Greenplum 应用开发

目录

1 GP 数据库设计

1.1 逻辑数据模型

1.1.1 支持通用的数据建模方式

1、星型模型
2、雪花模型
3、3NF,BCNF,4NF

1.1.2 Greenplum 支持

1、主键(Primary key)
2、单值约束(UNIQUE)
3、属性取值范围约束(空值，CHECK)

1.1.3 Greenplum 不支持

1、参照完整性约束(Foreign Key)

1.1.4 星型模型概述

1.1.4.1 事实表(Fact)

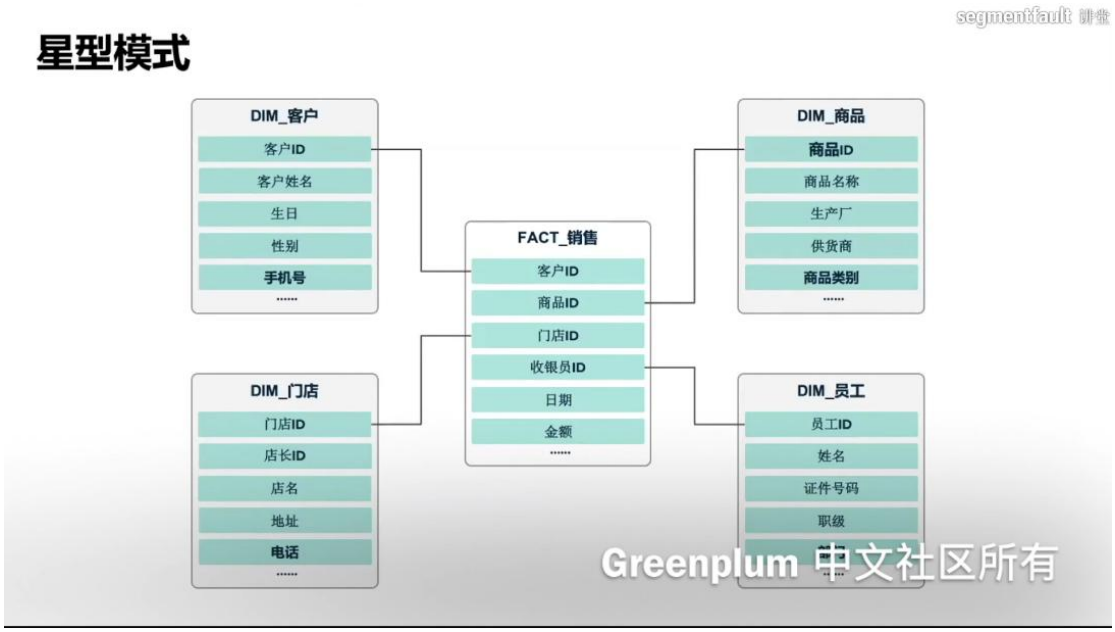
1、存放大量事实业务交易，事件
2、分析作业对事实表进行聚集
3、属性可以非常多，允许冗余
4、数据量不断增长，周期性通过 ETL 作业载入新数据
5、通常很少发生数据变更

1.1.4.2 维表(Dimension)

