text                     | text of the currently executing query, as submitted by the client (might contain more than one statement) |
| current_schema[]()           | name                     | name of current schema                                                                                    |
| current_schemas(boolean)     | name[]                   | names of schemas in search path, optionally including implicit schemas                                    |
| current_user                 | name                     | user name of current execution context                                                                    |
| inet_client_addr()           | inet                     | address of the remote connection                                                                          |
| inet_client_port()           | int                      | port of the remote connection                                                                             |
| inet_server_addr()           | inet                     | address of the local connection                                                                           |
| inet_server_port()           | int                      | port of the local connection                                                                              |
| pg_backend_pid()             | int                      | Process ID of the server process attached to the current session                                          |
| pg_conf_load_time()          | timestamp with time zone | configuration load time                                                                                   |
| pg_is_other_temp_schema(oid) | boolean                  | is schema another session's temporary schema?                                                             |
| pg_listening_channels()      | setof text               | channel names that the session is currently listening on                                                  |
| pg_my_temp_schema()          | oid                      | OID of session's temporary schema, or 0 if none                                                           |
| pg_postmaster_start_time()   | timestamp with time zone | server start time                                                                                         |
| session_user                 | name                     | session user name                                                                                         |
| user                         | name                     | equivalent to current_user                                                                                |
| version()                    | text                     | PostgreSQL version information                                                                            |

# Functions and Operators

## ■ 系统信息函数 - 访问权限函数

| Name                                                     | Return Type | Description                                               |
|----------------------------------------------------------|-------------|-----------------------------------------------------------|
| has_any_column_privilege(user, table, privilege)         | boolean     | does user have privilege for any column of table          |
| has_any_column_privilege(table, privilege)               | boolean     | does current user have privilege for any column of table  |
| has_column_privilege(user, table, column, privilege)     | boolean     | does user have privilege for column                       |
| has_column_privilege(table, column, privilege)           | boolean     | does current user have privilege for column               |
| has_database_privilege(user, database, privilege)        | boolean     | does user have privilege for database                     |
| has_database_privilege(database, privilege)              | boolean     | does current user have privilege for database             |
| has_foreign_data_wrapper_privilege(user, fdw, privilege) | boolean     | does user have privilege for foreign-data wrapper         |
| has_foreign_data_wrapper_privilege(fdw, privilege)       | boolean     | does current user have privilege for foreign-data wrapper |
| has_function_privilege(user, function, privilege)        | boolean     | does user have privilege for function                     |
| has_function_privilege(function, privilege)              | boolean     | does current user have privilege for function             |
| has_language_privilege(user, language, privilege)        | boolean     | does user have privilege for language                     |
| has_language_privilege(language, privilege)              | boolean     | does current user have privilege for language             |
| has_schema_privilege(user, schema, privilege)            | boolean     | does user have privilege for schema                       |
| has_schema_privilege(schema, privilege)                  | boolean     | does current user have privilege for schema               |
| has_sequence_privilege(user, sequence, privilege)        | boolean     | does user have privilege for sequence                     |
| has_sequence_privilege(sequence, privilege)              | boolean     | does current user have privilege for sequence             |
| has_server_privilege(user, server, privilege)            | boolean     | does user have privilege for foreign server               |
| has_server_privilege(server, privilege)                  | boolean     | does current user have privilege for foreign server       |
| has_table_privilege(user, table, privilege)              | boolean     | does user have privilege for table                        |
| has_table_privilege(table, privilege)                    | boolean     | does current user have privilege for table                |
| has_tablespace_privilege(user, tablespace, privilege)    | boolean     | does user have privilege for tablespace                   |
| has_tablespace_privilege(tablespace, privilege)          | boolean     | does current user have privilege for tablespace           |
| pg_has_role(user, role, privilege)                       | boolean     | does user have privilege for role                         |
| pg_has_role(role, privilege)                             | boolean     | does current user have privilege for role                 |

# Functions and Operators

## ■ 系统信息函数 - SCHEMA可见性函数

| Name                                     | Return Type | Description                                         |
|------------------------------------------|-------------|-----------------------------------------------------|
| pg_collation_is_visible(collation_oid)   | boolean     | is collation visible in search path                 |
| pg_conversion_is_visible(conversion_oid) | boolean     | is conversion visible in search path                |
| pg_function_is_visible(function_oid)     | boolean     | is function visible in search path                  |
| pg_opclass_is_visible(opclass_oid)       | boolean     | is operator class visible in search path            |
| pg_operator_is_visible(operator_oid)     | boolean     | is operator visible in search path                  |
| pg_table_is_visible(table_oid)           | boolean     | is table visible in search path                     |
| pg_ts_config_is_visible(config_oid)      | boolean     | is text search configuration visible in search path |
| pg_ts_dict_is_visible(dict_oid)          | boolean     | is text search dictionary visible in search path    |
| pg_ts_parser_is_visible(parser_oid)      | boolean     | is text search parser visible in search path        |
| pg_ts_template_is_visible(template_oid)  | boolean     | is text search template visible in search path      |
| pg_type_is_visible(type_oid)             | boolean     | is type (or domain) visible in search path          |

# Functions and Operators

## ■ 系统信息函数 - System Catalog Information 函数

| Name                                                     | Return Type  | Description                                                                                                                    |
|----------------------------------------------------------|--------------|--------------------------------------------------------------------------------------------------------------------------------|
| format_type(type_oid, typmod)                            | text         | get SQL name of a data type                                                                                                    |
| pg_describe_object(catalog_id, object_id, object_sub_id) | text         | get description of a database object                                                                                           |
| pg_get_constraintdef(constraint_oid)                     | text         | get definition of a constraint                                                                                                 |
| pg_get_constraintdef(constraint_oid, pretty_bool)        | text         | get definition of a constraint                                                                                                 |
| pg_get_expr(pg_node_tree, relation_oid)                  | text         | decompile internal form of an expression, assuming that any Vars in it refer to the relation indicated by the second parameter |
| pg_get_expr(pg_node_tree, relation_oid, pretty_bool)     | text         | decompile internal form of an expression, assuming that any Vars in it refer to the relation indicated by the second parameter |
| pg_get_functiondef(func_oid)                             | text         | get definition of a function                                                                                                   |
| pg_get_function_arguments(func_oid)                      | text         | get argument list of function's definition (with default values)                                                               |
| pg_get_function_identity_arguments(func_oid)             | text         | get argument list to identify a function (without default values)                                                              |
| pg_get_function_result(func_oid)                         | text         | get RETURNS clause for function                                                                                                |
| pg_get_indexdef(index_oid)                               | text         | get CREATE INDEX command for index                                                                                             |
| pg_get_indexdef(index_oid, column_no, pretty_bool)       | text         | get CREATE INDEX command for index, or definition of just one index column when column_no is not zero                          |
| pg_get_keywords()                                        | setof record | get list of SQL keywords and their categories                                                                                  |
| pg_get_ruledef(rule_oid)                                 | text         | get CREATE RULE command for rule                                                                                               |
| pg_get_ruledef(rule_oid, pretty_bool)                    | text         | get CREATE RULE command for rule                                                                                               |

# Functions and Operators

## ■ 系统信息函数 - System Catalog Information 函数

|                                                              |                           |                                                                                           |
|--------------------------------------------------------------|---------------------------|-------------------------------------------------------------------------------------------|
| <code>pg_get_serial_sequence(table_name, column_name)</code> | <code>text</code>         | get name of the sequence that a <code>serial</code> or <code>bigserial</code> column uses |
| <code>pg_get_triggerdef(trigger_oid)</code>                  | <code>text</code>         | get <code>CREATE [ CONSTRAINT ] TRIGGER</code> command for trigger                        |
| <code>pg_get_triggerdef(trigger_oid, pretty_bool)</code>     | <code>text</code>         | get <code>CREATE [ CONSTRAINT ] TRIGGER</code> command for trigger                        |
| <code>pg_get_userbyid(role_oid)</code>                       | <code>name</code>         | get role name with given OID                                                              |
| <code>pg_get_viewdef(view_name)</code>                       | <code>text</code>         | get underlying <code>SELECT</code> command for view ( <b>deprecated</b> )                 |
| <code>pg_get_viewdef(view_name, pretty_bool)</code>          | <code>text</code>         | get underlying <code>SELECT</code> command for view ( <b>deprecated</b> )                 |
| <code>pg_get_viewdef(view_oid)</code>                        | <code>text</code>         | get underlying <code>SELECT</code> command for view                                       |
| <code>pg_get_viewdef(view_oid, pretty_bool)</code>           | <code>text</code>         | get underlying <code>SELECT</code> command for view                                       |
| <code>pg_options_to_table(reloptions)</code>                 | <code>setof record</code> | get the set of storage option name/value pairs                                            |
| <code>pg_tablespace_databases(tablespace_oid)</code>         | <code>setof oid</code>    | get the set of database OIDs that have objects in the tablespace                          |
| <code>pg_typeof(any)</code>                                  | <code>regtype</code>      | get the data type of any value                                                            |

## 注释信息函数

| Name                                                     | Return Type       | Description                                             |
|----------------------------------------------------------|-------------------|---------------------------------------------------------|
| <code>col_description(table_oid, column_number)</code>   | <code>text</code> | get comment for a table column                          |
| <code>obj_description(object_oid, catalog_name)</code>   | <code>text</code> | get comment for a database object                       |
| <code>obj_description(object_oid)</code>                 | <code>text</code> | get comment for a database object ( <b>deprecated</b> ) |
| <code>shobj_description(object_oid, catalog_name)</code> | <code>text</code> | get comment for a shared database object                |

# Functions and Operators

## ■ 系统信息函数 - 事务ID与Snapshot 函数

| Name                                                         | Return Type                | Description                                                                 |
|--------------------------------------------------------------|----------------------------|-----------------------------------------------------------------------------|
| <code>txid_current()</code>                                  | <code>bigint</code>        | get current transaction ID                                                  |
| <code>txid_current_snapshot()</code>                         | <code>txid_snapshot</code> | get current snapshot                                                        |
| <code>txid_snapshot_xip(txid_snapshot)</code>                | <code>setof bigint</code>  | get in-progress transaction IDs in snapshot                                 |
| <code>txid_snapshot_xmax(txid_snapshot)</code>               | <code>bigint</code>        | get <code>xmax</code> of snapshot                                           |
| <code>txid_snapshot_xmin(txid_snapshot)</code>               | <code>bigint</code>        | get <code>xmin</code> of snapshot                                           |
| <code>txid_visible_in_snapshot(bigint, txid_snapshot)</code> | <code>boolean</code>       | is transaction ID visible in snapshot? (do not use with subtransaction ids) |

## ■ `txid_snapshot`结构

| Name                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>xmin</code>     | Earliest transaction ID (txid) that is still active. All earlier transactions will either be committed and visible, or rolled back and dead.                                                                                                                                                                                                                                                                                         |
| <code>xmax</code>     | First as-yet-unassigned txid. All txids greater than or equal to this are not yet started as of the time of the snapshot, and thus invisible.                                                                                                                                                                                                                                                                                        |
| <code>xip_list</code> | Active txids at the time of the snapshot. The list includes only those active txids between <code>xmin</code> and <code>xmax</code> ; there might be active txids higher than <code>xmax</code> . A txid that is $xmin \leq txid < xmax$ and not in this list was already completed at the time of the snapshot, and thus either visible or dead according to its commit status. The list does not include txids of subtransactions. |

# Functions and Operators

## ■ 系统管理函数 - 配置设置函数

| Name                                          | Return Type | Description                        |
|-----------------------------------------------|-------------|------------------------------------|
| current_setting(setting_name)                 | text        | get current value of setting       |
| set_config(setting_name, new_value, is_local) | text        | set parameter and return new value |

## ■ 信号函数

| Name                          | Return Type | Description                                                |
|-------------------------------|-------------|------------------------------------------------------------|
| pg_cancel_backend(pid int)    | boolean     | Cancel a backend's current query                           |
| pg_reload_conf()              | boolean     | Cause server processes to reload their configuration files |
| pg_rotate_logfile()           | boolean     | Rotate server's log file                                   |
| pg_terminate_backend(pid int) | boolean     | Terminate a backend                                        |

# Functions and Operators

## ■ 系统管理函数 - 备份控制函数

| Name                                                       | Return Type                | Description                                                                              |
|------------------------------------------------------------|----------------------------|------------------------------------------------------------------------------------------|
| <code>pg_create_restore_point(name text)</code>            | <code>text</code>          | Create a named point for performing restore (restricted to superusers)                   |
| <code>pg_current_xlog_insert_location()</code>             | <code>text</code>          | Get current transaction log insert location                                              |
| <code>pg_current_xlog_location()</code>                    | <code>text</code>          | Get current transaction log write location                                               |
| <code>pg_start_backup(label text [, fast boolean ])</code> | <code>text</code>          | Prepare for performing on-line backup (restricted to superusers or replication roles)    |
| <code>pg_stop_backup()</code>                              | <code>text</code>          | Finish performing on-line backup (restricted to superusers or replication roles)         |
| <code>pg_switch_xlog()</code>                              | <code>text</code>          | Force switch to a new transaction log file (restricted to superusers)                    |
| <code>pg_xlogfile_name(location text)</code>               | <code>text</code>          | Convert transaction log location string to file name                                     |
| <code>pg_xlogfile_name_offset(location text)</code>        | <code>text, integer</code> | Convert transaction log location string to file name and decimal byte offset within file |

## ■ 恢复信息函数

| Name                                         | Return Type                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pg_is_in_recovery()</code>             | <code>bool</code>                     | True if recovery is still in progress.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>pg_last_xlog_receive_location()</code> | <code>text</code>                     | Get last transaction log location received and synced to disk by streaming replication. While streaming replication is in progress this will increase monotonically. If recovery has completed this will remain static at the value of the last WAL record received and synced to disk during recovery. If streaming replication is disabled, or if it has not yet started, the function returns NULL.                                                                                                                                                          |
| <code>pg_last_xlog_replay_location()</code>  | <code>text</code>                     | Get last transaction log location replayed during recovery. If recovery is still in progress this will increase monotonically. If recovery has completed then this value will remain static at the value of the last WAL record applied during that recovery. When the server has been started normally without recovery the function returns NULL.                                                                                                                                                                                                             |
| <code>pg_last_xact_replay_timestamp()</code> | <code>timestamp with time zone</code> | Get time stamp of last transaction replayed during recovery. This is the time at which the commit or abort WAL record for that transaction was generated on the primary. If no transactions have been replayed during recovery, this function returns NULL. Otherwise, if recovery is still in progress this will increase monotonically. If recovery has completed then this value will remain static at the value of the last transaction applied during that recovery. When the server has been started normally without recovery the function returns NULL. |

# Functions and Operators

## ■ 系统管理函数 - 恢复控制函数

| Name                                    | Return Type       | Description                         |
|-----------------------------------------|-------------------|-------------------------------------|
| <code>pg_is_xlog_replay_paused()</code> | <code>bool</code> | True if recovery is paused.         |
| <code>pg_xlog_replay_pause()</code>     | <code>void</code> | Pauses recovery immediately.        |
| <code>pg_xlog_replay_resume()</code>    | <code>void</code> | Restarts recovery if it was paused. |

## ■ 对象大小查询函数

| Name                                                        | Return Type         | Description                                                                                                         |
|-------------------------------------------------------------|---------------------|---------------------------------------------------------------------------------------------------------------------|
| <code>pg_column_size(any)</code>                            | <code>int</code>    | Number of bytes used to store a particular value (possibly compressed)                                              |
| <code>pg_database_size(oid)</code>                          | <code>bigint</code> | Disk space used by the database with the specified OID                                                              |
| <code>pg_database_size(name)</code>                         | <code>bigint</code> | Disk space used by the database with the specified name                                                             |
| <code>pg_indexes_size(regclass)</code>                      | <code>bigint</code> | Total disk space used by indexes attached to the specified table                                                    |
| <code>pg_relation_size(relation regclass, fork text)</code> | <code>bigint</code> | Disk space used by the specified fork ('main', 'fsm' or 'vm') of the specified table or index                       |
| <code>pg_relation_size(relation regclass)</code>            | <code>bigint</code> | Shorthand for <code>pg_relation_size(..., 'main')</code>                                                            |
| <code>pg_size_pretty(bigint)</code>                         | <code>text</code>   | Converts a size in bytes into a human-readable format with size units                                               |
| <code>pg_table_size(regclass)</code>                        | <code>bigint</code> | Disk space used by the specified table, excluding indexes (but including TOAST, free space map, and visibility map) |
| <code>pg_tablespace_size(oid)</code>                        | <code>bigint</code> | Disk space used by the tablespace with the specified OID                                                            |
| <code>pg_tablespace_size(name)</code>                       | <code>bigint</code> | Disk space used by the tablespace with the specified name                                                           |
| <code>pg_total_relation_size(regclass)</code>               | <code>bigint</code> | Total disk space used by the specified table, including all indexes and TOAST data                                  |

# Functions and Operators

## ■ 系统管理函数 - 对象物理位置查询函数

| Name                                                 | Return Type | Description                               |
|------------------------------------------------------|-------------|-------------------------------------------|
| <code>pg_relation_filenode(relation regclass)</code> | oid         | Filenode number of the specified relation |
| <code>pg_relation_filepath(relation regclass)</code> | text        | File path name of the specified relation  |

## ■ 文件访问函数

| Name                                                                             | Return Type             | Description                        |
|----------------------------------------------------------------------------------|-------------------------|------------------------------------|
| <code>pg_ls_dir(dirname text)</code>                                             | <code>setof text</code> | List the contents of a directory   |
| <code>pg_read_file(filename text [, offset bigint, length bigint])</code>        | text                    | Return the contents of a text file |
| <code>pg_read_binary_file(filename text [, offset bigint, length bigint])</code> | bytea                   | Return the contents of a file      |
| <code>pg_stat_file(filename text)</code>                                         | record                  | Return information about a file    |

# Functions and Operators

## ■ 系统管理函数 - advisory锁函数

| Name                                                 | Return Type | Description                                                          |
|------------------------------------------------------|-------------|----------------------------------------------------------------------|
| pg_advisory_lock(key bigint)                         | void        | Obtain exclusive session level advisory lock                         |
| pg_advisory_lock(key1 int, key2 int)                 | void        | Obtain exclusive session level advisory lock                         |
| pg_advisory_lock_shared(key bigint)                  | void        | Obtain shared session level advisory lock                            |
| pg_advisory_lock_shared(key1 int, key2 int)          | void        | Obtain shared session level advisory lock                            |
| pg_advisory_unlock(key bigint)                       | boolean     | Release an exclusive session level advisory lock                     |
| pg_advisory_unlock(key1 int, key2 int)               | boolean     | Release an exclusive session level advisory lock                     |
| pg_advisory_unlock_all()                             | void        | Release all session level advisory locks held by the current session |
| pg_advisory_unlock_shared(key bigint)                | boolean     | Release a shared session level advisory lock                         |
| pg_advisory_unlock_shared(key1 int, key2 int)        | boolean     | Release a shared session level advisory lock                         |
| pg_advisory_xact_lock(key bigint)                    | void        | Obtain exclusive transaction level advisory lock                     |
| pg_advisory_xact_lock(key1 int, key2 int)            | void        | Obtain exclusive transaction level advisory lock                     |
| pg_advisory_xact_lock_shared(key bigint)             | void        | Obtain shared transaction level advisory lock                        |
| pg_advisory_xact_lock_shared(key1 int, key2 int)     | void        | Obtain shared advisory lock for the current transaction              |
| pg_try_advisory_lock(key bigint)                     | boolean     | Obtain exclusive session level advisory lock if available            |
| pg_try_advisory_lock(key1 int, key2 int)             | boolean     | Obtain exclusive session level advisory lock if available            |
| pg_try_advisory_lock_shared(key bigint)              | boolean     | Obtain shared session level advisory lock if available               |
| pg_try_advisory_lock_shared(key1 int, key2 int)      | boolean     | Obtain shared session level advisory lock if available               |
| pg_try_advisory_xact_lock(key bigint)                | boolean     | Obtain exclusive transaction level advisory lock if available        |
| pg_try_advisory_xact_lock(key1 int, key2 int)        | boolean     | Obtain exclusive transaction level advisory lock if available        |
| pg_try_advisory_xact_lock_shared(key bigint)         | boolean     | Obtain shared transaction level advisory lock if available           |
| pg_try_advisory_xact_lock_shared(key1 int, key2 int) | boolean     | Obtain shared transaction level advisory lock if available           |

# Functions and Operators

- 系统管理函数 - advisory锁函数
- 数据库锁(对应用来说不可控,隐锁)
  - 长事务不适合,降低了被锁记录的相关并发
  - 导致DEAD TUPLE无法回收.
- advisory锁(应用控制,显锁)
  - 特殊场景,如需要长时间持锁.
  - 但是又不能影响并发.
- 应用例子:
- <http://blog.163.com/digoal@126/blog/static/163877040201172492217830/>



# Functions and Operators

- 触发器函数
- test=# \sf suppress\_redundant\_updates\_trigger
- CREATE OR REPLACE FUNCTION  
pg\_catalog.suppress\_redundant\_updates\_trigger()
- RETURNS trigger
- LANGUAGE internal
- STRICT
- AS \$function\$suppress\_redundant\_updates\_trigger\$function\$
- 这个函数有什么用呢? 没有数据更新的更新操作不会产生新版本.
- test=# create table test (id int);
- test=# insert into test values (1),(2),(3);
- test=# select ctid,\* from test where id=1;
- ctid | id
- (0,1) | 1

# Functions and Operators

- 触发器函数
- test=# update test set id=1 where id=1;
- test=# select ctid,\* from test where id=1;
- ctid | id
- (0,4) | 1
  
- CREATE TRIGGER z\_min\_update
- BEFORE UPDATE ON test
- FOR EACH ROW EXECUTE PROCEDURE suppress\_redundant\_updates\_trigger();
- test=# update test set id=1 where id=1;
- **UPDATE 0**
- test=# select ctid,\* from test where id=1;
- ctid | id
- (0,4) | 1

# Type Conversion

- The PostgreSQL scanner/parser divides lexical elements into five fundamental categories: integers, non-integer numbers, strings, identifiers, and key words.
- 强类型指定可以提高性能.
- 以下4种SQL中的操作需要接受指定类型的数据
- Function
  - PostgreSQL中不是根据函数名来区分函数, 而是函数名和函数的参数类型. 同一个 SCHEMA中允许重名但参数个数不同或类型不完全相同的多个函数同时存在.
- Operator
  - 操作符在schema中也允许重名, 只要操作符的操作数不同
- Value Storage
  - INSERT, UPDATE
- UNION, CASE, ARRAY...
  - 合并(UNION)或选择性(CASE)多值输出时必须确保每列的输出类型一致, ARRAY也必须保证元素类型一致.
- 类型转换
  - CAST(value AS target\_type)
  - value::target\_type

# Index

## ■ Planner Methods

- #enable\_bitmapscan = on
- #enable\_hashagg = on
- #enable\_hashjoin = on
- #enable\_indexscan = on
- #enable\_material = on
- #enable\_mergejoin = on
- #enable\_nestloop = on
- #enable\_seqscan = on
- #enable\_sort = on
- #enable\_tidscan = on

## ■ 索引主要能干什么?

- 加速TUPLE定位
- 主键, 唯一约束
- 排序

# Index

## ■ Index Type OR Access Method

- b-tree
- hash
- gist
- gin
- spgist (PostgreSQL 9.2)

## ■ B-tree

- Operators
- $>$ ,  $<$ ,  $>=$ ,  $<=$ ,  $=$ , BETWEEN, IN, IS NULL, IS NOT NULL
- LIKE(开头匹配), ILIKE (大小写一致的字符开头匹配)

## ■ Hash

- Operator
- =
- 不记录WAL, 数据库crash后如果hash索引对应的表在恢复期间做过改动, hash需要重建.
- hash索引不支持基于wal或流复制的standby

# Index

## ■ GiST

- <<, &<, &>, >>, <<|, &<|, |&>, |>>, @>, <@, ~=, &&
- nearest-neighbor search

## ■ GIN

- <@, @>, =, &&

## ■ 建立索引的语法

- 注意, method, collation, opclass

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ name ] ON table [ USING method ]
  ( { column | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS { FIRST | LAST } ] [, ...] )
  [ WITH ( storage_parameter = value [, ...] ) ]
  [ TABLESPACE tablespace ]
  [ WHERE predicate ]
```

## ■ 建立operator class的语法, 通过系统表查询当前支持哪些operator class已经operator class中定义的operator和function, 是用于检索还是排序.

- pg\_am, pg\_amop, pg\_amproc, pg\_operator, pg\_opclass, pg\_opfamily

