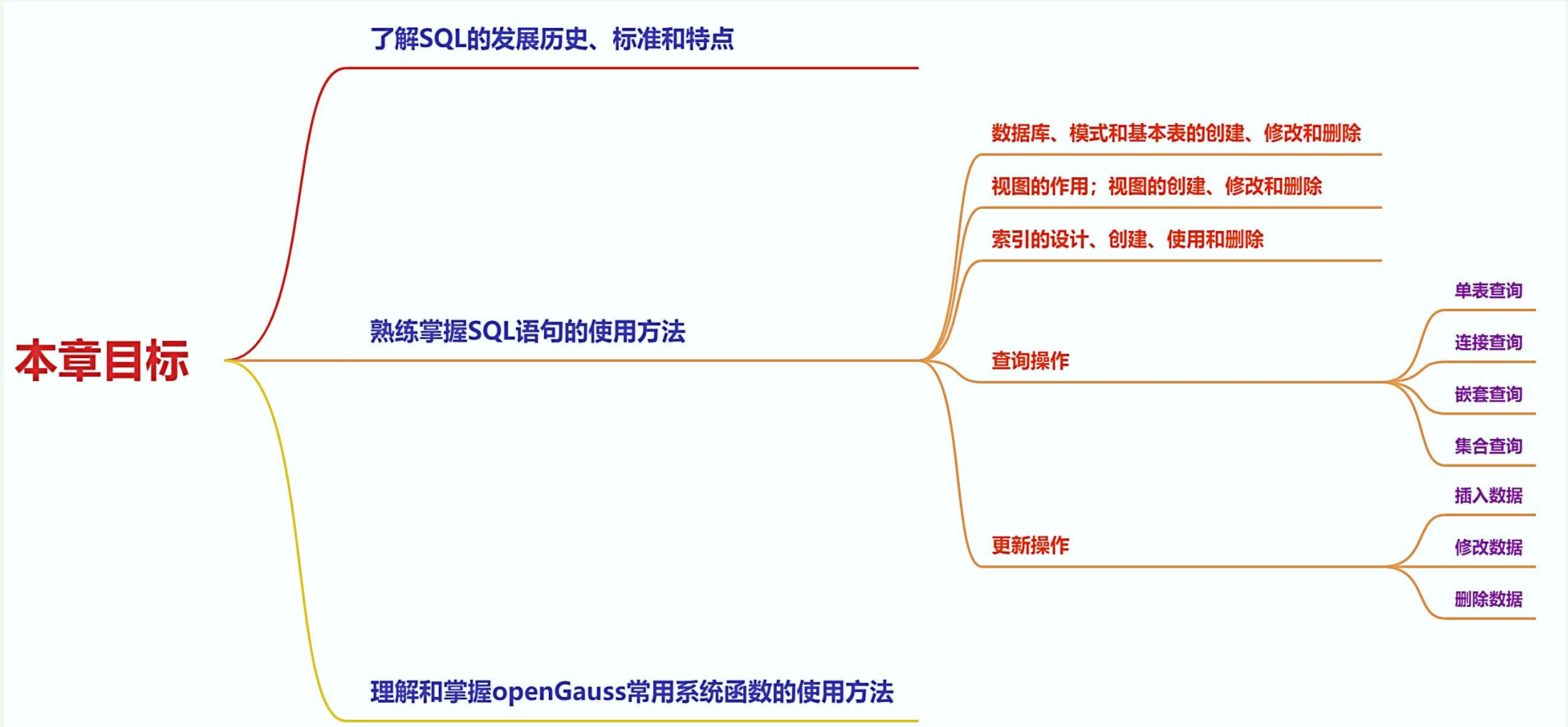

第3章 SQL之数据定义

1 本章导图


本章目标
了解SQL的发展历史、标准和特点
熟练掌握SQL语句的使用方法
数据库、模式和基本表的创建、修改和删除
视图的作用；视图的创建、修改和删除
索引的设计、创建、使用和删除
查询操作
单表查询
连接查询
嵌套查询
集合查询
更新操作
插入数据
修改数据
删除数据
理解和掌握openGauss常用系统函数的使用方法

2 大纲

- SQL概述
- 学生选课数据库
- 数据定义
- 数据查询
- 数据更新
- 空值的处理
- 视图
- 本章小结

3 SQL概述

- SQL语言(Structured Query Language)
  - Pronounced /ɛskjuːˈɛl /; unofficially /ˈsiːkwəl/
  - 结构化查询语言，是关系数据库的标准语言
  - SQL是一个通用的、功能极强的关系数据库语言
    - 支持数据定义(DDL)，数据操纵(DML)，数据控制(DCL)等功能
  - SQL作为共同的数据存取语言和标准接口，使不同数据库系统之间的互操作有了共同的基础