1、为事实表中一项或多项属性提供额外的描述信息
2、相对于事实表，数据量非常小
3、特定条件下会发生更新
4、一个事实表通常有多个维度表，构型星型模型

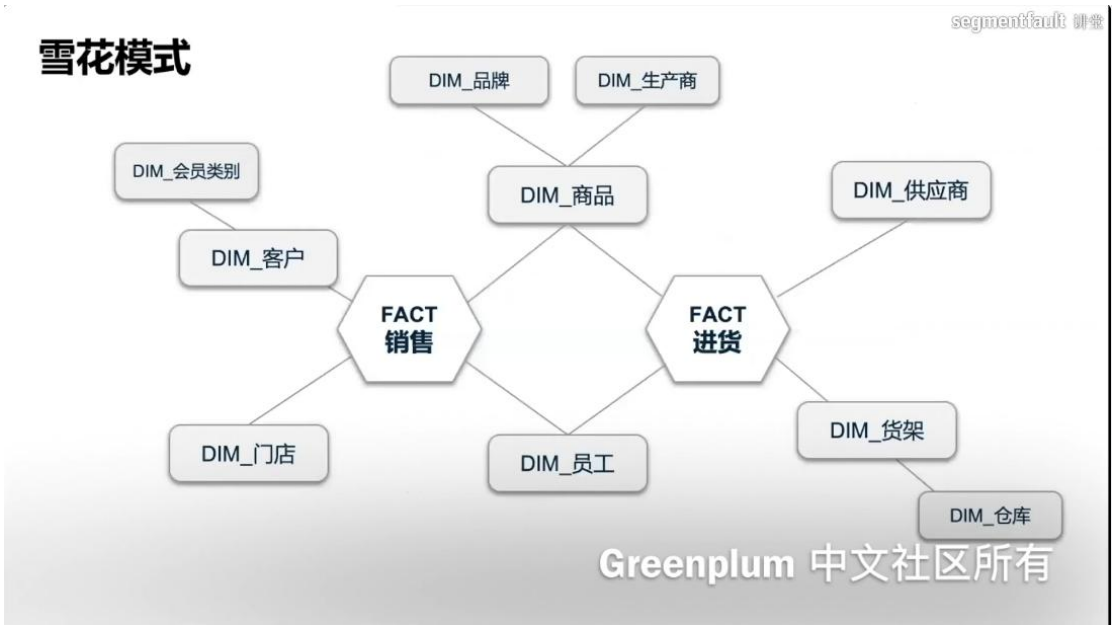
1.1.4.3 星型模型实例

FACT_销售表示事实表，其余的表都属于维表

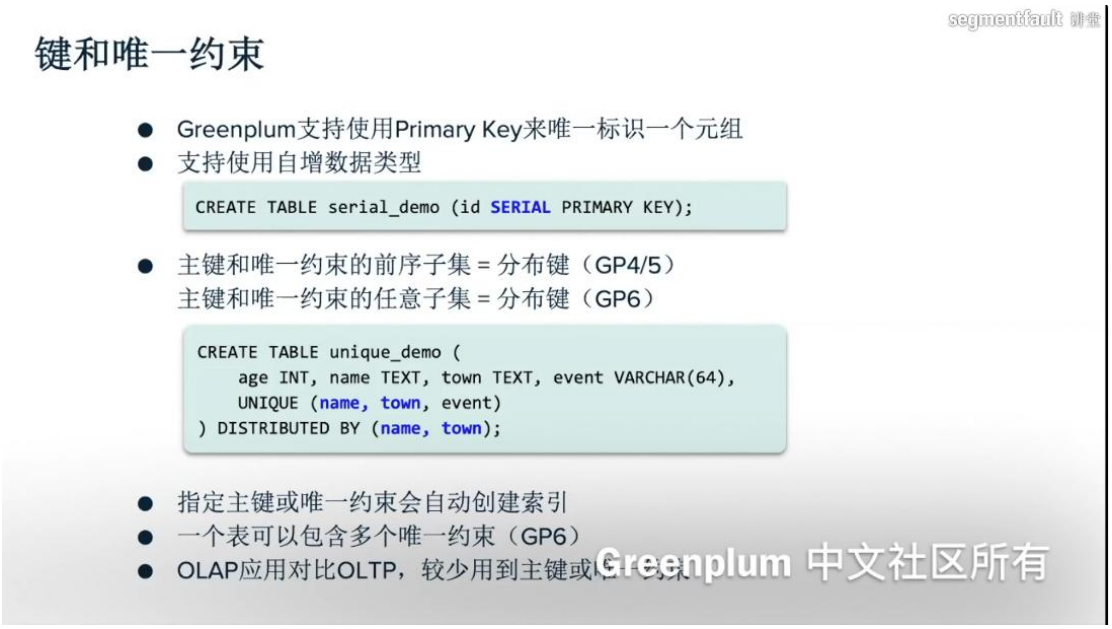


1.1.5 雪花模型实例

多个星型模型表就组成了雪花模型

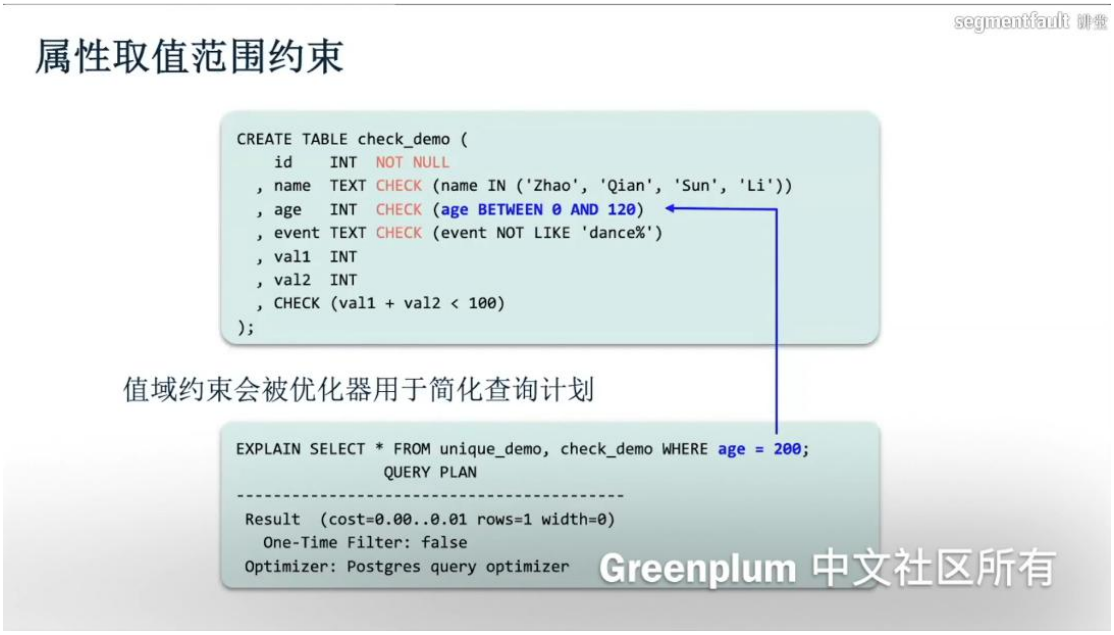


1.1.6 键和唯一约束

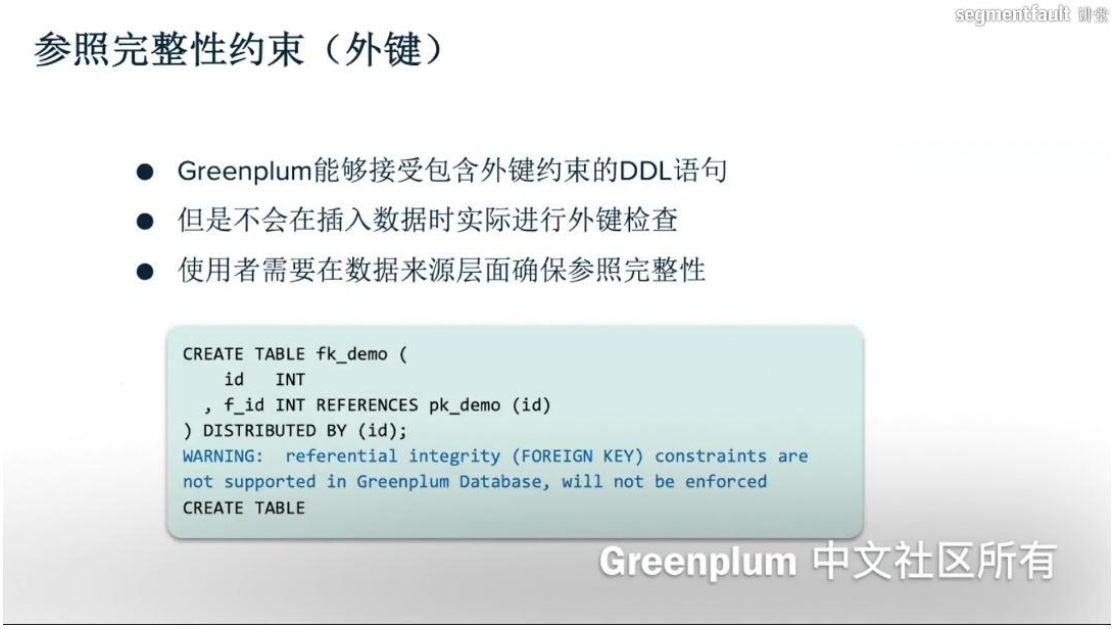


1.1.7 属性取值范围约束

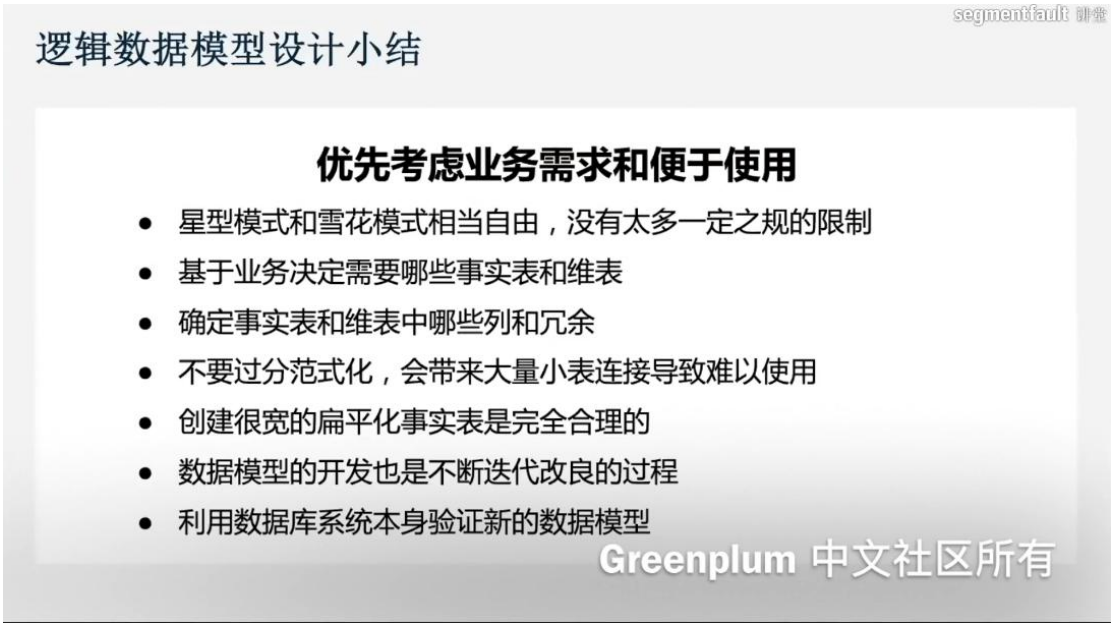
如果字段设置了范围约束，而查询时超过了范围 GP 优化器会认为条件不成立，就没有约束，这样会对查询的性能有极大的影响。



1.1.8 参照完整性约束(外键)



