**Курсовая работа**

**По курсу «Основы реляционных баз данных. MySQL»**

**Тема: «Организация хранения данных по учебным курсам»**

**Слушатель: Левшин АВ**

**1.Описание:** Учет учебных курсов по оперативным данным: наименование, когда проводится, лектор, по финансовым данным: приходы и расходы по курсам и направлениям план-факторный учет

**2.Описание таблиц:**

`direction` - направление курса

`event` - курс - мероприятие

`lecturer` - лектор на курс

`chapter` - группа статей

`article`-статьи доходов/расходов

`sub\_articles` - подстатьи доходов/расходов – дополнительная аналитика

`record` - фиксация фактического дохода и расхода

`fact\_cost` - конкретизация дохода и расхода в ‘record’ с возможностью расщепления суммы по курсу и подстатьи

`file` - файл для обоснования платежа (например, счет)

`firma` - контрагент – клиент/поставщик

`type\_receipt` - вид финансового документа

`payment` - финансовый документ - платеж

`plan\_cost` - фиксация плановых сумм по статье расхода

`user`- пользователь

`profile` - профиль пользователя

**3. скрипты создания структуры БД** – фрагмент;

CREATE TABLE `fact\_cost` (

`id` int unsigned NOT NULL AUTO\_INCREMENT,

`comment` varchar(45) DEFAULT NULL,

`summa` float(15,2) DEFAULT NULL,

`sub\_articles\_id\_receipt` int unsigned DEFAULT NULL,

`sub\_articles\_id\_charge` int unsigned DEFAULT NULL,

`record\_id` int unsigned NOT NULL,

`event\_id\_receipt` int unsigned DEFAULT NULL,

`event\_id\_charge` int unsigned DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `fk\_fact\_cost\_sub\_articles1\_idx` (`sub\_articles\_id\_receipt`),

KEY `fk\_fact\_cost\_sub\_articles2\_idx` (`sub\_articles\_id\_charge`),

KEY `fk\_fact\_cost\_record1\_idx` (`record\_id`),

KEY `fk\_fact\_cost\_event1\_idx` (`event\_id\_receipt`),

KEY `fk\_fact\_cost\_event2\_idx` (`event\_id\_charge`),

CONSTRAINT `fk\_fact\_cost\_record1` FOREIGN KEY (`record\_id`) REFERENCES `record` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=201;

CREATE TABLE `record` (

`id` int unsigned NOT NULL AUTO\_INCREMENT,

`profile\_user\_id` int unsigned NOT NULL,

`payment\_id` int unsigned NOT NULL,

`firma\_id` int unsigned NOT NULL,

`file\_id` int unsigned DEFAULT NULL,

`comment` varchar(245) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `fk\_record\_payment1\_idx` (`payment\_id`),

