
建库建表 sql 语句

1	Introduction	1
2	Installation	1
2.1	方法一：安装 MySQL 时选择 FULL 完全安装	1
2.2	方法二：下载后导入	2
3	Structure	3
3.1	表的对应关系	3
3.2	Tables	6
3.2.1	【01 演员表 actor】	7
3.2.2	【02 地址表 address】	7
3.2.3	【03 分类表 category】	8
3.2.4	【04 城市表 city】	8
3.2.5	【05 国家表 country】	8
3.2.6	【06 客户表 customer】	8
3.2.7	【07 电影表 film】	9
3.2.8	【08 film_actor 表】	9
3.2.9	【09 film_category 表】	9
3.2.10	【10 film_text 表】	9
3.2.11	【11 库存表 inventory】	10
3.2.12	【12 语言表 language】	10
3.2.13	【13 付款表 payment】	10
3.2.14	【14 租金表 rental】	11
3.2.15	【15 工作人员表 staff】	11
3.2.16	【16 商店表 store】	11

3.3	Views	12
3.3.1	The actor_info View	12
3.3.2	The customer_list View	13
3.3.3	The film_list View	13
3.3.4	The nicer_but_slower_film_list View	14
3.3.5	The sales_by_film_category View	16
3.3.6	The sales_by_store View	17
3.3.7	3.2.7. The staff_list View	17
3.4	Stored Procedures.....	18
3.4.1	The film_in_stock Stored Procedure.....	18
3.4.2	The film_not_in_stock Stored Procedure	19
3.4.3	The rewards_report Stored Procedure	20
3.5	Stored Functions.....	22
3.5.1	The get_customer_balance Function	22
3.5.2	The inventory_held_by_customer Function	24
3.5.3	The inventory_in_stock Function.....	24
3.6	Triggers	25
3.6.1	The customer_create_date Trigger.....	25
3.6.2	The payment_date Trigger.....	25
3.6.3	The rental_date Trigger.....	26
3.6.4	The ins_film Trigger	26
3.6.5	The upd_film Trigger.....	26
3.6.6	The del_film Trigger.....	26

1 Introduction

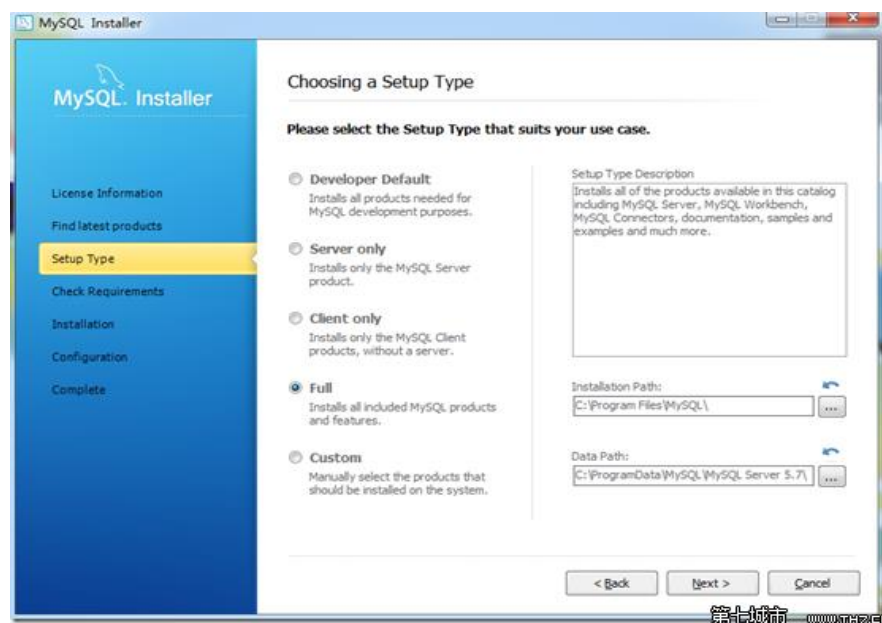
Sakila 可以作为数据库设计的参考,也可作为实验数据。我是用作数据仓库和 ODI 学习的实验数据。

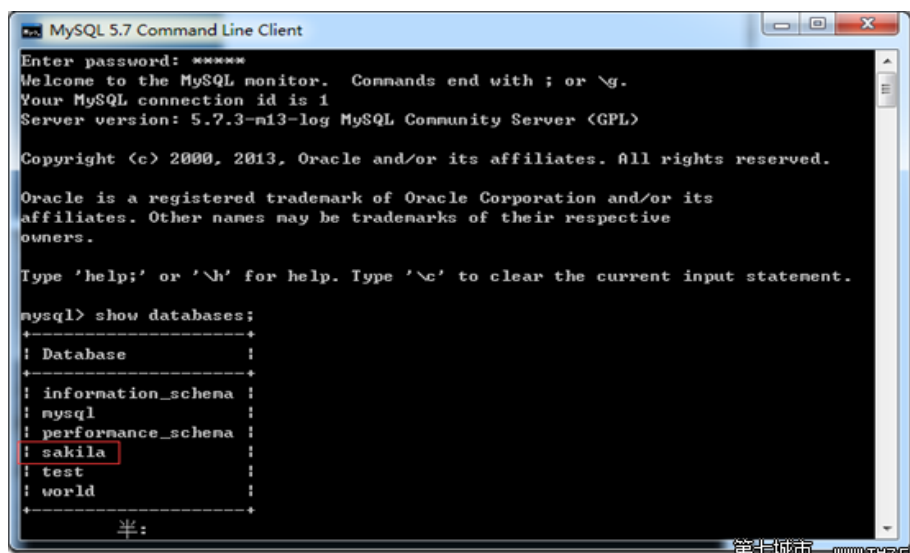
The Sakila sample database was developed by Mike Hillyer, a former member of the MySQL AB documentation team, and is intended to provide a standard schema that can be used for examples in books, tutorials, articles, samples, and so forth. Sakila sample database also serves to highlight the latest features of MySQL such as Views, Stored Procedures, and Triggers.

2 Installation

2.1 方法一：安装 MySQL 时选择 FULL 完全安装

安装 MySQL 时选择 FULL 完全安装, 默认安装了 Sakila sample database





2.2 方法二：下载后导入

The Sakila sample database is available from <http://dev.mysql.com/doc/index-other.html>. A downloadable archive is available in compressed file or Zip format. The archive contains three files: sakila-schema.sql, sakila-data.sql, and sakila.mwb.

The sakila-schema.sql file contains all the CREATE statements required to create the structure of the Sakila database including tables, views, stored procedures, and triggers.

