数据库实验指导书

实验三 DBMS应用

目录

1 实验目的

1、 理解视图、触发器、约束、存储过程和函数的基本概念，掌握它们的用法；
2、 能结合实例设计合理的视图、触发器和存储过程；
3、 结合实验加深对数据库完整性和安全性的理解。

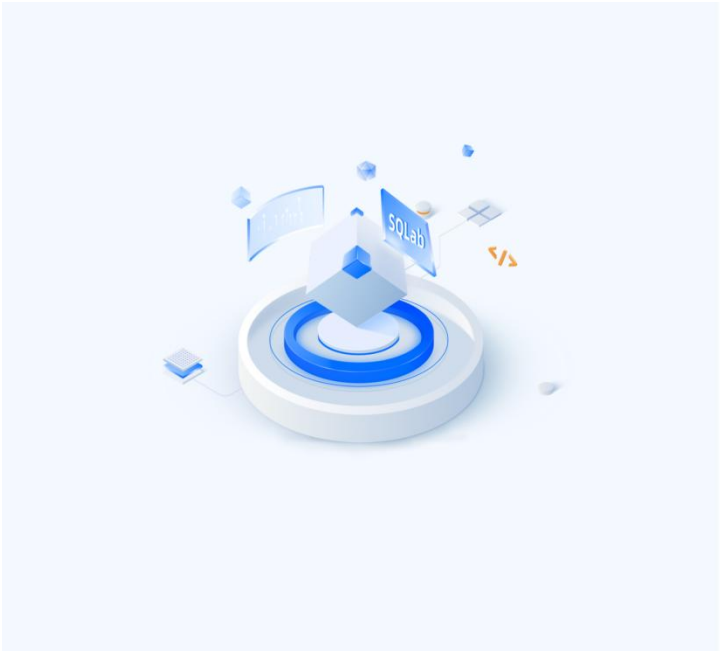
2 实验环境

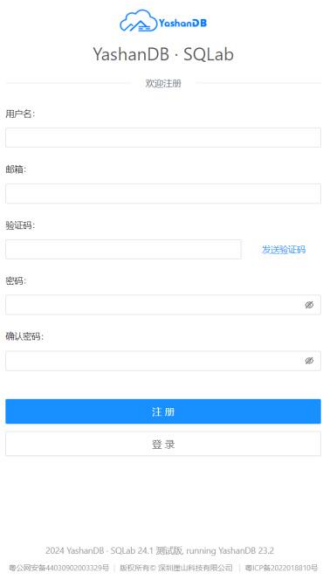
Windows 10 操作系统、SQLab平台。

2.1. 实验步骤

用户注册（如已注册，此步骤可忽略）

1. 访问SQLab地址（https://sqlab.yashandb.com）
2. 点击注册按钮，跳转到用户注册页面
3. 填写用户信息，点击发送验证码，然后查看邮箱中的验证码并填写
4. 点击注册按钮，完成注册，并成功登陆SQLab