KEY `fk\_record\_organization1\_idx` (`firma\_id`),

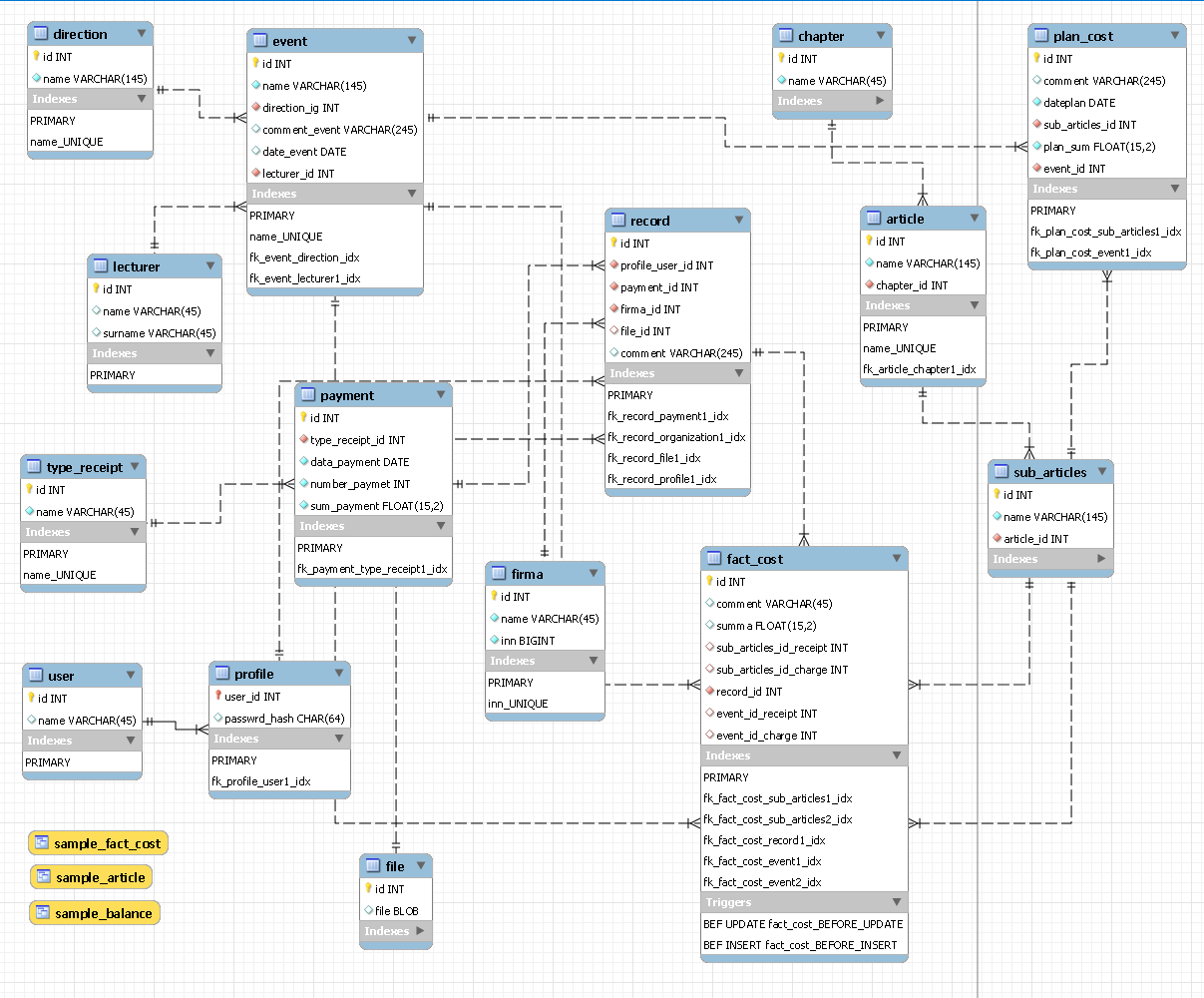
KEY `fk\_record\_file1\_idx` (`file\_id`),

KEY `fk\_record\_profile1\_idx` (`profile\_user\_id`),

CONSTRAINT `fk\_record\_payment1` FOREIGN KEY (`payment\_id`) REFERENCES `payment` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=201;

**4. создать ERDiagram для БД** - фрагмент;



**5. Скрипты наполнения БД данными** - фрагмент;

INSERT INTO `event` VALUES (1,'Курс 01',1,NULL,'2021-01-15',1),(2,'Курс 02',2,NULL,'2021-02-10',5),(3,'Курс 03',3,NULL,'2021-02-10',4),(4,'Курс 04',1,NULL,'2021-02-15',2),(5,'Курс 05',2,NULL,'2021-03-04',1),(6,'Курс 06',3,NULL,'2021-04-11',3),(7,'Курс 07',1,NULL,'2021-04-20',4),(8,'Курс 08',2,NULL,'2021-05-16',5),(9,'Курс 09',3,NULL,'2021-05-30',2),(10,'Курс 10',1,NULL,'2021-06-10',3);

INSERT INTO `lecturer` VALUES (1,'Артур','Смирнов'),(2,'Александр','Иванов'),(3,'Елена','Соколова'),(4,'Ирина','Кузнецова'),(5,'Сергей','Скоробогатов');