```
CREATE OPERATOR CLASS name [ DEFAULT ] FOR TYPE data_type
  USING index_method [ FAMILY family_name ] AS
  { OPERATOR strategy_number operator_name [ ( op_type, op_type ) ] [ FOR SEARCH | FOR ORDER BY sort_family_name ]
  | FUNCTION support_number [ ( op_type [, op_type] ) ] function_name ( argument_type [, ...] )
  | STORAGE storage_type
  } [, ... ]
```

# Index

- 用索引和操作符 $<->$ 快速检索与某point邻近的点举例
- 如果在location上建了一个GiST索引
- 那么可以通过这个索引快速的检索到离point '(101,456)'点最近的10个点的记录 如下:
- `SELECT * FROM places ORDER BY location <-> point '(101,456)' LIMIT 10;`

# Index

- 是否使用索引和什么有关?
- 首先是前面提到的Access Method, 然后是使用的operator class, 以及opc中定义的operator或function.
- 这些都满足后, 还要遵循CBO的选择.
  - #seq\_page\_cost = 1.0
  - #random\_page\_cost = 4.0
  - #cpu\_tuple\_cost = 0.01
  - #cpu\_index\_tuple\_cost = 0.005
  - #cpu\_operator\_cost = 0.0025
  - #effective\_cache\_size = 128MB
- 遵循完CBO的选择, 还需要符合当前配置的Planner 配置.
  - #enable\_bitmapscan = on
  - #enable\_hashagg = on
  - #enable\_hashjoin = on
  - #enable\_indexscan = on
  - #enable\_material = on
  - #enable\_mergejoin = on
  - #enable\_nestloop = on
  - #enable\_seqscan = on
  - #enable\_sort = on
  - #enable\_tidscan = on

# Index

- Multicolumn Index
  - only the B-tree, GiST and GIN index types support multicolumn indexes.
  - Up to 32 columns can be specified. (This limit can be altered when building PostgreSQL; see the file pg\_config\_manual.h.)
- B-tree
  - Query conditions that involve any subset of the index's columns, but the index is most efficient when there are constraints on the leading (leftmost) columns.
- GiST
  - Query conditions that involve any subset of the index's columns. Conditions on additional columns restrict the entries returned by the index, but the condition on the first column is the most important one for determining how much of the index needs to be scanned. A GiST index will be relatively ineffective if its first column has only a few distinct values, even if there are many distinct values in additional columns.
- GIN
  - A multicolumn GIN index can be used with query conditions that involve any subset of the index's columns. Unlike B-tree or GiST, **index search effectiveness is the same regardless of which index column(s) the query conditions use.**

# Index

## ■ Multicolumn Index

- 多列索引，使用任何列作为条件，只要条件中的操作符或函数能满足opclass的匹配，都可以使用索引，索引被扫描的部分还是全部基本取决于条件中是否有索引的第一列作为条件之一。

## ■ 例子

- postgres=# create table test (c1 int,c2 int);
- postgres=# insert into test select 1,generate\_series(1,100000);
- postgres=# create index idx\_test\_1 on test(c1,c2);
- postgres=# analyze test;
- postgres=# explain select \* from test where c2=100;
- Seq Scan on test (cost=0.00..1693.00 rows=1 width=8)
  - Filter: (c2 = 100)
- postgres=# set enable\_seqscan=off;
- postgres=# explain analyze select \* from test where c2=100;
- Index Scan using idx\_test\_1 on test (cost=0.00..1858.27 rows=1 width=8) (actual time=0.104..7.045 rows=1 loops=1)
  - Index Cond: (c2 = 100)

注意过滤条件  
不是驱动列。  
**看似不能走索引**

# Index

■ 使用索引来排序可以减少CPU对排序的开销, 特别是仅需返回少量行时. 使用索引效率会大大提高.

■ 例子

- postgres=# create table test (id int,info text);
- postgres=# insert into test select generate\_series(1,100000),'digoal'||generate\_series(1,100000);
- postgres=# explain analyze select \* from test order by id limit 10;
- Limit (cost=3701.96..3701.99 rows=10 width=36) (actual time=37.372..37.375 rows=10 loops=1)
  - -> Sort (cost=3701.96..3951.96 rows=100000 width=36) (actual time=37.370..37.371 rows=10 loops=1)
    - Sort Key: id
    - Sort Method: top-N heapsort Memory: 25kB
    - -> Seq Scan on test (cost=0.00..1541.00 rows=100000 width=36) (actual time=0.016..17.711 rows=100000 loops=1)
  - Total runtime: 37.405 ms
- postgres=# create index idx\_test\_id on test(id);
- postgres=# explain analyze select \* from test order by id limit 10;
- Limit (cost=0.00..0.48 rows=10 width=36) (actual time=0.052..0.058 rows=10 loops=1)
  - -> Index Scan using idx\_test\_id on test (cost=0.00..4768.26 rows=100000 width=36) (actual time=0.050..0.053 rows=10 loops=1)
- Total runtime: 0.085 ms

# Index

- 加速reference 表的reference colum的更新和删除操作
- postgres=# create table t1 (id int primary key,info text);
- postgres=# insert into t1 select generate\_series(1,100000),'digoal'||generate\_series(1,100000);
- postgres=# create table t2 (id int references t1(id) on update cascade,info text);
- postgres=# insert into t2 select generate\_series(1,100000),'digoal'||generate\_series(1,100000);
- postgres=# explain analyze update t1 set id=100001 where id=100000;
- Update on t1 (cost=4.27..8.28 rows=1 width=17) (actual time=0.066..0.066 rows=0 loops=1)
  - -> Bitmap Heap Scan on t1 (cost=4.27..8.28 rows=1 width=17) (actual time=0.021..0.022 rows=1 loops=1)
- Recheck Cond: (id = 100000)
  - -> Bitmap Index Scan on t1\_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.014..0.014 rows=1 loops=1)
- Index Cond: (id = 100000)
- Trigger for constraint t2\_id\_fkey on t1: time=21.082 calls=1
- Trigger for constraint t2\_id\_fkey on t2: time=0.094 calls=1
- Total runtime: 21.290 ms

# Index

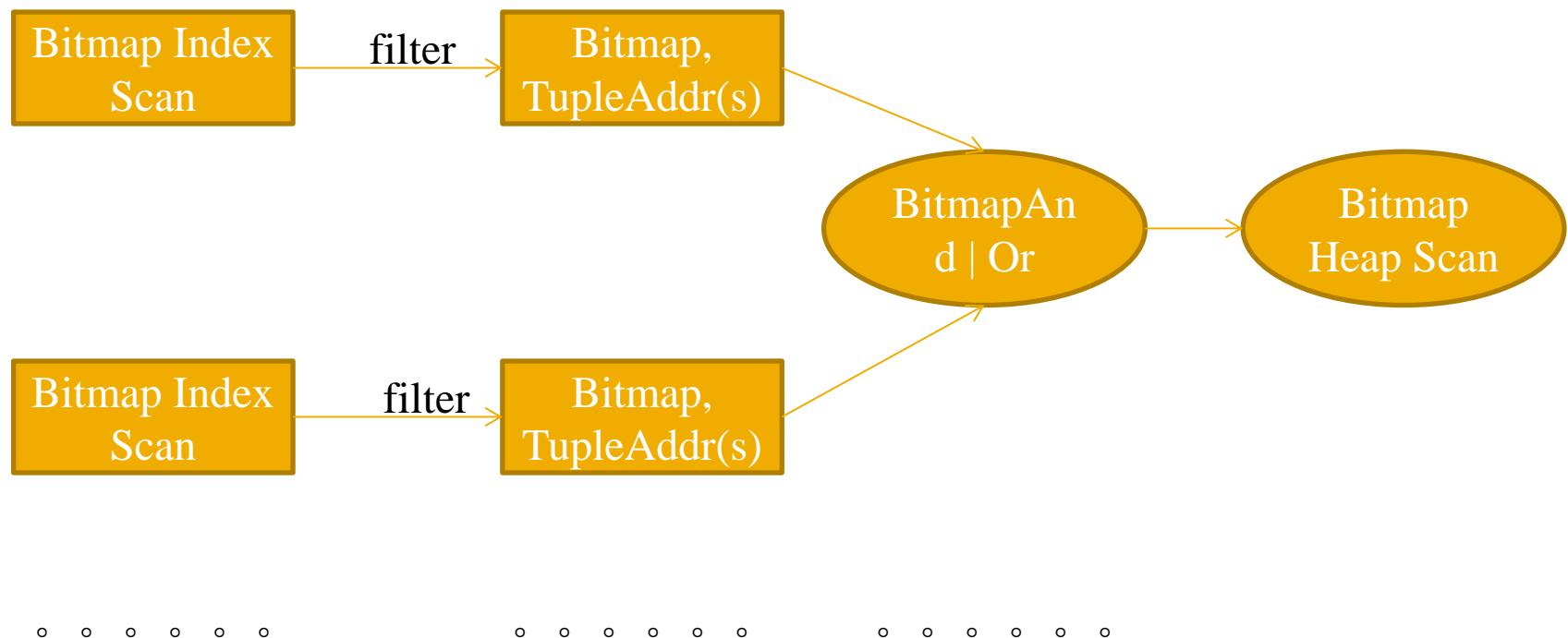
- postgres=# `create index idx_t2_id on t2(Id);`
- postgres=# `explain analyze update t1 set id=100002 where id=100001;`
- Update on t1 (cost=4.27..8.28 rows=1 width=17) (actual time=0.116..0.116 rows=0 loops=1)
  - > Bitmap Heap Scan on t1 (cost=4.27..8.28 rows=1 width=17) (actual time=0.031..0.032 rows=1 loops=1)
- Recheck Cond: (id = 100001)
- -> Bitmap Index Scan on t1\_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.020..0.020 rows=1 loops=1)
  - Index Cond: (id = 100001)
- Trigger for constraint t2\_id\_fkey on t1: time=0.516 calls=1
- Trigger for constraint t2\_id\_fkey on t2: time=0.058 calls=1
- Total runtime: 0.739 ms

# Index

- 唯一约束和主键字段创建唯一索引
- postgres=# create table test (id int primary key,info text unique);
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "test\_pkey" for table "test"
- NOTICE: CREATE TABLE / UNIQUE will create implicit index "test\_info\_key" for table "test"
- postgres=# \d test
  - Table "public.test"
  - Column | Type | Modifiers
  - -----+-----+-----
  - id | integer | not null
  - info | text |
- Indexes:
  - "test\_pkey" PRIMARY KEY, btree (id)
  - "test\_info\_key" UNIQUE CONSTRAINT, btree (info)

# Index

- Combining Multiple Indexes
- src/backend/executor
- 例如



# Index

## ■ Combining Multiple Indexes

### ■ one index combining

- postgres=# create table test (id int primary key,info text unique);
- postgres=# insert into test select generate\_series(1,1000000),'digoal'||generate\_series(1,1000000);
- postgres=# explain analyze select \* from test where id=1 or id=1000;
- **Bitmap Heap Scan** on test (cost=8.54..16.20 rows=2 width=36) (actual time=0.034..0.036 rows=2 loops=1)
  - Recheck Cond: ((id = 1) OR (id = 1000))
  - -> **BitmapOr** (cost=8.54..8.54 rows=2 width=0) (actual time=0.023..0.023 rows=0 loops=1)
    - -> **Bitmap Index Scan** on test\_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.012..0.012 rows=1 loops=1)
      - Index Cond: (id = 1)
      - -> **Bitmap Index Scan** on test\_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.009..0.009 rows=1 loops=1)
        - Index Cond: (id = 1000)

# Index

- multiple index combining
  - postgres=# explain analyze select \* from test where id=1 or info='digoal1000';
  - Bitmap Heap Scan on test (cost=8.55..16.22 rows=2 width=15) (actual time=0.038..0.040 rows=2 loops=1)
    - Recheck Cond: ((id = 1) OR (info = 'digoal1000'::text))
    - -> BitmapOr (cost=8.55..8.55 rows=2 width=0) (actual time=0.029..0.029 rows=0 loops=1)
      - -> Bitmap Index Scan on test\_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.012..0.012 rows=1 loops=1)
        - Index Cond: (id = 1)
      - -> Bitmap Index Scan on test\_info\_key (cost=0.00..4.28 rows=1 width=0) (actual time=0.017..0.017 rows=1 loops=1)
        - Index Cond: (info = 'digoal1000'::text)
    - Total runtime: 0.081 ms

# Index

- collection

- 例子

- CREATE TABLE test1c (
  - id integer,
  - content varchar COLLATE "x"
  - );
- CREATE INDEX test1c\_content\_index ON test1c (content);
- SELECT \* FROM test1c WHERE content > constant;
- -- 以下SQL不能使用索引test1c\_content\_index
- SELECT \* FROM test1c WHERE content > constant COLLATE "y";
- -- 需建立与y COLLATE对应的索引, 以上这条SQL才会走索引.
- CREATE INDEX test1c\_content\_y\_index ON test1c (content COLLATE "y");

# Index

- partial index
- 例子
- -- 部分约束
- --去除common值 id=1, 这个值有10W条, 走索引根本不合适. partial 索引很好的避免了此类情况.

- postgres=# create table test(id int,info text);
- postgres=# insert into test select 1,'digoal'||generate\_series(1,100000);
- postgres=# insert into test select generate\_series(1,1000),'digoal'||generate\_series(1,1000);
- postgres=# create index idx\_test\_1 on test(id) where id<>1;
- postgres=# explain select \* from test where id=1;
- Seq Scan on test (cost=0.00..1791.00 rows=100000 width=15)
  - Filter: (id = 1)
- postgres=# explain select \* from test where id=100;
- Index Scan using idx\_test\_1 on test (cost=0.00..8.27 rows=1 width=15)
  - Index Cond: (id = 100)

# Index

## ■ -- 非索引列的使用

- postgres=# explain select \* from test where info='digoal' and id=1;
- QUERY PLAN
- -----
- Seq Scan on test (cost=0.00..2041.00 rows=1 width=15)
- Filter: ((info = 'digoal')::text) AND (id = 1)
  
- postgres=# create index idx\_test\_2 on test(id) where info='digoal100';
- postgres=# explain select \* from test where info='digoal100';
- QUERY PLAN
- -----
- Index Scan using idx\_test\_2 on test (cost=0.00..8.27 rows=1 width=15)
- (1 row)

# Index

- -- 为什么要去除common 值
  - postgres=# drop index idx\_test\_1;
  - postgres=# drop index idx\_test\_2;
  - postgres=# explain select \* from test where id=1;
    - QUERY PLAN
    - -----
    - Seq Scan on test (cost=0.00..1791.00 rows=100000 width=15)
    - Filter: (id = 1)
    - -- 为什么会走全表扫描
  - postgres=# select id,count(\*) from test group by id order by count(\*) desc limit 10;
    - id | count
    - -----+-----
    - 1 | 100001
    - 120 | 1
    - 887 | 1
    - 681 | 1

# Index

- 函数索引和表达式索引
- 表达式索引
  - postgres=# explain select \* from test where id+1=100;
  - QUERY PLAN
  - -----
  - Seq Scan on test (cost=0.00..2059.86 rows=505 width=15)
  - Filter: ((id + 1) = 100)
  - postgres=# create index idx\_test\_1 on test((id+1));
  - CREATE INDEX
  - postgres=# explain select \* from test where id+1=100;
  - QUERY PLAN
  - -----
  - Bitmap Heap Scan on test (cost=12.18..577.45 rows=505 width=15)
  - Recheck Cond: ((id + 1) = 100)
  - -> Bitmap Index Scan on idx\_test\_1 (cost=0.00..12.05 rows=505 width=0)
  - Index Cond: ((id + 1) = 100)

# Index

## ■ 函数索引

- -- 以下区分大小写的场景无法使查询走普通的索引.
- postgres=# create table test (id int,info text,crt\_time timestamp(o));
- postgres=# insert into test select generate\_series(1,1000000),'digoal'||generate\_series(1,1000000),clock\_timestamp();
- postgres=# create index idx\_test\_info on test(info);
- postgres=# explain select \* from test where info ~\* '^a';  
Seq Scan on test (cost=0.00..1887.00 rows=10 width=23)  
Filter: (info ~\* '^a'::text)
- -- 忽略大小写的ilike和~\* 要走索引的话, 开头的字符只能是大小写一致的, 字母不行.  
数字可以. 例如字母a区分大小写, 数组o不区分大小写. 索引中的条目也就有差别.
- postgres=# explain select \* from test where info ~\* '^o';  
Index Scan using idx\_test\_info on test (cost=0.00..8.28 rows=10 width=23)  
Index Cond: ((info >= 'o'::text) AND (info < '1'::text))  
Filter: (info ~\* '^o'::text)

# Index

## ■ 函数索引

- -- 要让字母也可以走忽略大小写的索引如何做呢?
- -- 函数索引,但是函数必须是immutable状态的
- 过滤条件中也必须使用和创建的索引相同声明
- postgres=# select proname,provolatile from pg\_proc where proname='lower';
- proname | provolatile
- lower | i
  
- postgres=# create index idx\_test\_info\_1 on test(lower(info));
- CREATE INDEX
- postgres=# explain select \* from test where lower(info) ~ '^a';
- Bitmap Heap Scan on test (cost=13.40..648.99 rows=500 width=23)
- Filter: (lower(info) ~ '^a'::text)
- -> Bitmap Index Scan on idx\_test\_info\_1 (cost=0.00..13.27 rows=500 width=0)
- Index Cond: ((lower(info) >= 'a'::text) AND (lower(info) < 'b'::text))
- (4 rows)

# Index

## ■ 作为查询条件的函数 或 常量 或 变量 或 子查询

- 优化器需要知道给operator的参数值才能通过pg\_statistic中统计到的表柱状图来计算走索引还是走全表扫描或者其他planner的开销最小, 如果传入的是个变量则通常走默认的优先执行计划.
- postgres=# create index idx\_test\_1 on test (crt\_time);
- postgres=# select proname,proargtypes,provolatile from pg\_proc where prorettype in (1114,1184) order by proargtypes;
- | proname                              |              | proargtypes |  | provolatile |
|--------------------------------------|--------------|-------------|--|-------------|
| -----                                | -----+-----+ |             |  |             |
| transaction_timestamp                |              |             |  | s           |
| statement_timestamp                  |              |             |  | s           |
| pg_stat_get_bgwriter_stat_reset_time |              |             |  | s           |
| pg_conf_load_time                    |              |             |  | s           |
| pg_postmaster_start_time             |              |             |  | s           |
| pg_last_xact_replay_timestamp        |              |             |  | v           |

# Index

## ■ 作为查询条件的函数 或 常量 或 变量 或 子查询

- `clock_timestamp` | | v
- `now` | s

- `postgres=# explain select * from test where crt_time = clock_timestamp();`
- `Seq Scan on test (cost=0.00..2137.00 rows=100000 width=23)`
- `Filter: (crt_time = clock_timestamp())`
  
- `postgres=# explain select * from test where crt_time = now();`
- `Index Scan using idx_test_1 on test (cost=0.00..8.28 rows=1 width=23)`
- `Index Cond: (crt_time = now())`
  
- `postgres=# alter function now() strict volatile;`
- `postgres=# explain select * from test where crt_time = now();`
- `Seq Scan on test (cost=0.00..2137.00 rows=100000 width=23)`
- `Filter: (crt_time = now())`

# Index

## ■ 作为查询条件的函数 或 常量 或 变量 或 子查询

- postgres=# alter function clock\_timestamp() strict immutable;
- ALTER FUNCTION
- postgres=# explain select \* from test where crt\_time = clock\_timestamp();
- QUERY PLAN
- -----
- Index Scan using idx\_test\_1 on test (cost=0.00..8.28 rows=1 width=23)
- Index Cond: (crt\_time = '2012-04-30 15:32:02.559888+08'::timestamp with time zone)

## ■ 作为过滤条件的函数, immutable 和 stable 的函数在优化器开始计算COST前会把函数值算出来. 而volatile的函数, 是在执行SQL的时候运行的, 所以无法在优化器计算执行计划的阶段得到函数值, 也就无法和pg\_statistic中的信息比对到底是走索引呢还是全表扫描或其他执行计划.

# Index

- 表达式作为过滤条件时,同样的道理,表达式不会在优化器计算执行计划的过程中运算,所以也不能走最优的执行计划.

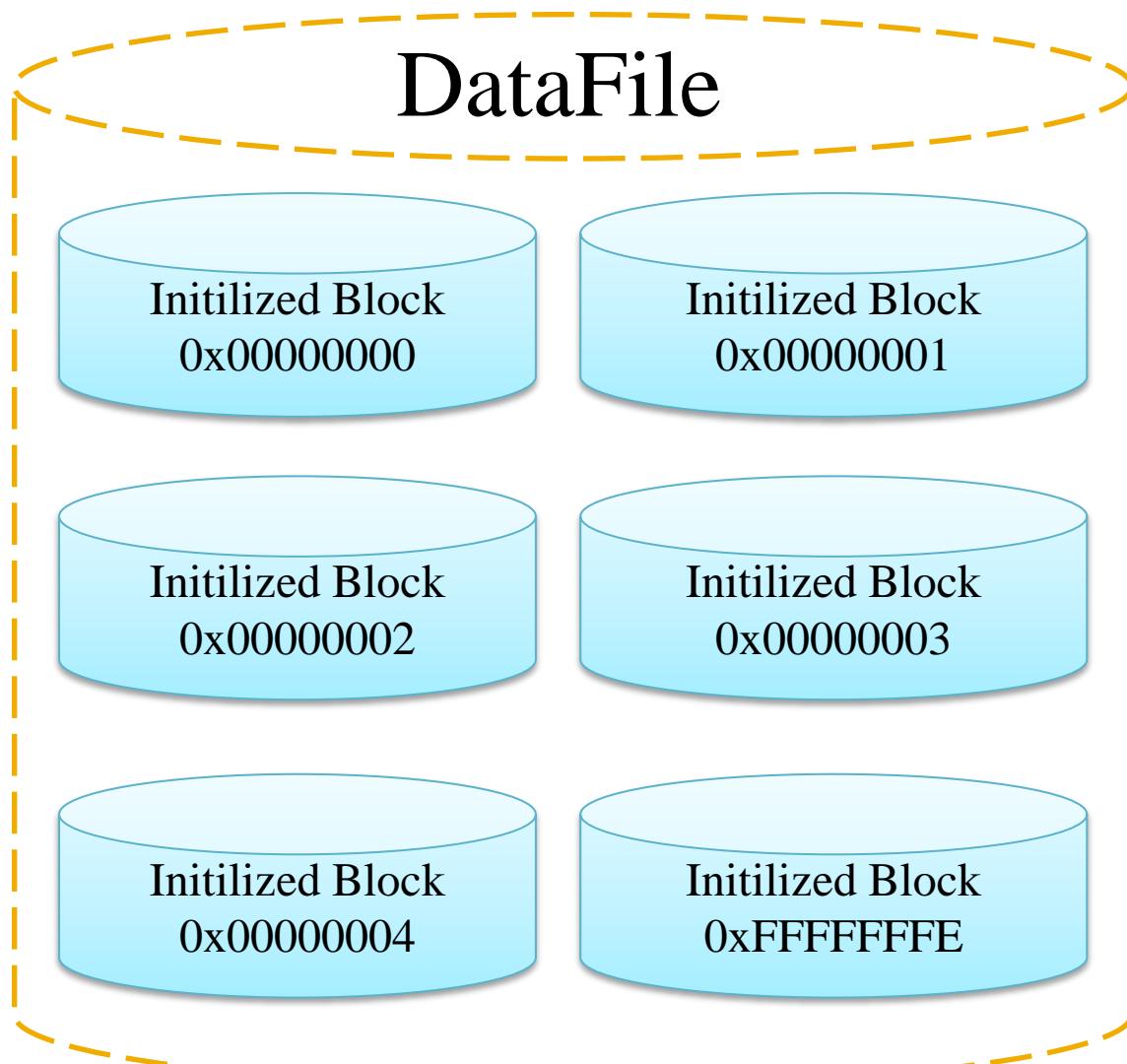
- postgres=# explain select \* from test where crt\_time = (select now());
  - QUERY PLAN
  - -----
  - Seq Scan on test (cost=0.01..1887.01 rows=100000 width=23)
  - Filter: (crt\_time = \$o)
  - InitPlan 1 (returns \$o)
  - -> Result (cost=0.00..0.01 rows=1 width=o)
  - (4 rows)

- 绑定变量是否走索引取决于驱动,机制和以上类似.(通常SESSION的第一个解析为硬解析,后面都是软解析.硬解析的时候的执行计划决定了后面的所有执行计划)

# Index

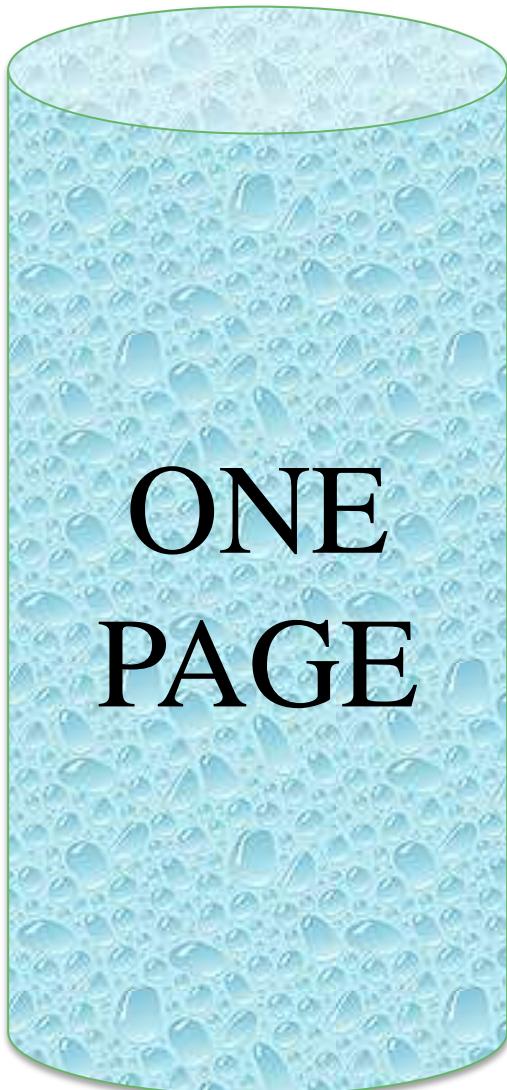
- 索引带来的Modify Overhead
- 降低Overhead的手段之一 : HOT
  
- 创建表时设置fillfactor < 100, 预留空间给HOT.
  
- 在了解HOT前先了解一下数据库的存储以及它们的数据结构.

# DataFile Storage Layout



One DataFile(s) Per Table or Index .  
BlockID :  
sequentially, 0 to 0xFFFFFFF4

# Page Layout



PageHeaderData(24 Bytes)

ItemIdData(Array of (offset,flag,length) pairs pointing to the actual items. 4 bytes per item)

Free space(The unallocated space.

New item pointers are allocated from the start of this area,  
new items from the end.)

Items (The actual items themselves.)

Special space (Index access method specific data.  
Different methods store different data. Empty  
in ordinary tables.)(an access method should always  
initialize its pages with PageInit  
and then set its own opaque fields.)

# PageHeader Layout

| Field               | Type          | Length  | Description                                                                |
|---------------------|---------------|---------|----------------------------------------------------------------------------|
| pd_lsn              | XLogRecPtr    | 8 bytes | LSN: next byte after last byte of xlog record for last change to this page |
| pd_tli              | uint16        | 2 bytes | TimeLineID of last change (only its lowest 16 bits)                        |
| pd_flags            | uint16        | 2 bytes | Flag bits                                                                  |
| pd_lower            | LocationIndex | 2 bytes | Offset to start of free space                                              |
| pd_upper            | LocationIndex | 2 bytes | Offset to end of free space                                                |
| pd_special          | LocationIndex | 2 bytes | Offset to start of special space                                           |
| pd_pagesize_version | uint16        | 2 bytes | Page size and layout version number information                            |
| pd_prune_xid        | TransactionId | 4 bytes | Oldest unpruned XMAX on page, or zero if none                              |

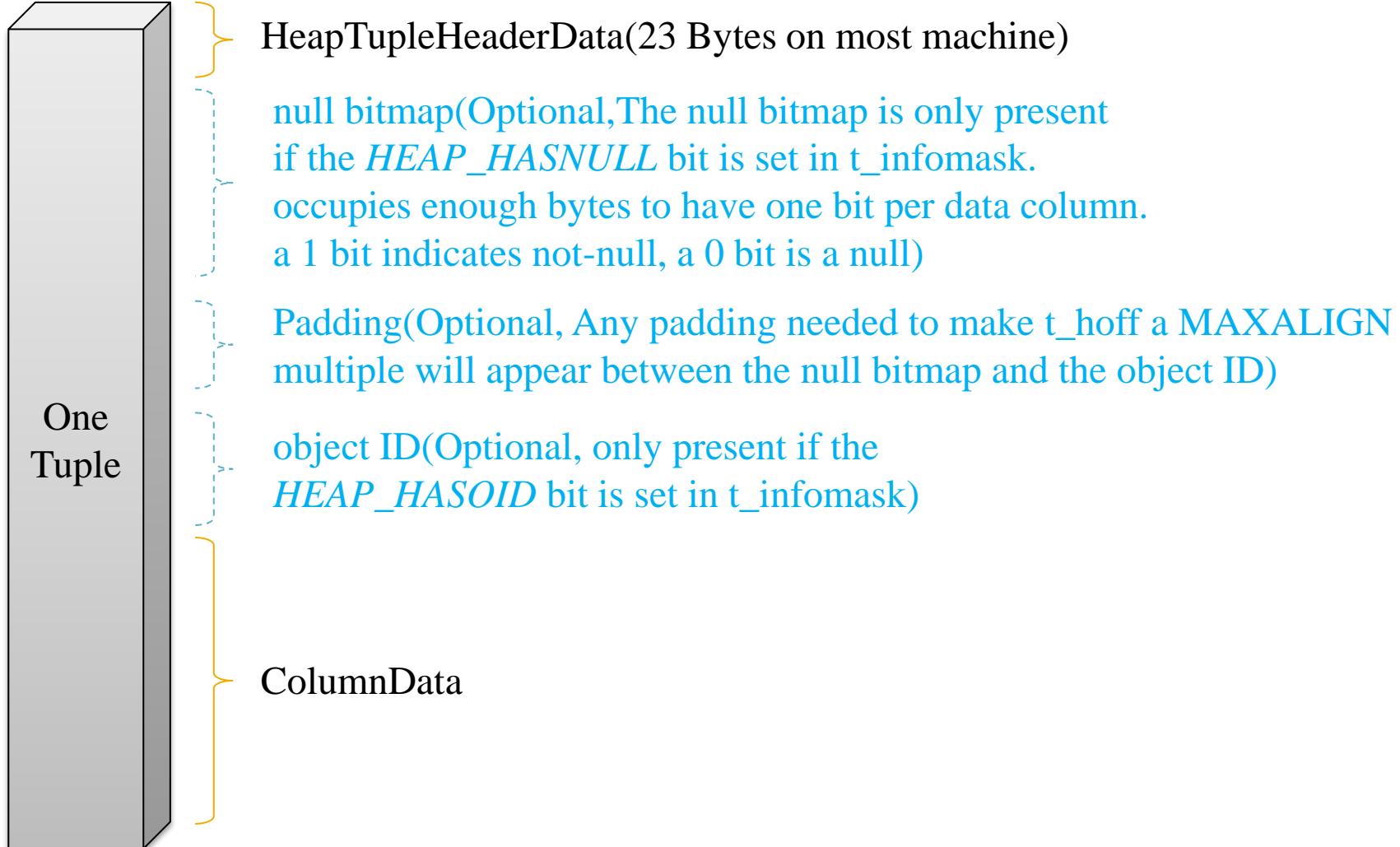
# pd\_flags define

```
■ /*
■ * pd_flags contains the following flag bits. Undefined bits are initialized
■ * to zero and may be used in the future.
■ *
■ * PD_HAS_FREE_LINES is set if there are any LP_UNUSED line pointers before
■ * pd_lower. This should be considered a hint rather than the truth, since
■ * changes to it are not WAL-logged.
■ *
■ * PD_PAGE_FULL is set if an UPDATE doesn't find enough free space in the
■ * page for its new tuple version; this suggests that a prune is needed.
■ * Again, this is just a hint.
■ */
■ #define PD_HAS_FREE_LINES      0x0001    /* are there any unused line pointers? */
■ #define PD_PAGE_FULL          0x0002    /* not enough free space for new tuple? */
■ #define PD_ALL_VISIBLE         0x0004    /* all tuples on page are visible to everyone */
■ #define PD_VALID_FLAG_BITS    0x0007    /* OR of all valid pd_flags bits */
```

# ItemIdData Layout