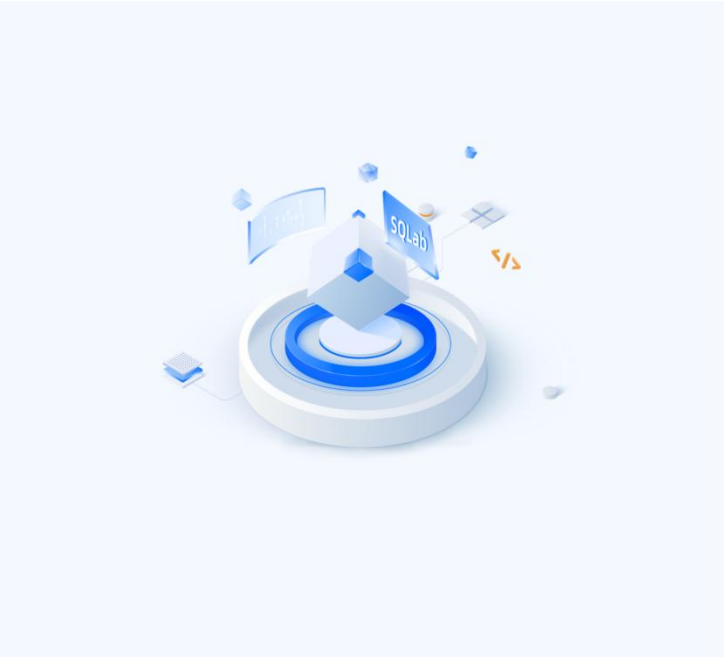




用户登录

1. 访问SQLab地址（https://sqlab.yashandb.com）

2. 填写已注册的邮箱和密码
3. 点击登录按钮，成功登陆SQLab



使用手册

1. 点击右上角如下图标，查看使用手册，可以查看完整的平台功能如何使用。

查看用户及对象

依次点击“崖山实验室”—>“我的对象”—>“SAKILA”，即可查看到sakila用户下所有的对象。

2.1.1关于视图

1、 回答问题：

（1） 分析以下 3 个视图，回答以下问题：

```
SELECT DBMS_METADATA.GET_DDL('VIEW', 'ACTOR_INFO','SAKILA') DDL FROM DUAL;
SELECT DBMS_METADATA.GET_DDL('VIEW', 'FILM_LIST','SAKILA') DDL FROM DUAL;
SELECT DBMS_METADATA.GET_DDL('VIEW', 'SALES_BY_FILM_CATEGORY','SAKILA') DDL FROM DUAL;
```

视图名	关联表	作用
actor_info		
film_list		
sales_by_film_category		

（2） 创建视图分别执行以下 3 句 SQL 语句：

创建视图：

```
CREATE OR REPLACE VIEW sakila.customer_simple_view AS
SELECT customer_id,
       first_name,
       last_name,
       email,
       address_id,
       active,
       create_date,
       last_update
FROM customer;
```

执行SQL语句：

```
(1)update sakila.customer_simple_view set email = 'newemail@example.com' where customer_id = 1;
(2)update sakila.staff_list set "zip code" = '518055' where ID   = '1';
(3)update sakila.film_list set price = 1.99 where FID = '1';
```

截图执行结果，并分析一下视图在什么情况下可以进行 update 操作，什么情况下不能？MYSQL和崖山数据库有什么区别？

2.1.2关于触发器

1、 回答问题：

（1） 触发器rental_date建在哪个表上？这个触发器实现什么功能？在这个表上新增一条数据，验证一下触发器是否生效。（截图语句和执行结果）

（2） 触发器 del_film 建在哪个表上？这个触发器实现什么功能？在这个表上删除一条记录，验证一下触发器是否生效。（截图语句和执行结果）

（3） 现在有这样一个建立触发器的语句：

```
CREATE OR REPLACE TRIGGER payment_date
BEFORE INSERT ON payment
FOR EACH ROW
BEGIN
   :NEW.payment_date := SYSTIMESTAMP;
END;
/
```

此时需要迁移payment表历史记录，请同学们思考，这个触发器是否会生效？如果生效的话，有什么办法可以避免迁移记录的payment_date被修改？

2.1.3关于约束

现有rental表的建表语句如下：

```
CREATE TABLE SAKILA.RENTAL
(RENTAL_ID BIGINT DEFAULT SAKILA.SEQ_RENTAL_RENTAL_ID.NEXTVAL NOT NULL ,
RENTAL_DATE TIMESTAMP(6) NOT NULL ,
INVENTORY_ID INTEGER NOT NULL ,
CUSTOMER_ID INTEGER NOT NULL ,
RETURN_DATE TIMESTAMP(6),
STAFF_ID SMALLINT NOT NULL ,
LAST_UPDATE TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP NOT NULL ,
CONSTRAINT FK_RENTAL_CUSTOMER FOREIGN KEY (CUSTOMER_ID) REFERENCES SAKILA.CUSTOMER (CUSTOMER_ID) ,
CONSTRAINT FK_RENTAL_INVENTORY FOREIGN KEY (INVENTORY_ID) REFERENCES SAKILA.INVENTORY (INVENTORY_ID) ,
CONSTRAINT FK_RENTAL_STAFF FOREIGN KEY (STAFF_ID) REFERENCES SAKILA.STAFF (STAFF_ID) ,
CONSTRAINT IDX_RENTAL_RENTAL_DATE_INVENTORY_ID_CUSTOMER_ID UNIQUE (RENTAL_DATE,INVENTORY_ID,CUSTOMER_ID)
PRIMARY KEY (RENTAL_ID)
) ;
```