INSERT INTO `record` VALUES (1,1,1,1,1,NULL),(2,2,2,2,2,NULL),(3,3,3,3,3,NULL),(4,4,4,4,4,NULL),(5,5,5,5,5,NULL),(6,6,6,6,6,NULL),(7,7,7,7,7,NULL),(8,8,8,8,8,NULL),(9,9,9,9,9,NULL),(10,10,10,10,10,NULL),(11,1,11,11,11,NULL),(12,2,12,12,12,NULL),(13,3,13,13,13,NULL),(14,4,14,14,14,NULL),(15,5,15,15,15,NULL),(16,6,16,16,16,NULL),(17,7,17,17,17,NULL),(18,8,18,18,18,NULL),(19,9,19,19,19,NULL),(20,10,20,20,20,NULL),(21,1,21,21,21,NULL),(22,2,22,22,22,NULL),(23,3,23,23,23,NULL),(24,4,24,24,24,NULL),(25,5,25,25,25,NULL),(26,6,26,26,26,NULL),(27,7,27,27,27,NULL),(28,8,28,28,28,NULL),(29,9,29,29,29,NULL),(30,10,30,30,30,NULL),(31,1,31,31,31,NULL),(32,2,32,32,32,NULL),(33,3,33,33,33,NULL),(34,4,34,34,34,NULL),(35,5,35,35,35,NULL),(36,6,36,36,36,NULL),(37,7,37,37,37,NULL),(38,8,38,38,38,NULL),(39,9,39,39,39,NULL),(40,10,40,40,40,NULL),(41,1,41,41,41,NULL),(42,2,42,42,42,NULL),(43,3,43,43,43,NULL),(44,4,44,44,44,NULL),(45,5,45,45,45,NULL),(46,6,46,46,46,NULL),(47,7,47,47,47,NULL),(48,8,48,48,48,NULL),(49,9,49,49,49,NULL),(50,10,50,50,50,NULL),(51,1,51,51,1,NULL),(52,2,52,52,2,NULL),(53,3,53,53,3,NULL),(54,4,54,54,4,NULL),(55,5,55,55,5,NULL),(56,6,56,56,6,NULL),(57,7,57,57,7,NULL),(58,8,58,58,8,NULL),(59,9,59,59,9,NULL),(60,10,60,60,10,NULL),(61,1,61,61,11,NULL),(62,2,62,62,12,NULL),(63,3,63,63,13,NULL),(64,4,64,64,14,NULL),(65,5,65,65,15,NULL),(66,6,66,66,16,NULL),(67,7,67,67,17,NULL),(68,8,68,68,18,NULL),(69,9,69,69,19,NULL),(70,10,70,70,20,NULL),(71,1,71,71,21,NULL),(72,2,72,72,22,NULL),(73,3,73,73,23,NULL),(74,4,74,74,24,NULL),(75,5,75,75,25,NULL),(76,6,76,76,26,NULL),(77,7,77,77,27,NULL),(78,8,78,78,28,NULL),(79,9,79,79,29,NULL),(80,10,80,80,30,NULL),(81,1,81,81,31,NULL),(82,2,82,82,32,NULL),(83,3,83,83,33,NULL),(84,4,84,84,34,NULL),(85,5,85,85,35,NULL),(86,6,86,86,36,NULL),(87,7,87,87,37,NULL),(88,8,88,88,38,NULL),(89,9,89,89,39,NULL),(90,10,90,90,40,NULL),(91,1,91,91,41,NULL),(92,2,92,92,42,NULL),(93,3,93,93,43,NULL),(94,4,94,94,44,NULL),(95,5,95,95,45,NULL),(96,6,96,96,46,NULL),(97,7,97,97,47,NULL),(98,8,98,98,48,NULL),(99,9,99,99,49,NULL),(100,10,100,100,50,NULL),(101,1,101,1,1,NULL),(102,2,102,2,2,NULL),(103,3,103,3,3,NULL),(104,4,104,4,4,NULL),(105,5,105,