**SQL的数据操纵功能**

·插入（Insertion）·更新（Updates）·删除（Deletion）

**插入数据**

插入子查询结果：将一个子查询结果集插入到表中，其语句的格式为：

**INSERT INTO** <表名>[(<属性列1>[，<属性列2>] [ ,...n ]) ]

**VALUES** (<属性值1>[，<属性值2>] [ ,...n ]

**单条数据插入**

使用INSERT INTO 语句插入数据至表AGENTS 中。

**insert into** agents(aid,aname,city,percent)

**values** ('a11', 'smith', 'new york',6)

向agents表插入一条记录。

**insert into** agents values ('a12', 'tom', 'new york',6) 只写了表名没有写属性名说明是对表中所有

insert into agents values ('a13', 'sam', null,5) 属性都插入记录，注意顺序必须与表中的顺序一致

**多条数据插入**

创建一个关系cust1，其关系模式与customers完全一样，将关系customers中的**所有元组**插入到关系cust1中去。

CREATE TABLE cust1 **LIKE** customers; //两个表的数据结构完全一样

INSERT INTO cust1 **SELECT \* FROM customers;**

**删除数据**

要从一个表中删除记录，使用DELETE命令。删除语句的命令格式为：

DELETE **FROM** <表名> [**WHERE**<条件>]

删除数据表agents中的居住在“New York”的所有代理商的记录。

delete **from** agents where city = 'new york’;

删除所有没有人订购的产品

delete **from** products where pid **not in** (select pid from orders)；

**更新功能**

UPDATE <表名>

SET <列名1>=<表达式1>[，<列名2>=<表达式2>][ ,...n ] [WHERE <条件>];

修改表AGENTS中的数据，把“a01”的“percent”值改成6。

**update** agents **set** percent = 6 **where** aid='a01’;

将所有订货总金额超过2000的顾客的折扣率增加10%。

update customers set discnt=discnt\*1.1 where **cid in**

(select **cid** from orders group by **cid** having sum(dollars)>2000);

DBMS在执行修改语句时会检查修改操作是否破坏表上已定义的完整性规则；

实体完整性；主键不允许修改；用户定义的完整性；NOT NULL约束； UNIQUE约束；值域约束

将所有余额超过$10000美元的账户增加6%，所有其他账户增加5%。

可以分成两步（更新顺序很重要，先更新大于10000的，防止小于10000的更新后大于10000又进行一次更新）

update account set balance = balance \* 1.06 where balance>10000

update account set balance = balance \* 1.05 where balance <=10000

或者一步进行：判断语句

update account set balance = case

**when** balance <= 10000

**then** balance \*1.05

**else** balance \* 1.06

**end**

**视图**

视图是从一个或几个基本不（或视图）导出一个虚拟表，数据库只存放试图的定义

视图的优点：提高安全性、简化操作、逻辑数据独立性

**视图的创建和使用**CREATE VIEW

CREATE VIEW <视图名>[(<列名>[,<列名>][ ,…..n ])]

AS<子查询语句>[WITH CHECK OPTION

创建一个视图custp01，列出订购了产品p01的顾客编号姓名、产品编号、产品数量和金额

create view custp01

as select c.cid,cname,pid,gty,dollars fromcustomers c, orders o

where c.cid = o.cid and o.pid ='p01’

视图的使用：select \* from custp01

**视图的更新**

视图的更新最终转化为对基本表的更新更新包括：**插入、删除、修改**

定义视图时加上**WITH CHECK OPTION**子句，当在视图上增删、改数据时，DBMS会进一步检查**视图定义**中的条件，若不满足条件（比如超过了视图定义时候的限制），则拒绝执行该操作。

创建agentsales视图，包含所有下过订单的代理商的aid值以及他们的销售总额。

create view agentsales(aid, totsales) as

select aid, **sum**(dollars) from orders **group by aid;【使用了聚合函数sum导致创建的试图是不可更新的】**

update agentsales set totsales = totsales + 1000

where aid ='a03'; **/\*illegal syntax\*/**

**嵌套视图**

创建视图IS\_S1

CREATE VIEWIS S1(Sno, Sname, Grade) AS

SELECT Student.Sno, Sname, Grade FROM Student, SC

WHERE Sdept ='IS' AND Student.Sno = SC.Sno AND SC.Cno ='1’.

创建视图IS\_S2

CREATE VIEW IS\_S2 AS

SELECT Sno, Sname, Grade FROM IS\_S1

WHERE Grade >= 90;

**建立索引**

索引是一个单独的、物理的**数据库结构**，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单。(指向数据记录的物理地址)

般来说索引本身也很大，不可能全部存储在内存中，因此索引往往以索引文件的形式存储在磁盘上。

索引的优点：**加快查询速度**

索引的缺点：①需要**占用物理空间**

②创建索引和维护索引要**耗费时间**，这种时间随着数据量的增加而增加

③当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，**降低了数据的维护速度**。

**索引分类**

MySQL的索引包括**普通索引、唯一性索引、全文索引、单列索引、多列索引和空间索引**等

·从**功能逻辑**上说：是普通索引、唯一索引、主键索引、全文索引

·按照**物理实现方式**：聚簇索引和非聚簇索引

·按照**作用字段个数**：单列索引和联合索引

（1）聚集索引(Primary index)

存储的数据按照**索引的顺序**存储称为聚集索引。一个表只能包含一个聚集索引。但该索引**可以包含多个列**。

（2）非聚集索引(Secondaryindex)