回答问题：

（1） rental表上建了哪几种约束？这些约束分别实现什么功能？

约束类型	功能

（2） 图中的ON DELETE RESTRICT 和ON UPDATE CASCADE 是什么意思？

2.1.4关于存储过程

观察 SAKILA用户里面的 rewards_report 存储过程：

PROCEDURE

```
PROCEDURE        rewards_report (
    min_monthly_purchases IN NUMBER,
    min_dollar_amount_purchased IN NUMBER,
    count_rewardees OUT NUMBER
) AUTHID CURRENT_USER IS last_month_start DATE;

last_month_end DATE;

TYPE customer_id_table IS TABLE OF customer.customer_id % TYPE INDEX BY PLS_INTEGER;

tmp_customers customer_id_table;

BEGIN

/* Some sanity checks... */
IF min_monthly_purchases = 0 THEN DBMS_OUTPUT.PUT_LINE(
    'Minimum monthly purchases parameter must be > 0'
);

RETURN;

END IF;

IF min_dollar_amount_purchased = 0.00 THEN DBMS_OUTPUT.PUT_LINE(
    'Minimum monthly dollar amount purchased parameter must be > $0.00'
);

RETURN;

END IF;
```

```
/* Determine start and end time periods */
last_month_start := ADD_MONTHS(TRUNC(SYSDATE, 'MM'), -1);
last_month_end := LAST_DAY(last_month_start);

/*
    Find all customers meeting the
    monthly purchase requirements
*/
SELECT
    p.customer_id BULK COLLECT INTO tmp_customers
FROM
    payment p
WHERE
    TRUNC(p.payment_date) BETWEEN last_month_start
    AND last_month_end
GROUP BY
    customer_id
HAVING
    SUM(p.amount) > min_dollar_amount_purchased
    AND COUNT(customer_id) > min_monthly_purchases;

/* Populate OUT parameter with count of found customers */
count_rewardees := tmp_customers.COUNT;

/*
    Output ALL customer information of matching rewardees.
    Customize output as needed.
*/
```

```
FOR i IN 1..tmp_customers.COUNT LOOP FOR c IN (
    SELECT
        *
    FROM
        customer
    WHERE
        customer_id = tmp_customers(i)
) LOOP
DBMS_OUTPUT.PUT_LINE(
    'Customer ID: ' || c.customer_id || ', Name: ' || c.first_name || ' ' || c.last_name
);

END LOOP;

END LOOP;

END;
```

回答问题：

（1） 这个存储过程实现了什么功能？输出参数 count_rewardees 是什么？

（2） last_month_start := ADD_MONTHS(TRUNC(SYSDATE, 'MM'), -1);具体是怎么获取到上个月第一天的日期的？请展开说明。