5,5,NULL),(106,6,106,6,6,NULL),(107,7,107,7,7,NULL),(108,8,108,8,8,NULL),(109,9,109,9,9,NULL),(110,10,110,10,10,NULL),(111,1,111,11,11,NULL),(112,2,112,12,12,NULL),(113,3,113,13,13,NULL),(114,4,114,14,14,NULL),(115,5,115,15,15,NULL),(116,6,116,16,16,NULL),(117,7,117,17,17,NULL),(118,8,118,18,18,NULL),(119,9,119,19,19,NULL),(120,10,120,20,20,NULL),(121,1,121,21,21,NULL),(122,2,122,22,22,NULL),(123,3,123,23,23,NULL),(124,4,124,24,24,NULL),(125,5,125,25,25,NULL),(126,6,126,26,26,NULL),(127,7,127,27,27,NULL),(128,8,128,28,28,NULL),(129,9,129,29,29,NULL),(130,10,130,30,30,NULL),(131,1,131,31,31,NULL),(132,2,132,32,32,NULL),(133,3,133,33,33,NULL),(134,4,134,34,34,NULL),(135,5,135,35,35,NULL),(136,6,136,36,36,NULL),(137,7,137,37,37,NULL),(138,8,138,38,38,NULL),(139,9,139,39,39,NULL),(140,10,140,40,40,NULL),(141,1,141,41,41,NULL),(142,2,142,42,42,NULL),(143,3,143,43,43,NULL),(144,4,144,44,44,NULL),(145,5,145,45,45,NULL),(146,6,146,46,46,NULL),(147,7,147,47,47,NULL),(148,8,148,48,48,NULL),(149,9,149,49,49,NULL),(150,10,150,50,50,NULL),(151,1,151,51,1,NULL),(152,2,152,52,2,NULL),(153,3,153,53,3,NULL),(154,4,154,54,4,NULL),(155,5,155,55,5,NULL),(156,6,156,56,6,NULL),(157,7,157,57,7,NULL),(158,8,158,58,8,NULL),(159,9,159,59,9,NULL),(160,10,160,60,10,NULL),(161,1,161,61,11,NULL),(162,2,162,62,12,NULL),(163,3,163,63,13,NULL),(164,4,164,64,14,NULL),(165,5,165,65,15,NULL),(166,6,166,66,16,NULL),(167,7,167,67,17,NULL),(168,8,168,68,18,NULL),(169,9,169,69,19,NULL),(170,10,170,70,20,NULL),(171,1,171,71,21,NULL),(172,2,172,72,22,NULL),(173,3,173,73,23,NULL),(174,4,174,74,24,NULL),(175,5,175,75,25,NULL),(176,6,176,76,26,NULL),(177,7,177,77,27,NULL),(178,8,178,78,28,NULL),(179,9,179,79,29,NULL),(180,10,180,80,30,NULL),(181,1,181,81,31,NULL),(182,2,182,82,32,NULL),(183,3,183,83,33,NULL),(184,4,184,84,34,NULL),(185,5,185,85,35,NULL),(186,6,186,86,36,NULL),(187,7,187,87,37,NULL),(188,8,188,88,38,NULL),(189,9,189,89,39,NULL),(190,10,190,90,40,NULL),(191,1,191,91,41,NULL),(192,2,192,92,42,NULL),(193,3,193,93,43,NULL),(194,4,194,94,44,NULL),(195,5,195,95,45,NULL),(196,6,196,96,46,NULL),(197,7,197,97,47,NULL),(198,8,198,98,48,NULL),(199,9,199,99,49,NULL),(200,10,200,100,50,NULL);