The sakila-data.sql file contains the INSERT statements required to populate the structure created by the sakila-schema.sql file, along with definitions for triggers that must be created after the initial data load.

The sakila.mwb file is a MySQL Workbench data model that you can open within MySQL Workbench to examine the database structure. For more information, see MySQL Workbench.

下载地址：

ZIP 格式：<http://downloads.mysql.com/docs/sakila-db.zip>

tar 格式 <http://downloads.mysql.com/docs/sakila-db.tar.gz>

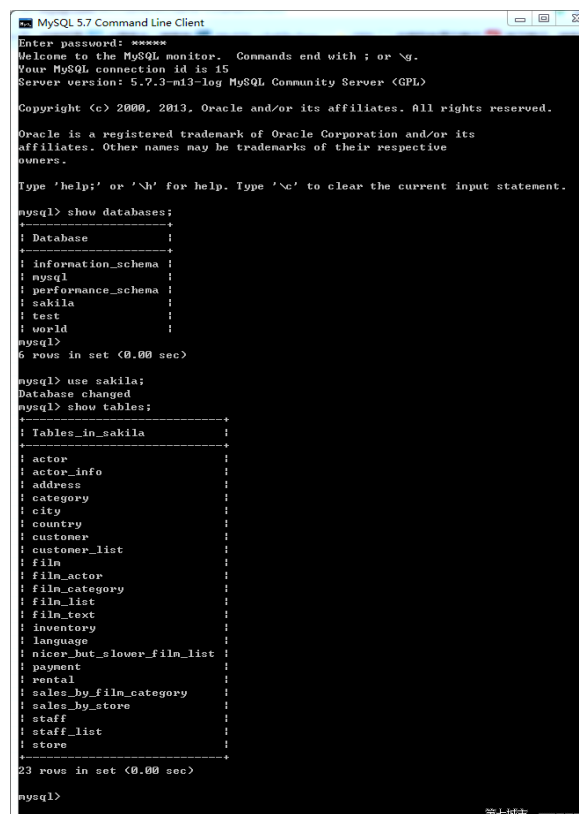
官方文档 <http://dev.mysql.com/doc/sakila/en/index.html>

解压后得到三个文件：

1. sakila-schema.sql 文件包含创建 Sakila 数据库的结构：表、视图、存储过程和触发器
2. sakila-data.sql 文件包含：使用 INSERT 语句填充数据及在初始数据加载后，必须创建的触发器的定义
3. sakila.mwb 文件是一个 MySQL Workbench 数据模型，可以在 MySQL 的工作台打开查看数据库结构。

导入参考：<http://dev.mysql.com/doc/sakila/en/sakila-installation.html>

```
shell>mysql -u root -p
mysql> SOURCE C:/temp/sakila-db/sakila-schema.sql;
mysql> SOURCE C:/temp/sakila-db/sakila-data.sql;
查看导入结果
show databases
use sakila  show datatable
```

A screenshot of the MySQL 5.7 Command Line Client window. The window title is "MySQL 5.7 Command Line Client". The terminal shows the following commands and output:
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.7.3-m13-log MySQL Community Server (GPL)
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sakila |
| test |
| world |
+-----+
mysql>
6 rows in set (0.00 sec)

mysql> use sakila;
Database changed
mysql> show tables;
+-----+
| Tables_in_sakila |
+-----+
| actor |
| actor_info |
| address |
| category |
| city |
| country |
| customer |
| customer_list |
| film |
| film_actor |
| film_category |
| film_list |
| film_text |
| inventory |
| language |
| nicer_but_slower_film_list |
| payment |
| rental |
| sales_by_film_category |
| sales_by_store |
| staff |
| staff_list |
| store |
+-----+
23 rows in set (0.00 sec)