- 本节主要内容
  - SQL的产生与发展
  - SQL的特点
  - SQL的基本概念

3 3.1 SQL的产生与发展

- 1974年，Boyce和Chamberlin提出 SQL 标准
- 1975年～1979年IBM公司在System R原型系统上实现
  - SQL 86和SQL 89是单个文档
  - SQL 92和SQL 99扩展为一系列开放的部分。SQL 92增加了SQL调用接口、SQL永久存储模块
  - SQL 99扩展为框架、SQL基础部分、SQL调用接口、SQL永久存储模块、SQL宿主语言绑定、SQL外部数据的管理和SQL对象语言绑定等
  - SQL2016扩展到了12个部分，引入XML类型、Window函数、TRUNCATE操作、时序数据以及JSON（JavaScript Object Notation）类型等
- 目前，没有一个数据库系统能够支持SQL标准的所有概念和特性，都只实现了SQL标准的一个子集。许多软件厂商对SQL基本命令集进行不同程度的扩充和修改，使之可以支持标准以外的一些功能
  - 使用具体的RDBMS时一定要查阅相应产品的用户手册

表3.1 SQL标准的发展过程

标准	篇幅(约)/页	发布日期/年	标准	大致页数	发布日期/年
SQL 86		1986年	SQL 2003	3600	2003
SQL 89(FIPS 127-1)	120页	1989年	SQL 2008	3777	2008
SQL 92	622页	1992年	SQL 2011	3817	2011
SQL 99（SQL 3）	1700页	1999年	SQL2016	4035	2016

3.2 SQL的特点

- 功能综合且风格统一
- 数据操纵高度非过程化
- 面向集合的操作方式
- 以统一的语法结构提供多种使用方式
- 语言简洁且易学易用

3 特点1：功能综合且风格统一

- 集数据定义语言，数据操纵语言，数据控制语言功能于一体
- 可以独立完成数据库生命周期中的全部活动：
  - 创建和删除数据库模式
  - 创建基本表，创建视图
  - 使用数据库，包括查询和增删改数据、事务处理等
  - 数据库控制，包括安全性控制、完整性控制和并发控制等
  - 数据库维护和重构，如修改和删除基本表、数据库备份与恢复等
- 用户在数据库投入运行后，可根据需要随时或逐步创建模式
- 数据操作符统一
  - 查找、插入、删除、更新等操作都只需要一种操作符

3 特点2：数据操纵高度非过程化

- 层次、网状模型的数据操纵语言面向过程，必须指定存取路径

- **SQL只要提出“做什么”，无须了解存取路径**
  - 存取路径的选择以及**SQL**的操作过程由**RDBMS**自动完成
  - 大大减轻了用户负担
  - 当数据存储结构发生变化时，数据操纵语句一般不用改变，提高了数据独立性

3 特点3：面向集合的操作方式

- 层次、网状模型采用面向记录的操作方式，操作对象是一条记录

- **SQL采用集合操作方式**
  - 操作对象、查找结果可以是元组的集合
  - 一次插入、删除、更新操作的对象可以是元组的集合

3 特点4：以统一的语法结构提供多种使用方式

厦門大學 XIAMEN UNIVERSITY

- SQL是独立的语言
  - 能够立地用于联机交互的使用方式
- SQL又是嵌入式语言
  - SQL能够嵌入到高级语言(例如C，C++，Java，Python)程序中，供程序员设计程序时使用。
  - 通常使用JDBC或ODBC中间件实现访问
- 在上述两种不同的使用方式下，SQL的语法结构基本一致，这提高了易用性和方便性