```
■ /* An item pointer (also called line pointer) on a buffer page */
■ /* In some cases an item pointer is "in use" but does not have any associated */
■ /* storage on the page. By convention, lp_len == 0 in every item pointer */
■ /* that does not have storage, independently of its lp_flags state. */
■ typedef struct ItemIdData
■ {
■     unsigned    lp_off:15,           /* offset to tuple (from start of page)*/
■                 lp_flags:2,         /* state of item pointer, see below */
■                 lp_len:15;          /* byte length of tuple */
■ }
■ } ItemIdData;
■ /* lp_flags has these possible states. An UNUSED line pointer is available */
■ /* for immediate re-use, the other states are not. */
■ #define LP_UNUSED          0           /* unused (should always have lp_len=0) */
■ #define LP_NORMAL           1           /* used (should always have lp_len>0) */
■ #define LP_REDIRECT          2           /* HOT redirect (should have lp_len=0) */
■ #define LP_DEAD              3           /* dead, may or may not have storage */
```

# Tuple Layout



# HeapTupleHeader Layout

| Field       | Type            | Length  | Description                                           |
|-------------|-----------------|---------|-------------------------------------------------------|
| t_xmin      | TransactionId   | 4 bytes | insert XID stamp                                      |
| t xmax      | TransactionId   | 4 bytes | delete XID stamp                                      |
| t_cid       | CommandId       | 4 bytes | insert and/or delete CID stamp (overlays with t_xvac) |
| t_xvac      | TransactionId   | 4 bytes | XID for VACUUM operation moving a row version         |
| t_ctid      | ItemPointerData | 6 bytes | current TID of this or newer row version              |
| t_infomask2 | int16           | 2 bytes | number of attributes, plus various flag bits          |
| t_infomask  | uint16          | 2 bytes | various flag bits                                     |
| t_hoff      | uint8           | 1 byte  | offset to user data                                   |

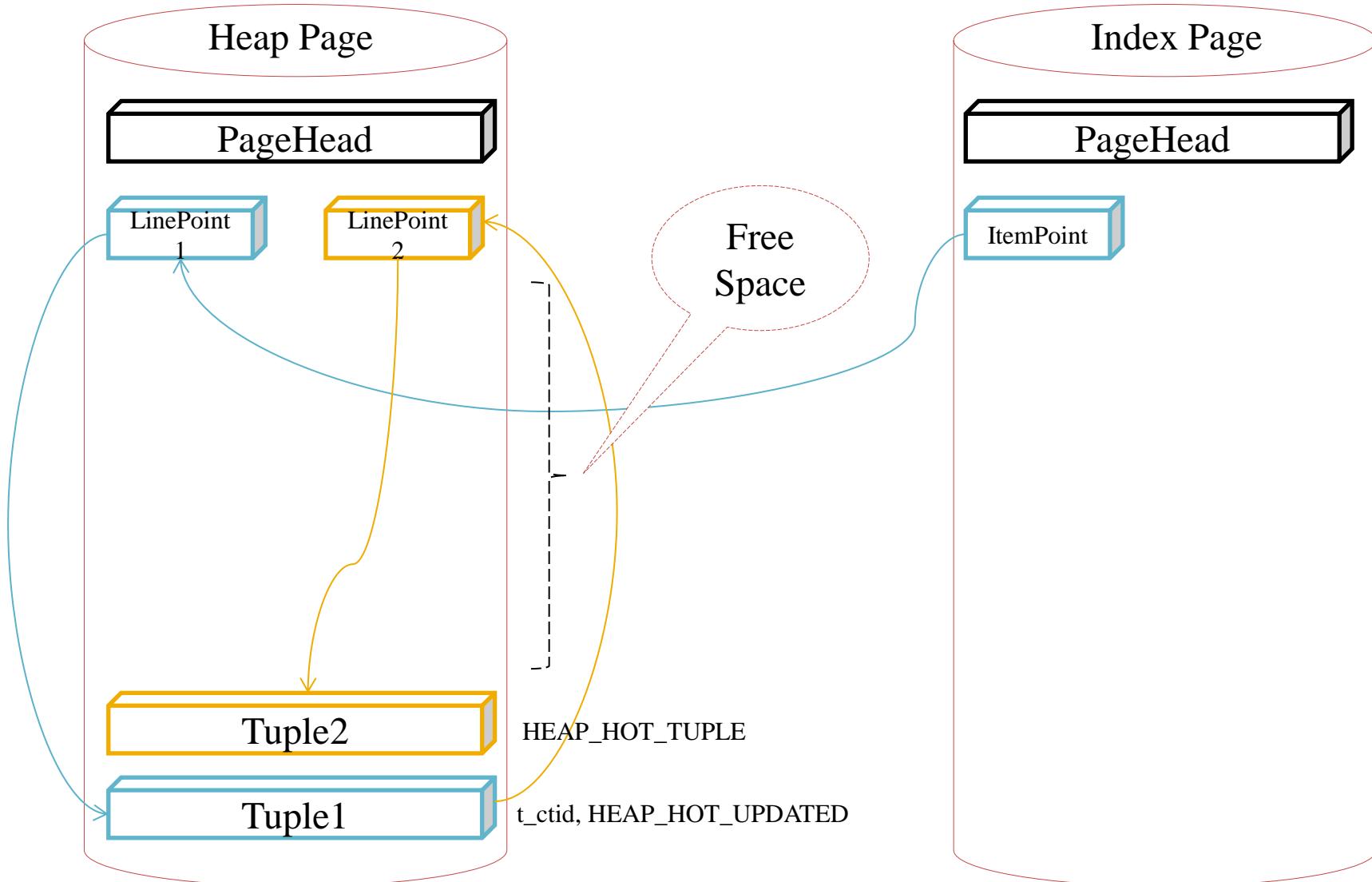
# INDEX Pointer Introduction

- ItemPointers (index) ->
- ItemId数据结构
  - (Array of (lp\_off:15bit,  
lp\_flags:2bit,lp\_len:15bit) pairs pointing to the  
actual items. 4 bytes per ItemId.)
- -> Item (tuple)

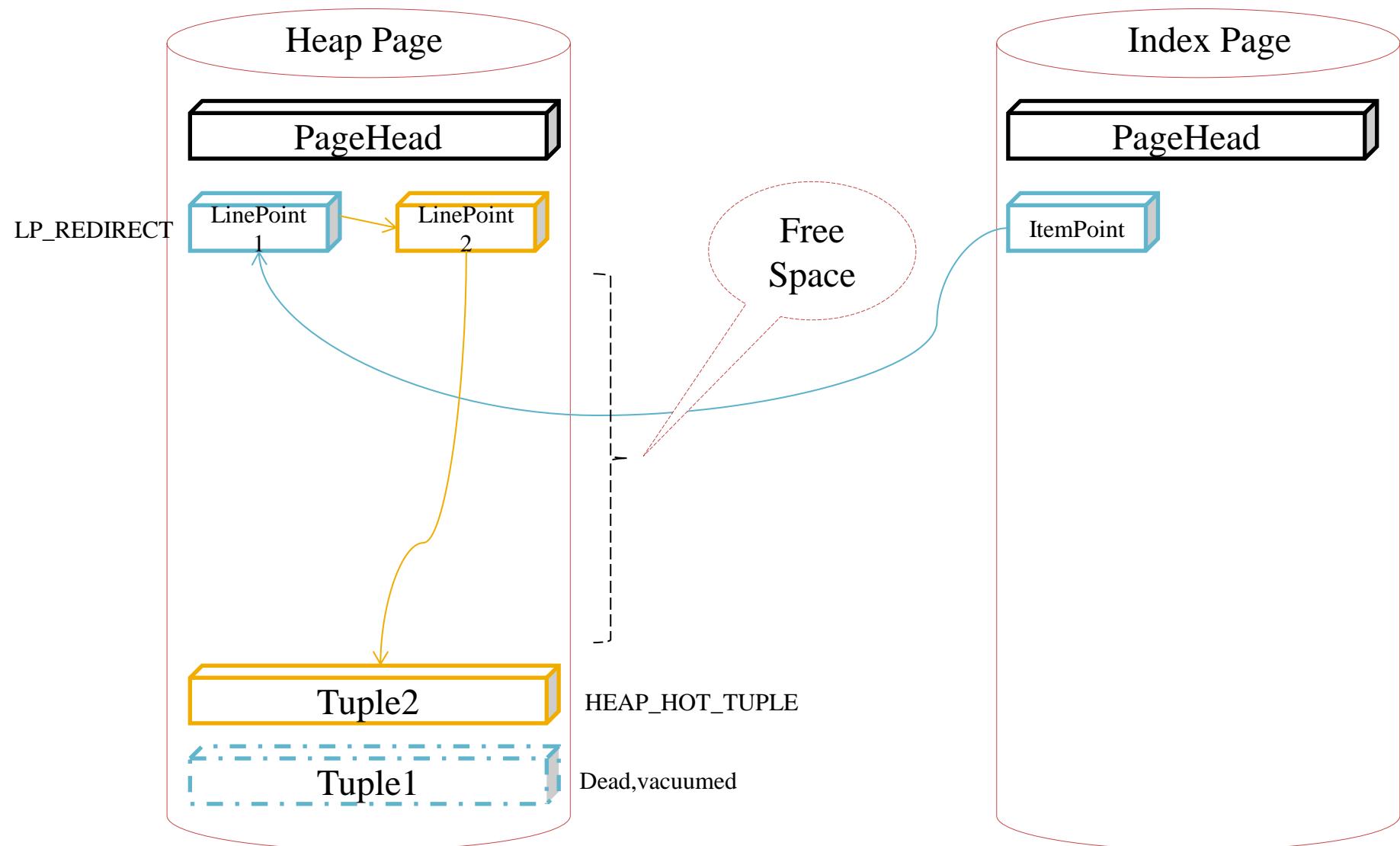
# HOT Introduction

- Heap-Only Tuple Benefit :
  - eliminates redundant index entries
  - allows the re-use of space taken by DELETED or obsoleted UPDATED tuples without performing a table-wide vacuum.
- Example
  - Update 1: Index points to 1
  - line points [1] [2]
  - Items [111111111]->[2222222222]
  - Update 2: Index points to 1
  - line point [1]->[2]
  - Items [222222222]
  - Update 3: Index points to 1
  - line points [1]->[2] [3]
  - Items [222222222]->[3333333333]
  - Update 4: Index points to 1
  - line points [1]----->[3]
  - Items [333333333]

# HOT Update



# HOT Update



# HOT Update

- 利用pageinspect extension 来观察HOT
  - postgres=# create extension pageinspect;
  - postgres=# create table hot\_test (id int primary key,info text);
  - postgres=# insert into hot\_test values (1,'digoal');
  - -- 因为是从0号page开始插入, 这里就省去了查询ctid等过程.直接切入0号page.
  - -- 当前的page信息
    - postgres=# select \* from page\_header(get\_raw\_page('hot\_test',0));

| lsn        | tli | flags | lower | upper | special | pagesize | version | prune_xid |
|------------|-----|-------|-------|-------|---------|----------|---------|-----------|
| 2/75B27878 | 1   | 0     | 28    | 8152  | 8192    | 8192     | 4       | 0         |
  - -- 当前的item信息
    - postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test',0));

| lp | lp_off | lp_flags | lp_len | t_xmin | t xmax | t_field3  | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid |
|----|--------|----------|--------|--------|--------|-----------|--------|-------------|------------|--------|--------|-------|
| 1  | 8152   | 1        | 35     | 1864   | 0      | 0   (0,1) | 2      | 2050        | 24         |        |        |       |



# HOT Update

# HOT Update

- -- 更新一次后
  - postgres=# update hot\_test set info='new' where id=1;
  - -- item信息
  - postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test',0));
  - lp | lp\_off | lp\_flags | lp\_len | t\_xmin | t xmax | t\_field3 | t\_ctid | t\_infomask2 | t\_infomask | t\_hoff | t\_bits | t\_oid
  - 1 | 8152 | 1 | 35 | 1864 | 1867 | 0 | (0,2) | 16386 | 258 | 24 | |
  - 2 | 8120 | 1 | 32 | 1867 | 0 | 0 | (0,2) | 32770 | 10242 | 24 | |
- itemID中的信息
- tuple中的信息,对应第一幅图
- -- 索引的item信息(没有变化)
  - postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test\_pkey',0));
  - lp | lp\_off | lp\_flags | lp\_len | t\_xmin | t xmax | t\_field3 | t\_ctid | t\_infomask2 | t\_infomask | t\_hoff | t\_bits | t\_oid
  - -- 内容略

# HOT Update

- -- vacuum 后
- postgres=# vacuum hot\_test ;
- VACUUM
- postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test',0));

|    |        |          |        |        |        |           |        |             |            |        |        |       |
|----|--------|----------|--------|--------|--------|-----------|--------|-------------|------------|--------|--------|-------|
| lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3  | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid |
| 1  | 2      | 2        | 0      |        |        |           |        |             |            |        |        |       |
| 2  | 8160   | 1        | 32     | 1867   | 0      | 0   (0,2) | 32770  | 10498       | 24         |        |        |       |
- -- 多次更新后
  - postgres=# update hot\_test set info='new' where id=1;
  - postgres=# update hot\_test set info='new' where id=1;

itemID中的信息  
对应第二幅图

# HOT Update

- postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test',0));  
lp | lp\_off | lp\_flags | lp\_len | t xmin | t xmax | t\_field3 | t\_ctid | t\_infomask2 | t\_infomask | t\_hoff | t\_bits | t\_oid

|   |      |   |    |      |      |           |       |       |    |  |  |  |
|---|------|---|----|------|------|-----------|-------|-------|----|--|--|--|
| 1 | 2    | 2 | 0  |      |      |           |       |       |    |  |  |  |
| 2 | 8160 | 1 | 32 | 1867 | 1868 | 0   (0,3) | 49154 | 9474  | 24 |  |  |  |
| 3 | 8128 | 1 | 32 | 1868 | 1869 | 0   (0,4) | 49154 | 9474  | 24 |  |  |  |
| 4 | 8096 | 1 | 32 | 1869 | 1870 | 0   (0,5) | 49154 | 9474  | 24 |  |  |  |
| 5 | 8064 | 1 | 32 | 1870 | 1871 | 0   (0,6) | 49154 | 9474  | 24 |  |  |  |
| 6 | 8032 | 1 | 32 | 1871 | 1872 | 0   (0,7) | 49154 | 9474  | 24 |  |  |  |
| 7 | 8000 | 1 | 32 | 1872 | 1873 | 0   (0,8) | 49154 | 8450  | 24 |  |  |  |
| 8 | 7968 | 1 | 32 | 1873 | 0    | 0   (0,8) | 32770 | 10242 | 24 |  |  |  |

注意redirect后,lp\_off的值表示第几条itemid, 而不是offset\_bytes.

# HOT Update

- -- vacuum后
- postgres=# vacuum hot\_test ;
- postgres=# select \* from heap\_page\_items(get\_raw\_page('hot\_test',0));
- lp | lp\_off | lp\_flags | lp\_len | t xmin | t xmax | t field3 | t ctid | t\_infomask2 | t\_infomask | t\_hoff | t\_bits | t\_oid

|   |      |   |    |      |   |           |       |       |    |  |  |  |
|---|------|---|----|------|---|-----------|-------|-------|----|--|--|--|
| 1 | 8    | 2 | 0  |      |   |           |       |       |    |  |  |  |
| 2 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 3 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 4 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 5 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 6 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 7 | 0    | 0 | 0  |      |   |           |       |       |    |  |  |  |
| 8 | 8160 | 1 | 32 | 1873 | 0 | 0   (0,8) | 32770 | 10498 | 24 |  |  |  |

- Use pageinspect EXTENSION view PostgreSQL Page's raw infomation
- <http://blog.163.com/digoal@126/blog/static/16387704020114273265960/>

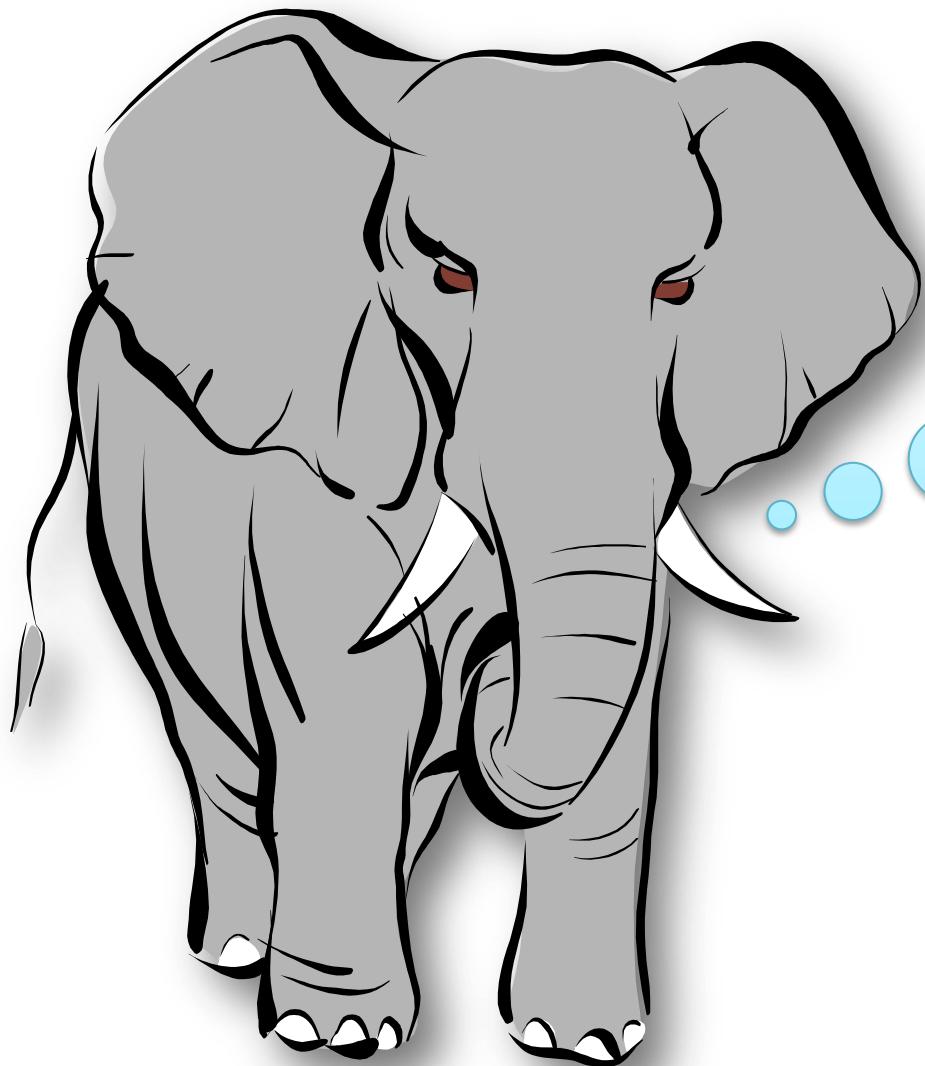
# Index

- 为什么要使用concurrently创建索引?
- 如果不使用concurrently, 创建索引时不允许对表进行增删改操作, 只允许查询操作.
- <http://blog.163.com/digoal@126/blog/static/163877040201231781923116/> 
- 索引快还是全表扫描快?
- 取决于缓存的大小,存储的IOPS能力, 是否使用索引排序以及SQL需要发起的IO次数(索引, 离散IO. 全表扫描, 顺序IO)等.
  
- 我建的索引到底有没有被系统用到, 还是说它就是个费索引
- explain
- pg\_statio\_all\_indexes
  - For each index in the current database, the table and index OID, schema, table and index name, numbers of disk blocks read and buffer hits in that index.
- pg\_stat\_all\_indexes
  - For each index in the current database, the table and index OID, schema, table and index name, number of index scans initiated on that index, number of index entries returned by index scans, and number of live table rows fetched by simple index scans using that index.

# Full Text Search

- 本期培训略
- 支持PostgreSQL的全文检索软件如sphinx.

# Concurrency Control



EveryOne Must Know  
Bussiness Rules &  
PostgreSQL  
ISOLATION LEVEL

# Concurrency Control

- SQL标准定义的隔离级别和读保护对应关系
- 注意SQL标准: **minimum** protections each isolation level must provide

| Isolation Level  | Dirty Read   | Nonrepeatable Read | Phantom Read |
|------------------|--------------|--------------------|--------------|
| Read uncommitted | Possible     | Possible           | Possible     |
| Read committed   | Not possible | Possible           | Possible     |
| Repeatable read  | Not possible | Not possible       | Possible     |
| Serializable     | Not possible | Not possible       | Not possible |

- PostgreSQL实现的隔离级别和读保护对应关系

| Isolation Level  | Dirty Read   | NonRepeatable Read | Phantom Read |
|------------------|--------------|--------------------|--------------|
| Read uncommitted | Not possible | Possible           | Possible     |
| Read committed   | Not possible | Possible           | Possible     |
| Repeatable read  | Not possible | Not possible       | Not possible |
| Serializable     | Not possible | Not possible       | Not possible |

- Real Serializable 除此以外还实现了并行事务串行化的组合检测.

# Concurrency Control

- dirty read
  - A transaction reads data written by a concurrent uncommitted transaction.
- nonrepeatable read
  - A transaction re-reads data it has previously read and finds that data has been modified by another transaction (that committed since the initial read).
  - 即一个事务中分两次执行同样的SQL, 查询某数据时, 前后得到的结果不一致. 也就是说在这两个SQL之间有另一个事务把该数据修改并提交了.
- phantom read
  - A transaction re-executes a query returning a set of rows that satisfy a search condition and finds that the set of rows satisfying the condition has changed due to another recently-committed transaction.
  - 即一个事务中分两次执行了同样的SQL, 查询一个区间的数据, 前后得到的结果不一致. 后面的SQL可能得到更多的数据, 例如在这两个SQL之间另一个事务插入了一些在这个查询的区间范围内的数据并提交了.

# Concurrency Control

- 为什么PostgreSQL实现的隔离级别对读保护超出了SQL标准定义的最小保护
- PostgreSQL并发控制的手段, MVCC
- xmin, xmax, xid
- INSERT, DELETE, UPDATE
  
- INSERT, xmin = current xid
- DELETE, xmax = current xid
- UPDATE, old tuple xmax = current xid, new tuple xmin = current xid
  
- 因此PostgreSQL很容易通过MVCC来实现不同的隔离级别.
- 可以理解为如下, 当然内部实现比这复杂得多
  - read committed
    - 拿系统已分配出去的最后一个事务ID作为比较, 去除未提交的那些事务ID, 能见到所有小于等于这个事务ID的所有行.
  - repeatable read
    - 记录下事务开始时有哪些未提交的事务, 事务中执行的SQL拿事务开始时的事务ID作为比较, 去除事务开始时未提交的事务, 能见到的记录范围是小于等于这个事务ID的所有行.

# Concurrency Control

## ■ txid\_current\_snapshot() 函数

| Name     | Description                                                                                                                                                                                                                                                                                                                                                                                        |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xmin     | Earliest transaction ID (txid) that is still active. All earlier transactions will either be committed and visible, or rolled back and dead.                                                                                                                                                                                                                                                       |
| xmax     | First as-yet-unassigned txid. All txids greater than or equal to this are not yet started as of the time of the snapshot, and thus invisible.                                                                                                                                                                                                                                                      |
| xip_list | Active txids at the time of the snapshot. The list includes only those active txids between xmin and xmax; there might be active txids higher than xmax. A txid that is $xmin \leq \text{txid} < xmax$ and not in this list was already completed at the time of the snapshot, and thus either visible or dead according to its commit status. The list does not include txids of subtransactions. |

# Concurrency Control

## ■ Session A:

- digoal=> create table tbl\_user (id int,firstname varchar(64),lastname varchar(64),corp varchar(64),age int);
- digoal=> insert into tbl\_user values (1,'zhou','digoal','sky-mobi',27);
- digoal=> select ctid,xmin,xmax,cmin,cmax,\* from tbl\_user;
- ctid | xmin | xmax | cmin | cmax | id | firstname | lastname | corp | age
- (0,1) | 3909 | o | o | o | 1 | zhou | digoal | sky-mobi | 27

## ■ Session B:

- digoal=> select ctid,xmin,xmax,cmin,cmax,\* from tbl\_user;
- ctid | xmin | xmax | cmin | cmax | id | firstname | lastname | corp | age
- (0,1) | 3909 | o | o | o | 1 | zhou | digoal | sky-mobi | 27

# Concurrency Control

## ■ Session A :

- digoal=> begin;
- digoal=> update tbl\_user set id=2 where id=1;
- digoal=> select ctid,xmin,xmax,cmin,cmax,\* from tbl\_user;
- ctid | xmin | xmax | cmin | cmax | id | firstname | lastname | corp | age
- (0,2) | 3910 | 0 | 0 | 2 | zhou | digoal | sky-mobi | 27
- digoal=> select txid\_current\_snapshot();
- 3910:3914:

## ■ Session B :

- select ctid,xmin,xmax,cmin,cmax,\* from tbl\_user;
- ctid | xmin | xmax | cmin | cmax | id | firstname | lastname | corp | age
- (0,1) | 3909 | 3910 | 0 | 0 | 1 | zhou | digoal | sky-mobi | 27
- digoal=> select txid\_current\_snapshot();
- 3910:3914:3910

# Concurrency Control

- read committed 隔离级别用例
  - BEGIN;
  - UPDATE accounts SET balance = balance + 100.00 WHERE acctnum = 12345;
  - UPDATE accounts SET balance = balance - 100.00 WHERE acctnum = 7534;
  - COMMIT;
  
- read committed隔离级别不适合以下场景
  - -- assume website is a two-row table with website.hits equaling 9 and 10
  - BEGIN;
  - UPDATE website SET hits = hits + 1;
  - -- 9改为10, 10改为11, 同时这两行被加锁
  - -- run from another session: DELETE FROM website WHERE hits = 10;
  - -- 另一个session它不能看到未提交的记录, 它等待的锁是老记录的行锁, hits实际上已经被修改为11, 也就是当前有两条记录
  - -- xmin 有值,xmax 有值, 10 -- xmin 有值,xmax=0, 11 -- 另一个session在等待的是xmin,xmax都有值的那条老的记录的锁释放.
  - COMMIT;
  - -- 当事务1提交后, 另一个事务同时获得了这个锁, 但是它所看到的这条记录的hits目前是11. 索引会导致delete0条记录的情况.

# Concurrency Control

- Repeatable Read Isolation Level
- 事务开始后同样的SQL不管执行多少次都返回同样的结果.
- 但是repeatable read事务不能修改在repeatable read事务执行过程中被其他事务修改并提交了的记录. 否则会抛出异常.
- ERROR: could not serialize access due to concurrent update
  
- 例如 :
- postgres=# insert into test values (1,'digoal1'),(100,'digoal100');
- commit;
- SESSION A:
- postgres=# begin transaction isolation level repeatable read;
- postgres=# select \* from test where id=1;
- id | info
- -----+-----
- 1 | digoal1

# Concurrency Control

- SESSION B:
  - postgres=# begin;
  - postgres=# update test set info='new\_digoal' where id=1;
  - postgres=# commit;
  
- SESSION A:
  - postgres=# select \* from test where id=1;
    - id | info
    - -----+-----
    - 1 | digoal1
  - postgres=# select count(\*) from test;
    - count
    - -----
    - 2

# Concurrency Control

- SESSION B:
  - postgres=# begin;
  - postgres=# insert into test select generate\_series(1000,1100),'digoal';
  - INSERT 0 101
  - postgres=# end;
  - -- 这个在其他数据库(如oracle)中需要serializable read隔离级别才能实现.
- SESSION A:
  - postgres=# select count(\*) from test;
  - count
    - -----
    - 2
  - postgres=# update test set info='session a' where id=1;
  - ERROR: could not serialize access due to concurrent update
  - postgres=# end;
  - ROLLBACK

# Concurrency Control

## ■ SERIALIZABLE READ

- In fact, this isolation level works exactly the same as Repeatable Read except that it monitors for conditions which could make execution of a concurrent set of serializable transactions behave in a manner inconsistent with all possible serial (one at a time) executions of those transactions. This monitoring does not introduce any blocking beyond that present in repeatable read, but there is some overhead to the monitoring, and detection of the conditions which could cause a serialization anomaly will trigger a serialization failure.
- SIReadLock
  - The particular locks acquired during execution of a query will depend on the plan used by the query, and multiple finer-grained locks (e.g., tuple locks) may be combined into fewer coarser-grained locks (e.g., page locks) during the course of the transaction to prevent exhaustion of the memory used to track the locks. A READ ONLY transaction may be able to release its SIRead locks before completion, if it detects that no conflicts can still occur which could lead to a serialization anomaly. In fact, READ ONLY transactions will often be able to establish that fact at startup and avoid taking any predicate locks. If you explicitly request a SERIALIZABLE READ ONLY DEFERRABLE transaction, it will block until it can establish this fact. (This is the **only** case where Serializable transactions block but Repeatable Read transactions don't.) On the other hand, SIRead locks often need to be kept past transaction commit, until overlapping read write transactions complete.
- 成功的提交并行的serializable read事务表示这些事务不管以什么顺序执行得到的结果都是一样的. 否则必将有事务会失败. 而应用必须要能够应对这种失败, 例如重新执行一遍失败的事务并再提交.

# Concurrency Control

## ■ SERIALIZABLE READ

■ 用例

■ class | value

-----+-----

■ 1 | 10

■ 1 | 20

■ 2 | 100

■ 2 | 200

## ■ SERIALIZABLE SESSION A:

■ SELECT SUM(value) FROM mytab WHERE class = 1;

■ inserts the result (30) as the value in a new row with class = 2;

## ■ SERIALIZABLE SESSION B:

■ SELECT SUM(value) FROM mytab WHERE class = 2;

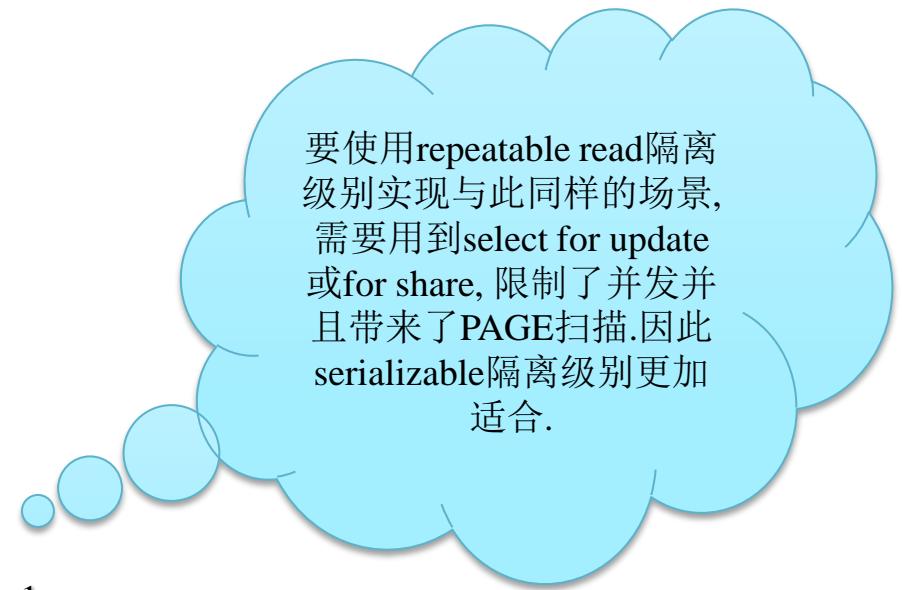
■ obtains the result 300, which it inserts in a new row with class = 1;

## ■ SERIALIZABLE SESSION A:

■ COMMIT;

## ■ SERIALIZABLE SESSION B:

■ COMMIT; 失败, 反之亦然



要使用repeatable read隔离级别实现与此同样的场景, 需要用到select for update或for share, 限制了并发并且带来了PAGE扫描.因此serializable隔离级别更加适合.

# Concurrency Control

- SERIALIZABLE READ
- 事务冲突检测条件：
  - 首先圈定发生冲突的事务的隔离级别 := serializable 。
  - 换句话说冲突只会在serializable 和serializable的事务之间发生，不会发生在serializable和read committed的事务之间等等。后面会有例子。
  - 2. 发生冲突的事务必须存在至少同一个表的操作交集。（一个事务查询了某些记录，另一个事务更新或删除了这些记录中的部分或全部。或者两个事务同时查询了相同的记录。）
  - 3. 发生冲突的事务必须对产生交集的表都有写的操作(insert,delete,update之一)，并且每个事务的写操作必须至少影响1行记录及以上。
  - 4. 发生冲突的事务都必须有2条或以上SQL(当DML和DSL没有已经存在的行交集时)，或者其中一个SESSION的DML（update,delete）与另一个SESSION有行交集。
- 当冲突发生时，第一时间提交的事务可以成功返回，在冲突域里面，后提交的所有事务都被自动 ROLLBACK。并且报错：
  - ERROR: could not serialize access due to read/write dependencies among transactions
  - DETAIL: Reason code: Canceled on identification as a pivot, during commit attempt .
  - HINT: The transaction might succeed if retried .

# Concurrency Control

- SERIALIZABLE READ
- 以下情况不会触发could not serialize access due to read/write dependencies among transactions错误
- 所以SERIALIZABLE不能帮你实现这种BUSSINESS RULE
- postgres=# create table t1 (class int, value int);
- postgres=# create table t2 (class int, value int);
- postgres=# insert into t1 values (1,10),(1,20),(2,100),(2,200);
  
- SESSION A:
- postgres=# begin transaction isolation level serializable;
- postgres=# select sum(value) from t1 where class=1;
- sum
- -----
- 30
- (1 row)
  
- SESSION B:
- postgres=# begin transaction isolation level serializable;

# Concurrency Control

- postgres=# select sum(value) from t1 where class=2;
- sum
- -----
- 300
- (1 row)
  
- SESSION A:
- postgres=# insert into t1 values(2,30);
- -- 注意下面要插入的是另一张表.
- SESSION B:
- postgres=# insert into t2 values (1,300);
  
- SESSION B:
- postgres=# end;
- 成功
  
- SESSION A:
- postgres=# end;
- 成功, -- 多希望他失败啊. 因为它也用到了B所改变的记录啊.

# Concurrency Control

- serializable使用注意实现以及优化建议
- Declare transactions as READ ONLY when possible.
- Control the number of active connections, using a connection pool if needed. This is always an important performance consideration, but it can be particularly important in a busy system using Serializable transactions.
- Don't put more into a single transaction than needed for integrity purposes.
- Don't leave connections dangling "idle in transaction" longer than necessary.
- Eliminate explicit locks, SELECT FOR UPDATE, and SELECT FOR SHARE where no longer needed due to the protections automatically provided by Serializable transactions.
- When the system is forced to combine multiple page-level predicate locks into a single relation-level predicate lock because the predicate lock table is short of memory, an increase in the rate of serialization failures may occur. You can avoid this by increasing max\_pred\_locks\_per\_transaction.
- A sequential scan will always necessitate a relation-level predicate lock. This can result in an increased rate of serialization failures. It may be helpful to encourage the use of index scans by reducing random\_page\_cost and/or increasing cpu\_tuple\_cost. Be sure to weigh any decrease in transaction rollbacks and restarts against any overall change in query execution time.
- hot standby最高只支持到repeatable read隔离级别.
- PostgreSQL 9.1 serializable isolation conflict occur condition and compared with Oracle
- <http://blog.163.com/digoal@126/blog/static/16387704020118162950691/>



# Concurrency Control

- 表级锁冲突模式
  - Command: LOCK
  - Description: lock a table
  - Syntax:  
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

# Concurrency Control

- 行锁
  - select for update
    - 相互冲突
  - select for share
    - 相互不冲突, 但是不允许修改或删除被锁的行.
    - \$PGDATA/pg\_multixact中存储了此类信息
  - 行锁 modifies selected rows to mark them locked, and so will result in disk writes.
- 页锁
  - In addition to table and row locks, page-level share/exclusive locks are used to control read/write access to table pages in the shared buffer pool. These locks are released immediately after a row is fetched or updated.
- 死锁
  - 事务直接形成了相互等待的局面, 可以发生在两个或以上事务中, 是应用设计的时候需要避免的

# Concurrency Control

- Lock and Index
- Though PostgreSQL provides nonblocking read/write access to table data, nonblocking read/write access is not currently offered for every index access method implemented in PostgreSQL. The various index types are handled as follows:
  - B-tree and GiST indexesShort-term share/exclusive page-level locks are used for read/write access. Locks are released immediately after each index row is fetched or inserted. These index types provide the highest concurrency without deadlock conditions.
  - Hash indexesShare/exclusive hash-bucket-level locks are used for read/write access. Locks are released after the whole bucket is processed. Bucket-level locks provide better concurrency than index-level ones, but deadlock is possible since the locks are held longer than one index operation.
  - GIN indexesShort-term share/exclusive page-level locks are used for read/write access. Locks are released immediately after each index row is fetched or inserted. But note that insertion of a GIN-indexed value usually produces several index key insertions per row, so GIN might do substantial work for a single value's insertion.
  - Currently, B-tree indexes offer the best performance for concurrent applications; since they also have more features than hash indexes, they are the recommended index type for concurrent applications that need to index scalar data. When dealing with non-scalar data, B-trees are not useful, and GiST or GIN indexes should be used instead.

# Performance Tips

## ■ SQL优化

- 执行计划
- 影响执行计划的参数
- #enable\_bitmapscan = on
- #enable\_hashagg = on
- #enable\_hashjoin = on
- #enable\_indexscan = on
- #enable\_material = on
- #enable\_mergejoin = on
- #enable\_nestloop = on
- #enable\_seqscan = on
- #enable\_sort = on
- #enable\_tidscan = on
- #seq\_page\_cost = 1.0
- #random\_page\_cost = 4.0
- #cpu\_tuple\_cost = 0.01
- #cpu\_index\_tuple\_cost = 0.005
- #cpu\_operator\_cost = 0.0025
- #effective\_cache\_size = 128MB
- #default\_statistics\_target = 100
- #constraint\_exclusion = partition
- #cursor\_tuple\_fraction = 0.1
- #fromCollapse\_limit = 8
- #joinCollapse\_limit = 8
- Genetic Query Optimizer

# Performance Tips

- 确保统计信息的及时更新对执行计划的优劣起到很大作用
- 点击进入
- PostgreSQL's statistics target and histogram\_bounds
- <http://blog.163.com/digoal@126/blog/static/16387704020111152495686/>
- PostgreSQL Statistics and Query Explain Introduction
- <http://blog.163.com/digoal@126/blog/static/163877040201041111454178/>
- PostgreSQL 行评估算法
- <http://blog.163.com/digoal@126/blog/static/163877040201041111499884/>
  
- Controlling the Planner with Explicit JOIN Clauses
  
- 举例
- CBO



# Performance Tips

## ■ CBO