1.1.9 逻辑数据模型设计小结

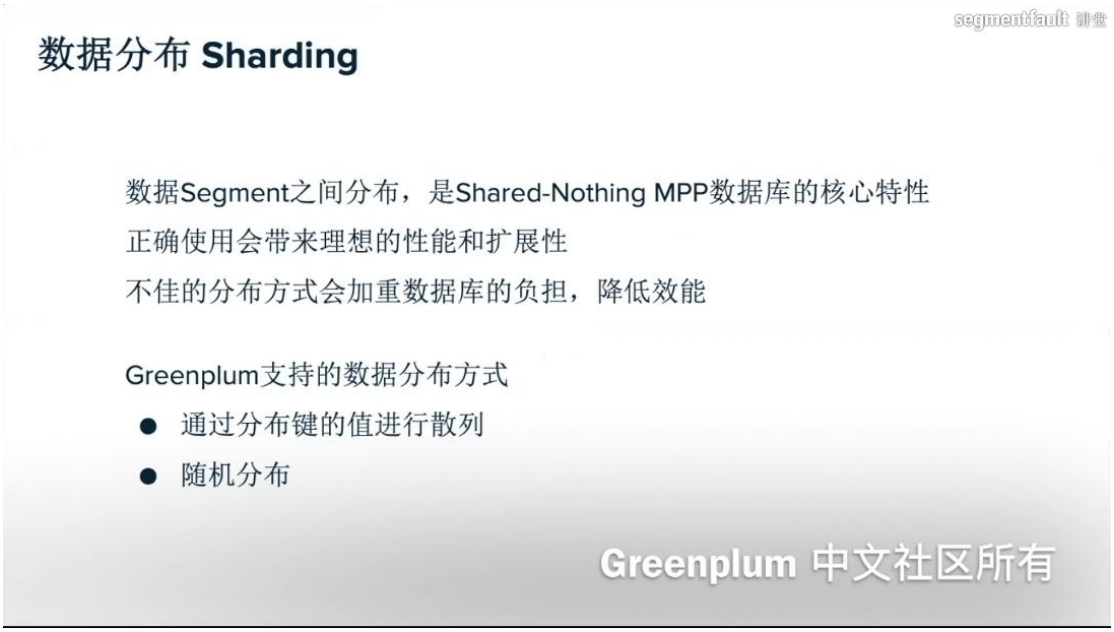


1.2 物理数据模型

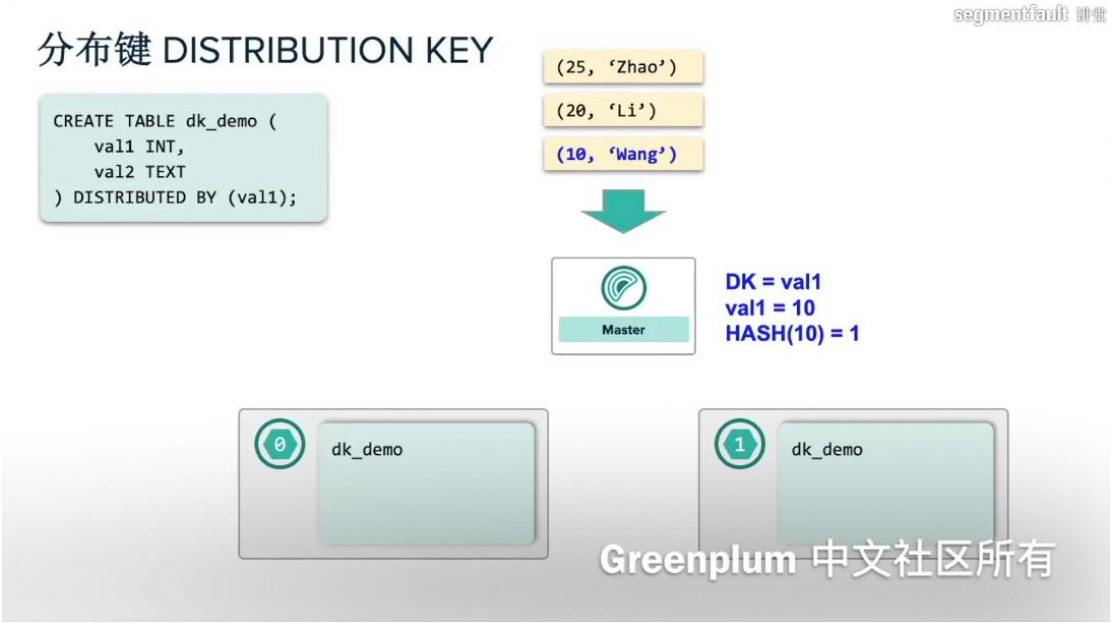
1.2.1 Greenplum 高性能之关键

1、分布键、随机分布
2、分布键的选择
3、储存引擎 Heap,AO,AOCO 和外部表
4、压缩
5、表分区
6、索引
7、多态分布

1.2.2 数据分布 Sharding

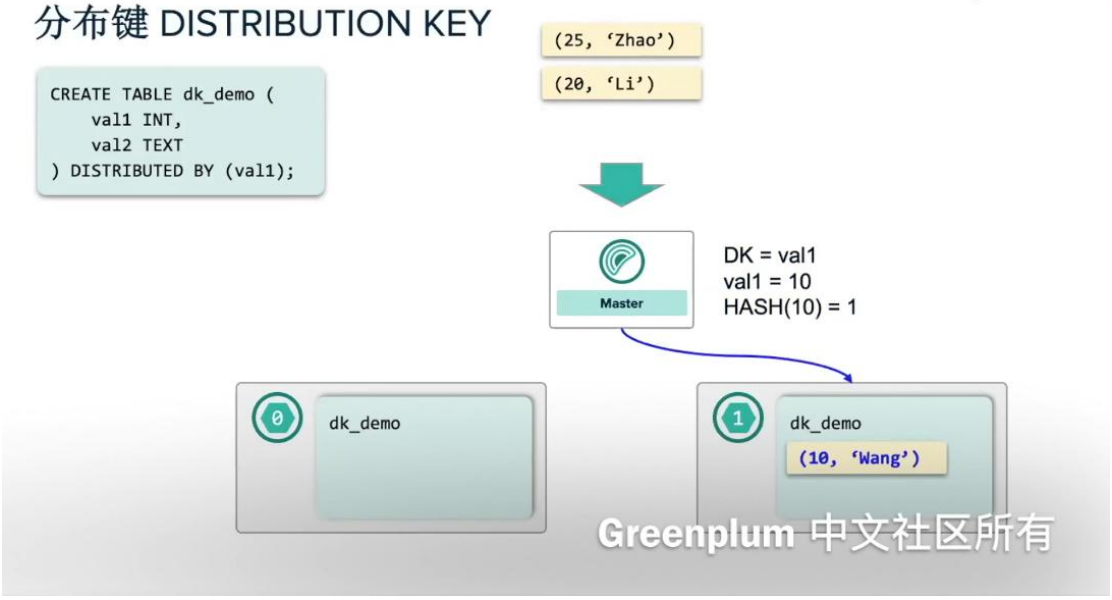


1.2.3 分布键(Distribution key)分布过程



分布键 DISTRIBUTION KEY

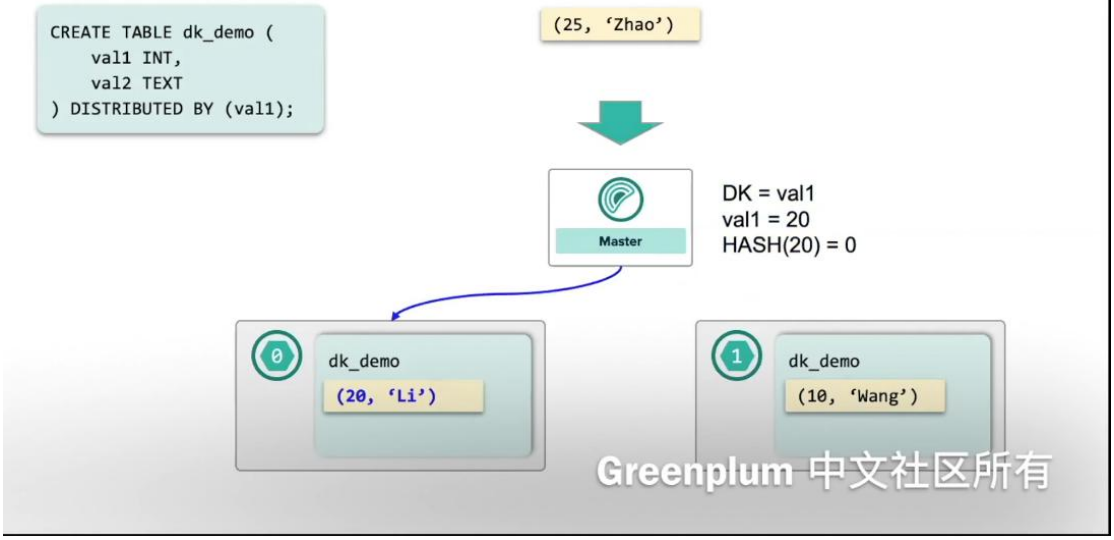
```
CREATE TABLE dk_demo (
    val1 INT,
    val2 TEXT
) DISTRIBUTED BY (val1);
```



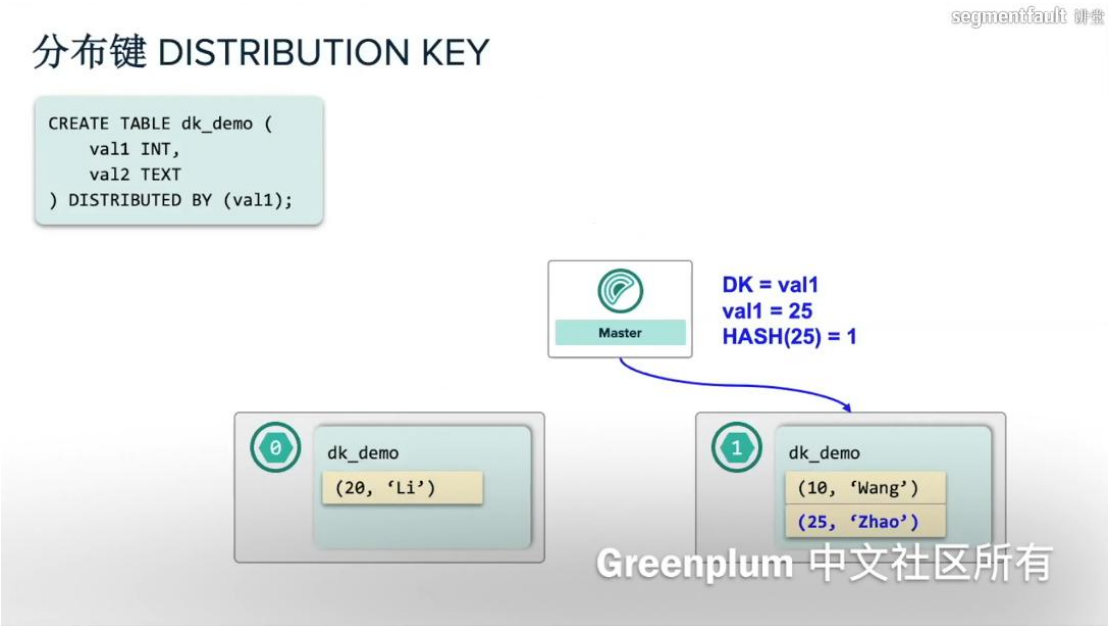
segmentfault 思否

分布键 DISTRIBUTION KEY

```
CREATE TABLE dk_demo (
    val1 INT,
    val2 TEXT
) DISTRIBUTED BY (val1);
```