2.1.5关于函数

观察 SAKILA用户里面的 get_customer_balance 函数：

```
 FUNCTION get_customer_balance(
    p_customer_id IN NUMBER,
    p_effective_date IN DATE
) RETURN NUMBER IS
v_rentfees NUMBER(5, 2); -- FEES PAID TO RENT THE VIDEOS INITIALLY
v_overfees NUMBER(5, 2); -- LATE FEES FOR PRIOR RENTALS
v_payments NUMBER(5, 2); -- SUM OF PAYMENTS MADE PREVIOUSLY
BEGIN

-- OK, WE NEED TO CALCULATE THE CURRENT BALANCE GIVEN A CUSTOMER_ID AND A DATE
-- THAT WE WANT THE BALANCE TO BE EFFECTIVE FOR. THE BALANCE IS:
--    1) RENTAL FEES FOR ALL PREVIOUS RENTALS
--    2) ONE DOLLAR FOR EVERY DAY THE PREVIOUS RENTALS ARE OVERDUE
--    3) IF A FILM IS MORE THAN RENTAL_DURATION * 2 OVERDUE, CHARGE THE REPLACEMENT_COST
--    4) SUBTRACT ALL PAYMENTS MADE BEFORE THE DATE SPECIFIED

SELECT
    NVL(SUM(film.rental_rate), 0) INTO v_rentfees
FROM
    film
    JOIN inventory ON film.film_id = inventory.film_id
    JOIN rental ON inventory.inventory_id = rental.inventory_id
WHERE
    rental.rental_date <= p_effective_date
    AND rental.customer_id = p_customer_id;
```

```
SELECT
    NVL(SUM(
        CASE WHEN (TRUNC(rental.return_date) - TRUNC(rental.rental_date)) > film.rental_duration
            THEN (TRUNC(rental.return_date) - TRUNC(rental.rental_date)) - film.rental_duration
            ELSE 0
        END),0
    ) INTO v_overfees
FROM
    rental
    JOIN inventory ON inventory.inventory_id = rental.inventory_id
    JOIN film ON film.film_id = inventory.film_id
WHERE
    rental.rental_date <= p_effective_date
    AND rental.customer_id = p_customer_id;

SELECT
    NVL(SUM(payment.amount), 0) INTO v_payments
FROM
    payment
WHERE
    payment.payment_date <= p_effective_date
    AND payment.customer_id = p_customer_id;

RETURN v_rentfees + v_overfees - v_payments;

END;
```