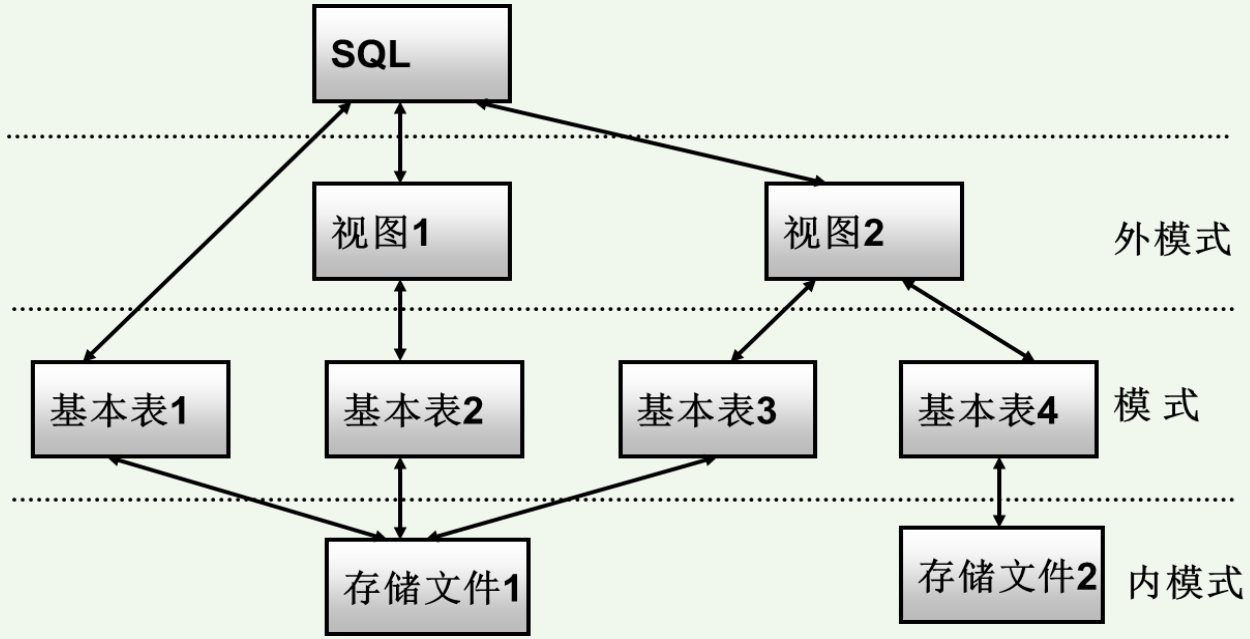
3 特点5：语言简洁且易学易用

- SQL功能极强，完成核心功能只用9个动词

SQL 功能	动词
数据定义	CREATE，DROP，ALTER
数据查询	SELECT
数据操纵	INSERT，UPDATE，DELETE
数据控制	GRANT，REVOKE

3.3 SQL的基本概念

- SQL支持关系数据库的三级模式结构



- **基本表(base table)**
  - 本身独立存在的表
  - 一个关系就对应一个基本表
  - 一个(或多个)基本表对应一个存储文件
  - 一个表可以带若干索引
- **存储文件(stored file)**
  - 逻辑结构和物理结构组成了关系数据库的内模式
  - 物理结构对用户是隐蔽的，是由**RDBMS**设计确定的
- **视图(view)**
  - 从一个或几个基本表导出的表
  - 数据库中只存放视图的定义而不存放视图对应的数据
  - 视图是一个虚表
  - 用户可以在视图上再定义视图

4 学生选课数据库

厦門大學 XIAMEN UNIVERSITY

- 学生选课模式S-C-SC：
  - Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
  - Course(Cno,Cname,Ccredit,Cpno)
  - SC(Sno,Cno,Grade,Semester,Teachingclass)

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

- SQL的数据定义功能：
  - 模式(Schema)定义
  - 表(Table)定义
  - 视图(View)和索引(Index)的定义

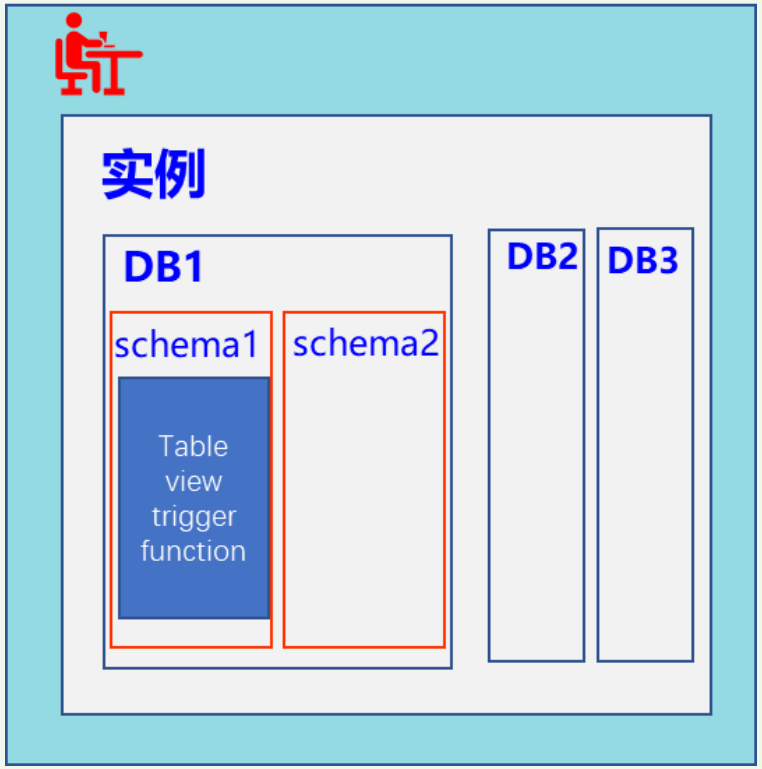
操作对象	操作方式		
	创建	删除	修改
数据库模式	CREATE SCHEMA	DROP SCHEMA	*SQL标准无修改语句
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	*CREATE INDEX	*DROP INDEX	*ALTER INDEX

5 数据定义(cont.)