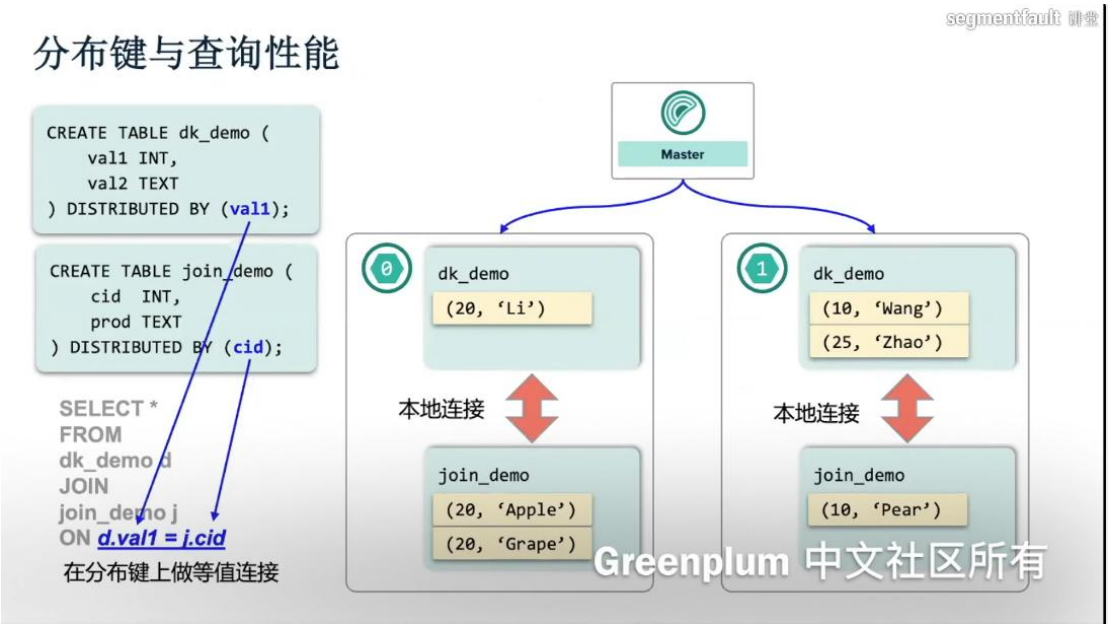


Greenplum 中文社区所有

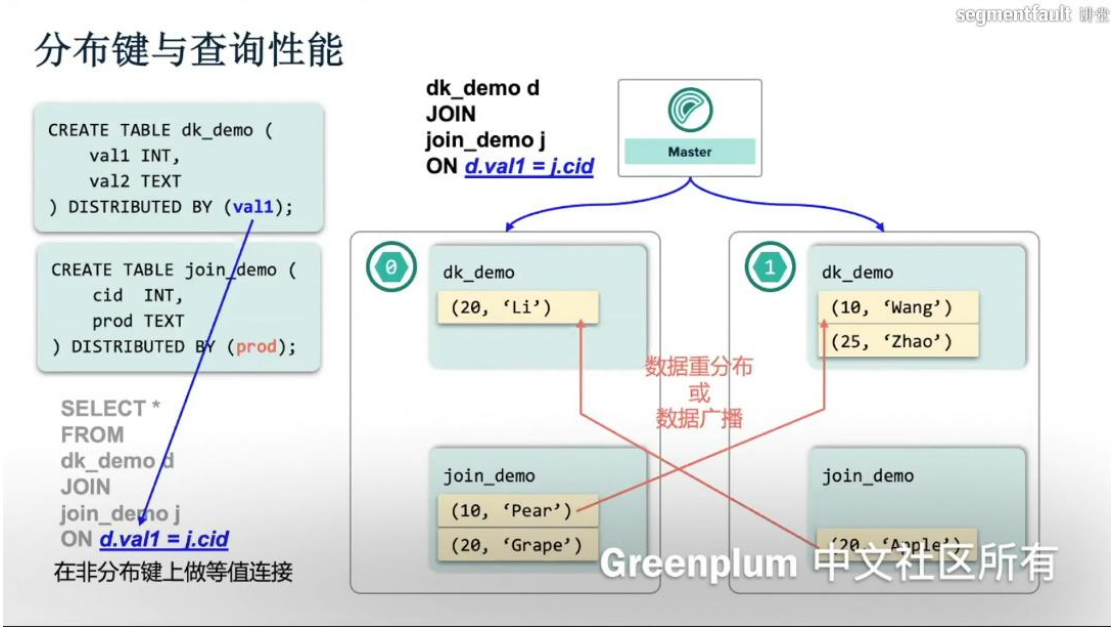


1.2.4 分布键与查询性能

1.2.4.1 最优查询性能

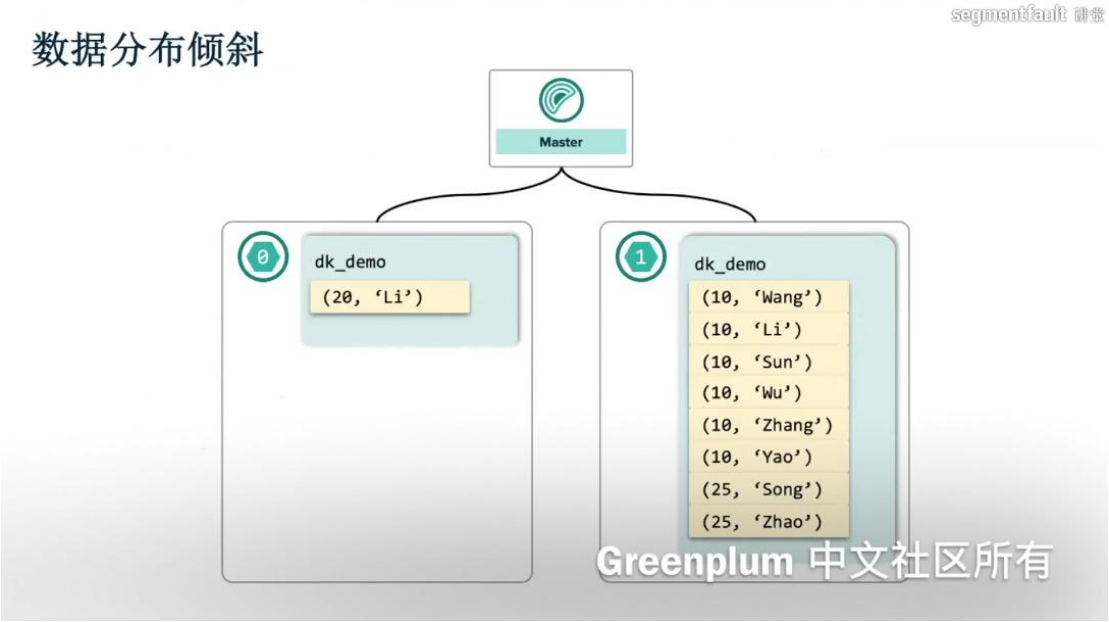


1.2.4.2 数据重分布查询

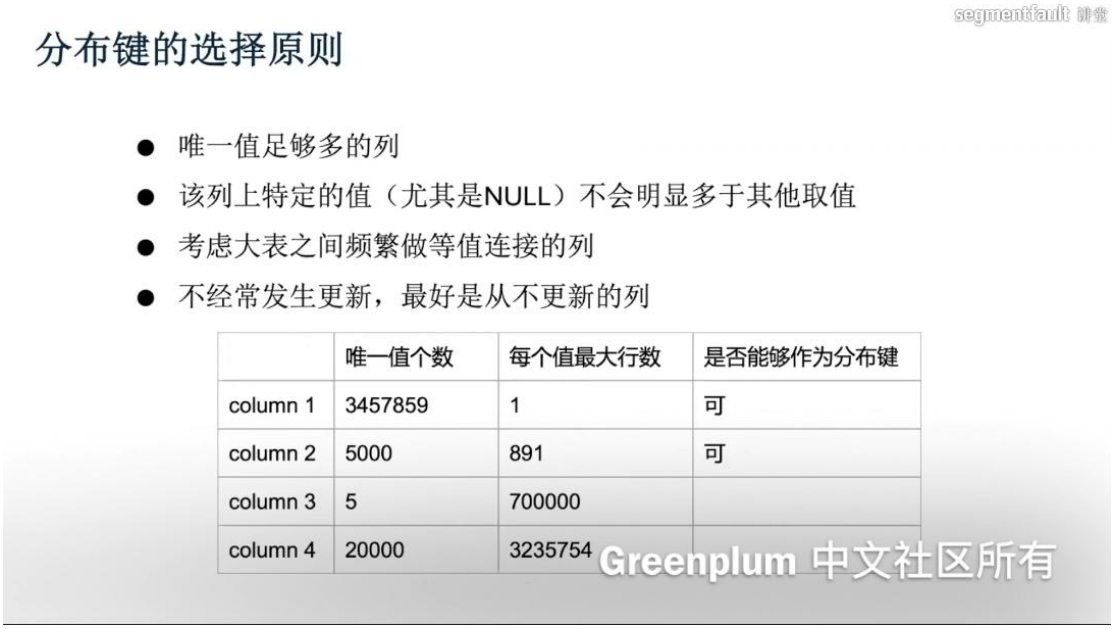


1.2.4.3 分布键与查询性能总结

1.2.5 数据倾斜



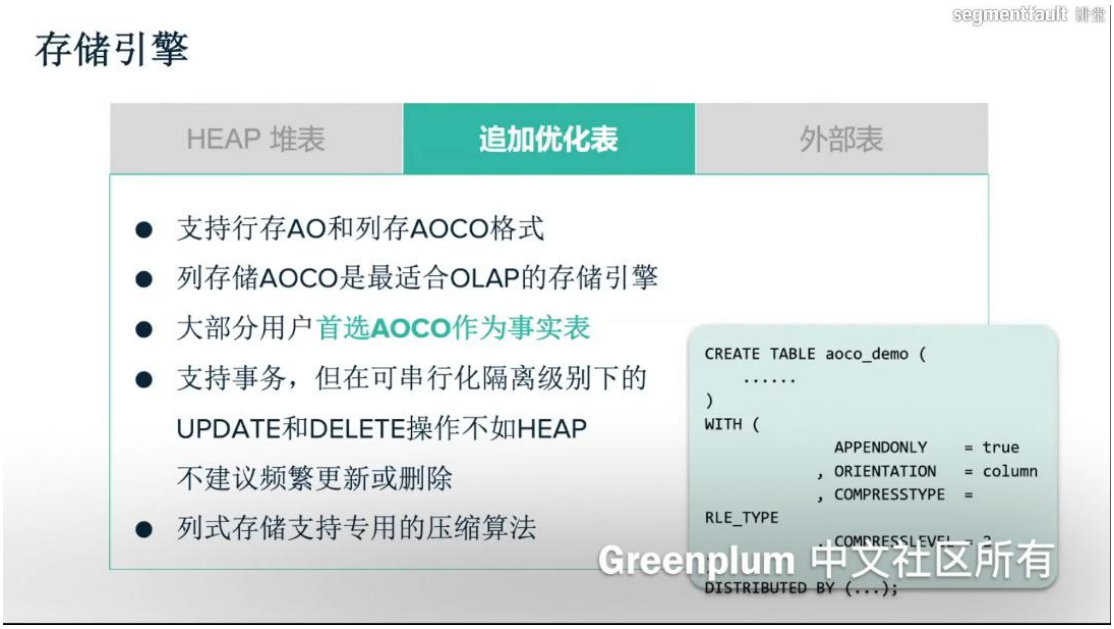
1.2.6 分布键的选择原则总结



1.3 Greenplum 储存引擎

