视图

一、 创建视图

1. 使用表中所有的字段创建简单视图

示例1: 查询 `product` 表中类别编号为 `11` 的所有记录，创建视图 `view_product`

2. 查看视图结构和定义

示例2：使用 `DESC[RIBE]` 查看视图 `view_product` 的结构

示例3: 使用 `SHOW CREATE VIEW | TABLE` 查看 `view_product` 的定义

3. 指定表中的某些字段构建视图

示例4：创建视图 `view_product2` ，包含 `product` 表中的 `product_id, product_name, product_place, subsort_id` 。

4. 为分组查询（ `GROUP BY` ）构建汇总视图

示例5：创建视图 `sum_product` ，包含 `product` 中 `product_place` 和各产地对应的产品数

5. 为多表查询建立视图

示例6：创建视图 `view_sort_product` ，包含类别名称和对应的产品数量

6. 视图字段的重命名

示例7: 创建视图 `sum_product` ，包含 `product` 中的 `product_place` ，各产地对应的产品数量，并命名为 `num_product`

二、查询视图

示例7: 查看视图 `sum_product3` 中的所有记录

三、修改或重新定义视图

1. 利用 `CREATE OR REPLACE VIEW` 重新定义视图

示例9：重新定义视图 `view_product` ，包含 `product_id, product_name, product_place, sort_id, subsort_id` 。

2. 使用 `ALTER` 语句修改视图定义

示例10: 修改视图结构 `view_product` ,包含 `product_id, product_name, product_place`

四、通过视图操作基本表的记录

1. 更新 `UPDATE`

示例11：通过 `view_product` 把 `product_place` 为 `'国产'` 的产品记录的对应值更改 `'中国'`

2. 插入 `INSERT`

示例12：通过 `view_product` 插入一条 `product_id = '12345', product_name='3d打印机', product_place='上海'` 的记录

3. 删除 `DELETE`

示例13：通过 `view_product` 删除 `product_name` 为 `'3d打印机'` 的记录

五、删除视图

示例14: 删除视图 `view_product`

补充

1. `algorithm: merge | temptable`
2. `with check option: cascaded | local`

练习

视图

```
USE purchase;
SET SQL_SAFE_UPDATES=0;
```

一、 创建视图

- **视图**是由一个或多个表（视图）导出来的虚拟表，其结构和数据都依赖于基本表。它不占用实际的存储空间，只是在数据字典中存储它的定义信息。通过视图不仅可查看基本表中的记录，也可查询、修改和删除基本表中的数据。
- 为什么需要使用视图?

> - 简化查询。例如基于一些多表或分组复杂查询构建视图。
> - 安全性。过滤掉一些敏感信息，如 `member` 表中的 `user_passw` 。
> - 逻辑数据独立性。

- 创建语法格式：

```
CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] VIEW view_name
[(column_list)]
AS
SELECT查询语句
[WITH [CASCADED | LOCAL | CHECK OPTION] ];
```

> `CREATE` ： 表示创建视图的关键字。
> `OR REPLACE` ： 如果已经有相同名称的视图，则替换已有视图定义。
> `ALGORITHM` ： 可选，表示创建视图选择的算法。
>
> - `UNDEFINED` ： 表示 `MySQL` 将自动选择所要使用的算法。一般偏向于 `merge` 。
> - `MERGE` ： 在涉及视图的SQL语句中，将视图定义取代查询语句的对应部分；在执行效率上比 `temptable` 更加高效。
> - `TEMPTABLE` :在涉及视图的SQL语句中，将视图结果存入临时表，然后在临时表的基础上再执行语句；可以不在视图定义涉及的表上加锁，利于并发。
>
> `view_name` ： 表示要创建的视图名称。
> `column_list` ： 可选，表示属性清单。
> `AS` ： 表示指定视图要执行的操作。
> `SELECT_statement` ： 表示从某个表或视图中查出满足条件的记录，并导入视图中。
> `WITH CHECK OPTION` ： 可选，表示创建视图时要保证在该视图的权限范围之内。
>
> - `CASCADED` ： 需要满足跟该视图有关的所有相关视图和表的条件，该参数为默认值。
> - `LOCAL` ： 可选。表示创建视图时，只要满足该视图本身定义的条件即可。

1. 使用表中所有的字段创建简单视图

示例1: 查询 `product` 表中类别编号为 `11` 的所有记录，创建视图 `view_product`

```
CREATE VIEW view_product
AS
SELECT * FROM product
WHERE sort_id = 11;

show tables;

-- 查看视图view_product的前10条记录
SELECT *
FROM view_product LIMIT 10;
```