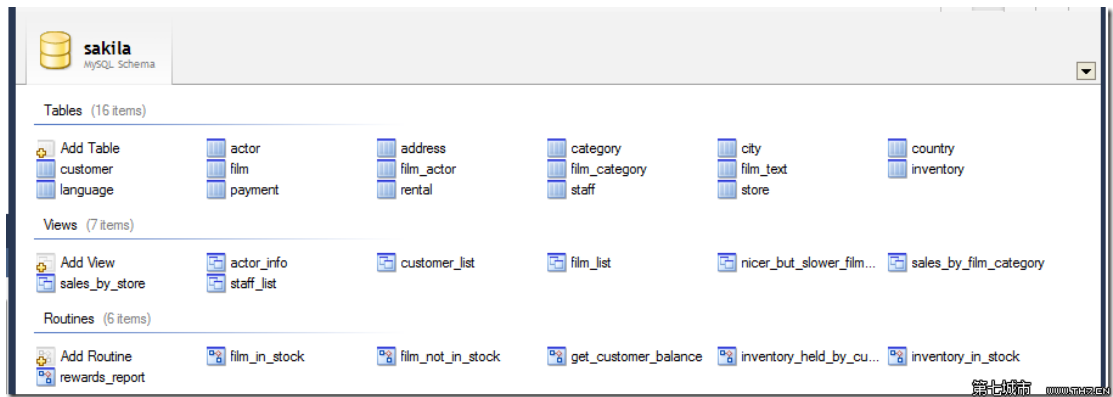
mysql>

sakila

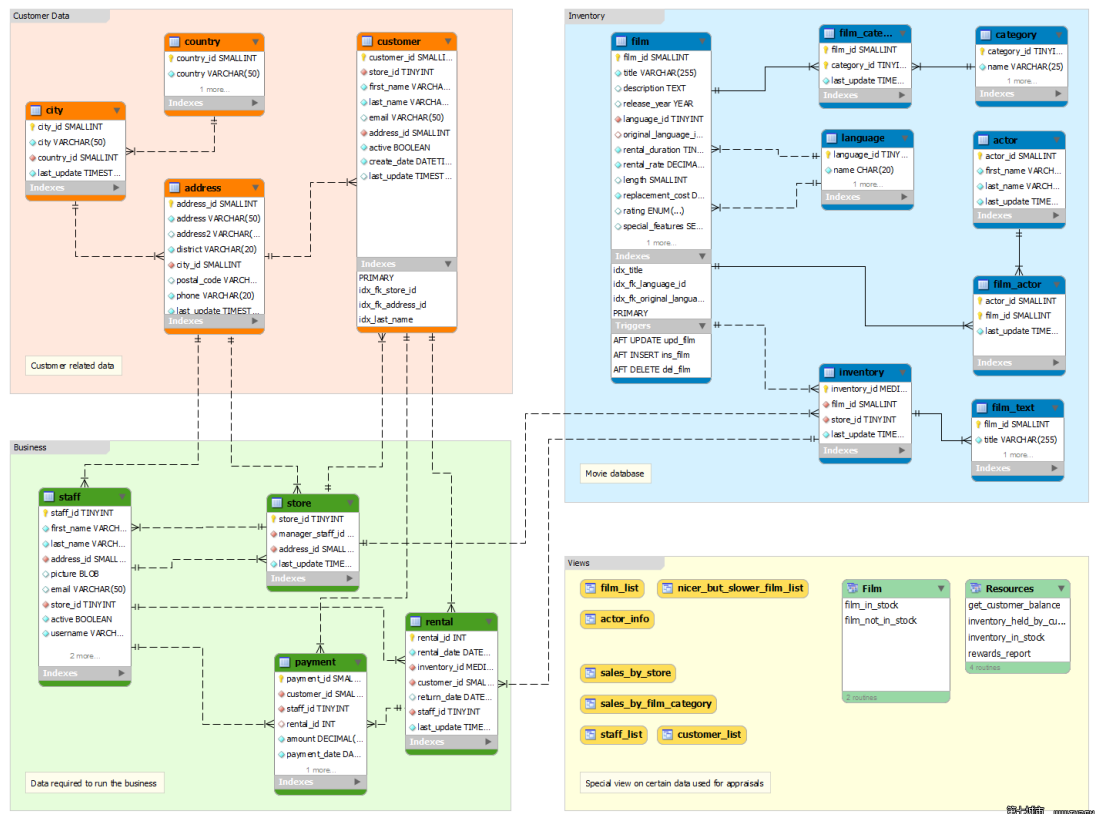
3 Structure

3.1 表的对应关系

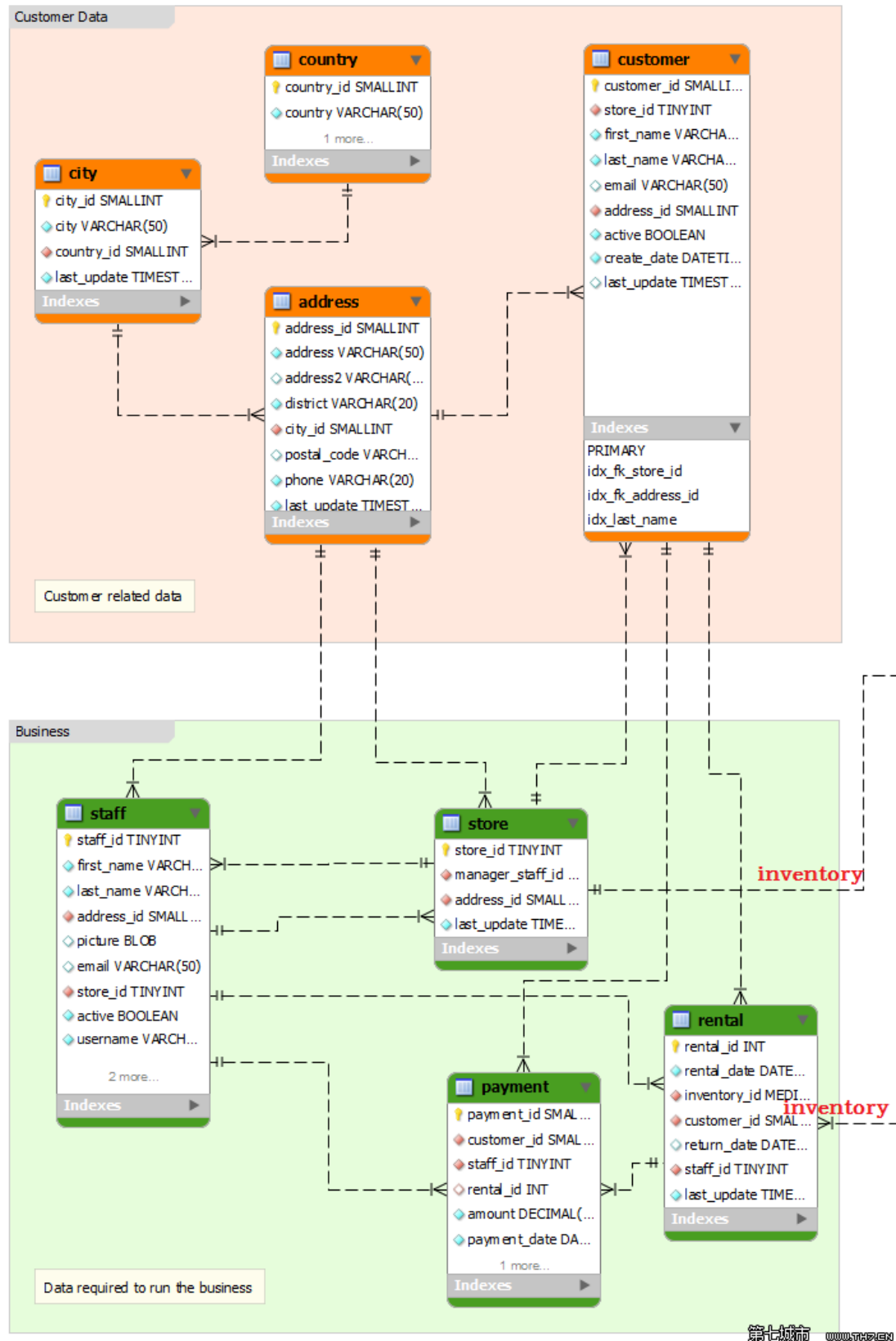
Sakila 样例数据库包括 16 张表格，7 个视图，3 个 Stored Procedures，3 个 Functions，6 个 Triggers。英文描述点开相应连接即可查看。