1.3.1 HEAD 堆表

1.3.2 AO 追加优化表



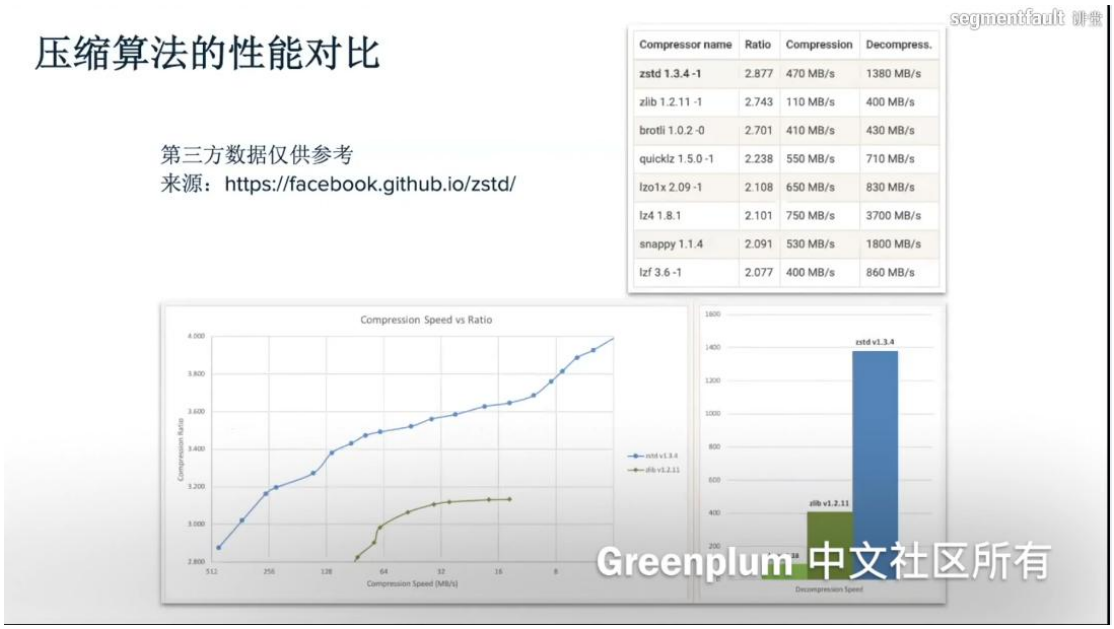
1.3.3 外部表

1.4 Greenplum 压缩

1.4.1 支持的压缩算法

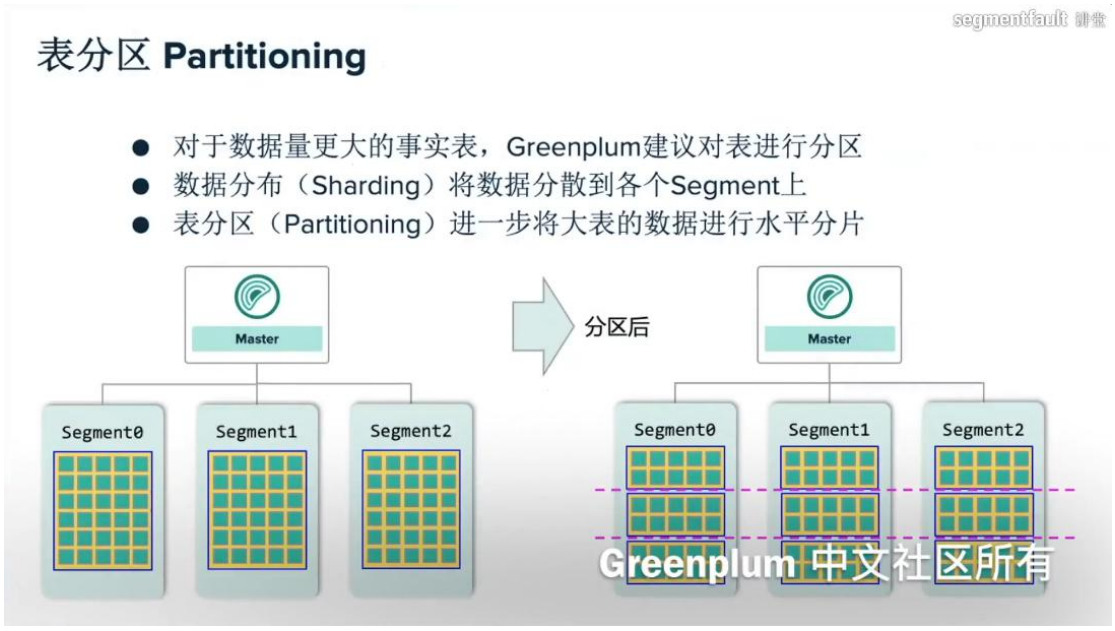
1.4.2 压缩算法的性能对比

详细信息请查看:
https://github.com/facebook/zstd



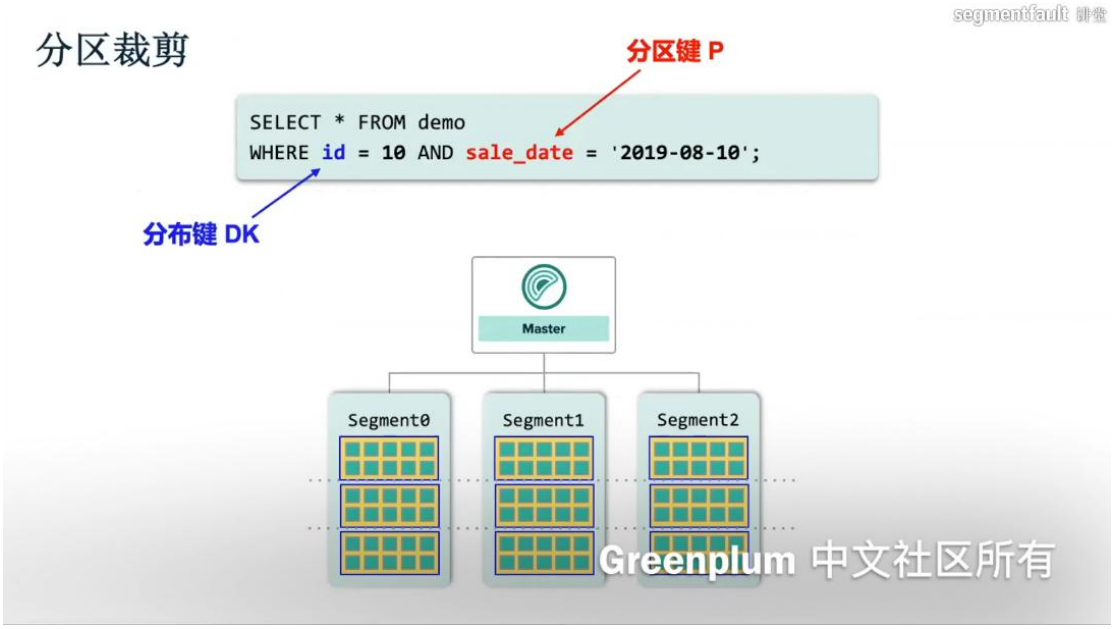
1.5 表分区 Partitioning

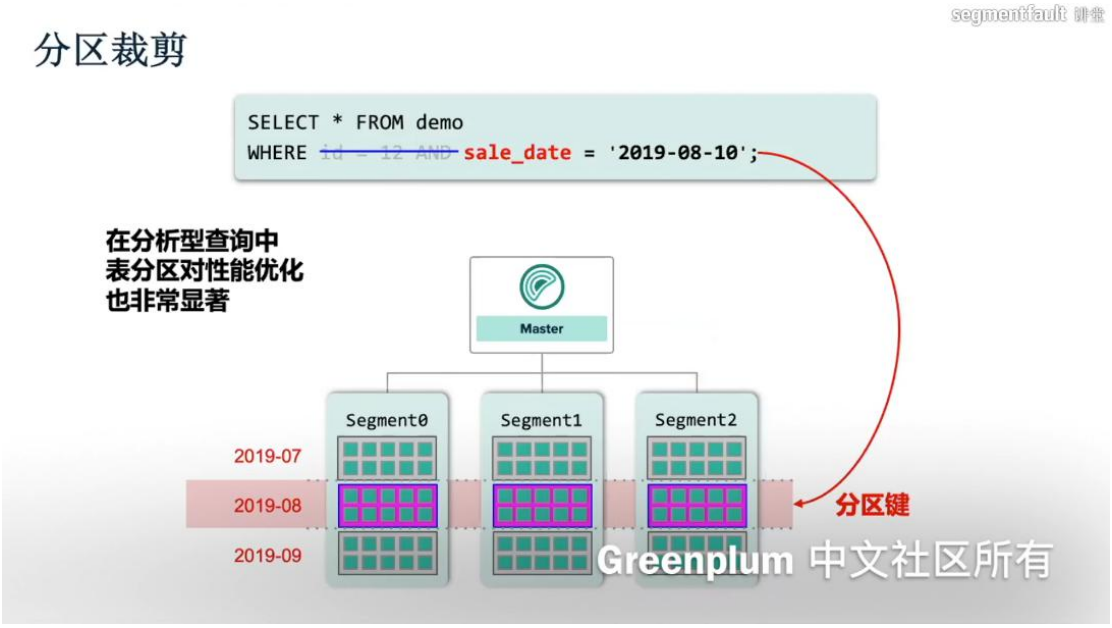
1.5.1 表分区概念