2. 查看视图结构和定义

- `DESC[RIBE] <view_name>` 可查看视图的结构

示例2：使用 `DESC[RIBE]` 查看视图 `view_product` 的结构

```
DESC view_product;
```

- `SHOW CREATE TABLE | VIEW <view_name>` 可查看视图的 `DDL`

示例3: 使用 `SHOW CREATE VIEW | TABLE` 查看 `view_product` 的定义

```
SHOW CREATE VIEW view_product;
-- 或者
SHOW CREATE TABLE view_product;
```

3. 指定表中的某些字段构建视图

示例4：创建视图 `view_product2`，包含 `product` 表中的 `product_id, product_name, product_place, subsort_id`。

```
CREATE VIEW view_product2
AS
SELECT product_id, product_name, product_place, subsort_id FROM product;

SELECT * FROM view_product2 LIMIT 10;
```

4. 为分组查询（`GROUP BY`）构建汇总视图

示例5：创建视图 `sum_product`，包含 `product` 中 `product_place` 和各产地对应的产品数

```
CREATE VIEW sum_product
AS
SELECT product_place, COUNT(product_id)
```

```
FROM product
GROUP BY product_place;

desc sum_product;

SELECT * FROM sum_product;

-- 或则

SELECT sort_name, `COUNT(product_id)` FROM sum_product;

```

5. 为多表查询建立视图

示例6：创建视图 `view_sort_product` ，包含类别名称和对应的产品数量

```
CREATE VIEW view_sort_product
AS
SELECT sort.sort_name, COUNT(product.product_id) AS num_product
FROM product JOIN sort ON product.sort_id = sort.sort_id
GROUP BY sort.sort_name;

SELECT * FROM view_sort_product;
```

6. 视图字段的重命名

如果未对视图中的字段进行重命名，则默认使用查询语句中的字段名称。若未对查询中的表达式重命名，则在对该字段进行查询时需加上反引号。

示例7: 创建视图 `sum_product` ，包含 `product` 中的 `product_place` ，各产地对应的产品数量，并命名为 `num_product`

```
CREATE VIEW sum_product2(product_place, num_product)
AS
SELECT product_place, COUNT(product_id)
FROM product
GROUP BY product_place;

desc sum_product2;

-- 或者
CREATE VIEW sum_product3
AS
SELECT product_place, COUNT(product_id) AS num_product
FROM product
GROUP BY product_place;

desc sum_product3;

SELECT * FROM sum_product3;
```

二、查询视图

视图具有与表相同的查询语法:

```
SELECT * | 视图字段列表
FROM 视图名称
WHERE 条件表达式;
```

> 等价于在AS后查询结构的基础上再进行筛选

示例7: 查看视图 `sum_product3` 中的所有记录

```
SELECT *
FROM sum_product3;
```

三、修改或重新定义视图

1. 利用 `CREATE OR REPLACE VIEW` 重新定义视图

示例9：重新定义视图 `view_product`，包含 `product_id, product_name, product_place, sort_id, subsort_id`。

```
CREATE OR REPLACE VIEW view_product3
AS
SELECT product_id, product_name, product_place, sort_id, subsort_id
FROM product;

SHOW TABLES;
```

2. 使用 `ALTER` 语句修改视图定义

语法:

```
ALTER [ALGORITHM = {UNIDIFIED | MERGE | TEMPTABLE}]
VIEW view_name [(column_list)]
AS SELECT查询语句
[WITH [CASCADE | LOCAL] CHECK OPTION];

注意：被修改的视图必须要存在
```

示例10: 修改视图结构 `view_product`,包含 `product_id, product_name, product_place`

```sql
ALTER VIEW view_product
AS
SELECT product_id, product_name, product_place
FROM product;
```

四、通过视图操作基本表的记录

注意，当视图包含以下结构时，则不能对视图执行插入、更新或删除操作不能执行：

> (1) 包含基本表中被定义为非空的列
> (2) `select` 语句后的字段列表中使用了数学表达式
> (3) `select` 语句后的字段列表中使用了聚合函数
> (4) `select` 语句使用了 `DISTINCT, TOP, GROUP BY, UNION, UNION ALL` 或 `HAVING` 子句
> (5) `algorithm=temptable` 的视图
> (6) 对基础表的某一列有多次引用

1. 更新 `UPDATE`

示例11：通过 `view_product` 把 `product_place` 为 `'国产'` 的产品记录的对应值更改 `'中国'`

```sql
UPDATE view_product
SET product_place = '中国'
WHERE product_place = '国产';

SELECT *
FROM view_product
WHERE product_place='中国' LIMIT 10;

SELECT *
FROM product
WHERE product_place='中国' LIMIT 10;
```