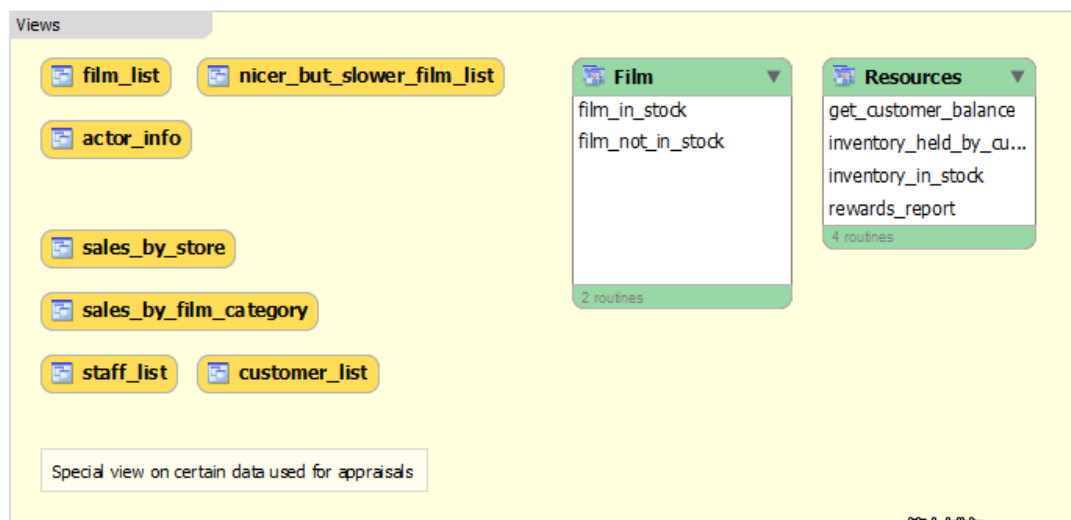
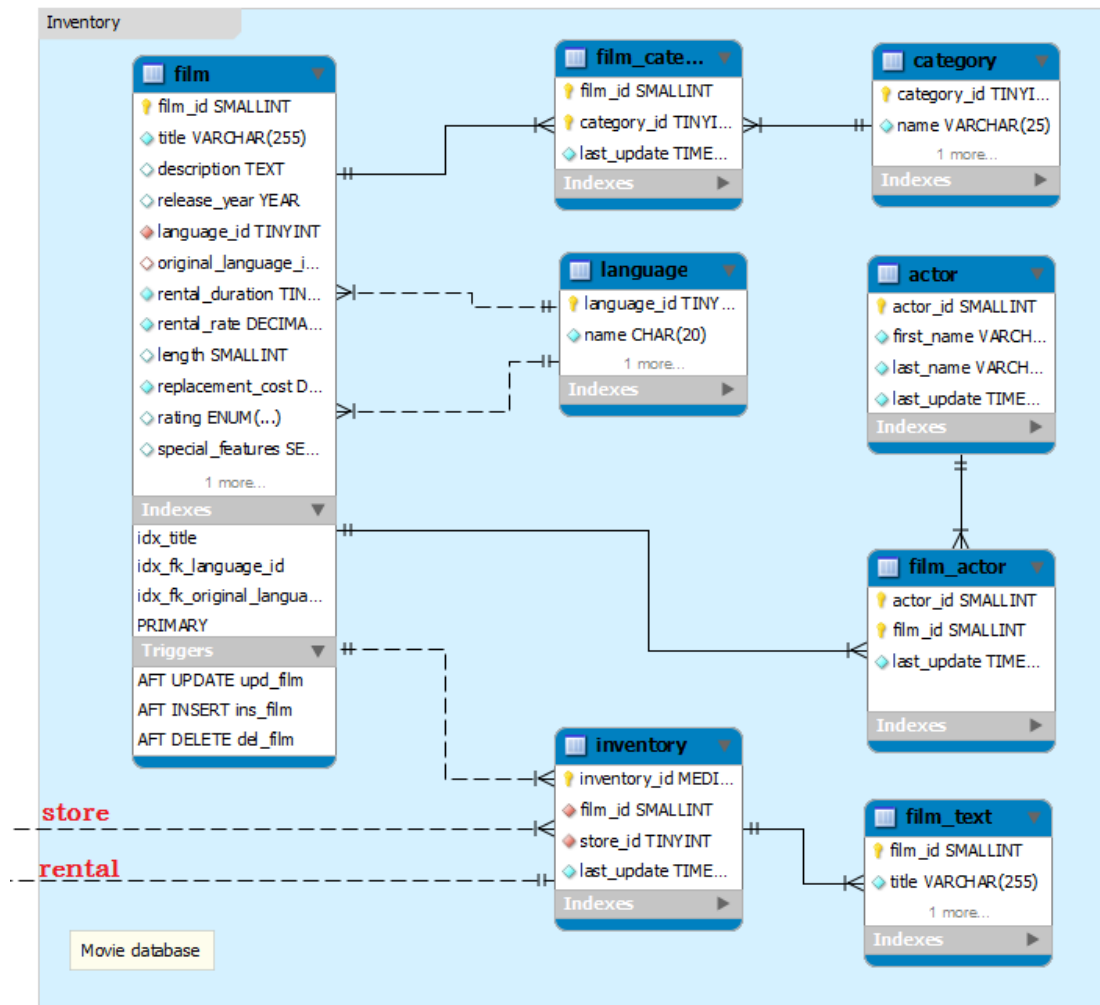
全图:



Customer Data 和 Business Data



Inventory Data



3.2 Tables

(1) The actor Table 演员表

-
- (2) The address Table 地址表
 - (3) The category Table 电影类别表
 - (4) The city Table 城市表
 - (5) The country Table 国家表
 - (6) The customer Table 客户表
 - (7) The film Table 电影表
 - (8) The film_actor Table 电影和演员中间对应表
 - (9) The film_category Table 电影和类别的中间对应表
 - (10) The film_text Table 只包含 title 和 description 两列的电影表子表，用以快速检索

- (11) The inventory Table 库存表
- (12) The language Table 语言表
- (13) The payment Table 付款表
- (14) The rental Table 租金表
- (15) The staff Table 工作人员表
- (16) The store Table 商店表

3.2.1 【01 演员表 actor】

演员表列出了所有演员的信息。演员表和电影表之间是多对多的关系，通过 film_actor 表建立关系

- (1) actor_id: 代理主键用于唯一标识表中的每个演员
- (2) first_name: 演员的名字
- (3) last_name: 演员的姓氏
- (4) last_update: 该行已创建或最近更新的时间

3.2.2 【02 地址表 address】

地址表包含客户、员工和商店的地址信息。地址表的主键出现在顾客、员工、和存储表的外键。

- (1) address_id: 代理主键用于唯一标识表中的每个地址
- (2) address: 地址的第一行
- (3) address2: 一个可选的第二行地址
- (4) district: 该地区的所属地区，这可以是国家，省，县等
- (5) city_id: 指向城市表的外键
- (6) postal_code: 邮政编码
- (7) phone: 地址的电话号码
- (8) last_update: 该行已创建或最近更新的时间

3.2.3 【03 分类表 category】

类别表列出了可以分配到一个电影类别。分类和电影是多对多的关系，通过表 film_category 建立关系

- (1) category_id: 代理主键用于唯一标识表中的每个类别
- (2) name: 类别名称
- (3) last_update: 该行已创建或最近更新的时间

3.2.4 【04 城市表 city】

城市表包含的城市名单。城市表使用外键来标示国家；在地址表中被作为外键来使用。

- (1) city_id: 代理主键用于唯一标识表中的每个城市
- (2) city: 城市的名字
- (3) country_id: 外键，用于标示城市所属的国家
- (4) last_update: 该行已创建或最近更新的时间