1.5.2 分区裁剪

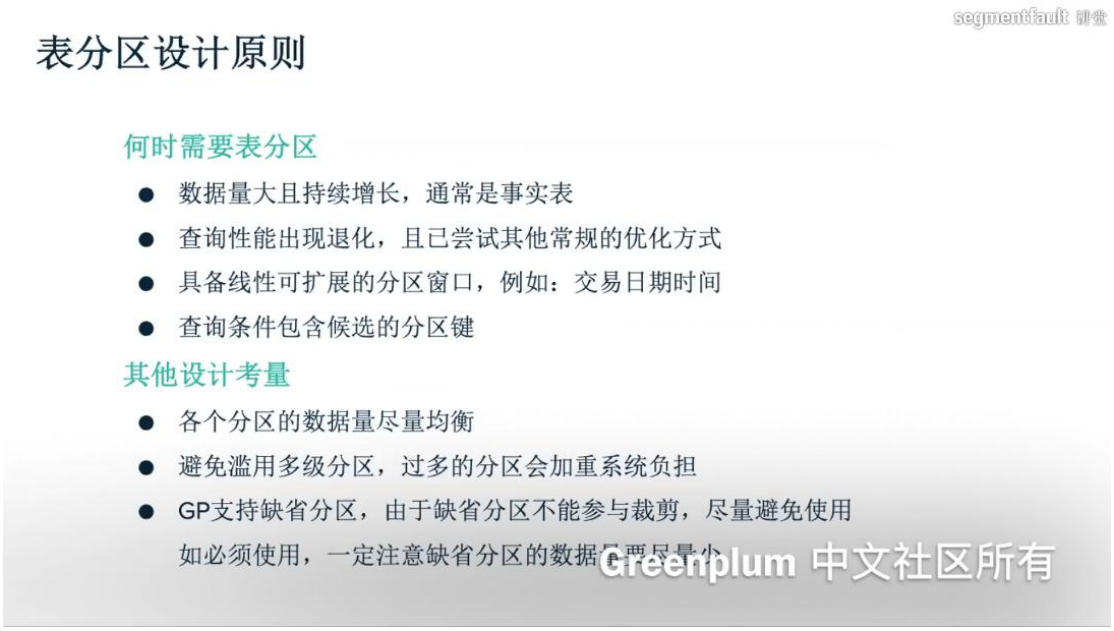
使用分区裁剪能够快速的查询到需要得到的数据



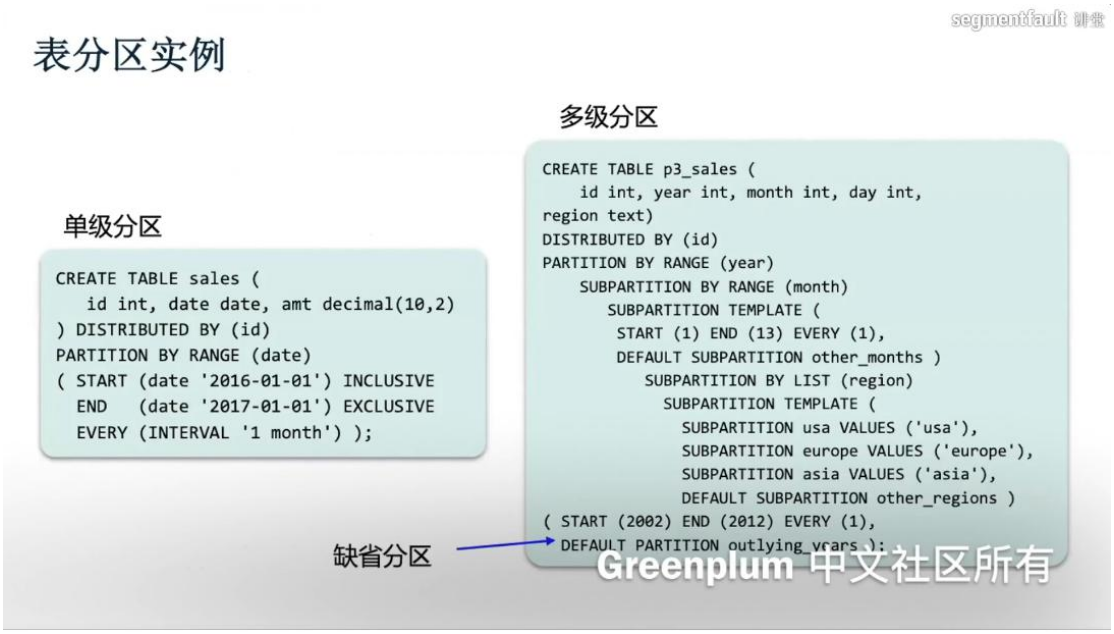


1.6 表分区设计

1.6.1 表分区设计原则

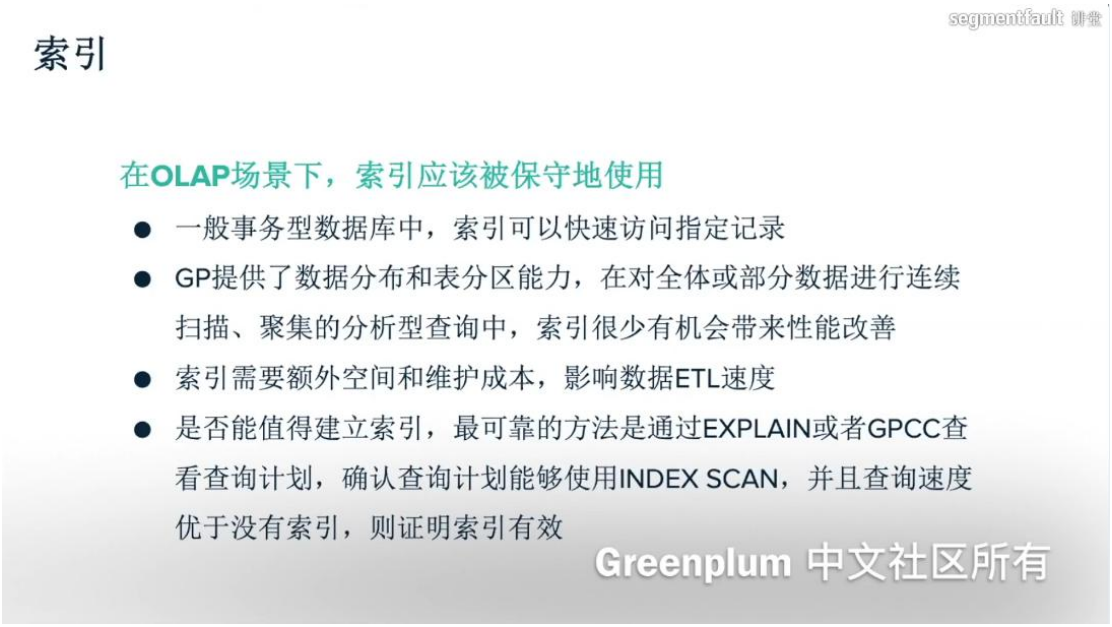


1.6.2 表分区实例

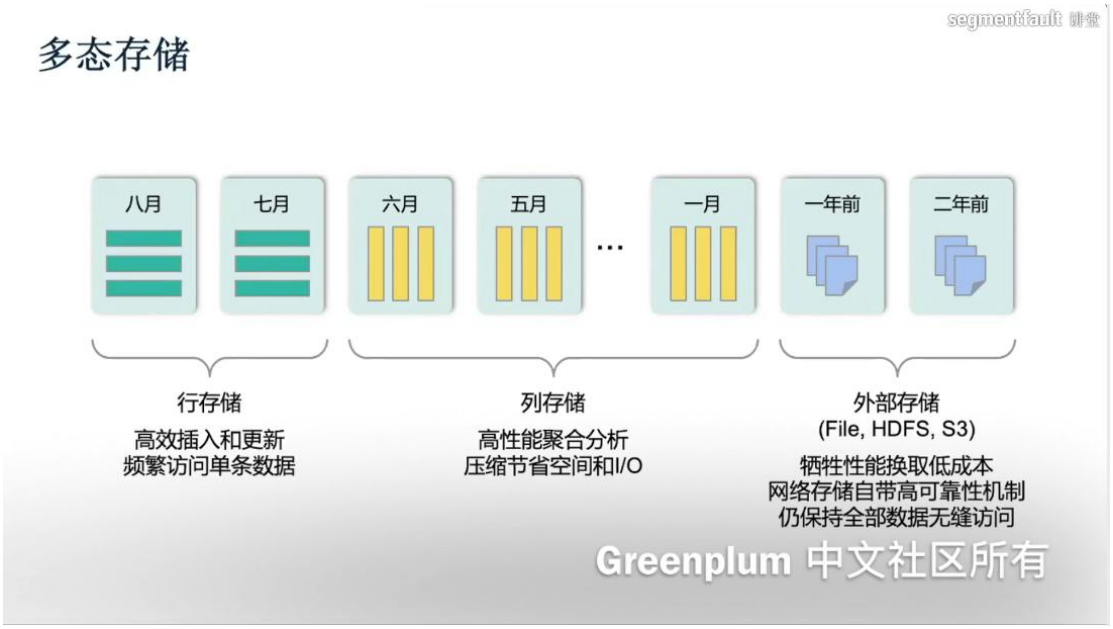


1.7 索引

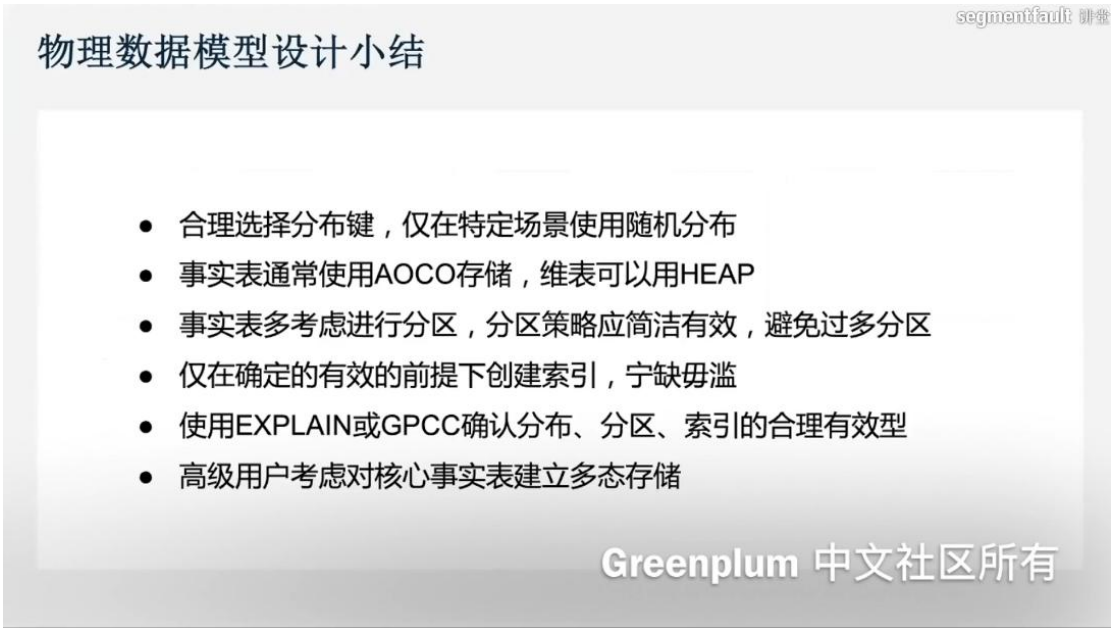
索引不是越创建越好，要通过查看 EXPLAIN 查看执行计划，使用使用了 INDEX SCAN