2. 插入 `INSERT`

示例12：通过 `view_product` 插入一条 `product_id = '12345', product_name='3d打印机', product_place='上海'` 的记录

```sql
INSERT INTO view_product(product_id, product_name, product_place)
VALUES ('12345', '3d打印机', '上海');

SELECT *
FROM view_product
WHERE product_name='3d打印机';

SELECT *
FROM product
WHERE product_name='3d打印机';
```

3. 删除 `DELETE`

示例13：通过 `view_product` 删除 `product_name` 为 `'3d打印机'` 的记录

```sql
DELETE FROM view_product
WHERE product_name = '3d打印机';

SELECT *
FROM view_product
WHERE product_name='3d打印机';

SELECT product_id, product_name, product_place
FROM product
WHERE product_name='3d打印机';
```

五、删除视图

语法:

```sql
DROP VIEW 视图名称;
```

示例14: 删除视图 `view_product`

```sql
DROP VIEW if exists view_product;
SHOW TABLES;
```

补充

1. `algorithm: merge | temptable`

例如：对于 `view_product` ，如果指定为 `merge` 算法

```
CREATE OR REPLACE ALGORITHM=MERGE VIEW view_product
AS
SELECT * FROM product
WHERE sort_id = 11;
```

则对其查询会发生对应定义的替换。如果我们要在该视图上查询价格大于1000的产品，即

```
SELECT * FROM view_product WHERE price > 1000;
```

以上语句被替换为

```
SELECT *
FROM product
WHERE sort_id = 11 AND price > 1000;
```

对于 `view_product` ，如果指定为 `temptable` 算法，则以上语句被转换为

```
SELECT *
FROM (SELECT *
      FROM view_product WHERE sort_id=11) a
WHERE price > 1000;
```

2. with check option: cascaded | local

`WITH CHECK OPTION` 用于规范可更新视图的更新行为，以限定视图定义中 `where` 中表达式的作用范围。它有两个选项，如果不定义，则默认为 `cascaded` ，即在执行视图插入或更新操作时，检查该视图定义中的 `where` 以及该视图依赖的所有视图定义中的 `where` 条件，如果插入的行不满足其中任意一个，则插入不成功。如果设定为 `local` 选项，则只检查当前视图定义中的 `where` 条件。

- 加与不加 `with check option` 的对比

```
-- 不加with check option选项
create or replace view view_product
as
select * from product where sort_id = 11;

start transaction;
insert into view_product(product_id, sort_id)
values (9999, 12); -- 执行成功，即使sort_id为12不满足view_product的定义

select * from product where product_id = 9999; -- 可以查看到相关记录
select * from view_product where product_id = 9999; -- 无结果，因为sort_id不为11
rollback;

-- 加with check option选项
create or replace view view_product
as
select * from product where sort_id = 11
with check option; -- 默认为cascade选项

```

```
start transaction;
insert into view_product(product_id, sort_id)
values (9999, 12); -- 执行不成功
rollback;
```

- `with local check option` 和 `with cascaded check option` 的对比

```
-- local
create or replace view view_product
as
select * from product where sort_id = 11;

create or replace view view_view_product
as
select * from view_product where price > 1000
with local check option;

start transaction;
insert into view_view_product(product_id, price, sort_id)
values (9999, 2000, 12); -- 执行成功, 虽然违背view_product的where条件

select * from product where product_id = 9999; -- 有结果
select * from view_product where product_id = 9999; -- 无结果
select * from view_view_product where product_id = 9999;  -- 无结果

rollback;

-- cascaded
create or replace view view_product
as
select * from product where sort_id = 11;

create or replace view view_view_product
as
select * from view_product where price > 1000
with cascaded check option;

start transaction;
insert into view_view_product(product_id, price, sort_id)
values (9999, 10000, 12); -- 出错, cascade选项级联检查view_product的条件
rollback;

```

练习

(1) 创建 `view_member` 视图, 包含 `member` 表中的 `user_name, sex, email` 等字段，并用查看视图结构和定义。

(2) 创建 `view_sort` 视图，包含 `sort` 表中的 `sort_name` ， 以及其对应的子类别的数量，并命名为 `num_subsort`

(3) 在 `view_sort` 视图查询类别名称中有 `'办公'` 两个字的所有记录

(4) 利用 `CREATE OR REPLACE VIEW` 或者 `ALTER VIEW` 修改视图 `view_member`，使其包含 `member` 表中的 `user_name, sex, email, address, phone` 等字段

() 通过 `view_memeber` 更新 `user_name` 为 `'饿狼'` 的记录的 `true_name` 为 `'胡颖'`

(6) 通过 `view_memeber` 插入一条记录， `user_name` 为 `'风清扬'`，`true_name` 为 `'张三丰'`，`sex` 为 `'女'`

() 通过 `view_memeber` 删除 `user_name` 为 `'风清扬'` 的记录

(8) 删除视图 `view_memeber`