3.2.5 【05 国家表 country】

国家表中包含的国家名单。国家表是指在城市表的外键。

- (1) country_id: 代理主键用于唯一标识表中的每个国家
- (2) country: 国家的名称
- (3) last_update: 该行已创建或最近更新的时间

3.2.6 【06 客户表 customer】

客户表包含了所有客户的列表。客户表在支付表和租金表被作为外键使用；客户表使用外键来表示地址和存储。

- (1) customer_id: 代理主键用于唯一标识表中的每个客户
- (2) store_id: 一个外键，确定客户所属的 store。
- (3) first_name: 客户的名字
- (4) last_name: 客户的姓氏
- (5) email: 客户的电子邮件地址
- (6) address_id: 使用在地址表的外键来确定客户的地址
- (7) active: 表示客户是否是活跃的客户
- (8) create_date: 顾客被添加到系统中的日期。使用 INSERT 触发器自动设置。
- (9) last_update: 该行已创建或最近更新的时间

说明： active: 此设置为“FALSE”作为替代客户彻底删除。大多数查询应该有一个 WHERE active = TRUE 字句。 store_id: 此处的客户不仅限于只由这家商店出租，而是包括客户常常去逛的商店

3.2.7 【07 电影表 film】

电影表是一个可能在商店库存的所有影片名单。每部影片的拷贝的实际库存信息保存在库存表。电影表指使用外键来标示语言表；在 film_category、film_actor 和库存表中作为外键使用。

- (1) film_id: 代理主键用于唯一标识表中的每个电影
 - (2) title: 影片的标题
 - (3) description: 一个简短的描述或电影的情节摘要
 - (4) release_year: 电影发行的年份
 - (5) language_id: 使用外键来标示语言
 - (6) original_language_id: 电影的原始语音。使用外键来标示语言
 - (7) rental_duration: 租赁期限的长短，以天作为单位
 - (8) rental_rate: 指定的期限内电影的租金
 - (9) length: 影片的长度，以分钟为单位。
 - (10) replacement_cost: 如果电影未被归还或损坏状态向客户收取的款项
 - (11) rating: 分配给电影评级。可以是 G, PG, PG - 13, R 或 NC - 17
 - (12) special_features: 包括 DVD 上常见的特殊功能的列表
 - (13) last_update: 该行已创建或最近更新的时间
- 特殊功能包括零个或多个拖车、评论、删剪片段、幕后。

3.2.8 【08 film_actor 表】

film_actor 表是用来支持许多电影和演员之间的多对多关系。对于每一个给定的电影演员，将有 film_actor 表中列出的演员和电影中的一个行。

film_actor 表指的是使用外键的电影和演员表。

- (1) actor_id: 用于识别演员的外键
- (2) film_id: 用于识别电影的外键
- (3) last_update: 该行已创建或最近更新的时间

3.2.9 【09 film_category 表】

film_category 表是用来支持许多电影和类别之间的多对多关系。应用于电影的每个类别中，将有 film_category 表中列出的类别和电影中的一个行。

film_category 表是指使用外键的电影和类别表。

- (1) film_id: 用于识别电影的外键
- (2) category_id: 用于识别类别的外键
- (3) last_update: 该行已创建或最近更新的时间

3.2.10 【10 film_text 表】

film_text 表是 Sakila 样例数据库唯一使用 MyISAM 存储引擎的表。

MyISAM 类型不支持事务处理等高级处理，而 InnoDB 类型支持。MyISAM 类型的表强调的是性能，其执行速度比 InnoDB 类型更快。此表提供允许全文搜索电影表中列出的影片的标题和描述。film_text 表包含的 film_id，标题和描述的列电影表，保存的内容与电影表上的内容同步（指电影表的插入、更新和删除操作）

- (1) film_id: 代理主键用于唯一标识表中的每个电影
- (2) title: 影片的标题
- (3) description: 一个简短的描述或电影的情节摘要

注意：film_text 表的内容不应该直接修改。所有的变更来自于电影表。

3.2.11 【11 库存表 inventory】

库存表的一行为存放在一个给定的商店里的一个给定的电影的 copy 副本。库存表是使用外键来识别电影和存储；在出租表中使用外键来识别库存。

- (1) inventory_id: 代理主键用于唯一标识每个项目在库存
- (2) film_id: 使用外键来识别电影
- (3) store_id: 使用外键来识别物品所在的商店
- (4) last_update: 该行已创建或最近更新的时间

3.2.12 【12 语言表 language】

语言表是一个查找表，列出可能使用的语言，电影可以有自己的语言和原始语言值。

语言表在电压表中被作为外键来使用。

- (1) language_id: 代理主键用于唯一标识每一种语言
- (2) name: 语言的英文名称
- (3) last_update: 该行已创建或最近更新的时间

3.2.13 【13 付款表 payment】

付款表记录每个客户的付款，如支付的金额和租金的资料。付款表使用外键来表示客户、出租、和工作人员。

- (1) payment_id: 代理主键用于唯一标识每个付款
- (2) customer_id: 使用外键来标识付款的客户
- (3) staff_id: 工作人员，负责处理支付。使用外键来标识
- (4) rental_id: 租借 ID, 外键，参照 rental 表
- (5) amount: 付款金额
- (6) payment_date: 处理付款的日期
- (7) last_update: 该行已创建或最近更新的时间

3.2.14 【14 租金表 rental】

租借表的一行表示每个 inventory 的租借客户、租借时间、归还时间

租借表是使用外键来标识库存，顾客 和工作人员；在支付表中使用了外键来标识租金。

- (1) rental_id: 代理主键唯一标识的租金
- (2) rental_date: 该项目租用的日期和时间
- (3) inventory_id: 该项目被租用
- (4) customer_id: 租用该项目的客户
- (5) return_date: 归还日期
- (6) staff_id: 处理该项业务的工作人员
- (7) last_update: 该行已创建或最近更新的时间

3.2.15 【15 工作人员表 staff】

工作人员表列出了所有的工作人员，包括电子邮件地址，登录信息和图片信息。

工作人员表是指使用外键来标识存储和地址表；在出租、支付和商店表中作为外键。

- (1) staff_id: 代理主键唯一标识的工作人员
- (2) first_name: 工作人员的名字
- (3) last_name: 工作人员的姓氏
- (4) address_id: 工作人员的地址在地址表的外键
- (5) picture: 工作人员的照片，使用了 BLOB 属性
- (6) email: 工作人员的电子邮件地址
- (7) store_id: 工作人员所在的商店，用外键标识
- (8) active: 是否是活跃的工作人员。
- (9) username: 用户名，由工作人员用来访问租赁系统
- (10) password: 工作人员访问租赁系统所使用的密码。使用了 SHA1 函数
- (11) last_update: 该行已创建或最近更新的时间
- (12) active: 是否有效，删除时设置为 False