```
# - Planner Cost Constants -  
  
#seq_page_cost = 1.0  
random_page_cost = 2.0  
#cpu_tuple_cost = 0.01  
#cpu_index_tuple_cost = 0.005  
#cpu_operator_cost = 0.0025  
effective_cache_size = 16384MB  
  
#default_statistics_target = 100  
#constraint_exclusion = partition  
#cursor_tuple_fraction = 0.1  
#fromCollapse_limit = 8  
#joinCollapse_limit = 8
```

```
# - Planner Method Configuration -  
  
#enable_bitmapscan = on  
#enable_hashagg = on  
#enable_hashjoin = on  
#enable_indexscan = on  
#enable_material = on  
#enable_mergejoin = on  
#enable_nestloop = on  
#enable_seqscan = on  
#enable_sort = on  
#enable_tidscan = on
```

# Performance Tips

- CBO Principle
  - autoanalyze

| Column            | Type     | Modifiers |
|-------------------|----------|-----------|
| schemaname        | name     |           |
| tablename         | name     |           |
| attname           | name     |           |
| inherited         | boolean  |           |
| null_frac         | real     |           |
| avg_width         | integer  |           |
| n_distinct        | real     |           |
| most_common_vals  | anyarray |           |
| most_common_freqs | real[]   |           |
| histogram_bounds  | anyarray |           |
| correlation       | real     |           |

是否被继承;

空值比例;

平均长度;

唯一值个数(-1唯一);

最常见的值;

最常见的值得占比;

记录分bucket边界值;

物理存储与该列的匹配顺性;

# Performance Tips

## ■ CBO Principle

### ■ autoanalyze

| Column          | Type      | Modifiers |
|-----------------|-----------|-----------|
| relname         | name      | not null  |
| relnamespace    | oid       | not null  |
| reltype         | oid       | not null  |
| reloftype       | oid       | not null  |
| relowner        | oid       | not null  |
| relam           | oid       | not null  |
| relfilenode     | oid       | not null  |
| reltablespace   | oid       | not null  |
| relpages        | integer   | not null  |
| reltuples       | real      | not null  |
| reltoastrelid   | oid       | not null  |
| reltoastidxid   | oid       | not null  |
| relhasindex     | boolean   | not null  |
| relisshared     | boolean   | not null  |
| relistemp       | boolean   | not null  |
| relkind         | "char"    | not null  |
| relnatts        | smallint  | not null  |
| relchecks       | smallint  | not null  |
| relhasoids      | boolean   | not null  |
| relhaskey       | boolean   | not null  |
| relhasexclusion | boolean   | not null  |
| relhasrules     | boolean   | not null  |
| relhastriggers  | boolean   | not null  |
| relhassubclass  | boolean   | not null  |
| relfrozenxid    | xid       | not null  |
| relacl          | aclitem[] |           |
| reloptions      | text[]    |           |

Indexes:

- "pg\_class\_oid\_index" UNIQUE, btree (oid)
- "pg\_class\_relname\_nsp\_index" UNIQUE, btree (relname, relnamespace)

# Performance Tips

```
SELECT relpages, reltuples FROM pg_class WHERE relname = 'tenk1';  
  
relpages | reltuples  
-----+-----  
 358 |     10000
```

```
EXPLAIN SELECT * FROM tenk1 WHERE unique1 < 1000;
```

## QUERY PLAN

```
-----  
Bitmap Heap Scan on tenk1  (cost=24.06..394.64 rows=1007 width=244)  
  Recheck Cond: (unique1 < 1000)  
    -> Bitmap Index Scan on tenk1_unique1  (cost=0.00..23.80 rows=1007 width=0)  
      Index Cond: (unique1 < 1000)
```

```
SELECT histogram_bounds FROM pg_stats  
WHERE tablename='tenk1' AND attname='unique1';
```

## histogram\_bounds

```
{0, 993, 1997, 3050, 4040, 5036, 5957, 7057, 8029, 9016, 9995}
```

```
selectivity = (1 + (1000 - bucket[2].min)/(bucket[2].max - bucket[2].min))/num_buckets  
= (1 + (1000 - 993)/(1997 - 993))/10  
= 0.100697
```

```
rows = rel_cardinality * selectivity  
= 10000 * 0.100697  
= 1007  (rounding off)
```

# Performance Tips

- 举例 : Change choice of the planer.
- digoal=> create table tbl.cbo\_test (id int primary key,firstname text,lastname text);
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl.cbo\_test\_pkey" for table "tbl.cbo\_test"
- CREATE TABLE
- digoal=> insert into tbl.cbo\_test select generate\_series(1,1000000),'zhou','digoal';
- INSERT 0 1000000
- digoal=> explain analyze select \* from tbl.cbo\_test where id=100000;
- **QUERY PLAN**
- -----  
Index Scan using tbl.cbo\_test\_pkey on tbl.cbo\_test (cost=0.00..4.32 rows=1 width=16)  
(actual time=0.014..0.015 rows=1 loops=1)
- Index Cond: (id = 100000)
- Total runtime: 0.035 ms
- (3 rows)

# Performance Tips

- digoal=> set enable\_indexscan=off;
- SET
- digoal=> explain analyze select \* from tbl.cbo\_test where id=100000;
- **QUERY PLAN**
- ---
- Bitmap Heap Scan on `tbl.cbo_test` (cost=2.31..4.33 rows=1 width=16) (actual time=0.030..0.031 rows=1 loops=1)
  - Recheck Cond: (`id = 100000`)
  - -> Bitmap Index Scan on `tbl.cbo_test_pkey` (cost=0.00..2.31 rows=1 width=0) (actual time=0.022..0.022 rows=1 loops=1)
    - Index Cond: (`id = 100000`)
  - **Total runtime: 0.065 ms**
  - (5 rows)

# Performance Tips

- digoal=> set enable\_bitmapscan=off;
- SET
- digoal=> explain analyze select \* from tbl.cbo\_test where id=100000;

## QUERY PLAN

---

- Seq Scan on `tbl.cbo_test` (cost=0.00..17906.00 rows=1 width=16) (actual time=17.524..149.940 rows=1 loops=1)
  - Filter: (id = 100000)
  - **Total runtime: 149.962 ms**
  - (3 rows)

# Performance Tips

## ■ 举例 : JOIN Tuning

```
digoal=> create table tbl_join_1 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_1_pkey" for table "tbl_join_1"
CREATE TABLE
digoal=> create table tbl_join_2 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_2_pkey" for table "tbl_join_2"
CREATE TABLE
digoal=> create table tbl_join_3 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_3_pkey" for table "tbl_join_3"
CREATE TABLE
digoal=> create table tbl_join_4 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_4_pkey" for table "tbl_join_4"
CREATE TABLE
digoal=> create table tbl_join_5 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_5_pkey" for table "tbl_join_5"
CREATE TABLE
digoal=> create table tbl_join_6 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_6_pkey" for table "tbl_join_6"
CREATE TABLE
digoal=> create table tbl_join_7 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_7_pkey" for table "tbl_join_7"
CREATE TABLE
digoal=> create table tbl_join_8 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_8_pkey" for table "tbl_join_8"
CREATE TABLE
digoal=> create table tbl_join_9 (id int primary key,info text);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_join_9_pkey" for table "tbl_join_9"
CREATE TABLE
```

# Performance Tips

## ■ 举例 : JOIN Tuning

```
digoal=> insert into tbl_join_1 select generate_series(1,10), '测试测试....';
INSERT 0 10
digoal=> insert into tbl_join_2 select generate_series(1,100), '测试测试....';
INSERT 0 100
digoal=> insert into tbl_join_3 select generate_series(1,1000), '测试测试....';
INSERT 0 1000
digoal=> insert into tbl_join_4 select generate_series(1,10000), '测试测试....';
INSERT 0 10000
digoal=> insert into tbl_join_5 select generate_series(1,100000), '测试测试....';
INSERT 0 100000
digoal=> insert into tbl_join_6 select generate_series(1,1000000), '测试测试....';
INSERT 0 1000000
digoal=> insert into tbl_join_7 select generate_series(1,2000000), '测试测试....';
INSERT 0 2000000
digoal=> insert into tbl_join_8 select generate_series(1,4000000), '测试测试....';
INSERT 0 4000000
digoal=> insert into tbl_join_9 select generate_series(1,8000000), '测试测试....';
INSERT 0 8000000
```

# Performance Tips

## ■ 举例 : JOIN Tuning

```
explain select t1.info,t5.info
from
tbl_join_1 t1,
tbl_join_2 t2,
tbl_join_3 t3,
tbl_join_4 t4,
tbl_join_5 t5,
tbl_join_6 t6,
tbl_join_7 t7,
tbl_join_8 t8,
tbl_join_9 t9
where
t1.id=t2.id
and t2.id=t3.id
and t3.id=t4.id
and t4.id=t5.id
and t5.id=t6.id
and t6.id=t7.id
and t7.id=t8.id
and t8.id=t9.id
and t9.id=10000;
```



```
explain select t1.info,t5.info
from
tbl_join_1 t1
join tbl_join_2 t2 on <t1.id=t2.id>
join tbl_join_3 t3 on <t2.id=t3.id>
join tbl_join_4 t4 on <t3.id=t4.id>
join tbl_join_5 t5 on <t4.id=t5.id>
join tbl_join_6 t6 on <t5.id=t6.id>
join tbl_join_7 t7 on <t6.id=t7.id>
join tbl_join_8 t8 on <t7.id=t8.id>
join tbl_join_9 t9 on <t8.id=t9.id>
where t9.id=10000;
```

Which SQL is better !

# Performance Tips

## ■ 举例 : JOIN Tuning

### QUERY PLAN

```
--  
Nested Loop  <cost=0.00..37.30 rows=1 width=49>  
  -> Nested Loop  <cost=0.00..33.02 rows=1 width=57>  
    -> Nested Loop  <cost=0.00..28.30 rows=1 width=61>  
      -> Nested Loop  <cost=0.00..23.92 rows=1 width=65>  
        -> Nested Loop  <cost=0.00..19.59 rows=1 width=61>  
          -> Nested Loop  <cost=0.00..15.09 rows=1 width=57>  
            -> Nested Loop  <cost=0.00..10.81 rows=1 width=36>  
              -> Nested Loop  <cost=0.00..8.55 rows=1 width=40>  
                -> Index Scan using tbl_join_1_pkey on tbl_join_1 t1  <cost=0.00..4.27 rows=1 width=36>  
>  
                  Index Cond: (id = 10000)  
                -> Index Scan using tbl_join_3_pkey on tbl_join_3 t3  <cost=0.00..4.27 rows=1 width=4>  
                  Index Cond: (t3.id = 10000)  
                -> Seq Scan on tbl_join_2 t2  <cost=0.00..2.25 rows=1 width=4>  
                  Filter: (t2.id = 10000)  
                -> Index Scan using tbl_join_5_pkey on tbl_join_5 t5  <cost=0.00..4.27 rows=1 width=21>  
                  Index Cond: (t5.id = 10000)  
                -> Index Scan using tbl_join_8_pkey on tbl_join_8 t8  <cost=0.00..4.49 rows=1 width=4>  
                  Index Cond: (t8.id = 10000)  
                -> Index Scan using tbl_join_6_pkey on tbl_join_6 t6  <cost=0.00..4.32 rows=1 width=4>  
                  Index Cond: (t6.id = 10000)  
                -> Index Scan using tbl_join_7_pkey on tbl_join_7 t7  <cost=0.00..4.38 rows=1 width=4>  
                  Index Cond: (t7.id = 10000)  
                -> Index Scan using tbl_join_9_pkey on tbl_join_9 t9  <cost=0.00..4.71 rows=1 width=4>  
                  Index Cond: (t9.id = 10000)  
                -> Index Scan using tbl_join_4_pkey on tbl_join_4 t4  <cost=0.00..4.27 rows=1 width=4>  
                  Index Cond: (t4.id = 10000)  
(26 rows)
```

Time: 15.671 ms

# Performance Tips

## ■ 举例 : JOIN Tuning

```
QUERY PLAN

Nested Loop  <cost=0.00..37.30 rows=1 width=49>
  -> Nested Loop  <cost=0.00..32.58 rows=1 width=53>
    -> Nested Loop  <cost=0.00..28.09 rows=1 width=53>
      -> Nested Loop  <cost=0.00..12.82 rows=1 width=44>
        -> Nested Loop  <cost=0.00..8.55 rows=1 width=40>
          -> Index Scan using tbl_join_1_pkey on tbl_join_1 t1  <cost=0.00..4.27 rows=1 width=36>
              Index Cond: <id = 10000>
            -> Index Scan using tbl_join_3_pkey on tbl_join_3 t3  <cost=0.00..4.27 rows=1 width=4>
                Index Cond: <t3.id = 10000>
            -> Index Scan using tbl_join_4_pkey on tbl_join_4 t4  <cost=0.00..4.27 rows=1 width=4>
                Index Cond: <t4.id = 10000>
  -> Nested Loop  <cost=0.00..15.25 rows=1 width=29>
    -> Nested Loop  <cost=0.00..10.87 rows=1 width=29>
      -> Nested Loop  <cost=0.00..6.53 rows=1 width=25>
        -> Seq Scan on tbl_join_2 t2  <cost=0.00..2.25 rows=1 width=4>
            Filter: <id = 10000>
        -> Index Scan using tbl_join_5_pkey on tbl_join_5 t5  <cost=0.00..4.27 rows=1 width=21>
            Index Cond: <t5.id = 10000>
        -> Index Scan using tbl_join_6_pkey on tbl_join_6 t6  <cost=0.00..4.32 rows=1 width=4>
            Index Cond: <t6.id = 10000>
    -> Index Scan using tbl_join_7_pkey on tbl_join_7 t7  <cost=0.00..4.38 rows=1 width=4>
        Index Cond: <t7.id = 10000>
  -> Index Scan using tbl_join_8_pkey on tbl_join_8 t8  <cost=0.00..4.49 rows=1 width=4>
      Index Cond: <t8.id = 10000>
-> Index Scan using tbl_join_9_pkey on tbl_join_9 t9  <cost=0.00..4.71 rows=1 width=4>
  Index Cond: <t9.id = 10000>

<26 rows>

Time: 6.044 ms
```

# Performance Tips

## ■ 举例 : JOIN Tuning

```
QUERY PLAN

Nested Loop  <cost=0.00..37.30 rows=1 width=49>
  -> Nested Loop  <cost=0.00..32.58 rows=1 width=53>
    -> Nested Loop  <cost=0.00..28.09 rows=1 width=53>
      -> Nested Loop  <cost=0.00..12.82 rows=1 width=44>
        -> Nested Loop  <cost=0.00..8.55 rows=1 width=40>
          -> Index Scan using tbl_join_1_pkey on tbl_join_1 t1  <cost=0.00..4.27 rows=1 width=36>
              Index Cond: <id = 10000>
            -> Index Scan using tbl_join_3_pkey on tbl_join_3 t3  <cost=0.00..4.27 rows=1 width=4>
                Index Cond: <t3.id = 10000>
            -> Index Scan using tbl_join_4_pkey on tbl_join_4 t4  <cost=0.00..4.27 rows=1 width=4>
                Index Cond: <t4.id = 10000>
  -> Nested Loop  <cost=0.00..15.25 rows=1 width=29>
    -> Nested Loop  <cost=0.00..10.87 rows=1 width=29>
      -> Nested Loop  <cost=0.00..6.53 rows=1 width=25>
        -> Seq Scan on tbl_join_2 t2  <cost=0.00..2.25 rows=1 width=4>
            Filter: <id = 10000>
        -> Index Scan using tbl_join_5_pkey on tbl_join_5 t5  <cost=0.00..4.27 rows=1 width=21>
            Index Cond: <t5.id = 10000>
        -> Index Scan using tbl_join_6_pkey on tbl_join_6 t6  <cost=0.00..4.32 rows=1 width=4>
            Index Cond: <t6.id = 10000>
    -> Index Scan using tbl_join_7_pkey on tbl_join_7 t7  <cost=0.00..4.38 rows=1 width=4>
        Index Cond: <t7.id = 10000>
  -> Index Scan using tbl_join_8_pkey on tbl_join_8 t8  <cost=0.00..4.49 rows=1 width=4>
        Index Cond: <t8.id = 10000>
-> Index Scan using tbl_join_9_pkey on tbl_join_9 t9  <cost=0.00..4.71 rows=1 width=4>
  Index Cond: <t9.id = 10000>

<26 rows>

Time: 6.044 ms
```

# Performance Tips



- 举例 : JOIN Tuning
- Explicit JOIN
- What happen when SET join\_collapse\_limit = 1 and use the join SQL;
- 如果不限制, 查询的关联的表越多, 关联的顺序组合就越多, 会带来很大的生成执行计划的开销(穷举).
- join\_collapse\_limit
  - 尽量把explicit JOIN(除了FULL JOIN)涉及的表都放到一个列表, 以这个列表进行JOIN顺序的排列组合得到最佳执行计划.(而join\_collapse\_limit就是限制这个列表有多大, 或者说有几个表会放到这里面来进行排列组合)
- fromCollapseLimit
  - 与join\_collapse\_limit功能类似, 只是他针对的是子查询, 例如
  - SELECT \* FROM x, y, (SELECT \* FROM a, b, c WHERE something) AS ss WHERE somethingelse;
  - 会把x,y,a,b,c放在一个列表中进行JOIN顺序的排列组合.前提是这个列表小于from\_collapse\_limit
- geqo\_threshold -- 不值得推荐, 因为会产生不可预估的执行计划, 随机产生. 虽然对复杂查询可以降低执行计划的时间.

# Performance Tips

## ■ 举例 : JOIN Tuning

### QUERY PLAN

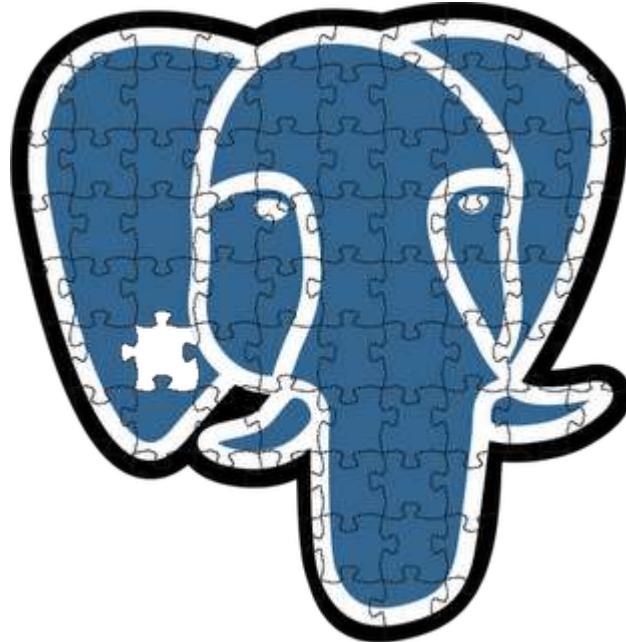
```
--  
Nested Loop <cost=0.00..37.30 rows=1 width=49>  
  -> Nested Loop <cost=0.00..32.58 rows=1 width=53>  
    -> Nested Loop <cost=0.00..28.09 rows=1 width=53>  
      -> Nested Loop <cost=0.00..23.70 rows=1 width=53>  
        -> Nested Loop <cost=0.00..19.37 rows=1 width=53>  
          -> Nested Loop <cost=0.00..15.08 rows=1 width=36>  
            -> Nested Loop <cost=0.00..10.81 rows=1 width=36>  
              -> Nested Loop <cost=0.00..6.53 rows=1 width=36>  
                -> Index Scan using tbl_join_1_pkey on tbl_join_1 t1 <cost=0.00..4.27 rows=1 width=36>  
>  
                  Index Cond: <id = 10000>  
                  -> Seq Scan on tbl_join_2 t2 <cost=0.00..2.25 rows=1 width=4>  
                    Filter: <t2.id = 10000>  
                  -> Index Scan using tbl_join_3_pkey on tbl_join_3 t3 <cost=0.00..4.27 rows=1 width=4>  
                    Index Cond: <t3.id = 10000>  
                  -> Index Scan using tbl_join_4_pkey on tbl_join_4 t4 <cost=0.00..4.27 rows=1 width=4>  
                    Index Cond: <t4.id = 10000>  
                  -> Index Scan using tbl_join_5_pkey on tbl_join_5 t5 <cost=0.00..4.27 rows=1 width=21>  
                    Index Cond: <t5.id = 10000>  
                  -> Index Scan using tbl_join_6_pkey on tbl_join_6 t6 <cost=0.00..4.32 rows=1 width=4>  
                    Index Cond: <t6.id = 10000>  
                  -> Index Scan using tbl_join_7_pkey on tbl_join_7 t7 <cost=0.00..4.38 rows=1 width=4>  
                    Index Cond: <t7.id = 10000>  
                  -> Index Scan using tbl_join_8_pkey on tbl_join_8 t8 <cost=0.00..4.49 rows=1 width=4>  
                    Index Cond: <t8.id = 10000>  
                  -> Index Scan using tbl_join_9_pkey on tbl_join_9 t9 <cost=0.00..4.71 rows=1 width=4>  
                    Index Cond: <t9.id = 10000>  
(26 rows)
```

# Performance Tips

- 数据迁移性能相关
  - 关闭autocommit
  - 使用COPY
  - 移除索引
  - 移除Foreign Key 约束
  - 加大maintenance\_work\_mem可以提高建索引速度
  - 加大checkpoint\_segments , checkpoint\_timeout
  - Disable WAL Archival and Streaming Replication
    - To do that, set archive\_mode to off, wal\_level to minimal, and max\_wal\_senders to zero before loading the dump. 数据导入完成后开启, standby需要重新从基础备份做.
  - 关闭autovacuum, 数据导入后运行analyze.
  - COPY commands will run fastest if you use a single transaction and have WAL archiving turned off.
  - 使用pg\_restore的并行参数

# Day 2

## PostgreSQL 9.1.3 2Day DBA QuickGuide



# Day 2

- PostgreSQL Client Applications
- PostgreSQL Server Applications
- Database Physical Storage
- Server Administration
  - [Database Layout](#)
  - [Reliability](#)
  - [Server Configuration](#)
  - [Routine Database Maintenance Tasks](#)
  - [Backup and Restore](#)
  - [HA and Replication](#)
  - [Stream Replication](#)
  - [Cascade Stream Replication](#)
  - [PostgreSQL-XC , PL/Proxy , pgpool](#)
- [Monitoring Database Activity](#)
- [Procedure Language and Debug](#)
- [PostgreSQL Distinguishing Feature](#)
  - [Additional Supplied Modules](#)
- [Database Performance Tuning Short Case](#)

# PostgreSQL Client Applications

- **clusterdb** -- clusterdb is a utility for reclustering tables in a PostgreSQL database. It finds tables that have previously been clustered, and clusters them again on the same index that was last used. Tables that have never been clustered are not affected.
- **clusterdb [connection-option...] [--verbose | -v] [--table | -t table ] [dbname]**
- **clusterdb [connection-option...] [--verbose | -v] [--all | -a]**
  
- **createdb** -- create a new PostgreSQL database
- **createdb [connection-option...] [option...] [dbname] [description]**
  
- **createlang** -- install a PostgreSQL procedural language
- **createlang [connection-option...] langname [dbname]**
- **createlang [connection-option...] --list | -l dbname**
  
- **createuser** -- define a new PostgreSQL user account
- **createuser [connection-option...] [option...] [username]**

# PostgreSQL Client Applications

- `dropdb` -- remove a PostgreSQL database
  - `dropdb [connection-option...] [option...] dbname`
- `droplang` -- remove a PostgreSQL procedural language
  - `droplang [connection-option...] langname [dbname]`
  - `droplang [connection-option...] --list | -l dbname`
- `dropuser` -- remove a PostgreSQL user account
  - `dropuser [connection-option...] [option...] [username]`
- `ecpg` -- embedded SQL C preprocessor
  - `ecpg [option...] file...`
- `pg_basebackup` -- take a base backup of a PostgreSQL cluster
  - `pg_basebackup [option...]`

# PostgreSQL Client Applications

- pg\_config -- retrieve information about the installed version of PostgreSQL
- pg\_config [option...]
  
- pg\_dump -- extract a PostgreSQL database into a script file or other archive file
- pg\_dump [connection-option...] [option...] [dbname]
  
- pg\_dumpall -- extract a PostgreSQL database cluster into a script file
- pg\_dumpall [connection-option...] [option...]
  
- pg\_restore -- restore a PostgreSQL database from an archive file created by pg\_dump
- pg\_restore [connection-option...] [option...] [filename]
  
- psql -- PostgreSQL interactive terminal
- psql [option...] [dbname [username]]

# PostgreSQL Client Applications

- `reindexdb` -- reindex a PostgreSQL database
- `reindexdb [connection-option...] [--table | -t table ] [--index | -i index ] [dbname]`
- `reindexdb [connection-option...] [--all | -a]`
- `reindexdb [connection-option...] [--system | -s] [dbname]`
  
- `vacuumdb` -- garbage-collect and analyze a PostgreSQL database
- `vacuumdb [connection-option...] [--full | -f] [--freeze | -F] [--verbose | -v] [--analyze | -z] [--analyze-only | -Z] [--table | -t table [( column [,....] )]] [dbname]`
- `vacuumdb [connection-option...] [--full | -f] [--freeze | -F] [--verbose | -v] [--analyze | -z] [--analyze-only | -Z] [--all | -a]`

# PostgreSQL Server Applications

- initdb -- create a new PostgreSQL database cluster
- initdb [option...] --pgdata | -D directory
  
- pg\_controldata -- display control information of a PostgreSQL database cluster
- pg\_controldata [option] [datadir]
  
- pg\_ctl -- initialize, start, stop, or control a PostgreSQL server
- pg\_ctl init[db] [-s] [-D datadir] [-o initdb-options]
- pg\_ctl start [-w] [-t seconds] [-s] [-D datadir] [-l filename] [-o options] [-p path] [-c]
- pg\_ctl stop [-W] [-t seconds] [-s] [-D datadir] [-m s[mart] | f[ast] | i[mmediate] ]
- pg\_ctl restart [-w] [-t seconds] [-s] [-D datadir] [-c] [-m s[mart] | f[ast] | i[mmediate] ] [-o options]
- pg\_ctl reload [-s] [-D datadir]
- pg\_ctl status [-D datadir]
- pg\_ctl promote [-s] [-D datadir]

# PostgreSQL Server Applications

- pg\_ctl kill signal\_name process\_id
- pg\_ctl register [-N servicename] [-U username] [-P password] [-D datadir] [-S a[uto] | d[emand] ] [-w] [-t seconds] [-s] [-o options]
- pg\_ctl unregister [-N servicename]
  
- pg\_resetxlog -- reset the write-ahead log and other control information of a PostgreSQL database cluster
- pg\_resetxlog [-f] [-n] [-oid ] [-x xid ] [-e xid\_epoch ] [-m mxid ] [-O mxoff ] [-l timelineid,fileid,seg ] datadir
  
- postgres -- PostgreSQL database server. postgres is the PostgreSQL database server. In order for a client application to access a database it connects (over a network or locally) to a running postgres instance. The postgres instance then starts a separate server process to handle the connection.
- postgres [option...]

# PostgreSQL Server Applications

- postmaster -- PostgreSQL database server. postmaster is a deprecated alias of postgres.
- postmaster [option...]
  
- 例子
- 控制文件
  - [src/bin/pg\\_controldata/pg\\_controldata.c](#)

# PostgreSQL Server Applications

## ■ 控制文件信息举例