1.8 多态储存



1.9 物理数据模型设计小结



2 GP 的应用开发

2.1 SQL 和事务

在 greenplum 中使用 BEGIN 开始事务，使用 COMMIT 提交事务

2.1.1 事务与应用的实例

事务与应用

- A和B两人各有1000元
- A给B转账100元
- 同时有一笔6%（共120元）的分红到账，需要分配给A和B

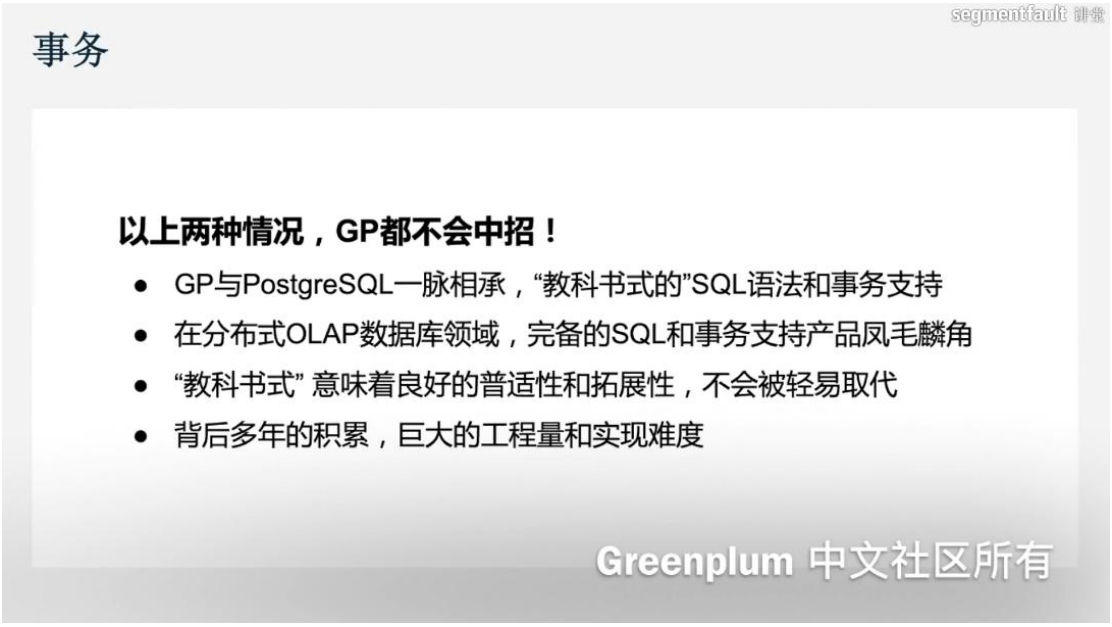
结果 A有954元，B有1160元　共计2114元，有6元不翼而飞

事务与应用

- A有100元
- 一笔交易支出了 10 元，但因其他原因交易提交失败了
- 另一笔交易支出了20元，成功提交

最后 A 余额 70 元

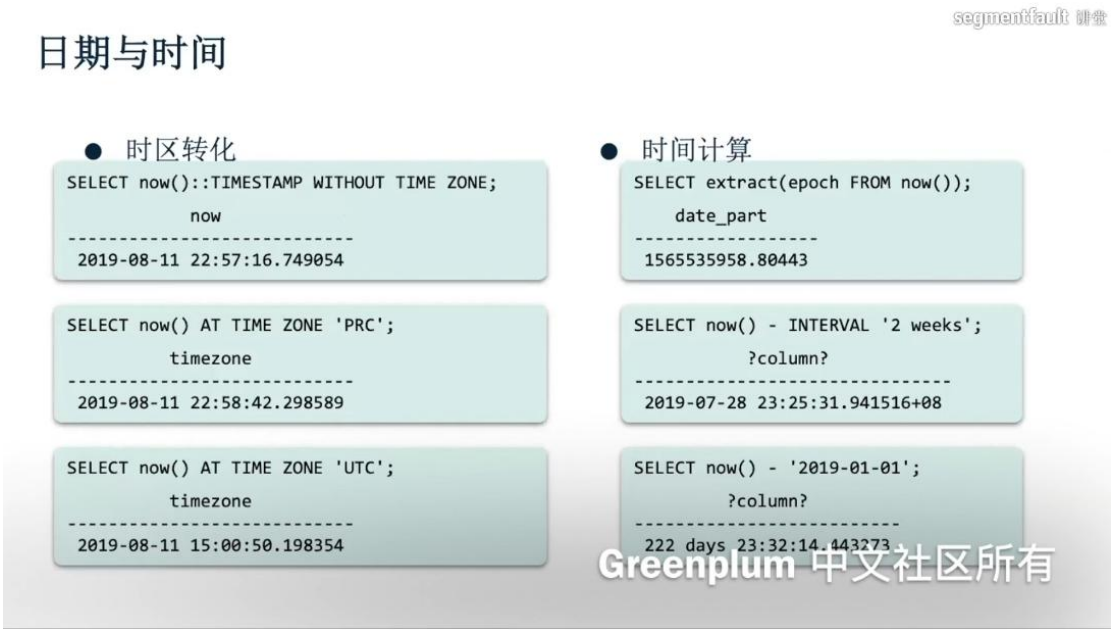
2.1.2 事务的总结



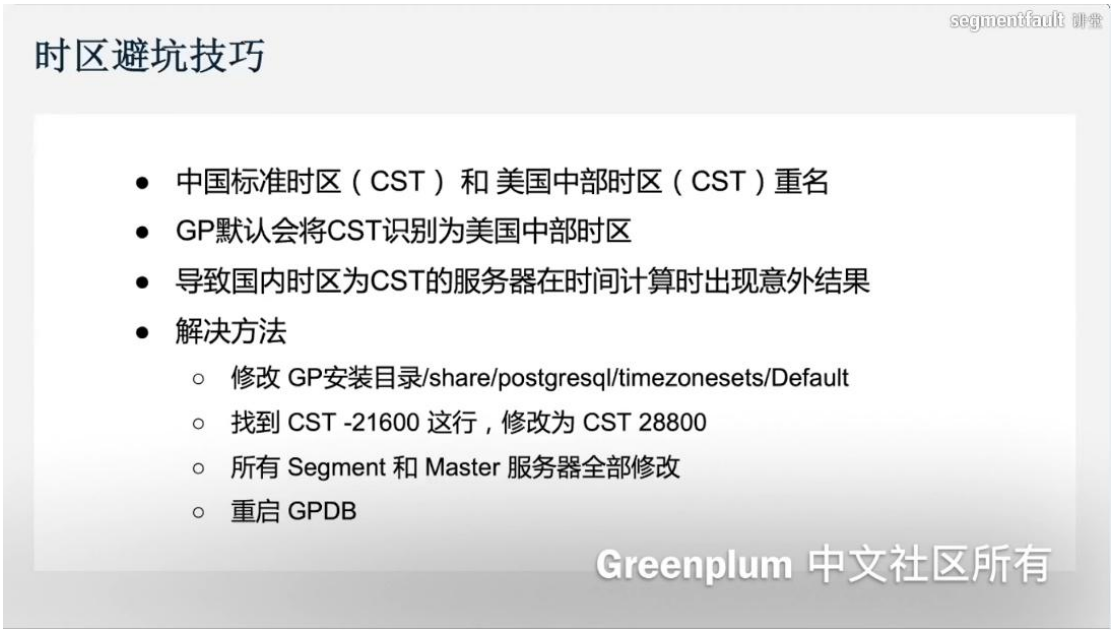
2.2 时间日期和内置函数

2.2.1 注意查看区别

2.2.2 常用的日期与实践 SQL



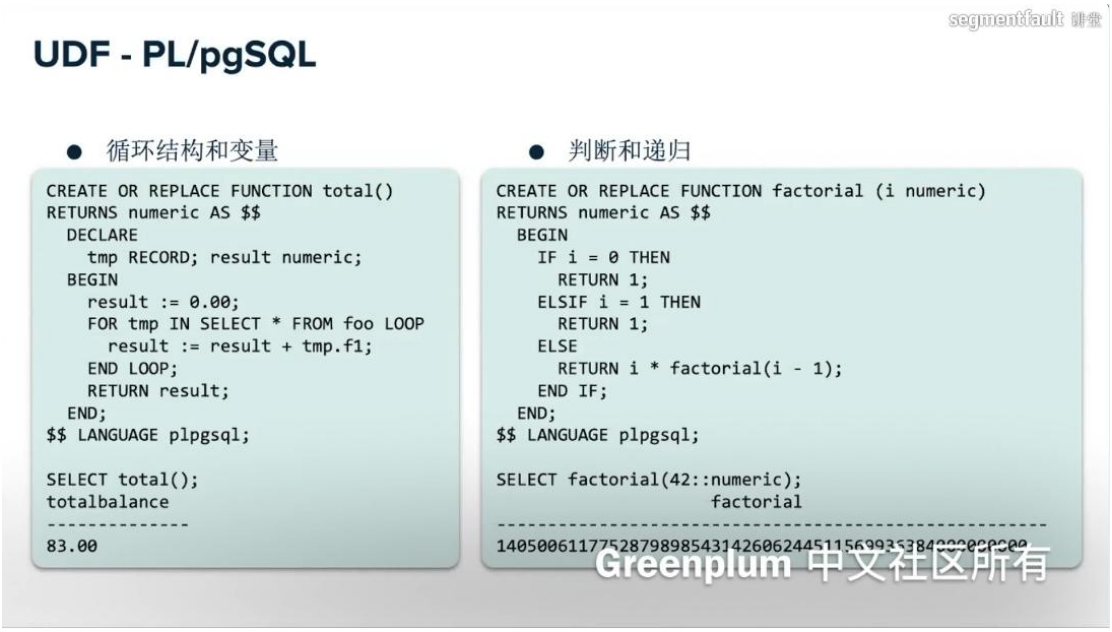
2.2.3 时区避坑技巧



2.3 UDF 和 UDT

2.3.1 内置的 UDF

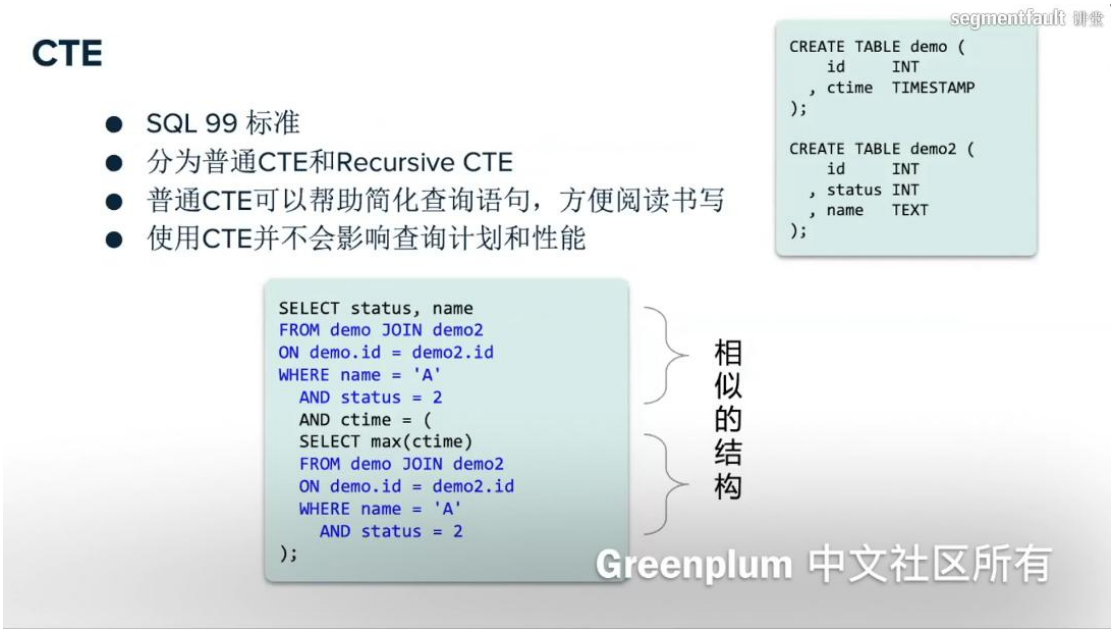
UDF 可以使用循环结构和变量,也可以支持判断和递归



2.3.2 常见的 UDF 语言

2.4 External Web Table

2.5 公用表表达式 CTE



CTE

- SQL 99 标准
- 分为普通CTE和Recursive CTE
- 普通CTE可以帮助简化查询语句，方便阅读书写
- 使用CTE并不会影响查询计划和性能

```
CREATE TABLE demo (
    id      INT
  , ctime   TIMESTAMP
);

CREATE TABLE demo2 (
    id      INT
  , status INT
  , name    TEXT
);
```

```
SELECT status, name
FROM demo JOIN demo2
ON demo.id = demo2.id
WHERE name = 'A'
  AND status = 2
  AND ctime = (
  SELECT max(ctime)
  FROM demo JOIN demo2
  ON demo.id = demo2.id
  WHERE name = 'A'
    AND status = 2
);
```

```
WITH cte_demo (ctime, status, name)
AS (
  SELECT ctime, status, name
  FROM demo JOIN demo2
  ON demo.id = demo2.id
  WHERE name = 'A'
    AND status = 2
)
SELECT * FROM cte_demo
WHERE ctime = (
  SELECT max(ctime) FROM cte_demo
);
```