回答问题：

（1） 这个函数实现了什么功能？返回值是什么？

（2） 这个函数体中用到了 3 个函数，是哪几个函数？这 3 个函数的作用分别是？

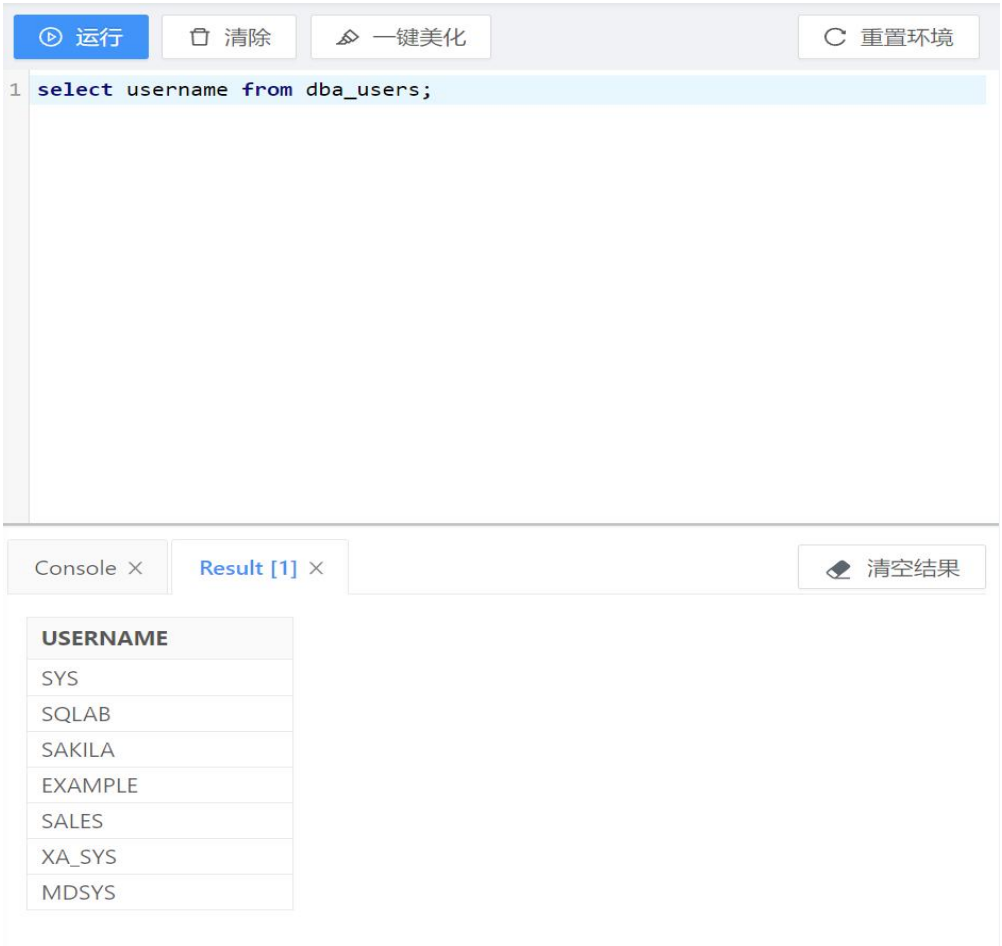
函数	作用

2.2 创建新用户并分配权限

（请自己创建一个以自己名字命名的用户！赋予该用户相应权限并验证，把过程截图填写到实验报告中）

1、 创建新用户

（1） 查看当前已有用户




```
select username from dba_users;
```

（2） 新建 SAKILA_TESE用户（密码 123456）

```
create user SAKILA_TEST identified by 123456;
```

（3） 再查看用户就可以看到新用户：

```
select username from dba_users;
```

2、 为新用户SAKILA_TEST赋予访问 sakila 的权限

（1） 先查看新用户当前的权限

```
select * from dba_sys_privs where grantee='SAKILA_TEST';
select * from dba_tab_privs where grantee='SAKILA_TEST';
select * from dba_role_privs where grantee='SAKILA_TEST';
```

赋予新用户connect角色（CONNECT角色具有CREATE SESSION权限，通过赋予CONNECT角色使用户能够登录会话。）

```
grant connect to SAKILA_TEST;
```

赋予新用户resource角色（RESOURCE角色具有CREATE TABLE、CREATE SEQUENCE、CREATE PROCEDURE、CREATE TRIGGER、CREATE TYPE的权限。）

```
grant resource to SAKILA_TEST;
```

（2） 看看当前的权限可以做什么

用新用户连接数据库：

输入 用户名、密码，点击“确认”

用新用户连接成功：

这时看不到原来的用户的表：

你可以用新用户新建表。

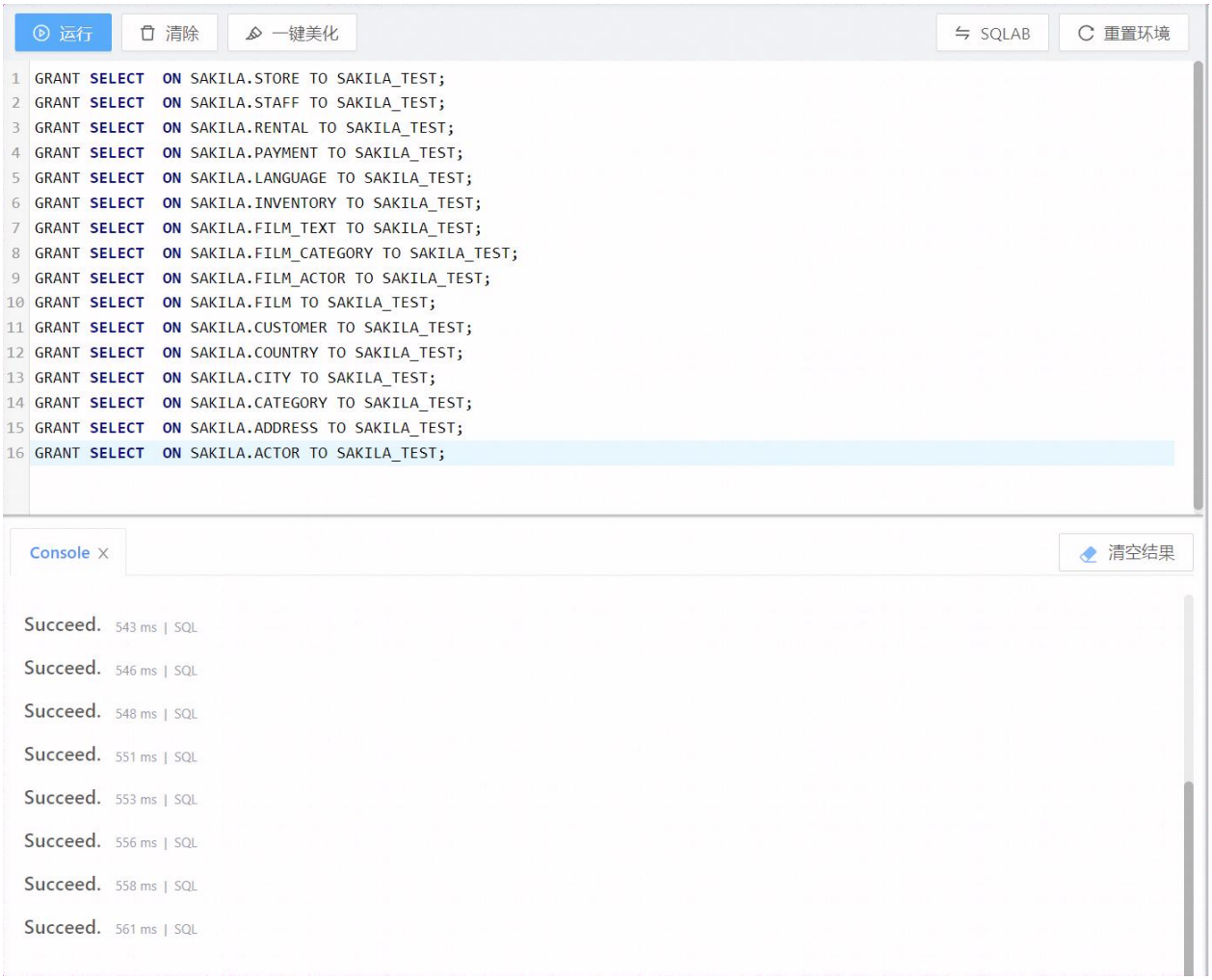
（3） 把原来的 SAKILA用户中所有的表的访问权限赋予 SAKILA_TEST 用户