```
pg_control version number:          903 -- ControlFile.pg_control_version
Catalog version number:            201105231 -- ControlFile.catalog_version_no
Database system identifier:        5736077078824723446 -- sysident_str
Database cluster state:           in production -- dbState(ControlFile.state)
                                  -- DB_STARTUP,DB_SHUTDOWNED,DB_SHUTDOWNED_IN_RECOVERY,DB_SHUTDOWNING
                                  -- DB_IN_CRASH_RECOVERY,DB_IN_ARCHIVE_RECOVERY,DB_IN_PRODUCTION

pg_control last modified:         Wed 02 May 2012 08:44:57 AM CST -- pgctime_str
Latest checkpoint location:       2/75B48D08 -- ControlFile.checkPoint.xlogid, ControlFile.checkPoint.xrecoff
Prior checkpoint location:        2/75B48C08 -- ControlFile.prevCheckPoint.xlogid, ControlFile.prevCheckPoint.xrecoff
Latest checkpoint's REDO location: 2/75B48D08 -- ControlFile.checkPointCopy.redo.xlogid, ControlFile.checkPointCopy.redo.xrecoff
Latest checkpoint's TimeLineID:    1 -- ControlFile.checkPointCopy.ThisTimeLineID
Latest checkpoint's NextXID:       0/1874 -- ControlFile.checkPointCopy.nextXidEpoch, ControlFile.checkPointCopy.nextXid
Latest checkpoint's NextOID:       6032792 -- ControlFile.checkPointCopy.nextOid
Latest checkpoint's NextMultiXactId: 1 -- ControlFile.checkPointCopy.nextMulti
Latest checkpoint's NextMultiOffset: 0 -- ControlFile.checkPointCopy.nextMultiOffset
Latest checkpoint's oldestXID:     1670 -- ControlFile.checkPointCopy.oldestXid
Latest checkpoint's oldestXID's DB: 1 -- ControlFile.checkPointCopy.oldestXidDB
Latest checkpoint's oldestActiveXID: 0 -- ControlFile.checkPointCopy.oldestActiveXid
Time of latest checkpoint:        Wed 02 May 2012 08:44:57 AM CST -- ckpttime_str
Minimum recovery ending location: 0/0 -- ControlFile.minRecoveryPoint.xlogid, ControlFile.minRecoveryPoint.xrecoff
Backup start location:           0/0 -- ControlFile.backupStartPoint.xlogid, ControlFile.backupStartPoint.xrecoff
Current wal_level setting:       minimal -- wal_level_str(ControlFile.wal_level)
                                  -- WAL_LEVEL_MINIMAL,WAL_LEVEL_ARCHIVE,WAL_LEVEL_HOT_STANDBY

Current max_connections setting:   100 -- ControlFile.MaxConnections
Current max_prepared_xacts setting: 0 -- ControlFile.max_prepared_xacts
Current max_locks_per_xact setting: 64 -- ControlFile.max_locks_per_xact
Maximum data alignment:           8 -- ControlFile.maxAlign
Database block size:              8192 -- ControlFile.blcksz
Blocks per segment of large relation: 131072 -- ControlFile.relsegs_size
WAL block size:                  8192 -- ControlFile.xlog_blkksz
Bytes per WAL segment:           67108864 -- ControlFile.xlog_seg_size
Maximum length of identifiers:    64 -- ControlFile.nameDataLen
Maximum columns in an index:      32 -- ControlFile.indexMaxKeys
Maximum size of a TOAST chunk:    1996 -- ControlFile.toast_max_chunk_size
Date/time type storage:          64-bit integers -- (ControlFile.enableIntTimes ? _("64-bit integers") : _("floating-point numbers"))
Float4 argument passing:          by value -- (ControlFile.float4ByWal ? _("by value") : _("by reference"))
Float8 argument passing:          by value -- (ControlFile.float8ByWal ? _("by value") : _("by reference"))
```

# PostgreSQL Server Applications

## ■ 单用户模式启动postgres

- 当系统遭遇如下错误时必须进入单用户模式修复数据库
- **ERROR:** database is not accepting commands to avoid wraparound data loss in database "mydb"
- **HINT:** Stop the postmaster and use a standalone backend to VACUUM in "mydb".
- 为什么会遭遇这个错误?
- 数据库中任何带refrozenxid标记的记录,年龄不能超过 $2^{31}$ (二十亿);当数据库中存在年龄大于 $\{(2^{31})-1\text{千万}\}$ 的记录时,数据库将报类似如下提示:
- **WARNING:** database "mydb" must be vacuumed within 177009986 transactions  
**HINT:** To avoid a database shutdown, execute a database-wide VACUUM in "mydb".
- 如果忽略上面的警告,当数据库中存在年龄大于 $\{(2^{31})-1\text{百万}\}$ 的记录时,数据库将报类似如下错误:
- **ERROR:** database is not accepting commands to avoid wraparound data loss in database "mydb"  
**HINT:** Stop the postmaster and use a standalone backend to VACUUM in "mydb".

# PostgreSQL Server Applications

- PostgreSQL single-user mode usage, like Repair Database
- <http://blog.163.com/digoal@126/blog/static/163877040201011152042497/> 
- bootstrapping模式启动postgres
- initdb 调用的就是bootstrapping模式, bootstrapping模式下的语法与普通模式下的语法也打不一样, 使用的是BKI接口. 例如initdb调用的\$PGHOME/share/postgres.bki.
  - postgres.bki文件结构
  - create bootstrap one of the critical tables
  - insert data describing at least the critical tables
  - close
  - Repeat for the other critical tables.
  - create (without bootstrap) a noncritical table
  - open
  - insert desired data
  - close
  - Repeat for the other noncritical tables.
  - Define indexes and toast tables.
  - build indices

# PostgreSQL Server Applications

- BKI commands
- 略
- <http://www.postgresql.org/docs/9.1/static/bki-commands.html>



- 我的数据"消失"了?
  - Use pg\_resetxlog simulate tuple disappear within PostgreSQL
  - <http://blog.163.com/digoal@126/blog/static/163877040201183043153622/>



# Database Physical Storage

## ■ \$PGDATA

```
postgres@db-172-16-3-150-> cd $PGDATA
postgres@db-172-16-3-150-> ll
total 88K
drwx----- 6 postgres postgres 4.0K Apr 30 11:00 base
drwx----- 2 postgres postgres 4.0K May  2 08:18 global
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 pg_clog
-rw----- 1 postgres postgres 4.4K Apr 27 21:48 pg_hba.conf
-rw----- 1 postgres postgres 1.6K Apr 27 21:48 pg_ident.conf
drwx----- 4 postgres postgres 4.0K Apr 27 21:48 pg_multixact
drwx----- 2 postgres postgres 4.0K Apr 30 09:39 pg_notify
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 pg_serial
drwx----- 2 postgres postgres 4.0K May  2 11:58 pg_stat_tmp
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 pg_subtrans
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 pg_tblspc
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 pg_twophase
-rw----- 1 postgres postgres     4 Apr 27 21:48 PG_VERSION
drwx----- 3 postgres postgres 4.0K Apr 30 14:40 pg_xlog
-rw----- 1 postgres postgres 19K Apr 28 08:37 postgresql.conf
-rw----- 1 postgres postgres   24 Apr 28 08:42 postmaster.opts
-rw----- 1 postgres postgres   92 Apr 30 09:39 postmaster.pid
```

# Database Physical Storage

## ■ base目录

- 对应pg\_default表空间，如果新建的数据库没有指定默认表空间，那么新建的数据库的默认表空间由参数#default\_tablespace决定，没配置的话就是pg\_default表空间，因此在这个数据库创建的对象未指定表空间的话都会创建在base目录下的数据库目录中。

```
postgres@db-172-16-3-150:~$ psql
psql (9.1.3)
Type "help" for help.

postgres=# \db
      List of tablespaces
   Name    | Owner    | Location
-----+-----+-----
 pg_default | postgres |
 pg_global  | postgres |
(2 rows)
```

# Database Physical Storage

## ■ base目录

- 查看当前的数据库对应的默认表空间
- 查看当前系统中的表空间

```
postgres=# \l+
   List of databases
   Name    |  Owner   | Encoding | Collate | Ctype | Access privileges |  Size   | Tablespace |          Description
----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
digoal  | postgres | UTF8    | C      | C      |                   | 6129 kB | test      |
postgres | postgres | UTF8    | C      | C      |                   | 6337 kB | pg_default |
template0 | postgres | UTF8    | C      | C      | =c/postgres     | 6025 kB | pg_default | unmodifiable empty database
          |          |          |          |          | postgres=CTc/postgres |
template1 | postgres | UTF8    | C      | C      | =c/postgres     | 6129 kB | pg_default | default template for new databases
          |          |          |          |          | postgres=CTc/postgres |
<4 rows>

postgres=# \db+
   List of tablespaces
   Name    |  Owner   |          Location          | Access privileges | Description
----+-----+-----+-----+-----+-----+
pg_default | postgres |                         |                   |
pg_global  | postgres |                         |                   |
test       | postgres | /pgdata/digoal/1921/data02/pg_tbs/test |                   |
<3 rows>
```

# Database Physical Storage

## ■ base目录

### ■ 查看数据库的oid

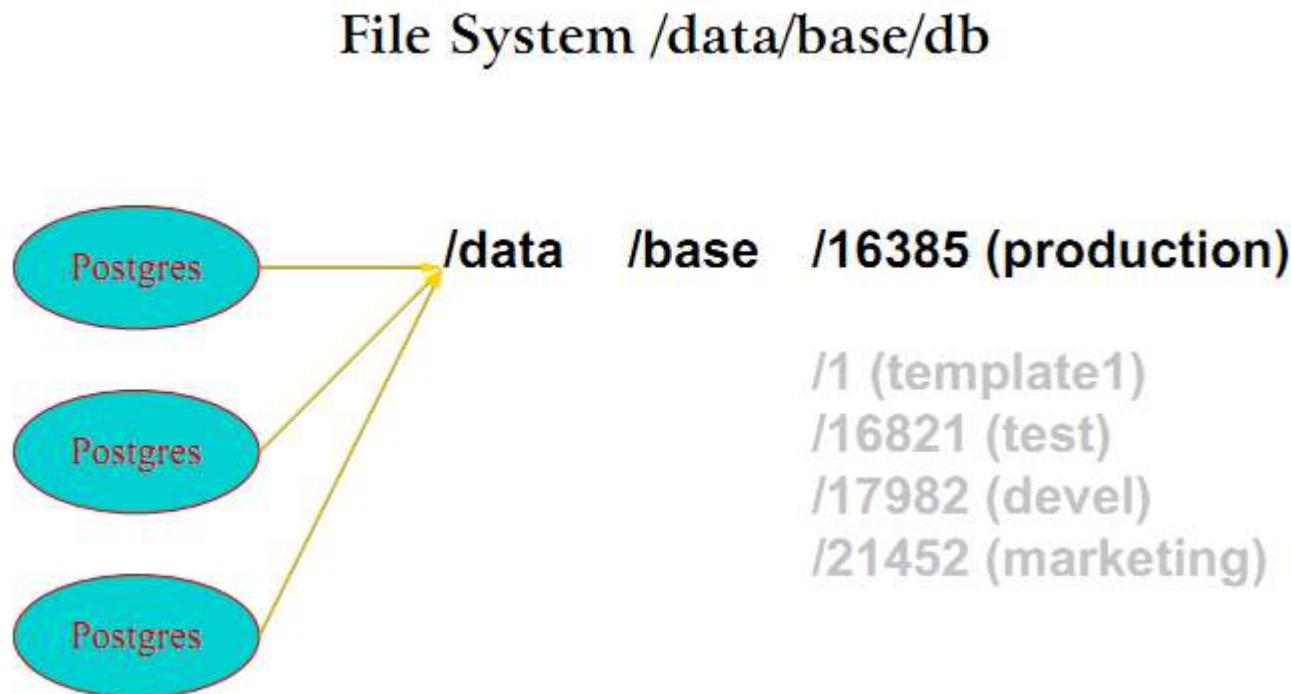
```
postgres=# select oid,datname from pg_database;
   oid   | datname
-----+-----
        1 | template1
  12691 | template0
6024600 | postgres
6024762 | digoal
...
```

- 数据库的oid为目录名，用默认表空间pg\_default的数据库的目录建在\$PGDATA/base下面
- 默认表空间不是pg\_default的，数据库目录建在数据库创建时的默认表空间内.

```
postgres@db-172-16-3-150-> cd $PGDATA/base
postgres@db-172-16-3-150-> ll
total 32K
drwx----- 2 postgres postgres 12K Apr 30 09:41 1
drwx----- 2 postgres postgres 4.0K Apr 27 21:48 12691
drwx----- 2 postgres postgres 12K May  2 08:36 6024600
drwx----- 2 postgres postgres 4.0K Apr 28 10:19 pgsql_tmp
postgres@db-172-16-3-150-> cd /pgdata/digoal/1921/data02/pg_tbs/test
postgres@db-172-16-3-150-> ll
total 4.0K
drwx----- 3 postgres postgres 4.0K May  2 12:02 PG_9.1_201105231
postgres@db-172-16-3-150-> cd PG_9.1_201105231/
postgres@db-172-16-3-150-> ll
total 4.0K
drwx----- 2 postgres postgres 4.0K May  2 12:02 6024762
```

# Database Physical Storage

- base目录
  - 以下图中表示production, template1, test, devel, marketing 库存放在base目录中的目录名.



# Database Physical Storage

- global 目录
  - 对应的是 pg\_global 表空间
  - 这里存放的是 PostgreSQL 集群的数据对象信息, 如 pg\_database, pg\_roles 等

```
postgres=# \db
   List of tablespaces
   Name    | Owner    |          Location
-----+-----+-----+
 pg_default | postgres |
 pg_global  | postgres |

```

- pg\_clog 目录
  - 存放数据库事务提交状态数据
- pg\_notify 目录
  - 存放 NOTIFY/LISTEN 状态数据
- pg\_multixact 目录
  - 存放 select for share 的事务状态数据, 用于共享行锁.

# Database Physical Storage

- pg\_serial 目录
  - PostgreSQL 9.1 带来的 serializable 隔离级别, 里面存储已提交的 serializable 事务的状态信息.
- pg\_stat\_tmp 目录
  - 收集统计信息如果产生临时文件将存放于此
- pg\_subtrans 目录
  - 存放子事务状态数据信息
- pg\_tblspc 目录
  - 存放新建的表空间的软链接信息
- pg\_twophase 目录
  - 存放 twophase 事务的状态信息
- pg\_xlog 目录或软链接(如果 initdb 时指定了 pg\_xlog 的位置)
  - 存放 WAL 日志文件
- PG\_VERSION 文件
  - PostgreSQL 的主版本号. 如 9.1

# Database Physical Storage

## ■ pg\_hba.conf文件

- 客户端认证配置文件

```
/etc/services
auth          113/tcp      authentication tap ident
auth          113/udp      authentication tap ident
```

## ■ pg\_ident.conf文件

- 和pg\_hba.conf结合使用, 存储操作系统用户和连接时使用的数据库用户的map用户信息, mapname将用于pg\_hba.conf 的ident认证方法.

- # MAPNAME SYSTEM-USERNAME PG-USERNAME

## ■ postgresql.conf文件

- 数据库配置文件