3.2.16 【16 商店表 store】

store 表列出了系统中的所有商店。

store 使用外键来标识工作人员和地址；在员工、客户、库存表被作为外键使用。

- (1) store_id: 代理主键唯一标识的商店
- (2) manager_staff_id: 使用外键来标识这家商店的经理
- (3) address_id: 使用外键来确定这家店的地址

(4) last_update: 该行已创建或最近更新的时间

3.3 Views

3.3.1 The actor_info View

actor_info 视图提供了所有演员的列表及所演的电影, 电影按 category 分组.

actor → film_actor → film → film_category → category

actor_id	1
first_name	PENELOPE
last_name	GUINESS
film_info	Animation: ANACONDA CONFESSIONS; Children: LANGUAGE COWBOY; Classics: COLOR PHILADELPHIA, WESTWARD SEABISCUIT; Comedy: VERTIGO NORTHWEST; Documentary: ACADEMY DINOSAUR; Family: KING EVOLUTION, SPLASH GUMP; Foreign: MULHOLLAND BEAST; Games: BULWORTH COMMANDMENTS, HUMAN GRAFFITI; Horror: ELEPHANT TROJAN, LADY STAGE, RULES HUMAN; Music: WIZARD COLDBLOODED; New: ANGELS LIFE, OKLAHOMA JUMANJI; Sci-Fi: CHEAPER CLYDE; Sports: GLEAMING JAWBREAKER

```
SELECT
a.actor_id,
a.first_name,
a.last_name,
GROUP_CONCAT(DISTINCT CONCAT(c.name, ': ',
(SELECT GROUP_CONCAT(f.title ORDER BY f.title SEPARATOR ', ')
FROM sakila.film f
INNER JOIN sakila.film_category fc
ON f.film_id = fc.film_id
INNER JOIN sakila.film_actor fa
ON f.film_id = fa.film_id
WHERE fc.category_id = c.category_id
AND fa.actor_id = a.actor_id
)
)
ORDER BY c.name SEPARATOR '; ')
AS film_info
FROM sakila.actor a
LEFT JOIN sakila.film_actor fa
ON a.actor_id = fa.actor_id
LEFT JOIN sakila.film_category fc
ON fa.film_id = fc.film_id
LEFT JOIN sakila.category c
ON fc.category_id = c.category_id
GROUP BY a.actor_id, a.first_name, a.last_name
```

3.3.2 The customer_list View

客户列表，firstname 和 lastname 连接成 fullname，将 address,city,country 集成在一个视图里

customer→address→city →country

ID	218
name	VERA MCCOY
address	1168 Najafabad Parkway
zip code	40301
phone	886649065861
city	Kabul
country	Afghanistan
notes	active
SID	1

```
SELECT
cu.customer_id AS ID,
CONCAT(
cu.first_name,
_utf8 ' ',
cu.last_name
) AS NAME,
a.address AS address,
a.postal_code AS `zip code`,
a.phone AS phone,
city.city AS city,
country.country AS country,
IF (
cu.active,
_utf8 'active',
_utf8 ''
) AS notes,
cu.store_id AS SID
FROM
customer AS cu
JOIN address AS a ON cu.address_id = a.address_id
JOIN city ON a.city_id = city.city_id
JOIN country ON city.country_id = country.country_id
```

3.3.3 The film_list View

电影列表视图，包含了每一部电影的信息及电影所对应的演员。电影对应

的演员以逗号作为分隔符。连接了 film,film_category,category,film_actorandactor 表的数据

```
SELECT
film.film_id AS FID,
film.title AS title,
film.description AS description,
category. NAME AS category,
film.rental_rate AS price,
film.length AS length,
film.rating AS rating,
GROUP_CONCAT(
CONCAT(
actor.first_name,
_utf8 ' ',
actor.last_name
) SEPARATOR ', '
) AS actors
FROM
category
LEFT JOIN film_category ON category.category_id =
film_category.category_id
LEFT JOIN film ON film_category.film_id = film.film_id
JOIN film_actor ON film.film_id = film_actor.film_id
JOIN actor ON film_actor.actor_id = actor.actor_id
GROUP BY
film.film_id
```

FID	1
title	ACADEMY DINOSAUR
description	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
category	Documentary
price	0.99
length	86
rating	PG
actors	PENELOPE GUINESS, CHRISTIAN GABLE, LUCILLE TRACY, SANDRA PECK, JOHNNY CAGE, MENA TEMPLE, WARREN NOLTE, OPRAH KILMER, ROCK DUKAKIS, MARY KEITEL

3.3.4 The nicer_but_slower_film_list View

电影列表视图，包含了每一部电影的信息及电影所对应的演员。电影对应

的演员以逗号作为分隔符。连接了 film,film_category,category,film_actorandactor 表的数据。和 The film_list View 不同，演员名字只有单词首字母大写了。

FID	1
title	ACADEMY DINOSAUR
description	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
category	Documentary
price	0.99
length	86
rating	PG
actors	Penelope Guinness, Christian Gable, Lucille Tracy, Sandra Peck, Johnny Cage, Mena Temple, Warren Nolte, Oprah Kilmer, Rock Dukakis, Mary Keitel

```

SELECT
film.film_id AS FID,
film.title AS title,
film.description AS description,
category. NAME AS category,
film.rental_rate AS price,
film.length AS length,
film.rating AS rating,
GROUP_CONCAT(
CONCAT(
CONCAT(
UCASE(
SUBSTR(actor.first_name, 1, 1)
),
LCASE(
SUBSTR(
actor.first_name,
2,
LENGTH(actor.first_name)
)
),
_utf8 ' ',
CONCAT(
UCASE(
SUBSTR(actor.last_name, 1, 1)
),
LCASE(

```

```

SUBSTR(
actor.last_name,
2,
LENGTH(actor.last_name)
)
)
)
)
)
) SEPARATOR ', '
) AS actors
FROM
category
LEFT JOIN film_category ON category.category_id =
film_category.category_id
LEFT JOIN film ON film_category.film_id = film.film_id
JOIN film_actor ON film.film_id = film_actor.film_id
JOIN actor ON film_actor.actor_id = actor.actor_id
GROUP BY
film.film_id