注意：要先切换回具有DBA权限(如默认用户SQLAB)的用户执行下面的 grant 操作

执行以下语句得到所有的权限授予语句

```
SELECT  'GRANT SELECT  ON SAKILA.'  || TABLE_NAME || ' TO SAKILA_TEST;' FROM ALL_TABLES WHERE OWNER = 'SAKILA';
```

赋予权限



切换到 sakila_test 用户，这时可以看到 sakila 用户的所有表，可以进行查询操作：

```
select * from all_tables;
```

Console × Result [1] × 清空结果

OWNER	TABLE_NAME	TABLE_TYPE	TABLESPACE_NAME	STATUS	PCT_FREE	I
SAKILA	STORE	HEAP	USERS	VALID	8	2
SAKILA	STAFF	HEAP	USERS	VALID	8	2
SAKILA	RENTAL	HEAP	USERS	VALID	8	2
SAKILA	PAYMENT	HEAP	USERS	VALID	8	2
SAKILA	LANGUAGE	HEAP	USERS	VALID	8	2
SAKILA	INVENTORY	HEAP	USERS	VALID	8	2
SAKILA	FILM_TEXT	HEAP	USERS	VALID	8	2
SAKILA	FILM_CATEGORY	HEAP	USERS	VALID	8	2
SAKILA	FILM_ACTOR	HEAP	USERS	VALID	8	2
SAKILA	FILM	HEAP	USERS	VALID	8	2

SQL | 执行耗时：767 ms 1 2 10 条/页 跳至 页

更多 grant 的用法参考附录 1。

2.3 设计并实现

根据应用场景，为 Sakila 数据库合理地设计并实现：

1. 设计 1 个视图，通过关联customer和address 2 个表，可以查看客户及其地址信息的列表，方便用户查看地址信息；

2. 设计 1 个触发器，融合ins_film、upd_film和del_film的功能，需要在报告里体现触发器生效；

3. 设计 1 个存储过程，用于获取特定客户在指定时间段内的租赁历史记录。它将返回客户的租赁信息，包括租赁日期、归还日期、租赁的电影标题以及租赁费用。须在报告里调用，并展示结果。

