NO
DATE

第3章 SQL.

3.1 概览

SQL
- 数据定义语言 DDL: 提供定义关系、删除/修改关系 (完整性)
- 数据操纵语言 DML: 查询, 从数据中插入/删除/修改元组 (Definition / Manipulation)
- 完整性 SQL DDL包含
- 视图定义 SQL DDL包含
- 非过程化 ~~事务控制~~: 定义事务的开始和结束.
- 嵌入式 SQL/动态 SQL: 到 C/Java...
- 授权: 访问权限设置

3.2 SQL数据定义 DDL.

	创建	删除	修改 (增加列)
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	(无)
索引	CREATE INDEX	DROP INDEX	(无)

- CREATE TABLE <关系名>
  (<属性名><属性类型>[NOT NULL], ...,
  <属性名><属性类型>[NOT NULL],
  完整性约束1, 完整性约束2, ...);

完整性约束: - primary key ($A_1, A_2, \cdots A_n$): 主码, 主码属性必须非空且唯一
- foreign key ($A_1, A_2, \cdots A_n$) references XXX
  声明外码
- not null: 非空

- ALTER TABLE <关系名> ADD|DROP <列名>[列类型]
- DROP TABLE <关系名>: 不仅删除关系的所有元组, 还删除关系模式

表示索引属性或属性组是主码属性,即不允许重复..

DATE

• CREATE [UNIQUE] INDEX <索引名> ON <关系名>(
<列名>[ORDER], ..., <列名>[ORDER]) [CLUSTER]

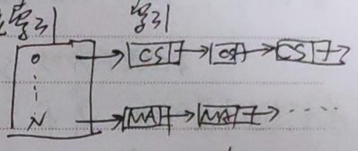
建立索引.　　ASC/DESC（递增/递减）　　　是否为聚簇索引.

例:在Student关系上以SSN属性为索引属性,建立一个聚簇索引,名字SSN-INDEX
并说明SSN是主码属性,索引按SSN值递增

CREATE UNIQUE INDEX SSN_INDEX ON Student (SSN ASC) CLUSTER

为什么要索引? 查找时要全表扫描吗?(顺序) 建立索引

能把一些具有特征的聚集起来



Simple Hash.

• DROP INDEX <索引名>

为什么要视图? 一让user看到整个逻辑不合适,安全,隐藏数据
SQL允许通过查询来定义"虚关系"一从一个或几个基本表(视图)
导出的虚关系 — 视图,本身不保存数据,仍存在基本表中.

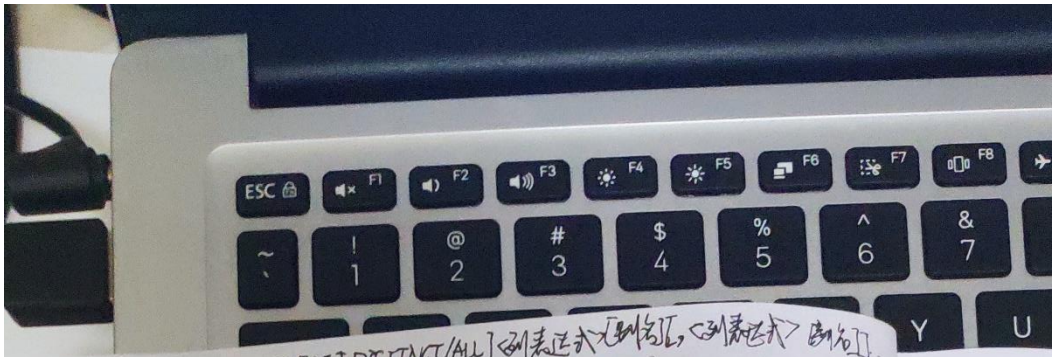
• CREATE VIEW <视图名> [(列名)[,<列名>]...)]
AS <子查询> [With CHECK OPTION]

组成视图的属性名 全部指定/全部省略　　对视图进行增删改时不得破坏谓词条件(即子查询中条件)

注: DBMS执行CREATE VIEW语句时只是把视图定义存入数据字典.
不执行其中SELECT语句. 只在对视图查询时,才按视图的定义从
基本表中将数据查出.

• DROP VIEW <视图名> [CASCADE]
如果没有CASCADE,而该视图上定义了其它视图,该语句拒绝执行
指定了CASCADE,视图及由它导出的所有视图都删除

(5). Select S#, SNAME



SELECT [DISTINCT/ALL] <列表达式>[别名],<列表达式> 别名]...
FROM <表名>[别名>][<表名>[别名>]],...
[WHERE <条件表达式>]
[GROUP BY <分组属性表> [HAVING <条件表达式>]]
[ORDER BY <列名> [ORDER],... <列名>[ORDER]];

3.3 SQL查询

3.3.1 单关系查询

Student (Sno, Sname, Ssex, Sage, Sdept)

SELECT <列表达式> FROM ...

<列表达式>中各列先后顺序可以不和表中逻辑一致，可以根据需要改变顺序

查询指定列 Select Sno, Sname From Student;

查询全部列 Select Sno, Sname, Sdept, Sage, Ssex From Student;
改变顺序

不改变顺序时 Select * From Student

查询经过计算的值：算术、字符串函数.

Select Sname, 2018-Sage From Student;

Select Sname, 2018-Sage, LOWER(Sdept) From Student;

使用列别名，可以使查询结果重命名.

Select Sname as NAME, 2018-Sage as BIRTHDAY From Student;

查询指定元组：WHERE子句；DISTINCT-消除取值重复行
ALL-不消除重复.