```

3.3.5 The sales_by_film_category View

每个电影种类的销售额 ,payment

→rental→inventory→film→film_category→category

category ▼	total_sales
► Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58
Action	4375.85
New	4351.62
Games	4281.33
Foreign	4270.67
Family	4226.07
Documentary	4217.52
Horror	3722.54
Children	3655.55
Classics	3639.59
Travel	3549.64
Music	3417.72

image

```
SELECT
c.name AS category
, SUM(p.amount) AS total_sales
FROM payment AS p
INNER JOIN rental AS r ON p.rental_id = r.rental_id
INNER JOIN inventory AS i ON r.inventory_id = i.inventory_id
INNER JOIN film AS f ON i.film_id = f.film_id
INNER JOIN film_category AS fc ON f.film_id = fc.film_id
INNER JOIN category AS c ON fc.category_id = c.category_id
GROUP BY c.name
ORDER BY total_sales DESC
```

3.3.6 The sales_by_store View

每个商店的 manager 及销售额。

payment → rental → inventory → store → staff

store	manager	total_sales
▶ Woodridge,Australia	Jon Stephens	33726.77
Lethbridge,Canada	Mike Hillyer	33679.79

第七城市 www.7c.cn

```
SELECT
CONCAT(c.city, _utf8',', cy.country) AS store
, CONCAT(m.first_name, _utf8' ', m.last_name) AS manager
, SUM(p.amount) AS total_sales
FROM payment AS p
INNER JOIN rental AS r ON p.rental_id = r.rental_id
INNER JOIN inventory AS i ON r.inventory_id = i.inventory_id
INNER JOIN store AS s ON i.store_id = s.store_id
INNER JOIN address AS a ON s.address_id = a.address_id
INNER JOIN city AS c ON a.city_id = c.city_id
INNER JOIN country AS cy ON c.country_id = cy.country_id
INNER JOIN staff AS m ON s.manager_staff_id = m.staff_id
GROUP BY s.store_id
ORDER BY cy.country, c.city
```

3.3.7 3.2.7. The staff_list View

工作人员的列表

ID	name	address	zip code	phone	city	country	SID
1	Mike Hillyer	23 Workhaven Lane		14033335568	Lethbridge	Canada	1
2	Jon Stephens	1411 Lillydale Drive		6172235589	Woodridge	Australia	2

```

SELECT
s.staff_id AS ID,
CONCAT(
s.first_name,
_utf8 ' ',
s.last_name
) AS NAME,
a.address AS address,
a.postal_code AS `zip code`,
a.phone AS phone,
city.city AS city,
country.country AS country,
s.store_id AS SID
FROM
staff AS s
JOIN address AS a ON s.address_id = a.address_id
JOIN city ON a.city_id = city.city_id
JOIN country ON city.country_id = country.country_id

```

3.4 Stored Procedures

3.4.1 The film_in_stock Stored Procedure

Description

The `film_in_stock` stored procedure is used to determine whether any copies of a given film are in stock at a given store.

判断在一个给定的商店里是否有一个给定电影的 copy.

Parameters

`p_film_id`: The ID of the film to be checked, from the `film_id` column of the `film` table.

`p_store_id`: The ID of the store to check for, from the `store_id` column of the `store` table.

`p_film_count`: An OUT parameter that returns a count of the copies of the film in stock.

Return Values

This procedure produces a table of inventory ID numbers for the copies of the film in stock, and returns (in the `p_film_count` parameter) a count that indicates the number of rows in that table.

```
BEGIN
SELECT inventory_id
FROM inventory
WHERE film_id = p_film_id
AND store_id = p_store_id
AND inventory_in_stock(inventory_id);
SELECT FOUND_ROWS() INTO p_film_count;
END
```



```
mysql> CALL film_in_stock(1,1,pcount);
+-----+
| inventory_id |
+-----+
|          1  |
|          2  |
|          3  |
|          4  |
+-----+
4 rows in set (0.18 sec)

Query OK, 1 row affected (0.20 sec)
```

3.4.2 The `film_not_in_stock` Stored Procedure

和 The `film_in_stock` Stored Procedure 相反，判断给定的电影，是否有 copy，不在一个给定的商店

Description

The `film_not_in_stock` stored procedure is used to determine whether there are any copies of a given film not in stock (rented out) at a given store.

Parameters

`p_film_id`: The ID of the film to be checked, from the `film_id` column of the `film` table.

`p_store_id`: The ID of the store to check for, from the `store_id` column of the `store` table.

`p_film_count`: An OUT parameter that returns a count of the copies of the film not

in stock.

Return Values

This procedure produces a table of inventory ID numbers for the copies of the film not in stock, and returns (in the `p_film_count` parameter) a count that indicates the number of rows in that table.

```
BEGIN
SELECT inventory_id
FROM inventory
WHERE film_id = p_film_id
AND store_id = p_store_id
AND NOT inventory_in_stock(inventory_id);
SELECT FOUND_ROWS() INTO p_film_count;
END
```

```
mysql> CALL film_not_in_stock(2,2,@count);
```

```
+-----+
| inventory_id |
+-----+
|          9 |
+-----+
1 row in set (0.03 sec)
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> SELECT @count;
```

```
+-----+
| @count |
+-----+
|       1 |
+-----+
1 row in set (0.00 sec)
```

第七城市 www.7c.cn

3.4.3 The rewards_report Stored Procedure

过去一个月，数量大于输入参数 `min_monthly_purchases`，消费金额大于输入参数 `min_dollar_amount_purchased` 的客户

Description

The `rewards_report` stored procedure generates a customizable list of the top customers for the previous month.

Parameters

`min_monthly_purchases`: The minimum number of purchases or rentals a customer needed to make in the last month to qualify.

min_dollar_amount_purchased: The minimum dollar amount a customer needed to spend in the last month to qualify.

count_rewardees: AnOUTparameter that returns a count of the customers who met the qualifications specified.

Return Values

This function returns a table of customers who met the qualifications specified. The table has the same structure as thecustomertable. The procedure also returns (in thecount_rewardeesparameter) a count that indicates the number of rows in that table.