注意：请将创建语句、执行结果截图记录到实验报告里。

2.4 思考题

在MySQL开发规范中，通常要求为每张表创建一个主键，但是在YashanDB中，表可以没有主键。请分析一下原因。主键是否没有存在的必要？

注意：请将答案写到实验报告里。

附录 1 GRANT 命令

一、**grant** 普通数据用户，查询、插入、更新、删除 数据库中某个表数据的权利。

```
grant select on testdb.test01 to common_user;

grant insert on testdb.test01 to common_user;

grant update on testdb.test01 to common_user;

grant delete on testdb.test01 to common_user;
```

或者，用一条YASHANDB命令来替代：

```
grant all privileges on testdb.test01 to common_user;
```

二、**grant** 数据库开发人员，创建表、索引、视图、存储过程、函数。。。等权限。

grant 创建、修改、删除YashanDB数据表结构权限。

```
grant create table to developer;

grant alter on testdb.test01 to developer;

grant drop any table to developer;
```

grant 操作 YashanDB 外键权限。

```
grant references on testdb.test01 to developer;
```

grant 操作 YashanDB 临时表权限。

```
grant create table to developer;
```

grant 操作 YashanDB 索引权限。

```
grant index on testdb.test01 to developer;
```

grant 操作 YashanDB 视图、查看视图源代码 权限。

```
grant create view to developer;

grant select any table to developer;--通过select text from
all_views;查看视图源代码
```

grant 操作 YashanDB 存储过程、函数 权限。

```
 grant  create any procedure  to developer; -- now, can show procedure status
 grant  alter any procedure  to developer;-- now, you can drop a procedure
```

```
 grant  EXECUTE ANY PROCEDURE  to developer;
```

三、**grant** 普通 **DBA** 管理拥有对表的所有对象特权。

```
grant all privileges on testdb.test01 to common_dba;
```

四、**grant** 高级 **DBA** 管理 **YashanDB** 具备所有的系统特权。

```
grant all privileges to high_dba;
```

五、**YashanDB grant** 权限，分别可以作用在多个层次上。

1. grant 作用在系统特权上：

```
grant select any table to high_dba; -- high_dba 具备对数据库中（sys schema除外）任意表、视图、动态视图、物化视图发起查询的权限；动态视图暂不受限
grant all privileges to high_dba; -- high_dba具备所有的系统特权
```

2. grant 作用在角色上：

```
grant connect to high_dba; -- CONNECT角色具有CREATE SESSION权限，通过赋予CONNECT角色使用户能够登录会话。
```

3. grant 作用在对象上：

```
grant select on testdb.test01 to high_dba; -- high_dba 可以查询 test01 中的所有数据。
```

这里在给一个用户授权多张表时，可以多次执行以上语句。例如：

```
grant select on testdb.test02 to high_dba;

grant update on testdb.test03 to high_dba;
```

4. grant 作用在表中的索引上：

```
grant index on testdb.test01 to high_dba;
```

5. grant 作用在存储过程上：

```
grant  create any procedure  to  high_dba;
```

六、查看**YashanDB**用户权限

查看用户系统权限：

```
select * from dba_sys_privs where grantee='common_user';
```

查看用户对象权限：

```
select * from dba_tab_privs where grantee='common_user';
```

查看用户拥有的角色：

```
select * from dba_role_privs where grantee='common_user';
```

七、撤销已经赋予给 **YashanDB** 用户权限的权限。

revoke 跟 grant 的语法差不多，只需要把关键字 to 换成 from 即可：

```
grant all privileges to high_dba;

revoke all privileges from high_dba;
```

八、**YashanDB grant**、**revoke** 用户权限注意事项

1. grant, revoke 用户权限后，该用户权限立即生效。

2. 如果想让授权的用户，也可以将这些对象或者系统权限 grant 给其他用户，需要选项 **grant option** 和 admin option

```
grant select on testdb.test01 to common_user  with grant option;
grant create table to common_user  with admin option;
```

这个特性一般用不到。实际中，数据库权限最好由 DBA 来统一管理。

（参考链接 GRANT | YashanDB Doc）