**6. Скрипты характерных выборок;**

SELECT

id,

name,

date\_event,

(SELECT CONCAT(name,' ',surname) FROM mydb.lecturer WHERE id= mydb.event.lecturer\_id) as lecturer,

(SELECT name FROM mydb.direction WHERE id= mydb.event.direction\_ig) as direction,

comment\_event as comment,

**balance(id) as balance**

FROM mydb.event

ORDER BY balance;

SELECT

id,

data\_payment as data,

number\_paymet as number,

sum\_payment as summa,

(SELECT name FROM mydb.type\_receipt WHERE id=type\_receipt\_id) as type\_receipt

FROM mydb.payment;

SELECT

sum(sum\_payment) as summa,

(SELECT name FROM mydb.type\_receipt WHERE id=type\_receipt\_id) as type\_receipt

FROM mydb.payment

GROUP BY type\_receipt

ORDER BY summa;

SELECT

id,

data\_payment as data,

number\_paymet as number,

sum\_payment as summa,

(SELECT name FROM mydb.type\_receipt WHERE id=type\_receipt\_id) as type\_receipt

FROM mydb.payment

WHERE sum\_payment>950000;

SELECT

art.id,

ch.name as chapter,

art.name as article

FROM mydb.article AS art

LEFT JOIN mydb.chapter ch ON ch.id=art.chapter\_id

ORDER BY chapter,article;

select

sar.id,

ar.chapter,

ar.article,

sar.name as sub\_article,

ifnull((SELECT plan\_sum FROM mydb.plan\_cost where sub\_articles\_id = sar.id order by dateplan desc limit 1),0) as plan\_sum

from mydb.sub\_articles sar

join (

SELECT

art.id as id,

ch.name as chapter,

art.name as article

FROM mydb.article AS art

LEFT JOIN mydb.chapter ch ON ch.id = art.chapter\_id

) ar on ar.id=sar.article\_id;

select

saba.id,

saba.name as sub\_article,

(SELECT plan\_sum FROM mydb.plan\_cost

where sub\_articles\_id = saba.id order by dateplan desc limit 1) as plan\_sum

from mydb.sub\_articles saba;

SELECT

plan\_sum

FROM mydb.plan\_cost

where sub\_articles\_id = 1

order by dateplan desc

limit 1;

#UPDATE mydb.plan\_cost SET comment = CONCAT('Комментарий ',mydb.plan\_cost.id);

#UPDATE mydb.plan\_cost SET event\_id = round(rand()\*(10-1)+1);

#explain

SELECT

rec.id,

us.name as user,

pay.data\_payment,

pay.number\_paymet,

pay.sum\_payment,

fir.name as firma,

fil.file

FROM mydb.record rec

LEFT JOIN mydb.user us ON us.id = rec.profile\_user\_id

LEFT JOIN mydb.payment pay ON pay.id = rec.payment\_id

LEFT JOIN mydb.firma fir ON fir.id = rec.firma\_id

LEFT JOIN mydb.file fil ON fil.id = rec.file\_id;

#explain

SELECT

profile\_user\_id,

(SELECT name FROM mydb.user WHERE id=record.profile\_user\_id) as user,

(SELECT data\_payment FROM mydb.payment WHERE id=record.payment\_id) as data\_paymet,

(SELECT number\_paymet FROM mydb.payment WHERE id=record.payment\_id) as number\_payment,

(SELECT sum\_payment FROM mydb.payment WHERE id=record.payment\_id) as sum\_paymet,

(SELECT name FROM mydb.firma WHERE id=record.firma\_id) as firma,

(SELECT file FROM mydb.file WHERE id=record.file\_id) as file

FROM mydb.record;

#explain

SELECT

profile\_user\_id,

(SELECT name FROM mydb.user WHERE id=record.profile\_user\_id) as user,

CONCAT('N ', (SELECT number\_paymet FROM mydb.payment WHERE id=record.payment\_id), ' от ',(SELECT data\_payment FROM mydb.payment WHERE id=record.payment\_id), ' на сумму: ', (SELECT sum\_payment FROM mydb.payment WHERE id=record.payment\_id)) as data\_paymet,

(SELECT name FROM mydb.firma WHERE id=record.firma\_id) as firma,

(SELECT file FROM mydb.file WHERE id=record.file\_id) as file

FROM mydb.record;

#explain

SELECT

profile\_user\_id,

(SELECT name FROM mydb.user WHERE id=record.profile\_user\_id) as user,

(SELECT name FROM mydb.firma WHERE id=record.firma\_id) as firma,

(SELECT file FROM mydb.file WHERE id=record.file\_id) as file,

CONCAT('N ', pay.number\_paymet, ' от ', pay.data\_payment, ' на сумму ', pay.sum\_payment) as Payment

FROM mydb.record

LEFT JOIN mydb.payment pay ON pay.id=record.payment\_id;

SELECT

fc.id,

fc.summa,

sa\_1.name AS sub\_articles\_receipt,

er\_1.name AS name\_event\_receipt,

fc.comment,

sa\_2.name AS sub\_article\_charge,

er\_2.name AS name\_event\_charge

FROM mydb.fact\_cost AS fc

LEFT JOIN mydb.sub\_articles sa\_1 ON fc.sub\_articles\_id\_receipt = sa\_1.id

LEFT JOIN mydb.sub\_articles sa\_2 ON fc.sub\_articles\_id\_charge = sa\_2.id

LEFT JOIN mydb.event er\_1 ON fc.event\_id\_receipt = er\_1.id

LEFT JOIN mydb.event er\_2 ON fc.event\_id\_charge = er\_2.id

LIMIT 10;