厦門大學 XIAMEN UNIVERSITY

- 本节主要内容
  - **模式的定义与删除**
  - **基本表的定义、修改与删除**
  - **索引的建立与删除**
  - **数据字典**

- **模式**
  - 当前RDBMS提供了**层次化**的数据库对象命名机制
    - 一个RDBMS的实例中可以建立多个数据库
    - 一个数据库中可以建立多个模式
    - 存储数据的基本表就位于某个模式下，由此形成了**数据的访问路径**



用户、实例、数据库、模式和数据库对象之间包含关系示意图

5.1.1定义模式

- 定义模式实际上就定义了一个命名空间
  - 在这个空间中可以定义该模式包含的数据库对象
    - 基本表、视图、索引、触发器、存储过程、函数和包等
  - 简而言之，模式就是数据库对象的集合
- **模式定义语句：**

  CREATE SCHEMA <模式名> AUTHORIZATION <用户名>;
  - 创建模式必须具有**DBA权限**，或获得了**DBA**授予的**CREATE SCHEMA**权限
  - 若模式名缺失，则模式名默认为用户名
  - **CREATE SCHEMA**可以接受**CREATE TABLE**，**CREATE VIEW**和**GRANT**子句

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>[<表定义子句>|<视图定义子句>|<授权定义子句>]

5 示例

[例3.1] 为用户WANG定义一个学生选课模式S-C-SC

```
CREATE SCHEMA“S-C-SC”AUTHORIZATION WANG;  /*标识符要么加双引号，要么不加任何符号*/
```

[例3.2] CREATE SCHEMA AUTHORIZATION WANG; /*没有指定<模式名>，<模式名>隐含为用户名WANG */

[例3.3] 为用户ZHANG创建了一个模式Test，并且在其中定义一个表Tab1

```
CREATE SCHEMA Test AUTHORIZATION ZHANG
CREATE TABLE Tab1(Col1   SMALLINT,
                  Col2   INT,
                  Col3   CHAR(20),
                  Col4   NUMERIC(10,3),
                  Col5   DECIMAL(5,2) );
```

5.1.2删除模式

- **删除模式语句：**

  DROP SCHEMA <模式名> <CASCADE | RESTRICT>;

  - **CASCADE(级联)**
    - 删除模式的同时把该模式中所有的数据库对象全部删除
  - **RESTRICT(限制)**
    - 如果该模式中定义了数据库对象(如表、视图等)，则拒绝该删除语句的执行
    - 仅当该模式中没有任何下属的对象时才能执行**Drop schema**命令

[例3.4] 删除[例3.3]中建立的模式Test

DROP SCHEMA Test CASCADE;

- 删除模式**TEST**及该模式中定义的表**TAB1**

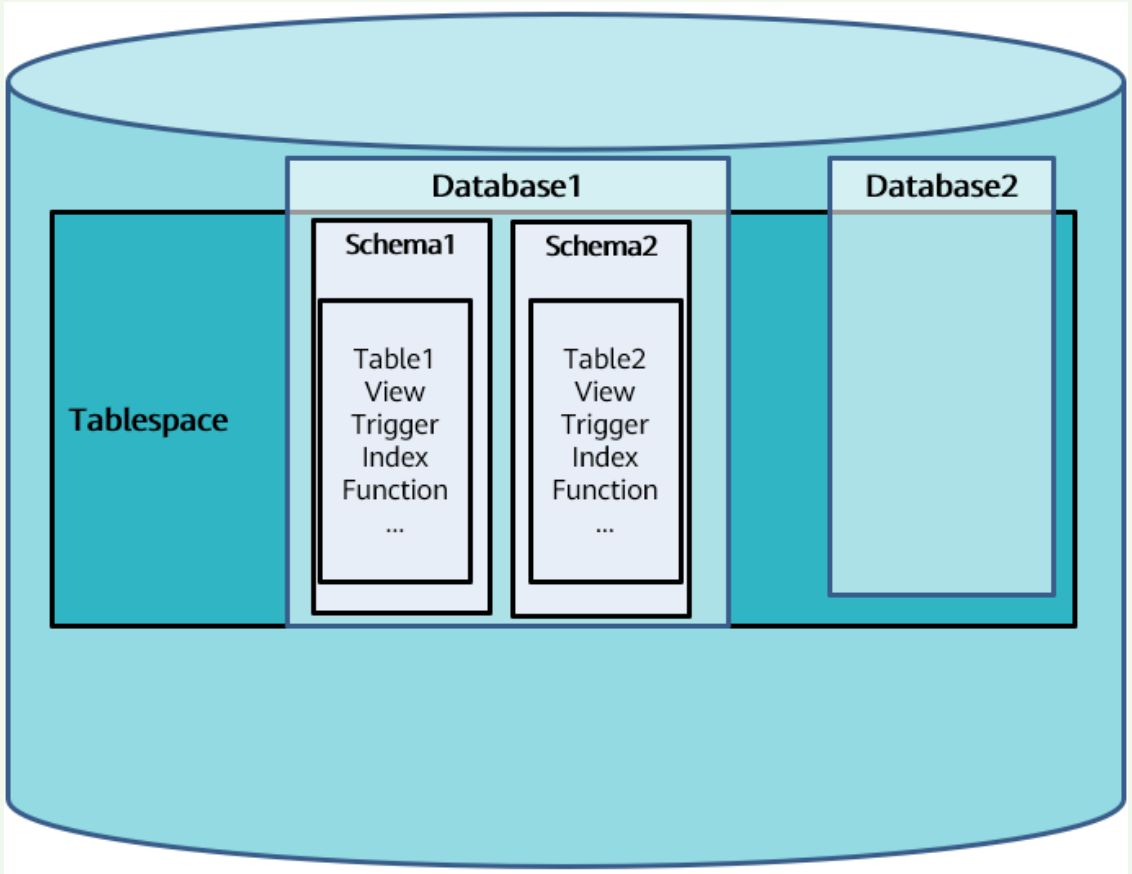
- 问题：如何修改模式?