非聚集索引表示数据存储在一个地方，索引存储在另一个地方（数据和索引分开存储），索引带有指针指向数据的存储位置。索引顺序与数据物理顺序不同，索引带有**指针**指向数据的存储位置。

**一般而言，应先创建聚集索引再创建非聚集索引**

**MySQL的默认存储引擎为InnoDB**

稠密索引：每个搜索码值都有一个索引项

稀疏索引：只为搜索码的某些值建立索引项

**索引的创建修改和删除**

**建立索引CREATE INDEX**

**CREATE** [UNIQUE |FULLTEXT|SPATIAL] **INDEX** index\_name

**ON** table\_name (col name[length],...) [ASC | DESC]

**ALTER TABLE** table name **ADD** [**UNIQUE** | **FULLTEXT** |**SPATIAL**] [**INDEX** | **KEY**]

[index\_name](col \_name[length],...) [ASC |DESC]

**索引的删除**

DROP INDEX<索引名> ON<表名>;

ALTER TABLE table name **DROP INDEX**<索引名>;

**查看索引**

SHOW INDEX FROM<表名>;

**适合创建索引的情况：**

1.主键自动建立唯一索引

2.频繁作为查询条件的字段应该创建索引

3.查询中与其它表关联的字段，外键关系建立索引(多表查询可以快速定位)

4.频繁更新的字段不适合创建索引;(因为每次更新不单单更新记录还会更新索引)

5.经常出现在Where子句中字段应该建立索引

6.单列/组合索引的选择问题?(在高并发下倾向创建组合索引)

7.查询中排序的字段，排序字段若通过索引去访问大大提高排序速度

8.查询中统计或者分组字段

不需要创建索引的情况：

1.在where中使用不到的字段，不要设置索引;

2.表记录太少;

3.**经常更新**的表;

4.有**大量重复数据**的列上不要建立索引。

**数据控制功能DCL**

**授子权限**

GRANT语句：对指定操作对象的指定操作权限授予指定的用户。

GRANT <权限>[，<杈限>][...n]

[ON<对象类型><对象名>]

TO <用户>[,<用户>][ ....n][WITH GRANT OPTION];

grant select on Agents to U1, U2, U3

**创建用户**WinTest，其口令为temp。

CREATE USER WinTest@localhost IDENTIFIED BY 'temp';

**授予用户**WinTest对数据库Sales的AGENTS表的**插入、更新**权限。

**GRANT** insert, update **on** table agents **to** WinTest@localhost;

【例5.75】授予用户WinTest对数据库Sales的CUSTOMERS表的列cid、cname的查询权限。

**GRANT select(cid,cname)** **on** table customers **to** WinTest@localhost;

**授子角色权限**

GRANT<权限>[,<权限>]

[on<对象类型><对象名>]

to<角色>[,<角色>][WITH GRANT OPTION]

**创建角色**Accounting.

CREATE ROLE 'Accouting'@'localhost';

**授子角色权限**

GRANT insert, update on table orders to Accouting;

**分配给用户所需角色**，如把用户Jill加入到Accounting角色:

GRANT 'Accouting'@"localhost' to Jil'@'%';

把查询视图cust1的**全部权限**授予**角色**WinTest。

GRANT **ALL PRIVlLlEGES** on table cust1 to 'wintest'@'localhost';

Accounting 角色，Jill 是Accounting 的成员，Jack 不是Accounting 的成员，授予Accounting角色查询Agents权限。

/\* User Jack \*/

GRANT SELECT ON Agents TO Accounting **WITH GRANT OPTION;**

/\* User Jill \*/

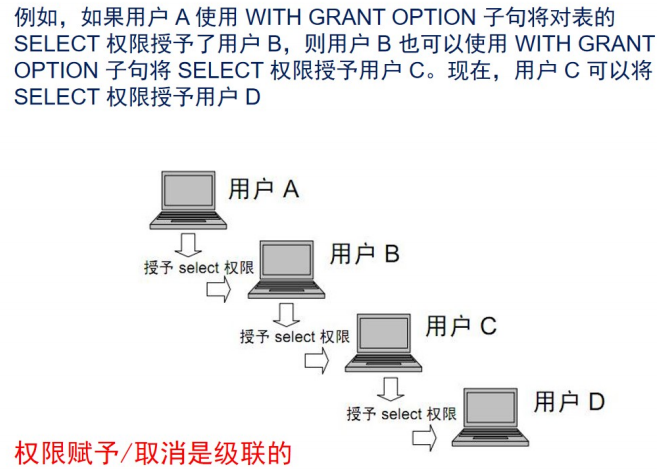
GRANT SELECT ON Agents To Jack **AS** Accounting

Jill 必须用 AS 子句来获得 Accounting 角色的授予权限。

**收回权限**

授予的权限可以由DBA或其他授权者用REVOKE语句收回

REVOKE<杈限>[，<杈限>][ ,...n]

[ON<对象类型><对象名>

FROM<用户>[，<用户>][ ,…..n ];

revoke select **on** branch **from** U1, U2, U3

**权限赋予/取消是级联的**

**视图机制保证安全性**

**视图机制与授权机制配合使用**

用户User1只能查询orders表顾客c001的订购信息。

①建立视图

create view view\_orders as

select \* from orders where cid='c001’

②对视图定义存取权限

grant select on view\_orders to user1