Recursive CTE

- 举例：分析公司员工的汇报关系

员工ID	姓名	职级	经理ID
10000	张三	CEO	--
20001	李四	副总	10000
20005	王五	副总	10000
20008	赵六	经理	20001
20231	路人甲	职员	20001
24122	路人乙	职员	20001
30123	路人丙	职员	20001
30211	路人丁	职员	20001
21001	小A	职员	20008
21002	小B	职员	20008
21003	小C	职员	20008
43421	东邪	职员	20005
31353	西毒	职员	20005
23112	南帝	职员	20005
28995	北丐	职员	20005
65432	黄蓉	职员	28995

Recursive CTE

- 怎样通过SQL分析得出如下结果？
  求每个员工的直属下属和间接下属总数

员工ID	姓名	下属总数
10000	张三	16
20001	李四	8
20005	王五	6
20008	赵六	3
21001	小A	0

.....

Recursive CTE

解答：

初始查询 →

递归查询 →

```
WITH RECURSIVE rcte_demo (id, name, manager_id, cnt)
AS (
    SELECT m.id, m.name, m.manager_id, count(*)
    FROM employee m JOIN employee r
      ON m.id = r.manager_id
    GROUP BY (m.id, m.name, m.manager_id)
  UNION ALL
    SELECT e.id, e.name, e.manager_id, r.cnt
    FROM employee e JOIN rcte_demo r
      ON e.id = r.manager_id
)
SELECT id, name, sum(cnt) rcnt
FROM rcte_demo
GROUP BY (id, name)
ORDER BY rcnt desc, id, name;
```

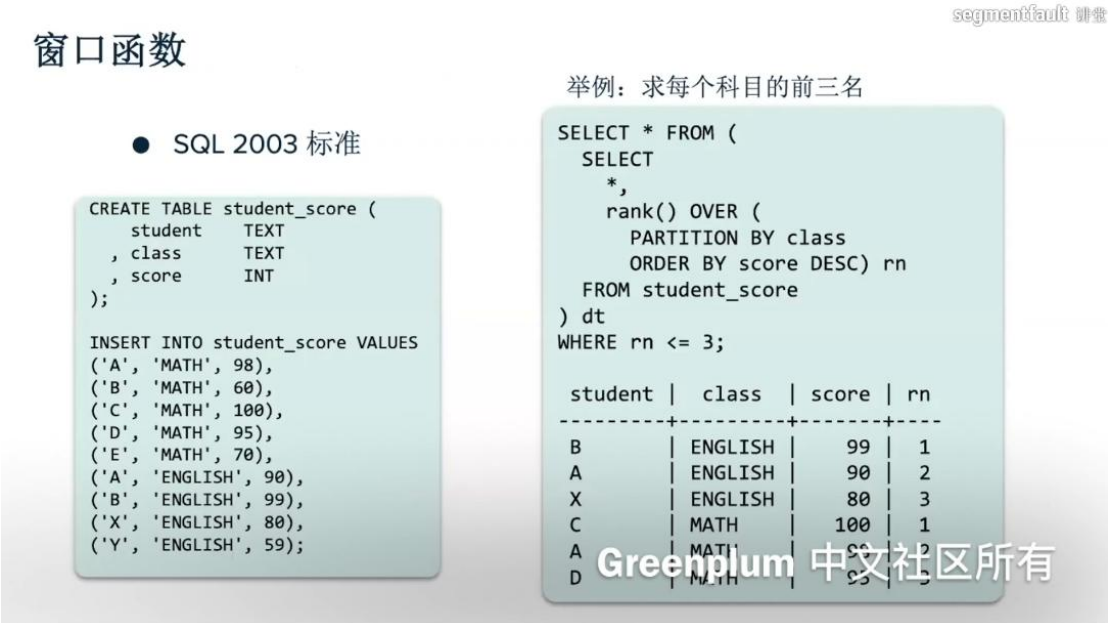
附：范例数据

```
CREATE TABLE employee (
    id         INT
  , name       TEXT
  , level      TEXT
  , manager_id INT
);

INSERT INTO employee VALUES
(10000, 'ZHANG SAN', 'CEO', NULL),
(20001, 'LI SI', 'VP', 10000);
(20005, 'WANG WU', 'VP', 10000);
(20008, 'ZHAO LIU', 'M', 20001);
(21001, 'A', 'E', 20008);
(21002, 'B', 'E', 20008);
(21003, 'C', 'E', 20008);
(20231, 'JIA', 'E', 20001);
(24122, 'YI', 'E', 20001);
(30123, 'BING', 'E', 20001);
(30211, 'DING', 'E', 20001);
(43421, 'EAST', 'E', 20005);
(31353, 'WEST', 'E', 20005);
(23112, 'SOUTH', 'E', 20005);
(28995, 'NORTH', 'E', 20005);
(63421, 'GUO JING', 'E', 28995);
(12345, 'HUANG ZH', 'E', 28995);
```

2.6 ROLLUP,CUBE,and GROUPING SETS

2.7 窗口函数



2.8 GP 扩展介绍

详细的请查看:

GP扩展框架

- PG有着丰富的扩展框架生态
  - https://github.com/postgres/postgres/tree/master/contrib
- 基于扩展框架产品
  - GPCC Metrics Collector 指标收集器
  - GPCC Workload Manager 工作负载管理
  - DiskQuota 磁盘配额
  - PostGIS 地理信息
  - pgcrypto 数据加密
  - pgaudit 数据审计

扩展框架举例

GP扩展组件的构成

- Makefile
- control文件
- sql文件
- C代码文件

my_extension.control

```
comment = 'Sample extension'
default_version = '1.0'
module_pathname = '$libdir/my_extension'
relocatable = true
```

my_extension--1.0.sql

```
CREATE FUNCTION my_extension_demo()
RETURNS TEXT STRICT
AS 'MODULE_PATHNAME'
LANGUAGE C;
```

Makefile

```
MODULES = my_extension

EXTENSION = my_extension
DATA = my_extension--1.0.sql

PG_CONFIG = pg_config
PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

my_extension.c

```
#include "postgres.h"
#include "utils/builtins.h"

PG_MODULE_MAGIC;

PG_FUNCTION_INFO_V1(my_extension_demo);

Datum
my_extension_demo(PG_FUNCTION_ARGS)
{
            PG_RETURN_TEXT_P(
    cstring_to_text("Hello, Greenplum!"));
}
```

扩展框架举例

- make

```
± |6X_STABLE → origin ?:3 ✗| → make
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -Wno-address -Wno-unused-command-line-argument -g -ggdb -O
 -std=gnu99   -Werror=uninitialized -Werror=implicit-function-declaration  -I. -I./ -I/Users/haowang/usr/local/greenplum-db-6X/include/postgresql/server -I/Users/haowang/usr/local/greenplum-db-6
/include/postgresql/internal    -c -o my_extension.o my_extension.c -MMD -MP -MF .deps/my_extension.Po
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -Wno-address -Wno-unused-command-line-argument -g -ggdb -O
 -std=gnu99   -Werror=uninitialized -Werror=implicit-function-declaration  -L/Users/haowang/usr/local/greenplum-db-6X/lib   -Wl,-dead_strip_dylibs   -bundle -bundle_loader /Users/haowang/usr/loc
l/greenplum-db-6X/bin/postgres -o my_extension.so my_extension.o
```

- make install

```
2019-08-12 20:22:57 ☆ Haos-MacBook-Pro in ~/workspace/gpdb/contrib/my_extension
± |6X_STABLE → origin ?:3 ✗| → make install
/bin/sh /Users/haowang/usr/local/greenplum-db-6X/lib/postgresql/pgxs/src/makefiles/../../config/install-sh -c -d '/Users/haowang/usr/local/greenplum-db-6X/share/postgresql/extension'
/bin/sh /Users/haowang/usr/local/greenplum-db-6X/lib/postgresql/pgxs/src/makefiles/../../config/install-sh -c -d '/Users/haowang/usr/local/greenplum-db-6X/share/postgresql/extension'
/bin/sh /Users/haowang/usr/local/greenplum-db-6X/lib/postgresql/pgxs/src/makefiles/../../config/install-sh -c -d '/Users/haowang/usr/local/greenplum-db-6X/lib/postgresql'
/usr/bin/install -c -m 644 my_extension.control '/Users/haowang/usr/local/greenplum-db-6X/share/postgresql/extension/'
/usr/bin/install -c -m 644 my_extension--1.0.sql '/Users/haowang/usr/local/greenplum-db-6X/share/postgresql/extension/'
/usr/bin/install -c -m 755  my_extension.so '/Users/haowang/usr/local/greenplum-db-6X/lib/postgresql/'
```

```
2019-08-12 20:26:38 ☆ Haos-MacBook-Pro in ~/workspace/gpdb/contrib/my_extension
± |6X_STABLE → origin ?:3 ✗| → psql -c "CREATE EXTENSION my_extension;"
CREATE EXTENSION

2019-08-12 20:30:22 ☆ Haos-MacBook-Pro in ~/workspace/gpdb/contrib/my_extension
± |6X_STABLE → origin ?:3 ✗| → psql -c "SELECT * FROM my_extension_demo();"
 my_extension_demo
-------------------
 Hello Greenplum
(1 row)
```