#наполнение данными

#UPDATE mydb.fact\_cost SET sub\_articles\_id\_charge = round(rand()\*(15-1)+1);

#UPDATE mydb.fact\_cost SET event\_id\_receipt = round(rand()\*(10-1)+1);

# с нарастающим итогом оборот по приходу

SET @row\_number1=0;

SET @row\_sum=0;

SELECT

(@row\_number1:=@row\_number1+1) as rows\_number,

fc.id,

fc.summa,

(@row\_sum:=@row\_sum+fc.summa) as turnover\_receipt, #оборот по приходу

fc.articles\_receipt,

fc.event\_receipt

FROM (

Select

id,

summa,

(SELECT name FROM mydb.sub\_articles WHERE id=sub\_articles\_id\_receipt) as articles\_receipt,

(SELECT name FROM mydb.event WHERE id=event\_id\_receipt) as event\_receipt

from mydb.fact\_cost

Where event\_id\_receipt=3

Order by event\_receipt, articles\_receipt) as fc;

# с нарастающим итогом оборот по расходу

SET @row\_number1=0;

SET @row\_sum=0;

SELECT

(@row\_number1:=@row\_number1+1) as rows\_number,

fc.id,

-fc.summa,

(@row\_sum:=@row\_sum-fc.summa) as turnover\_charge, #оборот по расходу

fc.articles\_charge,

fc.event\_charge

FROM (

Select

id,

summa,

(SELECT name FROM mydb.sub\_articles WHERE id=sub\_articles\_id\_charge) as articles\_charge,

(SELECT name FROM mydb.event WHERE id=event\_id\_charge) as event\_charge

from mydb.fact\_cost

Where event\_id\_charge=3

Order by event\_charge, articles\_charge) as fc;

#расчет итога по мероприятию вариант 1

Select

ev.event,

sum(summa\_receipt) as turnover\_receipt,

sum(summa\_charge) as turnover\_charge,

sum(balance) as balance

From (

Select

(SELECT name FROM mydb.event WHERE id=event\_id\_charge) as event,

-summa AS summa\_charge,

0 as summa\_receipt,

-summa as balance

from mydb.fact\_cost

UNION

Select

(SELECT name FROM mydb.event WHERE id=event\_id\_receipt) as event,

0 AS summa\_charge,

summa as summa\_receipt,

summa as balance

from mydb.fact\_cost) as ev

group by event

order by event;

#вариант 2

Select

ev.id,

(select name from mydb.direction dr where dr.id=ev.direction\_ig group by ev.id) as direction,

ev.name,

(select sum(summa) from mydb.fact\_cost fc where fc.event\_id\_receipt=ev.id group by ev.id) as summa\_receipt,

(select -sum(summa) from mydb.fact\_cost fc where fc.event\_id\_charge=ev.id group by ev.id) as summa\_charge,

((select sum(summa) from mydb.fact\_cost fc where fc.event\_id\_receipt=ev.id group by ev.id) + (select -sum(summa) from mydb.fact\_cost fc where fc.event\_id\_charge=ev.id group by ev.id)) as balance

from mydb.event ev

order by balance;

**7. представления VIEW**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `sample\_article` AS

SELECT

`sar`.`id` AS `id`,

`ar`.`chapter` AS `chapter`,

`ar`.`article` AS `article`,

`sar`.`name` AS `sub\_article`,

IFNULL((SELECT

`plan\_cost`.`plan\_sum`

FROM

`plan\_cost`

WHERE

(`plan\_cost`.`sub\_articles\_id` = `sar`.`id`)

ORDER BY `plan\_cost`.`dateplan` DESC

LIMIT 1),

0) AS `plan\_sum`

FROM

(`sub\_articles` `sar`

JOIN (SELECT

`art`.`id` AS `id`,

`ch`.`name` AS `chapter`,

`art`.`name` AS `article`

FROM

(`article` `art`

LEFT JOIN `chapter` `ch` ON ((`ch`.`id` = `art`.`chapter\_id`)))) `ar` ON ((`ar`.`id` = `sar`.`article\_id`)))