5 openGauss之用户、角色和用户组

- openGauss使用用户User和角色role来控制对数据库的访问
  - 根据角色自身的设置不同，一个角色可以看做是一个数据库用户，或者一组数据库用户
  - 在openGauss中角色和用户之间的区别只在于角色默认是没有LOGIN权限的。
  - 在openGauss中一个用户唯一对应一个角色，不过可以使用角色叠加来更灵活地进行管理

- 为了实现安装过程中安装帐户权限最小化及安装后openGauss的系统运行安全性。安装脚本在安装过程中会自动按照用户指定内容创建安装用户，并将此用户作为后续运行和维护openGauss的管理员帐户.
  - dbgrp组：初始化安装环境时，如果该用户组不存在，则会自动创建，属于操作系统
  - omm用户：在安装openGauss过程中运行“ gs_install”时，会创建与安装用户同名的数据库用户，即数据库用户omm。此用户具备数据库的最高操作权限，此用户初始密码由用户指定。属于OS
  - 关于omm的权限可参见：https://www.modb.pro/db/9151
  - 创建用户时，应当使用双引号或单引号将用户密码括起来

openGauss之表空间、数据库与模式



- 表空间Tablespace对应磁盘上的一个目录，里面存储的是它所包含的数据库的各种物理文件
- 可以存在多个表空间，每个表空间可以对应多个Database
- 表空间仅是起到了物理隔离的作用，其管理功能依赖于文件系统
- 两个默认表空间：pg_default、pg_global

- 数据库Database用于管理各类数据对象，各数据库间相互隔离。数据库管理的对象可分布在多个Tablespace上
- 创建数据对象时可以指定对应的表空间，如果不指定相应的表空间，相关的对象会默认保存在PG_DEFAULT空间中

openGauss之模式

- **openGauss**的模式是对数据库做一个逻辑分割，所有的数据库对象都建立在模式下面
  - 数据库对象包括：表、视图、索引、触发器、函数、存储过程等
- **openGauss**的模式和用户是弱绑定的
  - 所谓的弱绑定是指创建用户的同时会自动创建一个同名模式，但用户也可以单独创建模式，并且为用户指定其他的模式
- 通过管理模式，允许多个用户使用同一数据库而不相互干扰
- 每个数据库包含一个或多个模式

5 openGauss之SQL

- 支持标准的SQL92/SQL99/SQL2003/SQL2011规范，支持GBK和UTF-8字符集，支持SQL标准函数与分析函数，支持存储过程
- openGauss之SQL学习网址：

  https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXDR006+Self-paced/courseware/b86d3987a1294dc6b066a40b86e0032d/cad675e2e02e440587e1cbd6d518e187/

  - 请课后自行完成这部分内容的学习(这对顺利完成实验非常重要)
- openGauss之SQL语句书写规范建议：
  - 因SQL语句大小写不敏感，关键字大写，其他小写
  - 在gsql中，为了提升操作效率，可以全部使用小写
  - 使用行缩进增强可读性

- **openGauss的命名规范：**
  - openGauss的命名规范遵循postgreSQL规定
    - 库名、表名限制命名长度，建议表名及字段名字符总长度不超过63;【强制】
    - 对象名(表名、列名、函数名、视图名、序列名等对象名称)范，对象名务必只使用小写字母，下划线，数字。不要以pg开头，不要以数字开头，不要使用保留字;【强制】
    - 查询中的别名不要使用“小写字母，下划线，数字” 以外的字符, 如中文.【强制】
- 参考： https://www.cnblogs.com/panpanwelcome/p/12430122.html

5.2基本表的定义、修改与删除

- 基本表的定义
- 基本表的修改
- 基本表的删除

5.2.1基本表的定义

- **定义基本表语法：**

```
CREATE TABLE <表名> (<列名><数据类型>[列级完整性约束]
                    [,<列名><数据类型>[列级完整性约束]]
                    ...
                    [,<表级完整性约束>] );
```

  - 如果完整性约束涉及该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级
  - 完整性约束条件在定义后被存入RDBMS的数据字典中，并在对相关数据进行操作时被RDBMS用于自动检查是否满足这些约束条件

[例3.5] 建立学生Student表.

```
CREATE TABLE Student
        (Sno  char(8)  PRIMARY KEY,     /* 列级完整性约束, sno是主码 */
         Sname varchar(20)   UNIQUE,    /* sname取唯一值 */
         Ssex  char(6),
         Sbirthdate date,
         Smajor varchar(40)
         );
```