```
postgres@db-172-16-3-150-> cat postmaster.opts
/opt/pgsql/bin/postgres
postgres@db-172-16-3-150-> cat postmaster.pid
28732
/pgdata/digoal/1921/data02/pg_root
1335573739
1921
/tmp
localhost
1921001 7340037
```

## ■ postmaster.opts文件

- 最近一次数据库启动的时候创建的文件, 存储数据库启动时postgres的命令行选项参数等

## ■ postmaster.pid文件

- 存储数据库当前运行的postmaster.pid, 数据库集群目录位置, postmaster进程启动时间, 监听的端口号, Unix-socket 目录, 监听地址, 共享内存段信息

# Database Physical Storage

- Authenticate
- pg\_hba.conf
- pg\_shadow

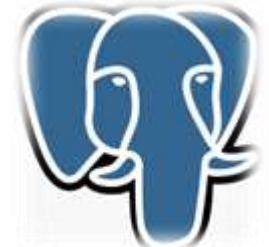
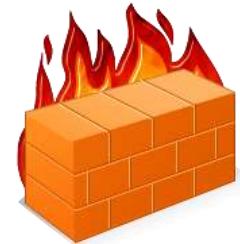


Roles

Connection Limit

Auth Method  
(Trust,  
Password,  
Ident,  
LDAP...)

PostgreSQL



Listene  
Which  
Address

PG\_HBA

TYPE DATABASE USER CIDR-ADDRESS METHOD

# Database Physical Storage

## ■ 数据对象文件

- 主数据文件，通过pg\_class.relfilenode或pg\_relation\_filenode()函数查看. 超过1GB或编译PostgreSQL时设置的 --with-segsize大小后, 会以相同的文件名加.1, ... 后缀新增文件.
- 每个表或索引都会有free space map, 记录page的空闲信息. 可通过pageinspect插件查看.
- 每个表或索引都会有visibility map, 记录没有dead tuple的page信息. 可通过pageinspect插件查看.
- unlogged table and index also have \_init suffix as initialiaztion fork. 可通过pageinspect插件查看.
- TOAST表, pg\_class.reltoastrelid.
  - TOAST,The Oversized-Attribute Storage Technique
  - <http://blog.163.com/digoal@126/blog/static/163877040201122910531988/>
  - how difference when update a table's column which it in TOAST or BASETABLE
  - <http://blog.163.com/digoal@126/blog/static/1638770402012116115354333/>
  - TOAST table with pgfincore
  - <http://blog.163.com/digoal@126/blog/static/16387704020120524144140/>



# Database Physical Storage



数据对象文件

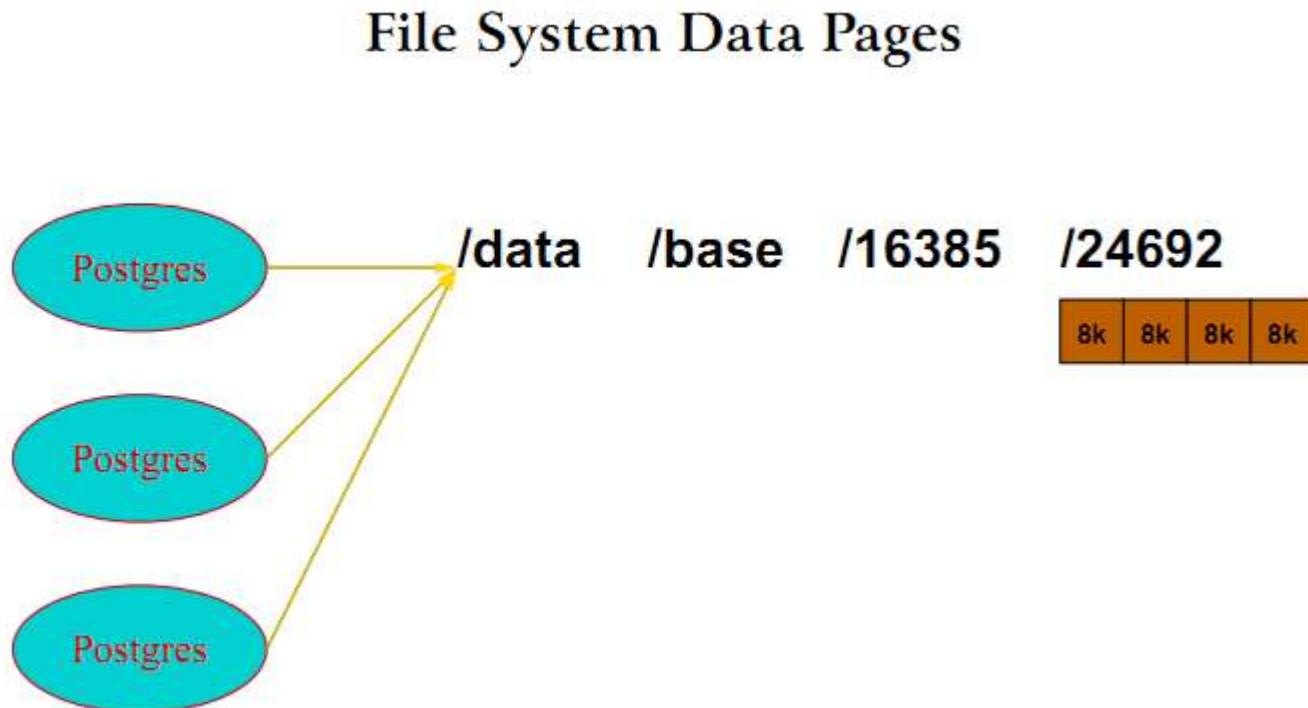
- 以下表示customer, order, product, employee, part存放在production库中的文件名.

File System /data/base/db/table



# Database Physical Storage

- 数据对象文件
  - 以下表示customer表的第一个数据文件的块信息.

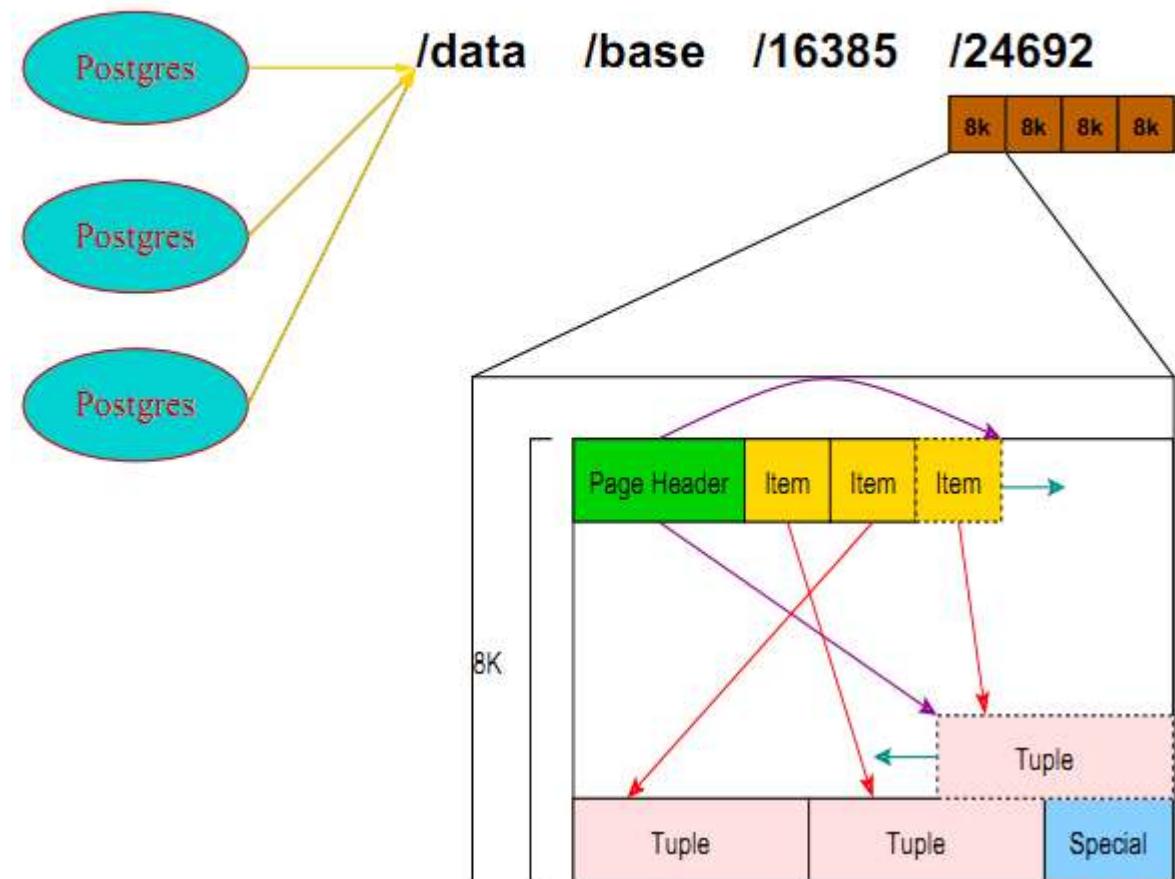


# Database Physical Storage

## ■ 数据对象文件

- 以下表示customer表的第一个数据文件的块内部的信息.

Data Pages

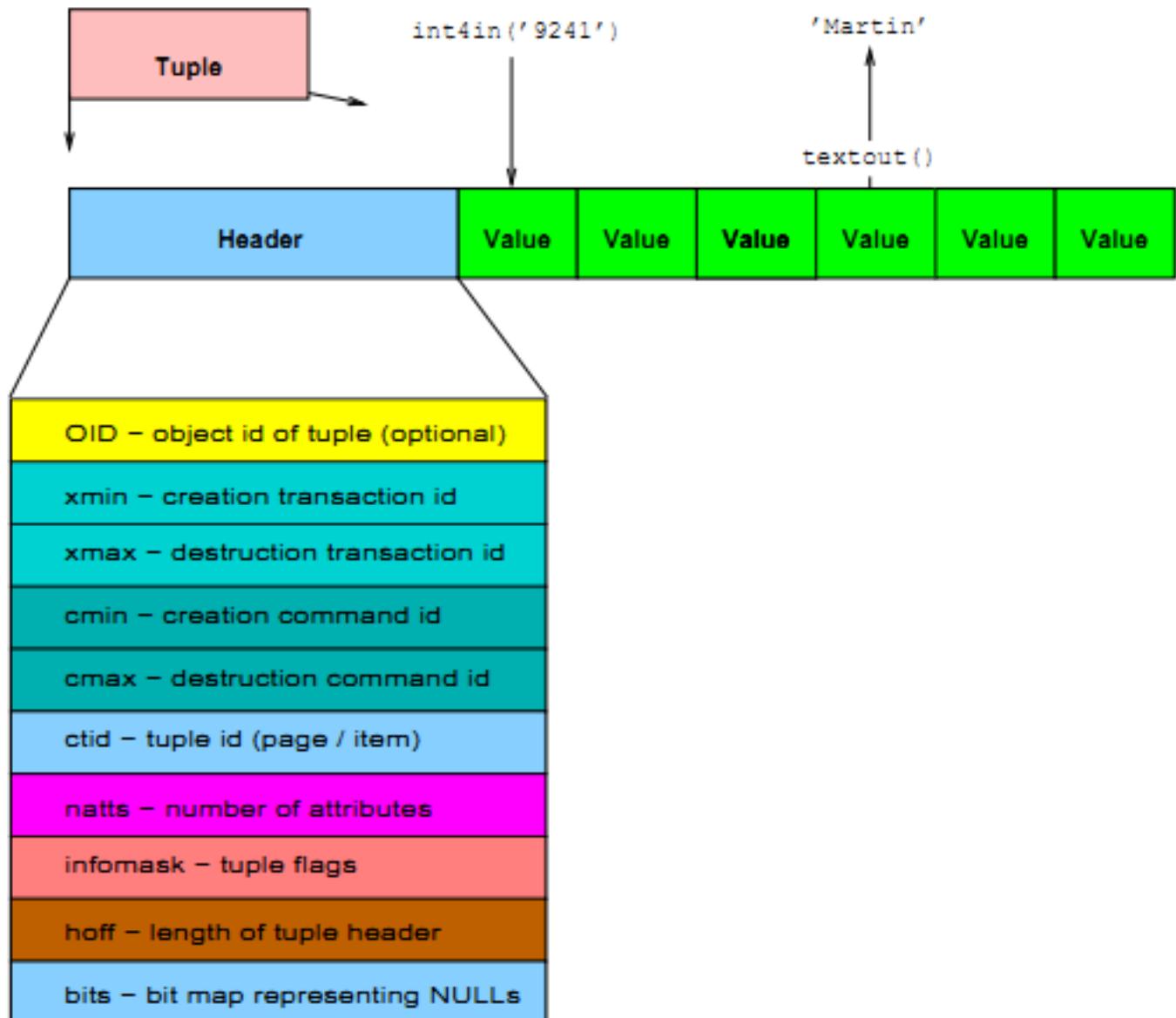


pageinspect

pg\_filedump-9.0.0

# Database Physical Storage

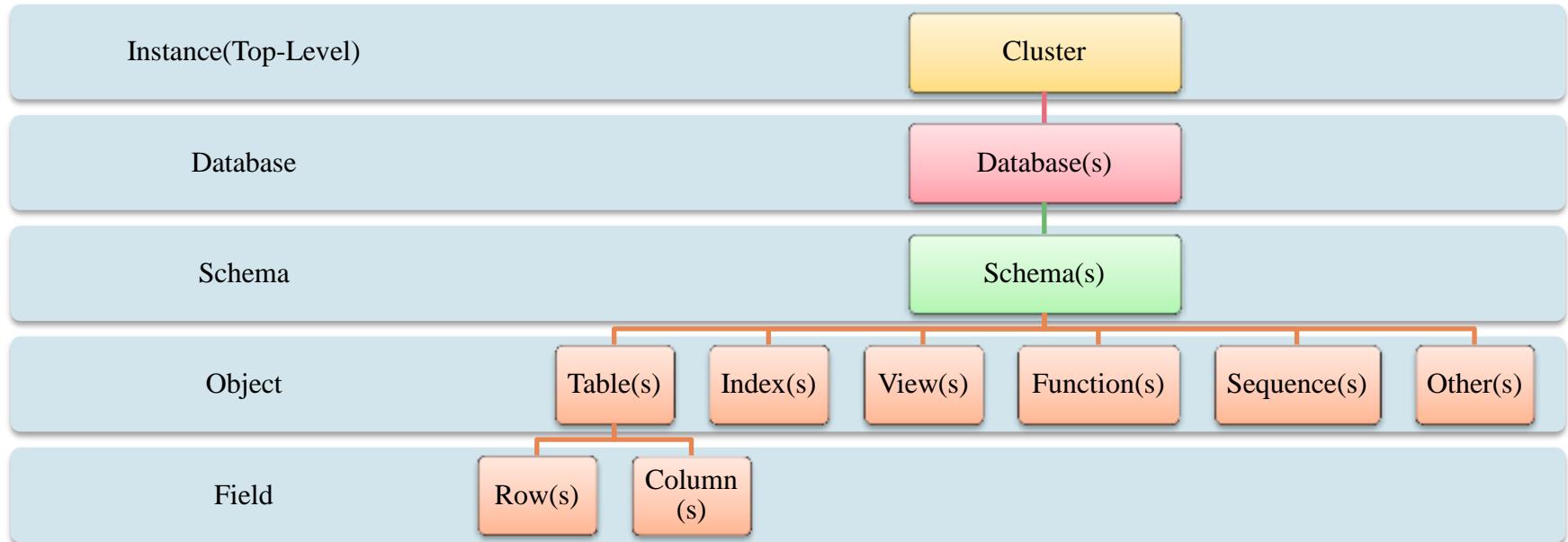
- 数据对象文件
  - tuple信息



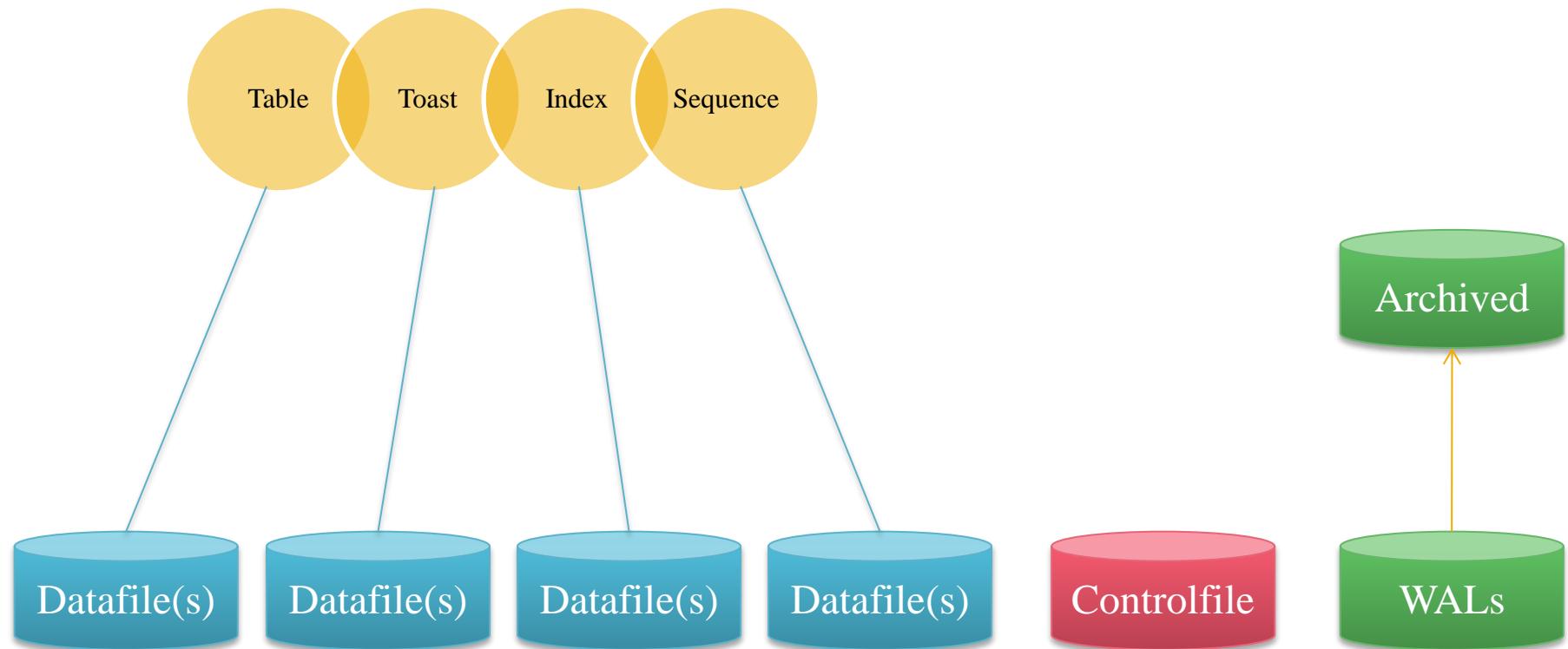
# Server Administration

- Database Layout
- Reliability
- Server Configuration
- Routine Database Maintenance Tasks
- Backup and Restore
- HA and Replication
  - Stream Replication
  - Cascade Stream Replication
  - PostgreSQL-XC , PL/Proxy , pgpool
- Monitoring Database Activity

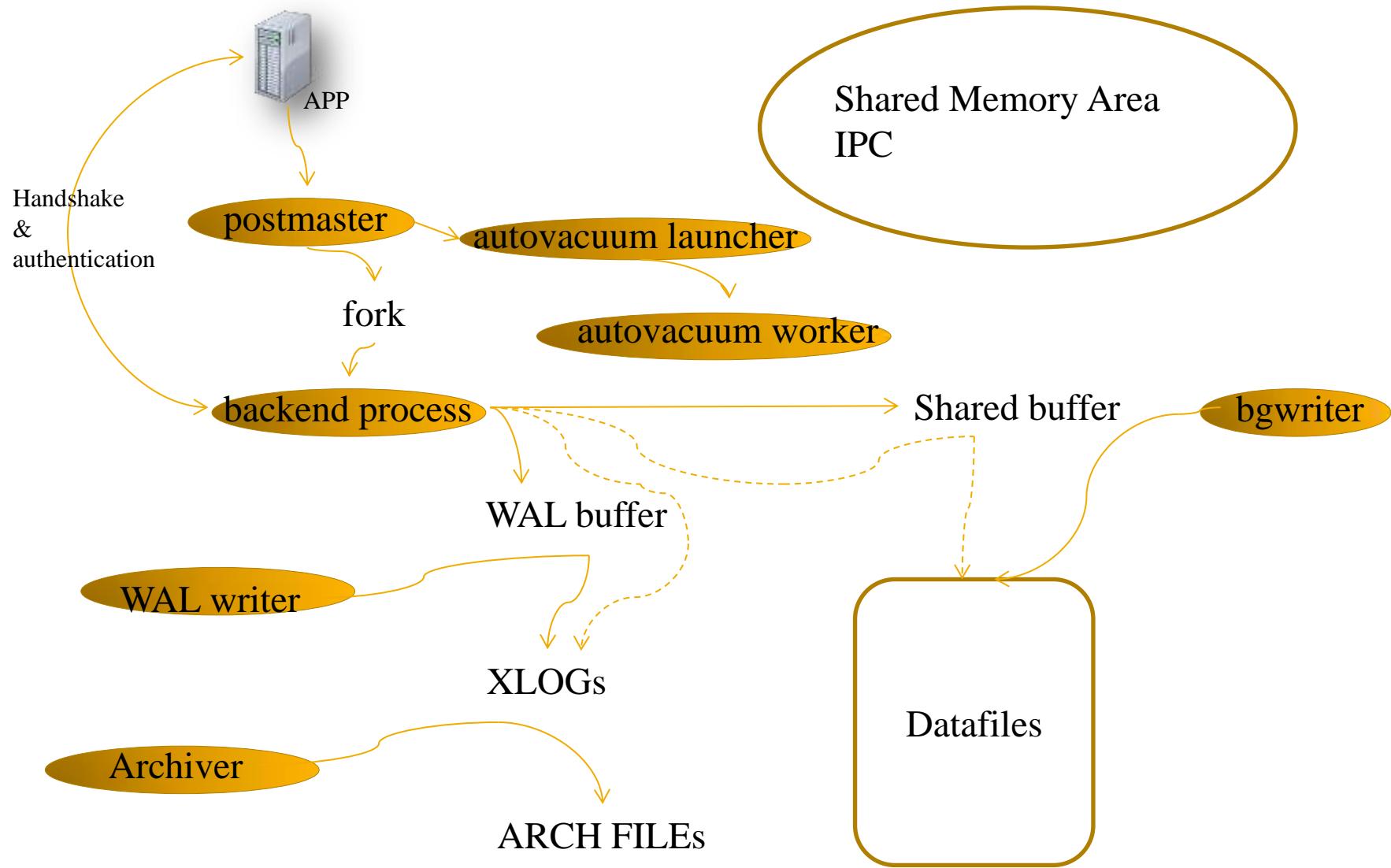
# Logical Layout



# Physical Layout



# Process Layout



# Reliability

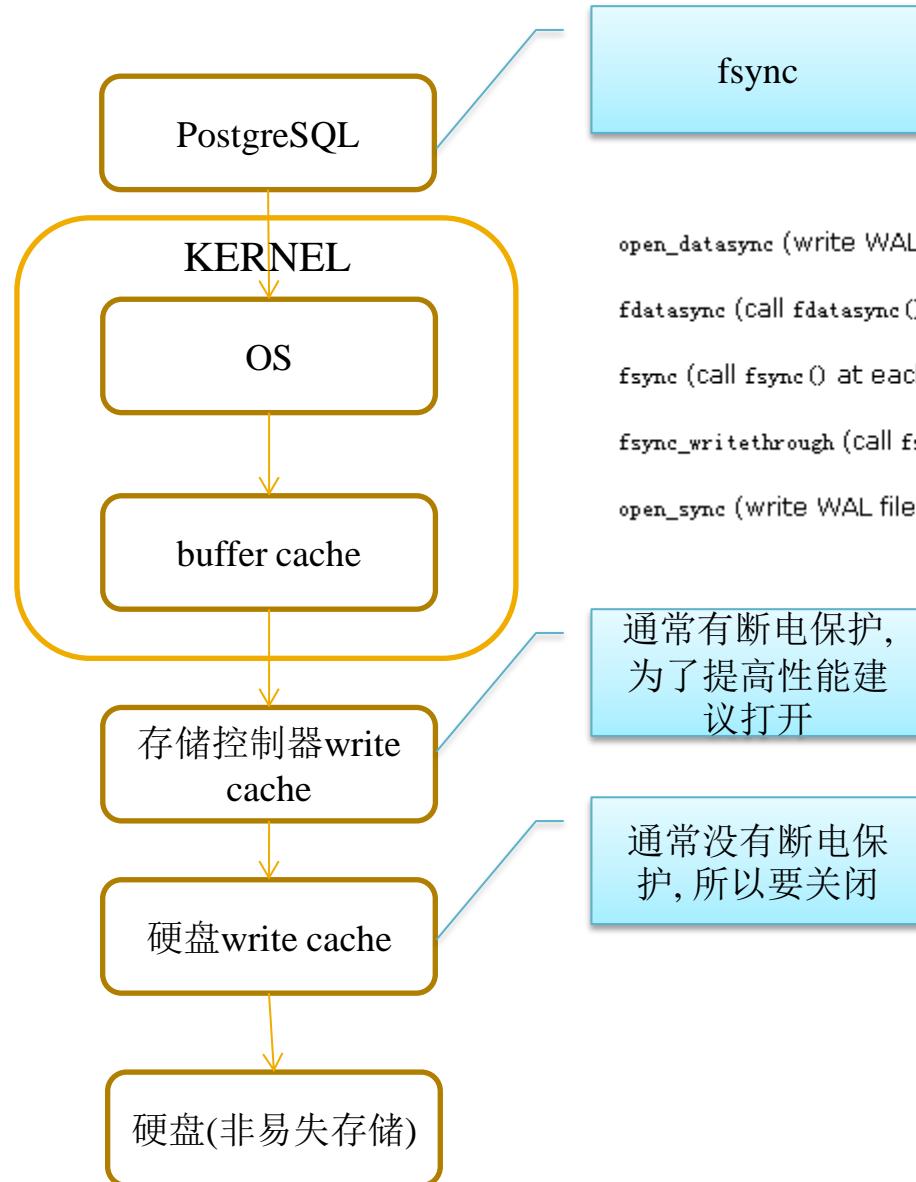
## ■ 让数据库可靠的注意事项

- 事务提交后确保这个事务未来可恢复吗?
  - 事务返回成功前, 事务日志(xlog)写入磁盘, synchronous\_commit = on
- 备份可恢复吗? 恢复后确保数据一致吗?
  - -- fsync = on . full\_page\_writes = on
- 必须写入非易失存储的数据已经写入到非易失存储了吗?
  - write - through , write - back
  - 关闭磁盘的write cache
  - 只允许有断电保护的write cache.
- 主机异常DOWN机后重启数据库能不能起到一个一致的状态?
  - PostgreSQL periodically writes full page images to permanent WAL storage **before** modifying the actual page on disk. -- full\_page\_writes = on
- 数据库异常DOWN机后重启数据库能不能起到一个一致的状态?
  - PostgreSQL periodically writes full page images to permanent WAL storage **before** modifying the actual page on disk. -- full\_page\_writes = on

# Reliability

- 让数据库可靠的注意事项
  - 事务日志可以用于恢复到任意时间点吗?
    - 开启归档, 并且有良好的备份策略.
    - wal\_level = archive 或 hot\_standby
  - 如果存储挂了怎么办?
    - 开启归档, 并且有良好的备份策略.
    - wal\_level = archive 或 hot\_standby
    - archive\_mode = on
    - archive\_command = 'cp %p /backup/%f'
  - 如果IDC挂了怎么办?
    - 开启归档, 并且有良好的备份策略.
    - wal\_level = archive 或 hot\_standby
    - 异地容灾, 如流复制.

# Reliability



PostgreSQL调用OS的sync WRITE函数。  
wal\_sync\_method=?

open\_datasync (write WAL files with `open()` option `O_DSYNC`)

`fdatasync` (call `fdatasync()` at each commit)

`fsync` (call `fsync()` at each commit)

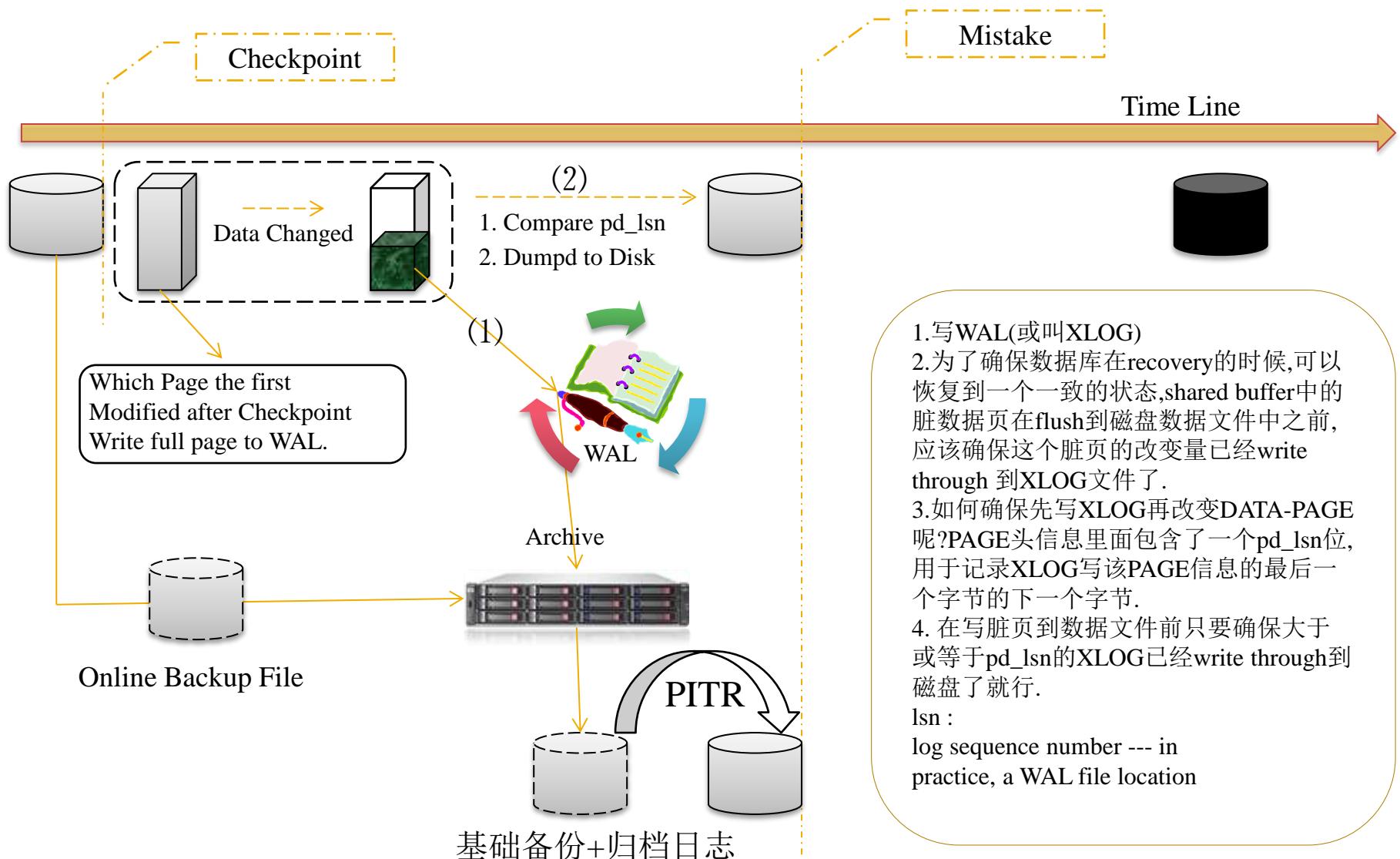
`fsync_writethrough` (call `fsync()` at each commit, forcing write-through of any disk write cache)

`open_sync` (write WAL files with `open()` option `O_SYNC`)

通常有断电保护,  
为了提高性能建  
议打开

通常没有断电保  
护, 所以要关闭

# Reliability



# Server Configuration

- 查看当前参数配置
  - SHOW ALL; SHOW ENABLE\_SEQSCAN; pg\_settings;
- 还原默认参数值(还原到优先级最高的默认参数值)
  - RESET configuration\_parameter; RESET ALL;
  - SET configuration\_parameter TO DEFAULT;
- -- 优先级从高到低
- 会话级参数配置
  - SET ENABLE\_SEQSCAN TO OFF;
- 用户级参数配置
  - ALTER ROLE SET ENABLE\_SEQSCAN TO OFF;
- 数据库级参数配置
  - ALTER DATABASE SET ENABLE\_SEQSCAN TO OFF;
- 命令行参数 -- postgres -c log\_connections=yes -c log\_destination='syslog'
- 环境变量参数 -- env PGOPTIONS='-c geqo=off'
- 默认参数配置
  - \$PGDATA/postgresql.conf

# Server Configuration

## ■ 参数值类型

- Boolean, integer, floating point, string or enum
- Boolean values can be written as on, off, true, false, yes, no, 1, 0 (all case-insensitive) or any unambiguous prefix of these.

## ■ 参数值单位

- Default units can be found by referencing pg\_settings.unit. For convenience, a different unit can also be specified explicitly.
- Valid memory units are kB (kilobytes), MB (megabytes), and GB (gigabytes);
- Note that the multiplier for memory units is 1024, not 1000.
- valid time units are ms (milliseconds), s (seconds), min (minutes), h (hours), and d (days).
- The allowed values can be found from pg\_settings.enumvals. Enum parameter values are case-insensitive.

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Autovacuum autovacuum\_freeze\_max\_age 200000000 100000000 2000000000  
\N
- Autovacuum autovacuum\_max\_workers 3 1 8388607 \N
- Autovacuum autovacuum\_vacuum\_cost\_delay 20 -1 100 \N ms
- Autovacuum autovacuum\_analyze\_scale\_factor 0.1 0 100 \N \N
- Autovacuum autovacuum\_analyze\_threshold 50 0 2147483647 \N
- Autovacuum autovacuum\_naptime 60 1 2147483 \N s
- Autovacuum autovacuum\_vacuum\_cost\_limit -1 -1 10000 \N
- Autovacuum autovacuum\_vacuum\_threshold 50 0 2147483647 \N
- Autovacuum autovacuum\_vacuum\_scale\_factor 0.2 0 100 \N \N
- Autovacuum autovacuum on \N \N \N \N

# Server Configuration

|                                                             |                                                   |          |                    |    |    |    |    |
|-------------------------------------------------------------|---------------------------------------------------|----------|--------------------|----|----|----|----|
| ■ category, name, setting, min_val, max_val, enumvals, unit |                                                   |          |                    |    |    |    |    |
| ■ Client Connection Defaults / Locale and Formatting        | server_encoding                                   | UTF8     | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | lc_collate                                        | C        | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | lc_ctype                                          | C        | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | lc_messages                                       | C        | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | timezone_abbreviations                            | Default  | \N                 | \N | \N |    |    |
|                                                             |                                                   | \N       |                    |    |    |    |    |
| ■ Client Connection Defaults / Locale and Formatting        | extra_float_digits                                | 0        | -15                | 3  | \N |    |    |
| ■ Client Connection Defaults / Locale and Formatting        | TimeZone                                          | PRC      | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | client_encoding                                   | UTF8     | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | DateStyle                                         | ISO, MDY | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | lc_time                                           | C        | \N                 | \N | \N | \N |    |
| ■ Client Connection Defaults / Locale and Formatting        | default_text_search_config                        |          | pg_catalog.english |    |    |    |    |
|                                                             | \N                                                | \N       | \N\N               |    |    |    |    |
| ■ Client Connection Defaults / Locale and Formatting        | lc_numeric                                        | C        | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | lc_monetary                                       | C        | \N                 | \N | \N | \N | \N |
| ■ Client Connection Defaults / Locale and Formatting        | IntervalStyle                                     | postgres | \N                 | \N |    |    |    |
|                                                             | {postgres,postgres_verbose,sql_standard,iso_8601} | \N       |                    |    |    |    |    |

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Client Connection Defaults / Other Defaults local\_preload\_libraries \N \N \N \N
- Client Connection Defaults / Other Defaults dynamic\_library\_path \$libdir \N \N \N \N
- Client Connection Defaults / Other Defaults tcp\_keepalives\_idle 0 0 2147483647 \N s
- Client Connection Defaults / Other Defaults gin\_fuzzy\_search\_limit 0 0 2147483647 \N
- Client Connection Defaults / Other Defaults tcp\_keepalives\_interval 0 0 2147483647 \N s
- Client Connection Defaults / Other Defaults tcp\_keepalives\_count 0 0 2147483647 \N
- Client Connection Defaults / Statement Behavior session\_replication\_role origin \N \N {origin,replica,local} \N
- Client Connection Defaults / Statement Behavior statement\_timeout 0 0 2147483647 \N ms
- Client Connection Defaults / Statement Behavior check\_function\_bodies on \N \N \N \N
- Client Connection Defaults / Statement Behavior vacuum\_freeze\_table\_age 150000000 0 2000000000 \N
- Client Connection Defaults / Statement Behavior xmlbinary base64 \N \N {base64,hex} \N
- Client Connection Defaults / Statement Behavior temp\_tablespaces \N \N \N \N
- Client Connection Defaults / Statement Behavior xmloption content \N \N {content,document} \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Client Connection Defaults / Statement Behavior bytea\_output hex \N \N {escape,hex} \N
- Client Connection Defaults / Statement Behavior vacuum\_freeze\_min\_age 500000000 0 1000000000 \N
- Client Connection Defaults / Statement Behavior search\_path "\$user",public \N \N \N \N
- Client Connection Defaults / Statement Behavior default\_tablespace \N \N \N \N
- Client Connection Defaults / Statement Behavior default\_transaction\_deferrable off \N \N \N \N
- Client Connection Defaults / Statement Behavior default\_transaction\_isolation read committed \N \N {serializable,"repeatable read","read committed","read uncommitted"} \N
- Client Connection Defaults / Statement Behavior default\_transaction\_read\_only off \N \N \N \N
- Client Connection Defaults / Statement Behavior transaction\_read\_only off \N \N \N \N
- Client Connection Defaults / Statement Behavior transaction\_isolation read committed \N \N \N \N
- Client Connection Defaults / Statement Behavior transaction\_deferrable off \N \N \N \N
- Connections and Authentication / Connection Settings max\_connections 100 1 8388607 \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Connections and Authentication / Connection Settings listen\_addresses localhost \N \N \N
- Connections and Authentication / Connection Settings unix\_socket\_group \N \N \N \N
- Connections and Authentication / Connection Settings unix\_socket\_directory \N \N \N \N
- Connections and Authentication / Connection Settings bonjour\_name \N \N \N \N
- Connections and Authentication / Connection Settings bonjour off \N \N \N \N
- Connections and Authentication / Connection Settings superuser\_reserved\_connections 3 0 8388607 \N
- Connections and Authentication / Connection Settings unix\_socket\_permissions 0777 0 511 \N
- Connections and Authentication / Connection Settings port 1921 1 65535 \N
- Connections and Authentication / Security and Authentication ssl off \N \N \N \N
- Connections and Authentication / Security and Authentication ssl\_ciphers ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Connections and Authentication / Security and Authentication db\_user\_namespace off \N \N \N
- Connections and Authentication / Security and Authentication authentication\_timeout 60 1 600 \N s
- Connections and Authentication / Security and Authentication krb\_server\_keyfile \N \N \N \N
- Connections and Authentication / Security and Authentication krb\_caseins\_users off \N \N \N \N \N
- Connections and Authentication / Security and Authentication krb\_srvname postgres \N \N \N \N \N \N \N
- Connections and Authentication / Security and Authentication ssl\_renegotiation\_limit 524288 0 2147483647 \N kB
- Connections and Authentication / Security and Authentication password\_encryption on \N \N \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Customized Options custom\_variable\_classes \N \N \N \N
- Developer Options ignore\_system\_indexes off \N \N \N \N
- Developer Options post\_auth\_delay 0 0 2147483647 \N s
- Developer Options allow\_system\_table\_mods on \N \N \N \N
- Developer Options trace\_recovery\_messages log \N \N {debug5,debug4,debug3,debug2,debug1,log,notice,warning,error} \N
- Developer Options pre\_auth\_delay 0 0 60 \N s
- Developer Options zero\_damaged\_pages off \N \N \N \N
- Developer Options debug\_assertions off \N \N \N \N
- Developer Options trace\_sort off \N \N \N \N
- Developer Options trace\_notify off \N \N \N \N
- Error Handling restart\_after\_crash on \N \N \N \N
- Error Handling exit\_on\_error off \N \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- File Locations config\_file /pgdata/digoal/1921/data02/pg\_root/postgresql.conf \N \N \N \N
- File Locations hba\_file /pgdata/digoal/1921/data02/pg\_root/pg\_hba.conf \N \N \N \N
- File Locations data\_directory /pgdata/digoal/1921/data02/pg\_root \N \N \N \N
- File Locations ident\_file /pgdata/digoal/1921/data02/pg\_root/pg\_ident.conf \N \N \N \N
- File Locations external\_pid\_file \N \N \N \N
- Lock Management max\_pred\_locks\_per\_transaction 640000 10 2147483647 \N
- Lock Management max\_locks\_per\_transaction 64 10 2147483647 \N
- Lock Management deadlock\_timeout 1000 1 2147483647 \N ms
- Preset Options server\_version 9.1.3 \N \N \N \N
- Preset Options wal\_block\_size 8192 8192 8192 \N
- Preset Options server\_version\_num 90103 90103 90103 \N
- Preset Options block\_size 8192 8192 8192 \N
- Preset Options segment\_size 131072 131072 131072 \N 8kB
- Preset Options integer\_datetimes on \N \N \N \N
- Preset Options max\_index\_keys 32 32 32 \N
- Preset Options wal\_segment\_size 8192 8192 8192 \N 8kB
- Preset Options max\_identifier\_length 63 63 63 \N
- Preset Options max\_function\_args 100 100 100 \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Query Tuning / Genetic Query Optimizer geqo\_effort 5 1 10 \N
- Query Tuning / Genetic Query Optimizer geqo\_on \N \N \N \N
- Query Tuning / Genetic Query Optimizer geqo\_generations 0 0 2147483647 \N
- Query Tuning / Genetic Query Optimizer geqo\_pool\_size 0 0 2147483647 \N
- Query Tuning / Genetic Query Optimizer geqo\_seed 0 0 1 \N \N
- Query Tuning / Genetic Query Optimizer geqo\_selection\_bias 2 1.5 2 \N \N
- Query Tuning / Genetic Query Optimizer geqo\_threshold 12 2 2147483647 \N
- Query Tuning / Other Planner Options constraint\_exclusion partition \N \N {partition,on,off}  
\N
- Query Tuning / Other Planner Options fromCollapse\_limit 8 1 2147483647 \N
- Query Tuning / Other Planner Options cursor\_tuple\_fraction 0.1 0 1 \N \N
- Query Tuning / Other Planner Options joinCollapse\_limit 8 1 2147483647 \N
- Query Tuning / Other Planner Options default\_statistics\_target 100 1 10000 \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Query Tuning / Planner Cost Constants cpu\_operator\_cost 0.0025 0 1.79769e+308 \N \N
- Query Tuning / Planner Cost Constants effective\_cache\_size 16384 1 2147483647 \N 8kB
- Query Tuning / Planner Cost Constants cpu\_index\_tuple\_cost 0.005 0 1.79769e+308 \N \N
- Query Tuning / Planner Cost Constants cpu\_tuple\_cost 0.01 0 1.79769e+308 \N \N
- Query Tuning / Planner Cost Constants seq\_page\_cost 1 0 1.79769e+308 \N \N
- Query Tuning / Planner Cost Constants random\_page\_cost 4 0 1.79769e+308 \N \N
- Query Tuning / Planner Method Configuration enable\_hashjoin on \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_indexscan on \N \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_material on \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_mergejoin on \N \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_tidscan on \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_sort on \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_nestloop on \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_seqscan off \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_bitmapscan on \N \N \N \N \N
- Query Tuning / Planner Method Configuration enable\_hashagg on \N \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Replication / Master Server max\_wal\_senders 0 0 8388607 \N
- Replication / Master Server wal\_keep\_segments 0 0 2147483647 \N
- Replication / Master Server synchronous\_standby\_names \N \N \N \N
- Replication / Master Server wal\_sender\_delay 1000 1 10000 \N ms
- Replication / Master Server replication\_timeout 60000 0 2147483647 \N ms
- Replication / Master Server vacuum\_defer\_cleanup\_age 0 0 1000000 \N
- Replication / Standby Servers hot\_standby off \N \N \N \N
- Replication / Standby Servers max\_standby\_streaming\_delay 30000 -1 2147483647 \N ms
- Replication / Standby Servers hot\_standby\_feedback off \N \N \N \N
- Replication / Standby Servers wal\_receiver\_status\_interval 10 0 2147483 \N s
- Replication / Standby Servers max\_standby\_archive\_delay 30000 -1 2147483647 \N ms

# Server Configuration

|                                                             |                             |         |    |            |                         |    |    |
|-------------------------------------------------------------|-----------------------------|---------|----|------------|-------------------------|----|----|
| ■ category, name, setting, min_val, max_val, enumvals, unit |                             |         |    |            |                         |    |    |
| ■ Reporting and Logging / What to Log                       | log_disconnections          | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_connections             | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_line_prefix             |         | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_autovacuum_min_duration | -1      | -1 | 2147483647 |                         | \N | ms |
| ■ Reporting and Logging / What to Log                       | log_hostname                | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_timezone                | PRC     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_checkpoints             | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_statement               | none    | \N | \N         | {none,ddl,mod,all}      |    | \N |
| ■ Reporting and Logging / What to Log                       | log_duration                | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_error_verbosity         | default | \N | \N         | {terse,default,verbose} |    | \N |
| ■ Reporting and Logging / What to Log                       | log_lock_waits              | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | log_temp_files              | -1      | -1 | 2147483647 |                         | \N | kB |
| ■ Reporting and Logging / What to Log                       | debug_pretty_print          | on      | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | debug_print_parse           | off     | \N | \N         | \N                      | \N |    |
| ■ Reporting and Logging / What to Log                       | debug_print_plan            | off     | \N | \N         | \N                      | \N |    |

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Reporting and Logging / What to Log application\_name psql \N \N \N \N
- Reporting and Logging / What to Log debug\_print\_rewritten off \N \N \N \N
- Reporting and Logging / When to Log log\_min\_duration\_statement -1 -1 2147483647 \N ms
- Reporting and Logging / When to Log log\_min\_messages warning \N \N {debug5,debug4,debug3,debug2,debug1,info,notice,warning,error,log,fatal,panic} \N
- Reporting and Logging / When to Log log\_min\_error\_statement error \N \N {debug5,debug4,debug3,debug2,debug1,info,notice,warning,error,log,fatal,panic} \N
- Reporting and Logging / When to Log client\_min\_messages notice \N \N {debug5,debug4,debug3,debug2,debug1,log,notice,warning,error} \N
- Reporting and Logging / Where to Log silent\_mode off \N \N \N \N
- Reporting and Logging / Where to Log logging\_collector off \N \N \N \N \N
- Reporting and Logging / Where to Log log\_rotation\_size 10240 0 2097151 \N kB
- Reporting and Logging / Where to Log log\_truncate\_on\_rotation off \N \N \N \N
- Reporting and Logging / Where to Log log\_destination stderr \N \N \N \N
- Reporting and Logging / Where to Log log\_filename postgresql-%Y-%m-%d\_%H%M%S.log \N \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Reporting and Logging / Where to Log log\_rotation\_age 1440 0 35791394 \N min
- Reporting and Logging / Where to Log log\_directory pg\_log \N \N \N \N
- Reporting and Logging / Where to Log syslog\_ident postgres \N \N \N \N
- Reporting and Logging / Where to Log syslog\_facility local0 \N {local0,local1,local2,local3,local4,local5,local6,local7} \N
- Reporting and Logging / Where to Log log\_file\_mode 0600 0 511 \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Resource Usage / Asynchronous Behavior effective\_ioConcurrency 1 0 1000 \N
- Resource Usage / Background Writer bgwriter\_lru\_multiplier 2 0 10 \N \N
- Resource Usage / Background Writer bgwriter\_lru\_maxpages 100 0 1000 \N
- Resource Usage / Background Writer bgwriter\_delay 200 10 10000 \N ms
- Resource Usage / Cost-Based Vacuum Delay vacuum\_cost\_page\_dirty 20 0 10000 \N
- Resource Usage / Cost-Based Vacuum Delay vacuum\_cost\_page\_hit 1 0 10000 \N
- Resource Usage / Cost-Based Vacuum Delay vacuum\_cost\_page\_miss 10 0 10000 \N
- Resource Usage / Cost-Based Vacuum Delay vacuum\_cost\_delay 0 0 100 \N ms
- Resource Usage / Cost-Based Vacuum Delay vacuum\_cost\_limit 200 1 10000 \N
- Resource Usage / Kernel Resources shared\_preload\_libraries \N \N \N \N
- Resource Usage / Kernel Resources max\_files\_per\_process 1000 25 2147483647 \N
- Resource Usage / Memory track\_activity\_query\_size 1024 100 102400 \N
- Resource Usage / Memory shared\_buffers 4096 16 1073741823 \N 8kB
- Resource Usage / Memory max\_prepared\_transactions 0 0 8388607 \N
- Resource Usage / Memory max\_stack\_depth 2048 100 2147483647 \N kB
- Resource Usage / Memory temp\_buffers 1024 100 1073741823 \N 8kB
- Resource Usage / Memory work\_mem 1024 64 2147483647 \N kB
- Resource Usage / Memory maintenance\_work\_mem 16384 1024 2147483647 \N kB

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Statistics / Monitoring log\_parser\_stats off \N \N \N \N
- Statistics / Monitoring log\_executor\_stats off \N \N \N \N
- Statistics / Monitoring log\_planner\_stats off \N \N \N \N
- Statistics / Monitoring log\_statement\_stats off \N \N \N \N
- Statistics / Query and Index Statistics Collector stats\_temp\_directory pg\_stat\_tmp \N \N \N  
\N
- Statistics / Query and Index Statistics Collector track\_functions none \N \N {none,pl,all} \N
- Statistics / Query and Index Statistics Collector track\_activities on \N \N \N \N
- Statistics / Query and Index Statistics Collector track\_counts on \N \N \N \N
- Statistics / Query and Index Statistics Collector update\_process\_title on \N \N \N \N

# Server Configuration

- category, name, setting, min\_val, max\_val, enumvals, unit
- Version and Platform Compatibility / Other Platforms and Clients transform\_null\_equals off \N  
\N \N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions lo\_compat\_privileges off \N  
\N \N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions quote\_all\_identifiers off \N  
\N \N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions synchronize\_seqscans on \N  
\N \N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions sql\_inheritance on \N \N  
\N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions escape\_string\_warning on \N  
\N \N \N
- Version and Platform Compatibility / Previous PostgreSQL Versions backslash\_quote safe\_encoding \N  
\N {safe\_encoding,on,off} \N
- Version and Platform Compatibility / Previous PostgreSQL Versions array\_nulls on \N \N \N  
\N
- Version and Platform Compatibility / Previous PostgreSQL Versions default\_with\_oids off \N  
\N \N \N

# Server Configuration

- Version and Platform Compatibility / Previous PostgreSQL Versions standard\_conforming\_strings on  
\N \N \N\N
- Write-Ahead Log / Archiving archive\_mode off \N \N \N \N
- Write-Ahead Log / Archiving archive\_command (disabled) \N \N \N \N
- Write-Ahead Log / Archiving archive\_timeout 0 0 2147483647 \N s
- Write-Ahead Log / Checkpoints checkpoint\_timeout 300 30 3600 \N s
- Write-Ahead Log / Checkpoints checkpoint\_warning 30 0 2147483647 \N s
- Write-Ahead Log / Checkpoints checkpoint\_completion\_target 0.5 0 1 \N \N
- Write-Ahead Log / Checkpoints checkpoint\_segments 3 1 2147483647 \N
- Write-Ahead Log / Settings wal\_level minimal \N \N {minimal,archive,hot\_standby} \N
- Write-Ahead Log / Settings wal\_buffers 128 -1 2147483647 \N 8kB
- Write-Ahead Log / Settings fsync on \N \N \N \N
- Write-Ahead Log / Settings wal\_sync\_method fdatasync \N \N {fsync,fdatasync,open\_sync} \N
- Write-Ahead Log / Settings wal\_writer\_delay 200 1 10000 \N ms
- Write-Ahead Log / Settings full\_page\_writes on \N \N \N \N
- Write-Ahead Log / Settings commit\_delay 0 0 100000 \N
- Write-Ahead Log / Settings synchronous\_commit on \N \N {local,on,off} \N
- Write-Ahead Log / Settings commit\_siblings 5 0 1000 \N

# Server Configuration

- 通常初始化完数据库后需要调整的参数
- `listen_addresses = '0.0.0.0'` -- 监听地址
- `port = 5432` -- 可更改为其他端口
- `max_connections = 1000` -- 最大允许的连接, 如果并发到达这么高可能需要500核的机器才能处理得过来, 否则性能会有下降. 有些业务不会释放连接, 所以可能导致连接占用多, 实际在处理请求的少. 因此这个数字看实际使用环境来设定.
- `superuser_reserved_connections = 13` -- 保留给超级用户的连接个数
- `unix_socket_directory = '/pgdata/digoal/1921/data02/pg_root'` -- 默认是/tmp, 不太安全. 放到\$PGDATA比较靠谱, 因为\$PGDATA的目录权限是700的.
- `unix_socket_permissions = 0700` -- 修改unix socket文件的权限为700
- `tcp_keepalives_idle = 60` -- tcp连接空闲多长时间后发出keepalive包
- `tcp_keepalives_interval = 10` -- 间隔多长时间再发一次
- `tcp_keepalives_count = 6` -- 总共发几次keepalive包
- `shared_buffers = 512MB` -- 在第一天的内容中包含了计算`shared_buffers`的方法
- `work_mem = 1MB` -- 默认是1MB, 如果发现数据经常使用临时文件排序或group by等, 可以考虑设置为一个比较大的值. 按需使用, 每个排序或merge JOIN用到的哈希表,DISTINCT,都需要消耗`work_mem`, 如果一个执行计划中有多个此类操作则最大需要使用多个`work_mem`.

# Server Configuration

- `maintenance_work_mem = 512MB` -- 用于创建索引的操作, vacuum操作. 按需使用 `maintenance_work_mem` 设置的内存, 当有并发的创建索引和 autovacuum 等操作时可能造成内存消耗过度.
- `max_stack_depth = 8MB` -- 一般设置为 ulimit 的 stack size 一致或略小.
- `shared_preload_libraries = 'pg_stat_statements'` -- 启动数据库集群时加载的库, 这里表示加载 pg\_stat\_statements, 一个用于统计 SQL 执行次数, CPU 开销等的模块.
- `vacuum_cost_delay = 10ms` -- VACUUM 操作比较消耗 IO, 设置延时是指 VACUUM 操作消耗的成本大于 `vacuum_cost_limit` 后延迟 10 毫秒再继续执行.
- `bgwriter_delay = 10ms` -- 每个 background writer 运行周期之间延迟 10 毫秒.
- `wal_level = hot_standby` -- WAL\_level 级别, 如果要开启备份必须设置为 archive 或 hot\_standby, 如果要建立 hot\_standby 则必须设置为 hot\_standby.
- `synchronous_commit = off` -- 关闭 XLOG 的同步写. 可以大大提高写事务的处理能力. 不会破坏数据库一致性, 但是如果数据库异常 DOWN 机需要 recovery 时, 恢复后的数据库可能丢失最后 10 毫秒 (wal\_writer\_delay) 的事务.
- `wal_sync_method = fdatasync` -- 使用 pg\_test\_fsync 测试出系统使用哪种 sync 接口效率最高.
- `wal_buffers = 16384kB` -- 一般繁忙的系统设置为 xlog 文件段的大小.

# Server Configuration

- wal\_writer\_delay = 10ms -- WAL日志写操作round之间延迟10毫秒
- commit\_delay = 0 -- 在事务提交的同时如果系统中有大于等于commit\_siblings个未提交事务时, 等待0毫秒. 合并这些提交事务的IO请求,降低IO请求次数.
- commit\_siblings = 5
- checkpoint\_segments = 256 -- 多少个xlog rotate后触发checkpoint, checkpoint segments一般设置为大于shared\_buffer的SIZE. 如shared\_buffer=1024MB, wal文件单个16MB, 则checkpoint\_segments>=1024/16;
- archive\_mode = on -- 开启归档, 修改这个配置需要重启, 所以一般安装好就开启
- archive\_command = '/bin/date' -- 这个可以RELOAD, 一般的做法是先设置一个空转命令
- max\_wal\_senders = 32 -- 修改这个配置需要重启数据库, 所以一般的做法是先设置一个数字.
- wal\_sender\_delay = 10ms -- In each round the WAL sender sends any WAL accumulated since the last round to the standby server. It then sleeps for wal\_sender\_delay milliseconds, and repeats. The sleep is interrupted by transaction commit, so the effects of a committed transaction are sent to standby servers as soon as the commit happens, regardless of this setting.
- wal\_keep\_segments = 0 -- 在主库中至少保留多少个xlog segment, 哪怕有一些XLOG已经不需要被数据库recovery使用.

# Server Configuration

- `#synchronous_standby_names = "` -- 如果打算配置同步流复制, 则需要配置这个参数. 同一时间只有一个同步复制角色standby, 如果这个节点挂了或者因为某些原因延迟了, 第二个配置节点将接替同步复制standby的角色.
- `hot_standby = on` -- 这个是standby节点的配置, 是否允许客户端连接standby进行readonly操作.
- `max_standby_archive_delay = 300s` -- 在规定的时间内必须完成archive standby的replay操作. 不影响接收操作. 计时从最近一次replay赶上receive的时间开始算.
- `max_standby_streaming_delay = 300s` -- 在规定的时间内必须完成streaming standby的replay操作. 不影响接收操作. 计时从最近一次replay赶上receive的时间开始算.
- `wal_receiver_status_interval = 10s` -- Specifies the minimum frequency for the WAL receiver process on the standby to send information about replication progress to the primary, where it can be seen using the `pg_stat_replication` view. The standby will report the last transaction log position it has written, the last position it has flushed to disk, and the last position it has applied. Updates are sent **each time the write or flush positions change, or at least as often as specified by this parameter**. Thus, the apply position may lag slightly behind the true position.
- `hot_standby_feedback = on` -- Specifies whether or not a hot standby will send feedback to the primary about queries currently executing on the standby.

# Server Configuration

- random\_page\_cost = 2.0 -- 调小后更倾向使用索引, 而非全表扫描.
- effective\_cache\_size = 12000MB -- 调大后更倾向使用索引, 而非全表扫描.
- log\_destination = 'csvlog' -- 便于导入到库中进行分析
- logging\_collector = on
- log\_directory = '/var/applog/pg\_log/集群名/port号'
- log\_filename = 'postgresql-%Y-%m-%d\_%H%M%S.log'
- log\_file\_mode = 0600
- log\_truncate\_on\_rotation = on -- 便于维护日志文件
- log\_rotation\_age = 1d -- 表示一天建立一个日志文件
- log\_rotation\_size = 10MB -- 表示大于10MB后新建一个日志文件
- log\_min\_duration\_statement = 1000ms -- 记录运行时间超过1秒的SQL
- log\_checkpoints = on -- 记录checkpoint的运行情况
- log\_lock\_waits = on -- 记录锁等待时间
- log\_statement = 'ddl' -- 记录DDLSQL

# Server Configuration

- `track_activity_query_size = 2048` -- 记录SQL长度最大限度改为2048, 可以记录更长的SQL
- `autovacuum = on` -- 开启自动vacuum
- `log_autovacuum_min_duration = 0` -- 记录所有的auto vacuum动作
- `deadlock_timeout = 1s` -- 死锁检测的最小值为1秒, 如果系统因为检测死锁造成压力较大可以调大这个值
- `custom_variable_classes = 'pg_stat_statements'` -- pg\_stat\_statements模块的定制参数
- `pg_stat_statements.max = 1000`
- `pg_stat_statements.track = all`
  
- 危险设置, 将导致数据库CRASH后不可恢复或数据不一致.
- `fsync = off`
- `full_page_writes = off`

# Server Configuration

## ■ 模块参数

- <http://www.postgresql.org/docs/9.1/static/runtime-config-custom.html>

## ■ 开发参数 -- 一般用于调试, 恢复等特殊场景.

- <http://www.postgresql.org/docs/9.1/static/runtime-config-developer.html>

## ■ 命令行选项



| Short Option                      | Equivalent                                                                                                                                                        |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -A <i>x</i>                       | debug_assertions = <i>x</i>                                                                                                                                       |
| -B <i>x</i>                       | shared_buffers = <i>x</i>                                                                                                                                         |
| -d <i>x</i>                       | log_min_messages = DEBUG <i>x</i>                                                                                                                                 |
| -e                                | datestyle = euro                                                                                                                                                  |
| -fb, -fh, -fi, -fm, -fn, -fs, -ft | enable_bitmapscan = off, enable_hashjoin = off, enable_indexscan = off, enable_mergejoin = off, enable_nestloop = off, enable_seqscan = off, enable_tidscan = off |
| -F                                | fsync = off                                                                                                                                                       |
| -h <i>x</i>                       | listen_addresses = <i>x</i>                                                                                                                                       |
| -i                                | listen_addresses = '*'                                                                                                                                            |
| -k <i>x</i>                       | unix_socket_directory = <i>x</i>                                                                                                                                  |
| -l                                | ssl = on                                                                                                                                                          |
| -N <i>x</i>                       | max_connections = <i>x</i>                                                                                                                                        |
| -O                                | allow_system_table_mods = on                                                                                                                                      |
| -p <i>x</i>                       | port = <i>x</i>                                                                                                                                                   |
| -P                                | ignore_system_indexes = on                                                                                                                                        |
| -s                                | log_statement_stats = on                                                                                                                                          |
| -S <i>x</i>                       | work_mem = <i>x</i>                                                                                                                                               |
| -tpa, -tpl, -te                   | log_parser_stats = on, log_planner_stats = on, log_executor_stats = on                                                                                            |
| -W <i>x</i>                       | post_auth_delay = <i>x</i>                                                                                                                                        |

# Routine Database Maintenance Tasks

## ■ Routine Vacuuming

### ■ 为什么要vacuum

- PostgreSQL的MVCC机制, 有很好的读些并发性以及极高的事务隔离性, 但是由于数据更新和删除操作后并没有在物理上从PAGE里面删除, 所以需要一种机制来回收这些费数据. 否则会导致膨胀. 一般的做法是让系统自动回收, 开启autovacuum.

- Preventing Transaction ID Wraparound Failures.

### ■ autovacuum

### ■ 如何跟踪哪些PAGE有脏数据需要回收

- PostgreSQL 8.3以及更老的版本

- max\_fsm\_pages
- Six bytes of shared memory are consumed for each page slot.
- max\_fsm\_relations
- Roughly seventy bytes of shared memory are consumed for each slot.
- 可能溢出, 跟踪不到.

- PostgreSQL 8.4以及更新的版本

- fsm, vm文件(对应每个对象). 不会溢出, vm(no dead tuple pages)加入后可以大大降低扫描的块的数量.

# Routine Database Maintenance Tasks

## ■ Routine Vacuuming

### ■ fsm 结构

- PostgreSQL 8.4 Free Space Map Principle
- <http://blog.163.com/digoal@126/blog/static/163877040201041115555401/>



### ■ autovacuum 在什么情况下会被触发

- autovacuum = on
- autovacuum\_vacuum\_threshold = 50
- autovacuum\_analyze\_threshold = 50
- autovacuum\_vacuum\_scale\_factor = 0.2
- autovacuum\_analyze\_scale\_factor = 0.1
- vacuum threshold = vacuum base threshold + vacuum scale factor \* number of tuples
- analyze threshold = analyze base threshold + analyze scale factor \* number of tuples
- -- Preventing Transaction ID Wraparound Failures.
- Autovacuum is invoked on any table that might contain XIDs older than the age specified by the configuration parameter autovacuum\_freeze\_max\_age. (This will happen even if autovacuum is disabled.)

# Routine Database Maintenance Tasks

## ■ Routine Vacuuming

### ■ autovacuum在什么情况下会被触发

- If for some reason autovacuum fails to clear old XIDs from a table, the system will begin to emit warning messages like this when the database's oldest XIDs reach ten million transactions from the wraparound point:
- WARNING: database "mydb" must be vacuumed within 177009986 transactions
- HINT: To avoid a database shutdown, execute a database-wide VACUUM in "mydb".
  
- If these warnings are ignored, the system will shut down and refuse to start any new transactions once there are fewer than 1 million transactions left until wraparound:
- ERROR: database is not accepting commands to avoid wraparound data loss in database "mydb"
- HINT: Stop the postmaster and use a standalone backend to VACUUM in "mydb".

# Routine Database Maintenance Tasks

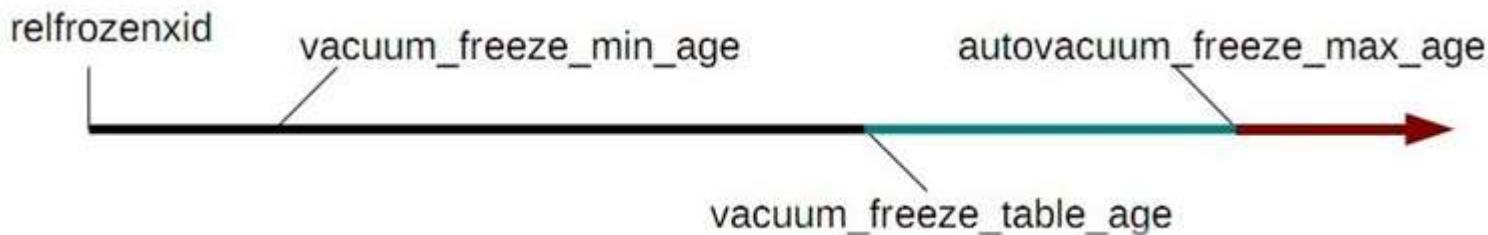
## ■ Routine Vacuuming

### ■ 不同的触发场景分别扫描哪些块

- 一个表对应的系统表pg\_class中的relfrozenxid字段的值表示这个表所有记录中存在的最老的记录. 只有发生扫全表的vacuum后(请区别于VACUUM FULL), 才会更新这个值.
- 当执行vacuum时, 决定要做什么?, 首先它会获取表的年龄age(pg\_class.relfrozenxid), 用这个年龄和下面的参数进行比较.
- vacuum根据vacuum\_freeze\_min\_age参数的值来决定要把哪些行的版本号更新为FrozenXID(比任何版本号都老的版本号).
- vacuum根据vacuum\_freeze\_table\_age参数的值来决定是否要扫表的所有块, 也就是扫完后可以更新pg\_class中的relfrozenxid字段.
- 还有两种情况是vacuum freeze, 或所有PAGE都有dead row版本需要扫描.
  - The whole table is scanned when
  - relfrozenxid is more than vacuum\_freeze\_table\_age transactions old, when VACUUM's FREEZE option is used,
  - or when all pages happen to require vacuuming to remove dead row versions.

# Routine Database Maintenance Tasks

## ■ Routine Vacuuming



## ■ 表级别的autovacuum参数