Stop Service net stop uxsms

Start Service net start uxsms

Disable Service sc config uxsms start= disabled


Enable Service sc config uxsms start= auto

```
proc: BEGIN
DECLARE last_month_start DATE;
DECLARE last_month_end DATE;
/* Some sanity checks... */
IF min_monthly_purchases = 0 THEN
SELECT 'Minimum monthly purchases parameter must be > 0';
LEAVE proc;
END IF;
IF min_dollar_amount_purchased = 0.00 THEN
SELECT 'Minimum monthly dollar amount purchased parameter must be >
$0.00';
LEAVE proc;
END IF;
/* Determine start and end time periods */
SET last_month_start = DATE_SUB(CURRENT_DATE(), INTERVAL 1
MONTH);
SET last_month_start = STR_TO_DATE(CONCAT(YEAR(last_month_start),'-
',MONTH(last_month_start),'-01'),'%Y-%m-%d');
SET last_month_end = LAST_DAY(last_month_start);
/*
Create a temporary storage area for
```

```

Customer IDs.
*/
CREATE TEMPORARY TABLE tmpCustomer (customer_id SMALLINT
UNSIGNED NOT NULL PRIMARY KEY);
/*
Find all customers meeting the
monthly purchase requirements
*/
INSERT INTO tmpCustomer (customer_id)
SELECT p.customer_id
FROM payment AS p
WHERE DATE(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 */
SELECT COUNT(*) FROM tmpCustomer INTO count_rewardees;
/*
Output ALL customer information of matching rewardees.
Customize output as needed.
*/
SELECT c.*
FROM tmpCustomer AS t
INNER JOIN customer AS c ON t.customer_id = c.customer_id;
/* Clean up */
DROP TABLE tmpCustomer;
END

```



```

mysql> CALL rewards_report(7,20.00,0count);
Empty set (0.24 sec)

Query OK, 0 rows affected (0.26 sec)

```

第七城市 www.7city.cn

因为不满足系统时间，所以结果为 0

3.5 Stored Functions

3.5.1 The get_customer_balance Function

返回指定客户到某一个日期欠的钱。所有的 rental_rate+超期费用-所有的 payment

Description

The `get_customer_balance` function returns the current amount owing on a specified customer's account.

Parameters

`p_customer_id`: The ID of the customer to check, from the `customer_id` column of the `customer` table.

`p_effective_date`: The cutoff date for items that will be applied to the balance. Any rentals, payments, and so forth after this date are not counted.

Return Values

This function returns the amount owing on the customer's account.

```
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
DECLARE v_rentfees DECIMAL(5,2); #FEES PAID TO RENT THE VIDEOS
INITIALLY
DECLARE v_overfees INTEGER; #LATE FEES FOR PRIOR RENTALS
DECLARE v_payments DECIMAL(5,2); #SUM OF PAYMENTS MADE
PREVIOUSLY
SELECT IFNULL(SUM(film.rental_rate),0) INTO v_rentfees
FROM film, inventory, rental
WHERE film.film_id = inventory.film_id
AND inventory.inventory_id = rental.inventory_id
AND rental.rental_date <= p_effective_date
AND rental.customer_id = p_customer_id;
SELECT IFNULL(SUM(IF((TO_DAYS(rental.return_date) -
TO_DAYS(rental.rental_date)) > film.rental_duration,
((TO_DAYS(rental.return_date) - TO_DAYS(rental.rental_date)) -
```

```

film.rental_duration),0)),0) INTO v_overfees
    FROM rental, inventory, film
    WHERE film.film_id = inventory.film_id
    AND inventory.inventory_id = rental.inventory_id
    AND rental.rental_date <= p_effective_date
    AND rental.customer_id = p_customer_id;SELECT
IFNULL(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

```

3.5.2 The inventory_held_by_customer Function

返回指定的 inventory 现在在那个客户的 customer_id

Theinventory_held_by_customerfunction returns thecustomer_idof the customer who has rented out the specified inventory item.

Parameters

p_inventory_id: The ID of the inventory item to be checked.

Return Values

This function returns thecustomer_idof the customer who is currently renting the item, orNULLif the item is in stock.

```

BEGIN
DECLARE v_customer_id INT;
DECLARE EXIT HANDLER FOR NOT FOUND RETURN NULL;
SELECT customer_id INTO v_customer_id
FROM rental
WHERE return_date IS NULL
AND inventory_id = p_inventory_id;
RETURN v_customer_id;
END

```

3.5.3 The inventory_in_stock Function

返回指定的 inventory 是否在商店里

Theinventory_in stock functionreturns a boolean value indicating whether the inventory item specified is in stock.

Parameters

p_inventory_id: The ID of the inventory item to be checked.

Return Values

This function returns TRUE or FALSE to indicate whether the item specified is in stock.

```
BEGIN
DECLARE v_rentals INT;
DECLARE v_out INT;
#AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental
TABLE
#FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED
SELECT COUNT(*) INTO v_rentals
FROM rental
WHERE inventory_id = p_inventory_id;
IF v_rentals = 0 THEN
RETURN TRUE;
END IF;
SELECT COUNT(rental_id) INTO v_out
FROM inventory LEFT JOIN rental USING(inventory_id)
WHERE inventory.inventory_id = p_inventory_id
AND rental.return_date IS NULL;
IF v_out > 0 THEN
RETURN FALSE;
ELSE
RETURN TRUE;
END IF;
END
```

3.6 Triggers

3.6.1 The customer_create_date Trigger

插入新客户时，设置 create_date 为当前日期

-- Trigger to enforce create dates on INSERT

```
CREATE TRIGGER customer_create_date BEFORE INSERT ON customer
FOR EACH ROW SET NEW.create_date = NOW();
```

3.6.2 The payment_date Trigger

payment 新增记录时，payment_date 设置为当前日期

-- Trigger to enforce payment_date during INSERT

```
CREATE TRIGGER payment_date BEFORE INSERT ON payment
```

```
FOR EACH ROW SET NEW.payment_date = NOW();
```

3.6.3 The rental_date Trigger

rental 表新增记录时，rental_date 设置为当前日期

-- Trigger to enforce rental_date on INSERT

```
CREATE TRIGGER rental_date BEFORE INSERT ON rental
FOR EACH ROW SET NEW.rental_date = NOW();
```

3.6.4 The ins_film Trigger

确保 film_text 和 film 同步插入

```
CREATE TRIGGER `ins_film` AFTER INSERT ON `film` FOR EACH ROW
BEGIN
    INSERT INTO film_text (film_id, title, description)
    VALUES (new.film_id, new.title, new.description);
END;;
```

3.6.5 The upd_film Trigger

确保 film_text 和 film 同步更新

```
CREATE TRIGGER `upd_film` AFTER UPDATE ON `film` FOR EACH
ROW BEGIN
    IF (old.title != new.title) OR (old.description != new.description) OR
    (old.film_id != new.film_id)
    THEN
        UPDATE film_text
        SET title=new.title,
        description=new.description,
        film_id=new.film_id
        WHERE film_id=old.film_id;
    END IF;
END;;
```

3.6.6 The del_film Trigger

确保 film_text 和 film 同步删除

```
CREATE TRIGGER `del_film` AFTER DELETE ON `film` FOR EACH ROW
BEGIN
    DELETE FROM film_text WHERE film_id = old.film_id;
END;;
```