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `sample\_fact\_cost` AS

SELECT

`fc`.`id` AS `id`,

`fc`.`summa` AS `summa`,

`sa\_1`.`name` AS `sub\_articles\_receipt`,

`er\_1`.`name` AS `name\_event\_receipt`,

`fc`.`comment` AS `comment`,

`sa\_2`.`name` AS `sub\_article\_charge`,

`er\_2`.`name` AS `name\_event\_charge`

FROM

((((`fact\_cost` `fc`

LEFT JOIN `sub\_articles` `sa\_1` ON ((`fc`.`sub\_articles\_id\_receipt` = `sa\_1`.`id`)))

LEFT JOIN `sub\_articles` `sa\_2` ON ((`fc`.`sub\_articles\_id\_charge` = `sa\_2`.`id`)))

LEFT JOIN `event` `er\_1` ON ((`fc`.`event\_id\_receipt` = `er\_1`.`id`)))

LEFT JOIN `event` `er\_2` ON ((`fc`.`event\_id\_charge` = `er\_2`.`id`)))

**8. хранимые процедуры / триггеры;**

CREATE DEFINER=`root`@`localhost` PROCEDURE `Balance`(IN id\_int int)

BEGIN

SELECT

((select sum(summa) from mydb.fact\_cost fc where fc.event\_id\_receipt = ev.id group by ev.id) + (select -sum(summa) from mydb.fact\_cost fc where fc.event\_id\_charge = ev.id group by ev.id)) as fact\_balance,

(Select sum(plan\_sum) From plan\_cost where event\_id=ev.id group by ev.id) as plan\_balance

FROM mydb.event ev

Where ev.id = id\_int;

END

call balance(3)

CREATE DEFINER=`root`@`localhost` FUNCTION `Balance`(id\_int int) RETURNS float

READS SQL DATA

BEGIN

DECLARE balance float;

SET balance = (SELECT

((select sum(summa) from mydb.fact\_cost fc where fc.event\_id\_receipt = ev.id group by ev.id) + (select -sum(summa) from mydb.fact\_cost fc where fc.event\_id\_charge = ev.id group by ev.id))

FROM mydb.event ev

Where ev.id = id\_int);

RETURN balance;

END

CREATE DEFINER=`root`@`localhost` TRIGGER `fact\_cost\_BEFORE\_INSERT` BEFORE INSERT ON `fact\_cost` FOR EACH ROW BEGIN

IF NEW.event\_id\_receipt = NEW.event\_id\_charge THEN

IF NEW.sub\_articles\_id\_receipt = NEW.sub\_articles\_id\_charge THEN

SIGNAL sqlstate '45000'

SET message\_text = 'identical sub-aticles';

END IF;

END IF;

IF NEW.sub\_articles\_id\_receipt = NEW.sub\_articles\_id\_charge THEN

IF NEW.event\_id\_receipt = NEW.event\_id\_charge THEN

SIGNAL sqlstate '45000'

SET message\_text = 'identical events';

END IF;

END IF;

END

CREATE DEFINER=`root`@`localhost` TRIGGER `fact\_cost\_BEFORE\_UPDATE` BEFORE UPDATE ON `fact\_cost` FOR EACH ROW BEGIN

IF NEW.event\_id\_receipt = NEW.event\_id\_charge THEN

IF NEW.sub\_articles\_id\_receipt = NEW.sub\_articles\_id\_charge THEN

SET NEW.sub\_articles\_id\_receipt = OLD.sub\_articles\_id\_receipt;

SET NEW.sub\_articles\_id\_charge = OLD.sub\_articles\_id\_charge;

END IF;

END IF;

IF NEW.sub\_articles\_id\_receipt = NEW.sub\_articles\_id\_charge THEN

IF NEW.event\_id\_receipt = NEW.event\_id\_charge THEN

SET NEW.event\_id\_receipt = OLD.event\_id\_receipt;

SET NEW.event\_id\_charge = OLD.event\_id\_charge;

END IF;

END IF;

END