```
autovacuum_enabled, toast.autovacuum_enabled <boolean>
autovacuum_vacuum_threshold, toast.autovacuum_vacuum_threshold <integer>
autovacuum_vacuum_scale_factor, toast.autovacuum_vacuum_scale_factor <float4>
autovacuum_analyze_threshold <integer>
autovacuum_analyze_scale_factor <float4>
autovacuum_vacuum_cost_delay, toast.autovacuum_vacuum_cost_delay <integer>
autovacuum_vacuum_cost_limit, toast.autovacuum_vacuum_cost_limit <integer>
autovacuum_freeze_min_age, toast.autovacuum_freeze_min_age <integer>
autovacuum_freeze_max_age, toast.autovacuum_freeze_max_age <integer>
autovacuum_freeze_table_age, toast.autovacuum_freeze_table_age <integer>
```

# Routine Database Maintenance Tasks

- Routine Reindexing
  - 为什么要reindex b-tree 索引
    - 频繁的non-HOT Update后b-tree索引会膨胀, B-tree index pages that have become completely empty are reclaimed for re-use.
    - 一个全新的btree索引和一个频繁non-HOT Update后的b-tree索引的page逻辑顺序和物理顺序顺性不一样, 效率也不一样.
    - concurrently rebuild bloated indexes
    - <http://blog.163.com/digoal@126/blog/static/163877040201231781923116/>
  - 为什么要reindex non-b-tree 索引
    - 如果发现non-b-tree索引膨胀比较厉害, 并且性能下降严重的时候需要reindex他们.
- 监控
  - [http://bucardo.org/check\\_postgres/check\\_postgres.pl.html](http://bucardo.org/check_postgres/check_postgres.pl.html)
  - Use PostgreSQL collect and analyze Operation System statistics
  - <http://blog.163.com/digoal@126/blog/static/163877040201211354145701/>

# Backup and Restore

## ■ 备份数据或SQL

- 数据或SQL的备份, 支持不同版本的备份和恢复, 如果要将低版本的数据备份后还原到高版本的数据库中, 一般建议使用高版本的pg\_dump备份, 高版本的pg\_restore还原.
- pg\_dump
  - 输出支持两种格式, 一种是纯文本格式, 另一种是PostgreSQL bin格式.
  - pg\_dump [connection-option...] [option...] [dbname]
- pg\_dumpall
  - 输出仅支持纯文本格式.
  - pg\_dumpall [connection-option...] [option...]
- COPY
  - 类似纯文本格式的备份, 但是可以支持定制化备份列和行的信息.

# Backup and Restore

- 备份数据或SQL
  - pg\_dump

## General options:

|                                          |                                                         |
|------------------------------------------|---------------------------------------------------------|
| <code>-f, --file=FILENAME</code>         | output file or directory name                           |
| <code>-F, --format=c d t p</code>        | output file format (custom, directory, tar, plain text) |
| <code>-v, --verbose</code>               | verbose mode                                            |
| <code>-Z, --compress=0-9</code>          | compression level for compressed formats                |
| <code>--lock-wait-timeout=TIMEOUT</code> | fail after waiting TIMEOUT for a table lock             |
| <code>--help</code>                      | show this help, then exit                               |
| <code>--version</code>                   | output version information, then exit                   |

## Connection options:

|                                  |                                                     |
|----------------------------------|-----------------------------------------------------|
| <code>-h, --host=HOSTNAME</code> | database server host or socket directory            |
| <code>-p, --port=PORT</code>     | database server port number                         |
| <code>-U, --username=NAME</code> | connect as specified database user                  |
| <code>-w, --no-password</code>   | never prompt for password                           |
| <code>-W, --password</code>      | force password prompt (should happen automatically) |
| <code>--role=ROLENAME</code>     | do SET ROLE before dump                             |

If no database name is supplied, then the PGDATABASE environment variable value is used.

# Backup and Restore

## ■ 备份数据或SQL

## ■ pg\_dump

| Options controlling the output content: |                                                                                         |
|-----------------------------------------|-----------------------------------------------------------------------------------------|
| -a, --data-only                         | dump only the data, not the schema                                                      |
| -b, --blobs                             | include large objects in dump                                                           |
| -c, --clean                             | clean (drop) database objects before recreating                                         |
| -C, --create                            | include commands to create database in dump                                             |
| -E, --encoding=ENCODING                 | dump the data in encoding ENCODING                                                      |
| -n, --schema=SCHEMA                     | dump the named schema(s) only                                                           |
| -N, --exclude-schema=SCHEMA             | do NOT dump the named schema(s)                                                         |
| -o, --oids                              | include OIDs in dump                                                                    |
| -O, --no-owner                          | skip restoration of object ownership in plain-text format                               |
| -s, --schema-only                       | dump only the schema, no data                                                           |
| -S, --superuser=NAME                    | superuser user name to use in plain-text format                                         |
| -t, --table=TABLE                       | dump the named table(s) only                                                            |
| -T, --exclude-table=TABLE               | do NOT dump the named table(s)                                                          |
| -x, --no-privileges                     | do not dump privileges (grant/revoke)                                                   |
| --binary-upgrade                        | for use by upgrade utilities only                                                       |
| --column-inserts                        | dump data as INSERT commands with column names                                          |
| --disable-dollar-quoting                | disable dollar quoting, use SQL standard quoting                                        |
| --disable-triggers                      | disable triggers during data-only restore                                               |
| --inserts                               | dump data as INSERT commands, rather than COPY                                          |
| --no-security-labels                    | do not dump security label assignments                                                  |
| --no-tablespaces                        | do not dump tablespace assignments                                                      |
| --no-unlogged-table-data                | do not dump unlogged table data                                                         |
| --quote-all-identifiers                 | quote all identifiers, even if not key words                                            |
| --serializable-deferrable               | wait until the dump can run without anomalies                                           |
| --use-set-session-authorization         | use SET SESSION AUTHORIZATION commands instead of ALTER OWNER commands to set ownership |

# Backup and Restore

- 备份数据或SQL
  - pg\_dumpall

## General options:

|                                    |                                             |
|------------------------------------|---------------------------------------------|
| <b>-f, --file=FILENAME</b>         | output file name                            |
| <b>--lock-wait-timeout=TIMEOUT</b> | fail after waiting TIMEOUT for a table lock |
| <b>--help</b>                      | show this help, then exit                   |
| <b>--version</b>                   | output version information, then exit       |

## Connection options:

|                              |                                                     |
|------------------------------|-----------------------------------------------------|
| <b>-h, --host=HOSTNAME</b>   | database server host or socket directory            |
| <b>-l, --database=DBNAME</b> | alternative default database                        |
| <b>-p, --port=PORT</b>       | database server port number                         |
| <b>-U, --username=NAME</b>   | connect as specified database user                  |
| <b>-w, --no-password</b>     | never prompt for password                           |
| <b>-W, --password</b>        | force password prompt (should happen automatically) |
| <b>--role=ROLENAME</b>       | do SET ROLE before dump                             |

If **-f**/**--file** is not used, then the SQL script will be written to the standard output.

# Backup and Restore

## ■ 备份数据或SQL

- ## ■ pg\_dumpall

**Options controlling the output content:**

|                                        |                                                                                            |
|----------------------------------------|--------------------------------------------------------------------------------------------|
| <b>-a, --data-only</b>                 | dump only the data, not the schema                                                         |
| <b>-c, --clean</b>                     | clean (drop) databases before recreating                                                   |
| <b>-g, --globals-only</b>              | dump only global objects, no databases                                                     |
| <b>-o, --oids</b>                      | include OIDs in dump                                                                       |
| <b>-O, --no-owner</b>                  | skip restoration of object ownership                                                       |
| <b>-r, --roles-only</b>                | dump only roles, no databases or tablespaces                                               |
| <b>-s, --schema-only</b>               | dump only the schema, no data                                                              |
| <b>-S, --superuser=NAME</b>            | superuser user name to use in the dump                                                     |
| <b>-t, --tablespaces-only</b>          | dump only tablespaces, no databases or roles                                               |
| <b>-x, --no-privileges</b>             | do not dump privileges (grant/revoke)                                                      |
| <b>--binary-upgrade</b>                | for use by upgrade utilities only                                                          |
| <b>--column-inserts</b>                | dump data as INSERT commands with column names                                             |
| <b>--disable-dollar-quoting</b>        | disable dollar quoting, use SQL standard quoting                                           |
| <b>--disable-triggers</b>              | disable triggers during data-only restore                                                  |
| <b>--inserts</b>                       | dump data as INSERT commands, rather than COPY                                             |
| <b>--no-security-labels</b>            | do not dump security label assignments                                                     |
| <b>--no-tablespaces</b>                | do not dump tablespace assignments                                                         |
| <b>--no-unlogged-table-data</b>        | do not dump unlogged table data                                                            |
| <b>--quote-all-identifiers</b>         | quote all identifiers, even if not key words                                               |
| <b>--use-set-session-authorization</b> | use SET SESSION AUTHORIZATION commands instead of<br>ALTER OWNER commands to set ownership |

# Backup and Restore

## ■ 备份数据或SQL

### ■ COPY

```
COPY table_name [ <column [, ...]> ]  
    FROM <'filename' : STDIN>  
    [ [ WITH ] <option [, ...]> ]  
  
COPY <table_name [ <column [, ...]> ] + <query>>  
    TO <'filename' : STDOUT>  
    [ [ WITH ] <option [, ...]> ]  
  
where option can be one of:  
  
    FORMAT format_name  
    OIDS [ boolean ]  
    DELIMITER 'delimiter_character'  
    NULL 'null_string'  
    HEADER [ boolean ]  
    QUOTE 'quote_character'  
    ESCAPE 'escape_character'  
    FORCE_QUOTE <<column [, ...]>> | * >  
    FORCE_NOT_NULL <column [, ...]> |  
    ENCODING 'encoding_name'
```

The following example copies a table to the client using the vertical bar <|> as the field delimiter:

```
COPY country TO STDOUT <DELIMITER '|'|>;
```

To copy data from a file into the country table:

```
COPY country FROM '/usr1/proj/bray/sql/country_data';
```

To copy into a file just the countries whose names start with 'A':

```
COPY <SELECT * FROM country WHERE country_name LIKE 'A%'> TO '/usr1/proj/bray/sql/a_list_countries.copy';
```

# Backup and Restore

- 备份数据文件, 增量备份, 可用于做基于时间点的恢复, 基于xid的恢复, 基于定制还原点的恢复
  - 有效备份数据文件的前提
    - full\_page\_writes = on
    - fsync = on
    - wal\_level = archive 或 hot\_standby
    - archive\_mode = on
    - archive\_command = 'cp %p /backup/%f'
  - pg\_start\_backup -- 排他. 同一时间只允许一个pg\_start\_backup运行.
  - 备份\$PGDATA, pg\_tblspc中软链接对应的表空间目录
  - pg\_xlog 目录不需要备份
  - pg\_stop\_backup -- 停止备份.
  - CHECKPOINT;
  - pg\_switch\_xlog();
  - 备份在备份过程中产生的wal\_archive
- pg\_basebackup -- 一般被用于创建standby.

# Backup and Restore

- 执行pg\_start\_backup后, \$PGDATA目录生成一个backup\_label文件
- 文件内容类似
  - START WAL LOCATION: o/B0000020 (file 0000000100000000000000002C)
  - CHECKPOINT LOCATION: o/B0000058
  - BACKUP METHOD: pg\_start\_backup
  - START TIME: 2012-05-03 12:07:32 CST
  - LABEL: test
- 执行pg\_stop\_backup后会在pg\_xlog中生成一个备份完成标记的文件, 文件及内容如下

```
64M Apr 26 11:16 0000000200000001C0000002E
274 Apr 26 11:16 0000000200000001C0000002E.00000020.backup
```

```
postgres@dh-172-16-3-150-> less 0000000200000001C0000002E.00000020.backup
START WAL LOCATION: 1C/B8000020 <file 0000000200000001C0000002E>
STOP WAL LOCATION: 1C/B80001F8 <file 0000000200000001C0000002E>
CHECKPOINT LOCATION: 1C/B8000058
BACKUP METHOD: pg_start_backup
START TIME: 2012-04-26 11:03:27 CST
LABEL: test
STOP TIME: 2012-04-26 11:16:08 CST
```

# Backup and Restore

## ■ pg\_basebackup

```
pg_basebackup takes a base backup of a running PostgreSQL server.