[例3.6] 建立课程Course表.

```
CREATE TABLE Course
        (Cno  char(5)  PRIMARY KEY,        /*列级完整性约束, cno是主码*/
         Cname varchar(40)   NOT NULL,  /*列级完整性约束, cname非空*/
         Ccredit  smallint,
         Cpno char(5),
         FOREIGN KEY (Cpno) REFERENCES Course(Cno)
           /*表级完整性约束, Cpno是外码，其值参照course.cno值*/
         );
```

[例3.7] 建立学生选课SC表

```
CREATE TABLE SC
        (Sno char(8),
         Cno  char(5),
         Grade smallint,
         Semester char(5),
         Teachingclass char(8),
         PRIMARY KEY (Sno, Cno),  /*两个属性以上的主码必须定义为表级*/
         FOREIGN KEY (Sno) REFERENCES Student(Sno),
         FOREIGN KEY (Cno) REFERENCES Course(Cno)
         );
```

- SQL中域的概念用数据类型来实现
  - 定义表的属性时需要指明其数据类型及长度
  - 选用哪种数据类型
    - 取值范围
    - 要做哪些运算
- **注意！**
  - 不同的**RDBMS**中支持的数据类型不完全相同
  - **openGauss**的内置数据类型可参考华为**openGauss**文档

5 ~数据类型

厦門大學 XIAMEN UNIVERSITY

数据类型	含义
CHAR(*n*), CHARACTER(*n*)	长度为n的定长字符串
VARCHAR(*n*), CHARACTERVARYING(*n*)	最大长度为n的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT，INTEGER	长整数（4字节）
SMALLINT	短整数（2字节）
BIGINT	大整数（8字节）
NUMERIC(*p*, *d*)	定点数，由*p* 位数字（不包括符号、小数点）组成，小数后面有*d* 位数字
DECIMAL(*p*, *d*), DEC(*p*, *d*)	同NUMERIC
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(*n*)	可选精度的浮点数，精度至少为*n*位数字
BOOLEAN	逻辑布尔量
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

- 每一个基本表都属于某一个模式，一个模式包含多个基本表

- 定义基本表所属模式
  - 方法一：在表名中明显地给出模式名
    - **Create table"S-C-SC".Student(…);　/* Student所属的模式是S-C-SC */**
    - **Create table"S-C-SC".Course(…);　/* Course所属的模式是S-C-SC */**
    - **Create table"S-C-SC".SC(…);　/* SC所属的模式是S-C-SC */**
  - 方法二：在创建模式语句中同时创建表
  - 方法三：设置所属的模式

Tips：在openGauss中，指明表时应把该表所属的模式也加上去

- 创建基本表(其他数据库对象)时，若没有指定模式，系统根据搜索路径来确定该对象所属的模式
  - RDBMS会使用模式列表中第一个存在的模式作为数据库对象的模式名
  - 若搜索路径中的模式名都不存在，系统将给出错误
    - 显示当前的搜索路径： **SHOW SEARCH_PATH;**
    - 搜索路径的当前默认值是：**$user，PUBLIC**
      - 表示首先搜索与用户名相同的模式名，若该模式不存在则使用PUBLIC模式

- **DBA也可以设置搜索路径**
  - 例如，**SET SEARCH_PATH TO “S-C-SC”, PUBLIC;**
  - 然后定义表**CREATE TABLE Student(…);**
  - 模式与表关系的结果为： **S-C-SC.student**

5.2.2基本表的修改

- **修改基本表语句：**

```
ALTER TABLE <表名>
[ADD[COLUMN]<新列名><数据类型>[完整性约束]]
[ADD<表级完整性约束>]
[DROP[COLUMN]<列名>[CASCADE|RESTRICT]]
[DROP CONSTRAINT<完整性约束名> [RESTRICT|CASCADE]]
[RENAME COLUMN<列名>TO<新列名>]
[ALTER COLUMN<列名>TYPE<数据类型>];
```

- <表名>：要修改的基本表
- DROP COLUMN子句：删除指定的列，如果指定了CASCADE短语，则自动删除引用了该列的其他对象，如果指定了RESTRICT短语且若该列被其他对象引用，RDBMS将拒绝删除该列
- DROP CONSTRAINT子句：删除指定的完整性约束
- RENAME COLUMN子句：修改列名
- ALTER COLUMN子句：修改列的数据类型

5 示例

[例3.8] 向Student表增加邮箱地址列Semail，其数据类型为字符型

```
ALTER TABLE Student ADD Semail VARCHAR(30);
```

- 注意：不论基本表中原来是否已有数据，新增加的列一律为空值

[例3.9] 将Student表中出生日期Sbirthdate的数据类型由DATE型改为字符型

```
ALTER TABLE Student ALTER COLUMN Sbirthdate TYPE VARCHAR(20);
```

- 注意：DATE类型占用19字节，所以修改为VARCHAR时长度要大于或等于19

[例3.10] 增加课程名称必须取唯一值的约束条件

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

5.2.3基本表的删除

- **删除基本表语句：**

  DROP TABLE <表名> [RESTRICT| CASCADE] ;