条件	谓词
比较	=, >, <, >=, <=, !=, <>, !<, !>, NOT+
确定范围	BETWEEN AND, NOT BETWEEN AND
确定集合	IN, NOT IN
空值	IS NULL, IS NOT NULL
多重条件	AND, OR
字符串	%, _, upper(), Caver(), like, escape

不重复

Select distinct Sno From SC.

Select Sname, Sex From Student Where Sage between 19 and 21;

Select * From Student Where Sname like "李%"

沈神说的 ← 多 零到 — 还是一个? → 是一, csy说的

通配符%: 替代多个字符. "_"仅代表一个

ESCAPE短语: 当用户要查询的字符串本身就含有%或_时,要使用 ESCAPE '<换码字符>' 短语对通配符进行转义

Eg: 查询以"DB_"开头,且倒数第3个字符为i的课程的全部情况

```
SELECT * FROM Course WHERE Cname LIKE 'DB\_%i__'
                  ESCAPE '\';
```

```
Select * From Student Where sdept in('MA', 'CS').
```

查询结果排序. ORDER BY子句. 可按一个或多个属性列排序 → 先按第一个排,第一个相同时,按第二个排

ASC升. DESC降. 缺省值升序

→ 用在SELECT [表达式] FROM中, Having中

聚集函数. COUNT([DISTINCT|ALL] *) 统计元组个数.

计算由查询中from语句和where子句所创建的关系中的元组个数.

COUNT([DISTINCT|ALL] <列名>) 统计一列中值的个数(不计NULL)

SUM([DISTINCT|ALL]<列名>) 计算一列值总和

AVG/MAX/MIN   Select count(*) From student.

查询结果分组. Group By (细化聚集函数作用对象)

Eg 查询各课程号与相应的选课人数. Select Cno, count(Sno). From SC.
GROUP BY Cno.

查询选修了4门课以上学生学号

Select Sno From SC Group by Sno Having count(*) >4

注意/需要谨记: 任何没出现在group by中的属性如果出现在select子句中的话,只能出现在聚集函数内部

Eg Select dept_name, ID, avg(salary) from instructor
group by dept_name ✗

(4) Select S#, SNAME From S Where SNAME like '

(5) Select S#, SNAME From S Order by S#;

NO
DATE

3.3.2 连接查询 一同时涉及多个表的查询
连接条件-用来连接两个表的条件称为连接条件/谓词
·[<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>
比较运算符: =、>、<、>=、<=、!=.
·[<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>

Select Sname From student, sc Where Student.Sno=SC.Sno and
Cno='2' and grade>90;

3.3.3 嵌套查询 一个SELECT-FROM-WHERE语句称为一个查询块.
将一个查询块嵌套在另一个查询块的WHERE/FROM/HAVING短语的条件中的查询称为嵌套查询. 由里向外处理, 子查询的结果用于父查询

Select Sname From Student Where Sno In
(Select Sno From SC Where Cno='2');

子查询不能使用 Order By 子句

有些嵌套查询可以用连接运算替代
查询与"刘一"在同一系的学生
Eg: 自身连接 Select S2.Sno, S2.Sname, S2.Sdept
From Student as S1, Student as S2
Where S1.dept=S2.Sdept AND
S1.name='刘一'

相关子查询: 子查询的查询条件依赖于外层父查询的某个属性值.
Eg 查询选修了课学生姓名
Select Sname From Student as S
Where exists(Select * From SC Where Sno=S.Sno and
Cno='1');

使用存在量词后，若内层查询结果非空，则外层的where子句返回真值，否则假

Eg. From子句中的子查询

查询系平均工资超过42000的系名与平均工资

```
Select dept_name, avg_salary
From ( Select dept_name, Avg(salary) as avg_salary
       From instructor
       Group by dept_name)
Where avg_salary > 42000;
```

3.3.4 集合查询

并 UNION 交 INTERSECT 差 EXCEPT

参加集合操作的各结果表列数必须相同；对应的数据类型也必须相同

系统自动去重

Eg：查询CS系或年龄大于19的学生

```
(Select * From Student Where Sdept='CS') UNION
(Select * From Student Where Sage>19);
```

定义视图 CREATE VIEW <视图名> [(<列名>[, <列名>]...)]

AS <子查询>

[WITH CHECK OPTION];

NO
DATE

3.4 SQL语言数据更新

·插入单个数据
INSERT INTO <表名>[(<属性列1>[,<属性列2>...)]
VALUES(<常量1>[,<常量2>]...)

插入子查询结果：
INSERT INTO <表名>[(<属性列1>[,<属性列2>...)]
子查询;

对每个系，求学生平均年龄，把结果存入数据库
CREATE TABLE Deptage(Sdept CHAR(15),
Avgage SMALLINT);
INSERT INTO Deptage(Sdept, Avgage)
SELECT Sdept, AVG(Sage) FROM Student GROUP By Sdept;

·修改数据
UPDATE <表名> SET <列名>=<表达式>[,<列名2>=<表达式>]...
[WHERE <条件>];
SET指定，要修改的列，修改后的取值

·删除数据(元组)
DELETE FROM <表名> [WHERE <条件>];

NO
DATE

SQL事务 - Commit 提交当前事务. Rollback 回滚到事务第一条语句之前

SQL触发器 一段一条语句，当对数据修改时自动被执行.
一指明什么条件下执行触发器 十指明触发器的动作
Create trigger check1 after insert on SC
referencing new row as nrow
for each row
when (nrow.cno not in (select cno from course))
begin
rollback;
end.

SQL授权 select/insert/update/delete.
GRANT <privilege list> ON <object> TO <user ID list>.
[WITH GRANT OPTION].

收回权限
REVOKE <privilege list> ON <object> FROM <user ID list>
[Restrict/Cascade]
防止级联收回 支持级联收回(默认)