Usage:
  pg_basebackup [OPTION]...

Options controlling the output:
  -D, --pgdata= DIRECTORY      receive base backup into directory
  -F, --format= {plain, tar}    output format {plain, tar}
  -x, --xlog                   include required WAL files in backup
  -z, --gzip                    compress tar output
  -Z, --compress=0-9           compress tar output with given compression level

General options:
  -c, --checkpoint= {fast, spread}   set fast or spread checkpointing
  -l, --label=LABEL              set backup label
  -P, --progress                 show progress information
  -v, --verbose                  output verbose messages
  --help                         show this help, then exit
  --version                      output version information, then exit

Connection options:
  -h, --host=HOSTNAME           database server host or socket directory
  -p, --port=PORT                database server port number
  -U, --username=NAME            connect as specified database user
  -w, --no-password              never prompt for password
  -W, --password                 force password prompt (should happen automatically)
```

# Backup and Restore

## ■ xlog.c

```
/*
 * do_pg_start_backup is the workhorse of the user-visible pg_start_backup()
 * function. It creates the necessary starting checkpoint and constructs the
 * backup label file.
 *
 * There are two kind of backups: exclusive and non-exclusive. An exclusive
 * backup is started with pg_start_backup(), and there can be only one active
 * at a time. The backup label file of an exclusive backup is written to
 * $PGDATA/backup_label, and it is removed by pg_stop_backup().
 *
 * A non-exclusive backup is used for the streaming base backups (see
 * src/backendreplication/basebackup.c). The difference to exclusive backups
 * is that the backup label file is not written to disk. Instead, its would-be
 * contents are returned in *labelfile, and the caller is responsible for
 * including it in the backup archive as 'backup_label'. There can be many
 * non-exclusive backups active at the same time, and they don't conflict
 * with an exclusive backup either.
 *
 * Every successfully started non-exclusive backup must be stopped by calling
 * do_pg_stop_backup() or do_pg_abort_backup().
 */
```

# Backup and Restore

- 还原
- 纯文本备份解读

- postgres@db-172-16-3-150-> pg\_dump -F p -f ./test.dmp.20120503 -C -E UTF8 -b -h 127.0.0.1 -U postgres test
- postgres@db-172-16-3-150-> cat test.dmp.20120503

```
--  
-- PostgreSQL database dump  
  
SET statement_timeout = 0;  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;  
SET check_function_bodies = false;  
SET client_min_messages = warning;  
--  
-- Name: test; Type: DATABASE; Schema: -; Owner: postgres  
--  
  
CREATE DATABASE test WITH TEMPLATE = template0 ENCODING = 'SQL_ASCII' LC_COLLATE = 'C' LC_CTYPE = 'C';  
  
ALTER DATABASE test OWNER TO postgres;  
  
\connect test  
  
SET statement_timeout = 0;  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;  
SET check_function_bodies = false;  
SET client_min_messages = warning;
```

# Backup and Restore

## ■ 纯文本备份解读

```
--  
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:  
  
CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;  
  
--  
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:  
  
COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';  
SET search_path = public, pg_catalog;  
  
--  
-- Name: f_void(); Type: FUNCTION; Schema: public; Owner: postgres  
  
CREATE FUNCTION f_void() RETURNS void  
    LANGUAGE plpgsql  
    AS $$  
declare  
begin  
return;  
end;  
$$;
```

# Backup and Restore

## ■ 纯文本备份解读

```
ALTER FUNCTION public.f_void() OWNER TO postgres;

-- 
-- Name: seq; Type: SEQUENCE; Schema: public; Owner: postgres
-- 

CREATE SEQUENCE seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.seq OWNER TO postgres;
-- 
-- Name: seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
-- 

SELECT pg_catalog.setval('seq', 1, false);

SET default_tablespace = '';
SET default_with_oids = false;

-- 
-- Name: userinfo; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
-- 

CREATE TABLE userinfo (
    id integer NOT NULL,
    info text
);
;
```

# Backup and Restore

## ■ 纯文本备份解读

```
ALTER TABLE public.userinfo OWNER TO postgres;

-- 
-- Name: v_test; Type: VIEW; Schema: public; Owner: postgres
-- 

CREATE VIEW v_test AS
    SELECT userinfo.id, userinfo.info FROM userinfo;

ALTER TABLE public.v_test OWNER TO postgres;

-- 
-- Data for Name: userinfo; Type: TABLE DATA; Schema: public; Owner: postgres
-- 

COPY userinfo (id, info) FROM stdin;
COPY userinfo (id, info) FROM stdin;
1      digoal
2      digoal
100    digoal
\.

-- 
-- Name: userinfo_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
-- 

ALTER TABLE ONLY userinfo
    ADD CONSTRAINT userinfo_pkey PRIMARY KEY (id);

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;
```

# Backup and Restore

- 使用psql -f还原纯文本格式的备份
  - postgres=# drop database test;
  - DROP DATABASE
  - postgres@db-172-16-3-150-> psql -f ./test.dmp.20120503
- 使用pg\_restore还原BIN格式的备份
  - postgres@db-172-16-3-150-> pg\_dump -F c -f ./test.dmp.20120503.c -C -E UTF8 -b -h 127.0.0.1 -U postgres test
  - postgres=# drop database test;
  - DROP DATABASE
  - postgres@db-172-16-3-150-> pg\_restore -v -d postgres -C -F c -h 127.0.0.1 -U postgres ./test.dmp.20120503.c
  - pg\_restore: connecting to database for restore
  - pg\_restore: creating DATABASE test
  - pg\_restore: connecting to new database "test"
  - pg\_restore: connecting to database "test" as user "postgres"
  - pg\_restore: creating SCHEMA public

# Backup and Restore

## ■ 使用pg\_restore还原BIN格式的备份

- pg\_restore: creating COMMENT SCHEMA public
- pg\_restore: creating EXTENSION plpgsql
- pg\_restore: creating COMMENT EXTENSION plpgsql
- pg\_restore: creating FUNCTION f\_void()
- pg\_restore: creating SEQUENCE seq
- pg\_restore: executing SEQUENCE SET seq
- pg\_restore: creating TABLE userinfo
- pg\_restore: creating VIEW v\_test
- pg\_restore: restoring data for table "userinfo"
- pg\_restore: creating CONSTRAINT userinfo\_pkey
- pg\_restore: setting owner and privileges for DATABASE test
- pg\_restore: setting owner and privileges for SCHEMA public
- pg\_restore: setting owner and privileges for COMMENT SCHEMA public
- pg\_restore: setting owner and privileges for ACL public
- pg\_restore: setting owner and privileges for EXTENSION plpgsql

# Backup and Restore

- 使用pg\_restore还原BIN格式的备份
  - pg\_restore: setting owner and privileges for COMMENT EXTENSION plpgsql
  - pg\_restore: setting owner and privileges for FUNCTION f\_void()
  - pg\_restore: setting owner and privileges for SEQUENCE seq
  - pg\_restore: setting owner and privileges for SEQUENCE SET seq
  - pg\_restore: setting owner and privileges for TABLE userinfo
  - pg\_restore: setting owner and privileges for VIEW v\_test
  - pg\_restore: setting owner and privileges for TABLE DATA userinfo
  - pg\_restore: setting owner and privileges for CONSTRAINT userinfo\_pkey
- 通过编辑TOC文件定制还原
- <http://blog.163.com/digoal@126/blog/static/16387704020123129649342/>

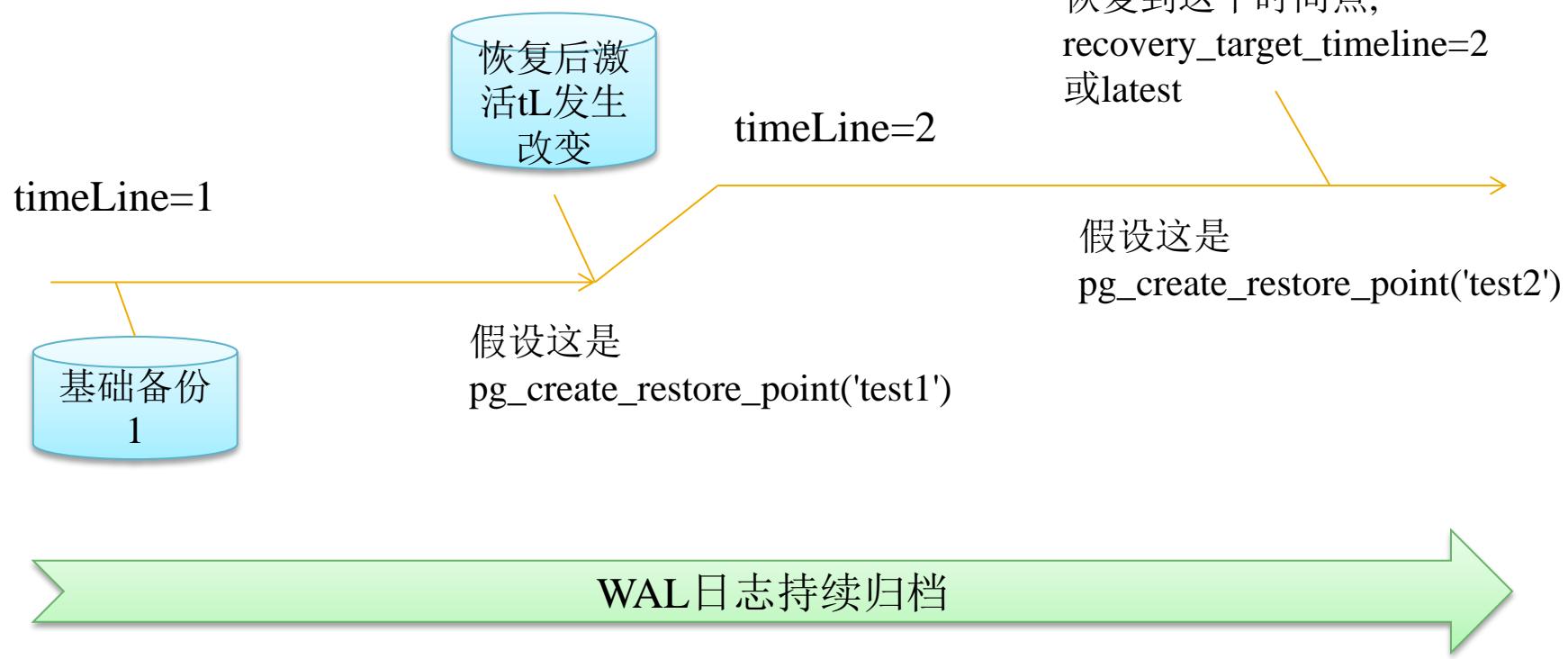
# Backup and Restore

- PITR
- 解读recovery.conf文件
  - 示例文件在 \$PGHOME/share/recovery.conf.sample
  - restore\_command -- 还原归档文件的命令
  - recovery\_target\_name | recovery\_target\_time | recovery\_target\_xid
    - 恢复到什么目标,其中recovery\_target\_name是使用pg\_create\_restore\_point()生成的.
  - recovery\_target\_inclusive ( 目标为时间和xid时可配置是否恢复到(包含|接近但不包含)指定的时间点或XID)
  - recovery\_target\_timeline (恢复到哪个时间线,或latest表示直到最大的时间线)
  - pause\_at\_recovery\_target (恢复到指定点后暂停恢复,一般可用于连到数据库去检测是否已经到达了想要恢复的时间点,没到达的话可以关闭数据库调整恢复目标点,继续恢复,直到到了想要的点后,使用pg\_xlog\_replay\_resume()来停止恢复并激活数据库) 打开hot\_standby才能使用.
- 基于单个表空间 / 数据库的还原和恢复
- <http://blog.163.com/digoal@126/blog/static/16387704020123261422581/>



# Backup and Restore

- 例子
- <http://blog.163.com/digoal@126/blog/static/1638770402012431063591/>



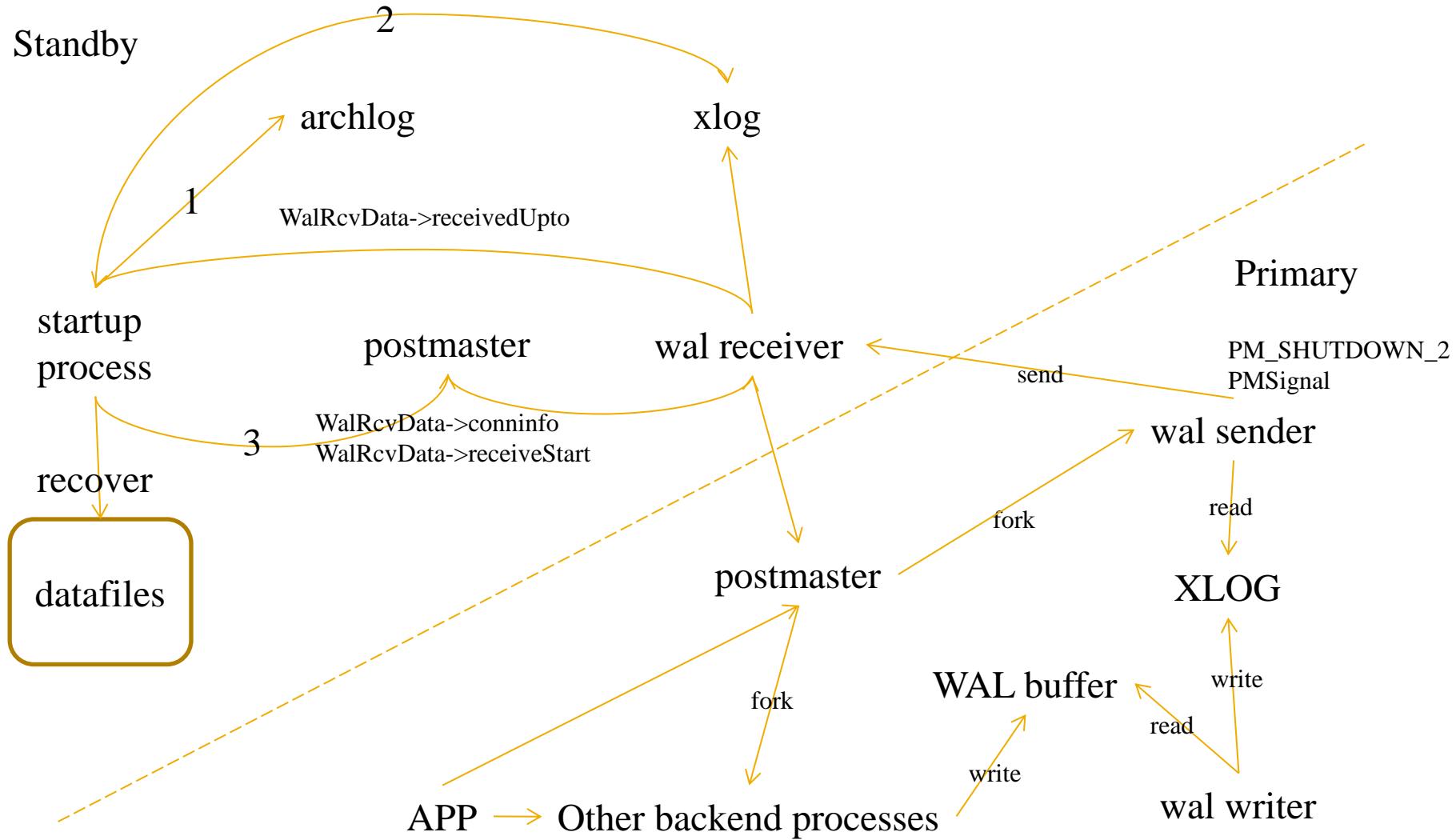
# HA and Replication

- 传统HA, 共享存储
  - 缺点, 存储或者链接存储的线路甚至HBA卡都会成为故障点, 所以HA并不HA.
  - 通常需要靠存储复制或文件系统异机镜像来应对存储层面的故障.
- PostgreSQL流复制带来了新的解决方案
  - LVS+PG流复制可以组成读负载均衡的场景
  - PG流复制还可以应用于异地容灾的场景
  - PG流复制可以作为HA的一部分, 通过VIP漂移和激活同步 standby可以做到极高的数据可靠性和高可用.
- PgCloud, 一种不依赖虚拟化和集中式存储的思路
- <http://blog.163.com/digoal@126/blog/static/1638770402011111422518103/>



# HA and Replication

## ■ 异步流复制原理



## Parameter Tuning :

### Primary

max\_wal\_senders

wal\_sender\_delay ( The sleep is interrupted by transaction commit )

wal\_keep\_segments

vacuum\_defer\_cleanup\_age ( the number of transactions by which VACUUM and HOT updates will defer cleanup of dead row versions. )

### Standby

hot\_standby

# wal apply & SQL on standby conflict reference parameter

max\_standby\_archive\_delay

( the maximum total time allowed to apply any one WAL segment's data. )

max\_standby\_streaming\_delay

( the maximum total time allowed to apply WAL data once it has been received from the primary server )

wal\_receiver\_status\_interval

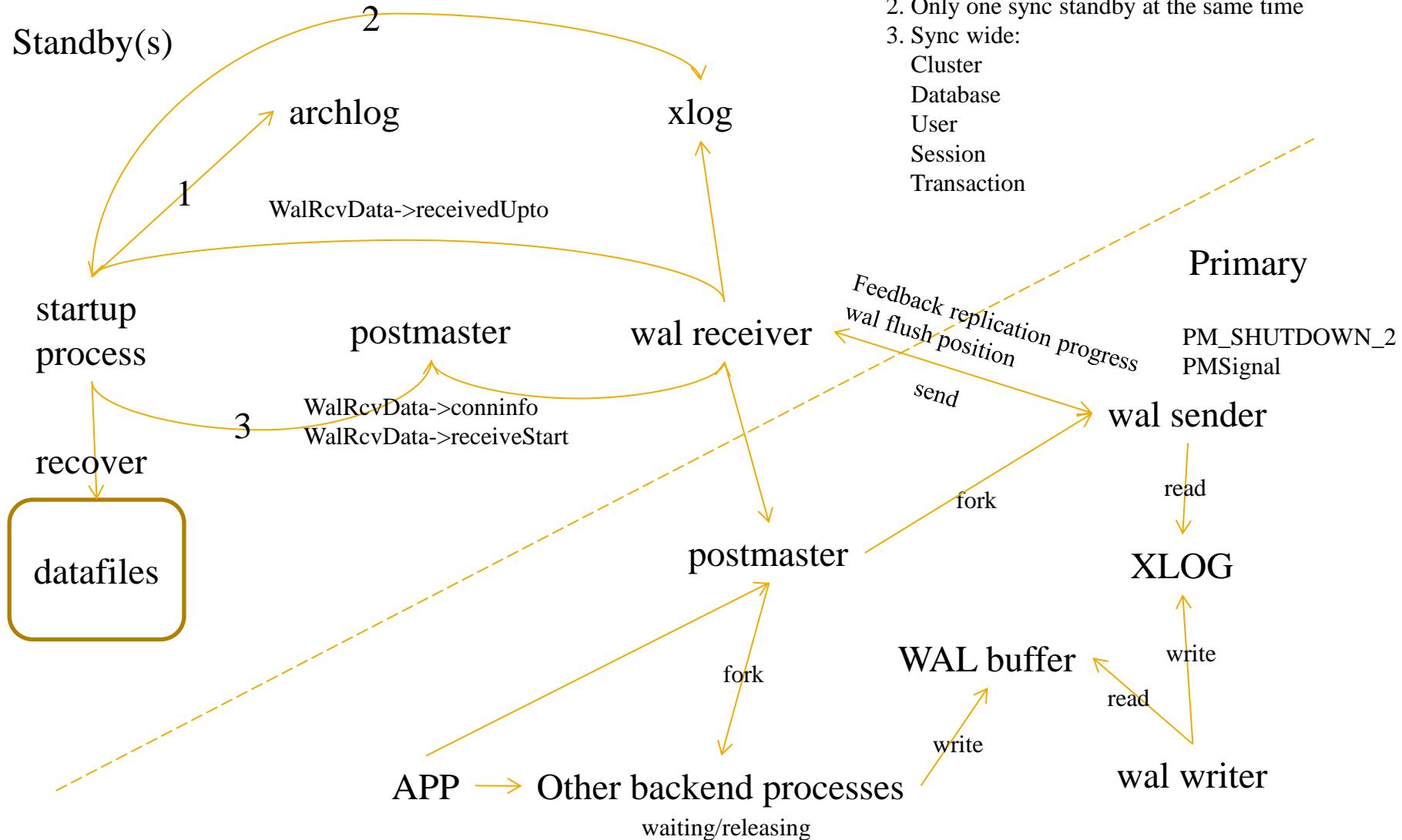
( minimum frequency, The standby will report the last transaction log position it has written, the last position it has flushed to disk, and the last position it has applied.)

hot\_standby\_feedback

(send feedback to the primary about queries currently executing on the standby. )

# HA and Replication

## 同步流复制原理



# HA and Replication

Parameter Tuning :

Primary

max\_wal\_senders  
wal\_sender\_delay  
wal\_keep\_segments  
vacuum\_defer\_cleanup\_age  
**synchronous\_replication**  
**synchronous\_standby\_names**  
( primary\_conninfo in standby's primary\_conninfo )

Standby

hot\_standby  
max\_standby\_archive\_delay  
max\_standby\_streaming\_delay  
wal\_receiver\_status\_interval  
hot\_standby\_feedback

# HA and Replication

- 流复制hot\_standby演示
  - 规划主机, 网络, 存储, 同步主备机器的时间
  - 生成主库
  - 配置主库postgresql.conf, pg\_hba.conf
  - 新建replication角色
  - 配置hot\_standby .pgpass, 数据目录
  - 使用pg\_basebackup创建备库基础备份
  - 配置备库recovery.conf, postgresql.conf
  - 启动hot\_standby
  - 测试, 新建用户, 表空间, 数据库, schema, 数据表.
  - 使用pgbench进行压力测试
  - 角色切换测试
- PostgreSQL 9.2 级联流复制
- <http://blog.163.com/digoal@126/blog/static/1638770402012012361519/>



# HA and Replication

## ■ 数据库复制技术

- 基于触发器的复制
  - slony-I, bucardo, londiste
- 基于SQL分发的复制
  - pgpool, continue
- 基于PAGE改变日志的复制
  - 流复制, 归档复制

# Scale-Out



- PG-XC
  - <http://blog.163.com/digoal@126/blog/static/16387704020121952051174/>
- pl/proxy
  - <http://blog.163.com/digoal@126/blog/static/163877040201192535630895/>
  - <http://www.tudou.com/programs/view/TcluEJ4ZfPA/>
- pgpool-II
  - <http://pgfoundry.org/projects/pgpool/>

# Monitoring Database Activity

- long sql
- lock
- unused index
- dead tuples ratio
- server load
- server rtps
- server wtps
- server iowait
- server swap page in/out
- server process/s
- error | warning log
- pg\_stat\_statements
- CPU used by one SQL
  
- Use PostgreSQL collect and analyze Operation System statistics
- <http://blog.163.com/digoal@126/blog/static/163877040201211354145701/>



# Procedure Language

- 支持多种语言, perl, python, tcl , plpgsql等.
  - plpgsql函数在PostgreSQL中作为一个事务处理, 当触发了exception时, exception中的内容作为另一个事务.
- 
- Debug plpgsql Function
  - <http://blog.163.com/digoal@126/blog/static/163877040201222011550296/> 
  - PostgreSQL 2-PC Transaction
  - <http://blog.163.com/digoal@126/blog/static/16387704020111141103578/>

# Additional Supplied Modules

## ■ 去哪里找模块

- <http://pgxn.org/>
- <http://pgfoundry.org/>
- <http://www.postgresql.org/docs/9.1/static/contrib.html>



## ■ 比较常用的模块

- `auto_explain` -- 自动记录超过设定运行时间的SQL执行时的执行计划
- <http://blog.163.com/digoal@126/blog/static/16387704020115825612145/>
- `dblink` -- 数据库链接, 可用于链接远程数据库
- <http://www.postgresql.org/docs/9.1/static/dblink.html>
- `file_fdw` -- 基于文件创建外部表
- <http://blog.163.com/digoal@126/blog/static/163877040201141641148311/>
- `pageinspect` -- 用于查看表或索引的PAGE以及ITEM的信息
- <http://blog.163.com/digoal@126/blog/static/16387704020114273265960/>
- `pg_archivecleanup` -- 清除归档的模块
- <http://blog.163.com/digoal@126/blog/static/16387704020110445753526/>

# Additional Supplied Modules

## ■ 比较常用的模块

- pgbench -- 压力测试模块
- <http://blog.163.com/digoal@126/blog/static/163877040201151534631313/>
- pg\_buffercache -- 查看buffer信息的模块
- <http://blog.163.com/digoal@126/blog/static/16387704020115149458640/>
- pg\_freespacemap -- 查看freespacemap信息的模块
- <http://www.postgresql.org/docs/9.1/static/pgfreespacemap.html>
- pgrowlocks -- 查看行锁的模块
- <http://blog.163.com/digoal@126/blog/static/16387704020115105557166/>
- pg\_stat\_statements -- 统计数据库执行的SQL语句的次数以及CPU开销的模块
- <http://www.postgresql.org/docs/9.1/static/pgstatstatements.html>
- pgstattuple -- 获得tuple级统计信息的模块
- <http://www.postgresql.org/docs/9.1/static/pgstattuple.html>

# Additional Supplied Modules

## ■ 比较常用的模块

- pg\_test\_fsync -- 调用各种OS同步写接口的测试模块
- <http://blog.163.com/digoal@126/blog/static/163877040201141795025354/>
- pg\_trgm -- 可用于近似度匹配的模块
- <http://blog.163.com/digoal@126/blog/static/163877040201191882553803/>
- pg\_upgrade -- 基于catalog的变更升级模块, 通常比pg\_dump升级速度快很多倍.
- <http://www.postgresql.org/docs/9.1/static/pgupgrade.html>
- Foreign data wrapper -- 建立外部表的模块
- -- 参见day1的fdw章节
- pgfincore -- 模拟持久化缓存的模块
- <http://blog.163.com/digoal@126/blog/static/163877040201062944945126/>
- <http://blog.163.com/digoal@126/blog/static/1638770402011630102117658/>
- <http://blog.163.com/digoal@126/blog/static/16387704020120524144140/>

## ■ 其他常见的支持PostgreSQL的外围软件

- 全文检索Sphinx, 地理信息PostGIS

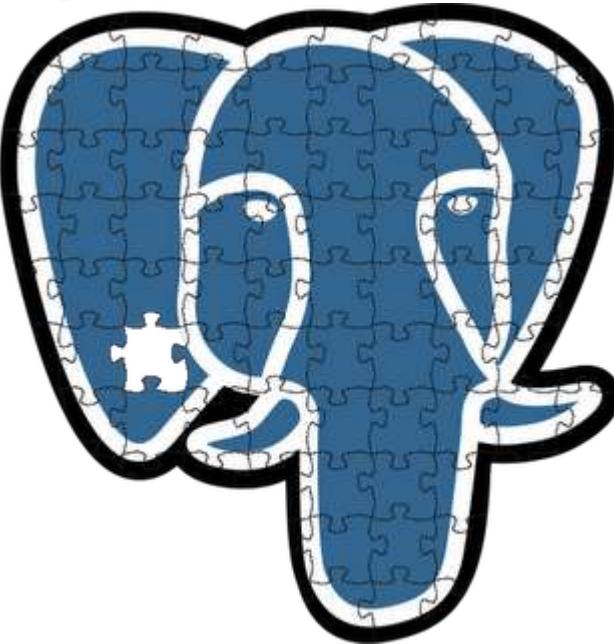
# Tuning case

- PostgreSQL性能优化综合案例
- <http://blog.163.com/digoal@126/blog/static/163877040201221382150858/>
- <http://blog.163.com/digoal@126/blog/static/163877040201221333411196/>



# Thanks

## PostgreSQL 9.1.3 2Day DBA QuickGuide



不积跬步，  
无以至千里。

荀子

- 关于本PPT有问题请发邮件至 [digoal@126.com](mailto:digoal@126.com)
  - 保持联系, 个人QQ: 276732431
  - 群: 3336901
  - 【参考】
  - 《PostgreSQL 9.1.3 Manual》
- 《PostgreSQL 9 Administration Cookbook》
- 《PostgreSQL 9.0 High Performance》
  - 【更多内容请关注】
- <http://blog.163.com/digoal@126/>

# 中国2012 PostgreSQL用户大会

■ 地点: 北京人民大学

■ 时间: 6月14-17号

■ 内容简介:

- 本次大会将邀请到社区的核心组员MagnusHagander,主要开发人员Simon Rigg、PG-XC的首席架构师铃木幸一(Suzuki Koichi)及其他海外专家为本次大会分享最前沿的PostgreSQL方面的信息，同时还有业界的资深人员作相关演讲。
- PG-XC峰会
- 数据库应用分会场
- 内核开发分会场
- 管理与性能调优分会场
- PostgreSQL ACE颁奖

■ 主题: 开放征集中