  - **RESTRICT**：删除表是有限制的
    - 欲删除的基本表不能被其他表的约束所引用
    - 如果存在依赖该表的对象，则此表不能被删除
  - **CASCADE**：删除该表没有限制
    - 在删除基本表的同时，相关的依赖对象也一起被删除

[例3.11] 删除**Student**表，选择**CASCADE**

```
DROP TABLE  Student  CASCADE;
```

- 基本表定义被删除，数据被删除
- 表上建立的索引、视图、触发器等一般也将被删除

5 示例

[例3.12] 删除Student表，若表上建有视图，选择RESTRICT时表不能删除；选择CASCADE时可以删除表，视图也自动删除

```
CREATE VIEW CS_Student   --在Student表上创建名为CS_Student的视图
AS
SELECT Sno, Sname, Ssex, Sbirthdate, Smajor  FROM Student
WHERE Smajor='计算机科学与技术';
```

```
DROP TABLE Student RESTRICT; --采用RESTRICT方式删除Student表
    --ERROR: cannot drop table Student because other objects depend on it
DROP TABLE Student CASCADE; --采用CASCADE方式删除Student表
    --NOTICE: drop cascades to view CS_Student
SELECT * FROM CS_Student;
    --ERROR: relation "CS_Student"does not exist
```

- 不同产品DROP TABLE处理策略比较
  - 不同RDBMS产品在遵循SQL标准的基础上，具体实现和处理策略会与标准有所差别，所以在实际应用中一定要阅读产品手册

表 1　DROP TABLE 时，SQL2011 标准与 3 个关系数据库管理系统的处理策略比较

序号	标准及主流数据库的处理方式 / 依赖基本表的对象	SQL2011 标准		KingbaseES V8		Oracle 19c		MS SQL Server 2012
		R	C	R	C	无 R	C	无 R/C
1	索引	无规定		√	√	√	√	√
2	视图	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留
3	DEFAULT，PRIMARY KEY，CHECK（只含该表的列）NOT NULL 等约束	√	√	√	√	√	√	√
4	外码 FOREIGN KEY	×	√	×	√	×	√	×
5	触发器 TRIGGER	×	√	√	√	√	√	√
6	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

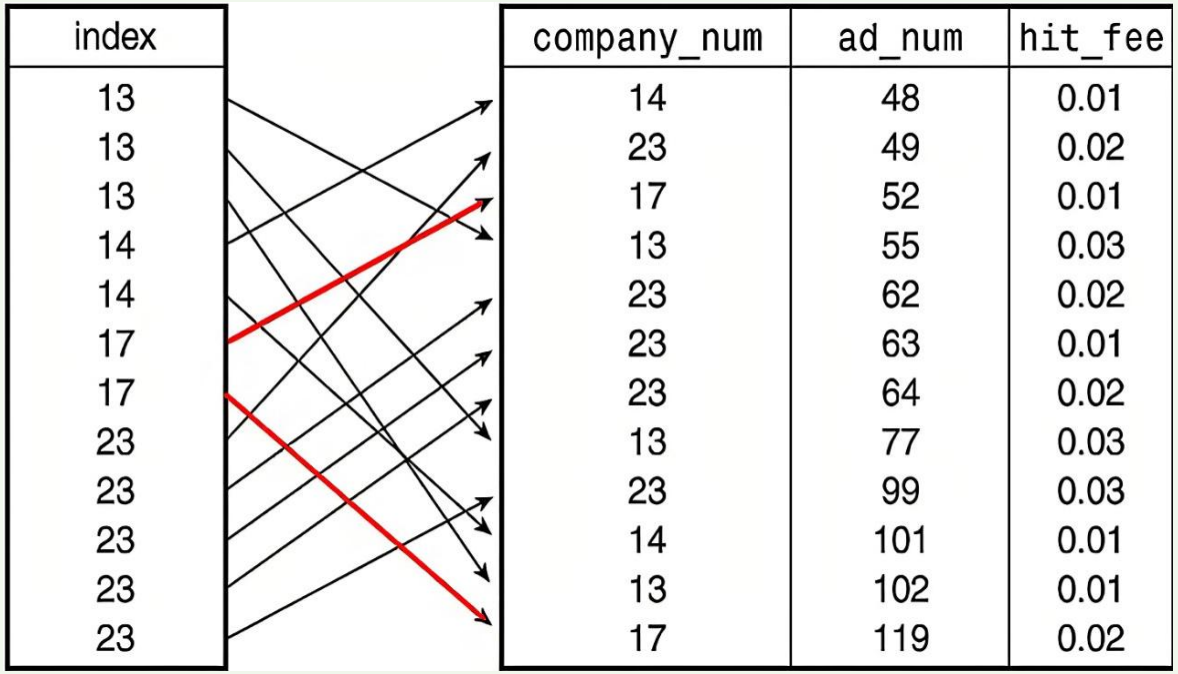
- R表示RESTRICT
- C表示CASCADE
- '×'表示不能删除基本表
- '√'表示能删除基本表
- '保留'表示删除基本表后，还保留依赖对象

5.3索引的定义与删除

- 什么是索引?
- 创建索引
- 修改索引
- 删除索引

5.3.1什么是索引?

- 索引(index)是一种提高数据检索速度的数据结构，属于内模式
  - 数据结构：search-key | pointer
  - 可选的
  - 建立在基本表上
- RDBMS常见索引类型：
  - 顺序文件上的索引
  - B+树索引
    - 具有动态平衡的优点
  - 散列(hash)索引
    - 具有查找速度快的特点
  - 位图(Bitmap)索引



index
13
13
13
14
14
17
17
23
23
23
23
23

company_num	ad_num	hit_fee
14	48	0.01
23	49	0.02
17	52	0.01
13	55	0.03
23	62	0.02
23	63	0.01
23	64	0.02
13	77	0.03
23	99	0.03
14	101	0.01
13	102	0.01
17	119	0.02

5 ~(cont.)

- 谁可以建立索引?
  - 数据库管理员(DBA)或表的属主(即建立表的人Owner)
  - RDBMS一般会自动建立以下列上的索引：PRIMARY KEY，UNIQUE
- 谁维护索引?
  - RDBMS自动完成
- 使用索引
  - RDBMS自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引

5.3.2创建索引

- **建立索引语句：**

CREATE [UNIQUE][CLUSTER] INDEX <索引名> ON<表名>(<列名>[<次序>][,<列名>[次序]]...);

- 表名：要建索引的基本表的名字
- 索引：可以建立在该表的一列或多列上，各列名之间用逗号分隔
- 次序：指定索引值的排列次序，升序：**ASC**，降序：**DESC**。默认值：**ASC**
- **UNIQUE**：此索引的每一个索引值只对应唯一的数据记录
  - 对于已含重复值的属性列不能建**UNIQUE**索引
  - 对某个列建立**UNIQUE**索引后，插入新记录时**DBMS**会自动检查新记录在该列上是否取了重复值，这相当于增加了一个**UNIQUE**约束
- **CLUSTER**：表示要建立的索引是聚簇索引
  - 一张表只能建立一个聚簇索引

5 示例

[例3.13] 为学生-课程数据库中的Student，Course，SC三个表建立索引。Student表按学生姓名升序建唯一索引，Course表按课程名称升序建唯一索引，SC表按学号升序和课程号降序建唯一索引.

```
CREATE UNIQUE INDEX Idx_StuSname ON Student(Sname);
CREATE UNIQUE INDEX Idx_CouCname ON Course(Cname);
CREATE UNIQUE INDEX Idx_SCCno ON SC(Sno ASC,Cno DESC);
```

- 不同RDBMS产品的索引语法可能不同，openGauss索引示例：

```
CREATE UNIQUE INDEX pg_job_id_index ON pg_id USING btree (job_id) TABLESPACE pg_global;
CREATE INDEX gs_asp_sampletime_index ON gs_asp  USING btree (sample_time) TABLESPACE pg_default;
```

5.3.3修改索引

- **修改索引名称语句：**

  ALTER INDEX <旧索引名> RENAME TO <新索引名>;

  – 为已建立的索引重新命名

[例3.14] 将SC表的Idx_SCCno索引名改为Idx_SCSnoCno

ALTER INDEX Idx_SCCno RENAME TO Idx_SCSnoCno;

5.3.4删除索引

- **删除索引语句：**

DROP INDEX <索引名>;

  - 删除索引时，系统会从数据字典中删去有关该索引的描述

[例3.15] 删除**Student**表的**Idx_StuSname**索引

DROP INDEX Idx_StuSname;

- 可以通过先删后建的方式实现对索引内容的修改

5.4数据字典

- 数据字典(Data dictionary)是RDBMS内部的一组系统表，它记录了数据库中所有定义信息：
  - 关系模式定义
  - 视图定义
  - 索引定义
  - 完整性约束定义
  - 各类用户对数据库的操作权限
  - 统计信息等
- RDBMS在执行SQL的数据定义语句时，就是在更新数据字典表中的相应信息
- 查询处理和查询优化时，RDBMS要根据数据字典中的信息执行处理算法和优化算法

5 openGauss的系统表

- 除了创建的表以外，数据库还包含很多系统表
  - 这些系统表包含openGauss安装信息以及在openGauss上运行的各种查询和进程信息
- 可以通过查询系统表来收集有关数据库的信息
- openGauss提供了以下类型的系统表和视图：
  - 继承自PG的系统表和视图
    - 这类系统表和视图具有PG前缀
  - openGaussl新增的系统表和视图
    - 这类系统表和视图具有GS前缀

5 openGauss系统表的查询

- 可以通过查询**openGauss**系统表来了解**openGauss**数据库的相关信息.

[例1] 在**PG_TABLES**表中查看**public**模式中包含的所有基本表

```
SELECT distinct(tablename) FROM pg_tables WHERE SCHEMANAME='public';
```

[例2] 通过**PG_USER**表查询数据库中所有用户列表及其权限

```
SELECT * FROM pg_user;   ---用户权限在字段ID(USESYSID)
```

数据定义部分结束！