sakila数据库

sakila数据库是MySQL给出的可以用于教导演示用的样例数据库

以下，是笔者的一些翻译以及笔记：

可以在官网阅读该文档的声明:

Sakila Sample Database (mysql.com)

注意事项!

有一些异常不符合文档里面描述的现象可能是因为数据库版本落后，有些属性是更新版本才支持的

比如sakila.address.loaction的属性

在翻译中对于这部分内容不会做翻译，请自行翻译官方文档

生词本

1. integrity:完整性
2. compatity:兼容性
3. surrogate primary key:代理主键

   英文解释为:

   ```
   In a database, a surrogate primary key is a system-generated identifier used
   as the primary key for a table.
   It is not derived from any data in the table,
   but rather is generated by the system, often as an integer value or a UUID.
   The purpose of a surrogate key is to provide a simple, unique identifier
   that can be used to refer to a specific record in the table, without relying
   on the values of any of its attributes.
    Surrogate keys can be useful in situations where the natural keys of a
   table are complex or may change over time, as well as in situations where a
   table may not have a clear candidate key.
   ```

   可以理解为

   ```
   在一个数据库中代理主键时一个系统自动生成的作为表的
   主键的标识符。
   它不能够通过表中任何数据推导得到，而是系统自动生成的，总是一个
   整数值或者UUID (Universally Unique Identifier,通用唯一标识符)
   代理键的目的是用来提供一个简单的独特的用来唯一指向表中一个确定的
   记录的标识符，而且不依赖于这个记录的任何其他属性。
   当表中自然的主键复杂或者可能随时间改变的时候，或者表可能没有一个清晰
   的候选键的时候，代理键是有用的。
   ```

4. inventory和stock的区别:

   inventory范围更大，

   在这里，inventory表示存储商店拥有的碟片的记录，及时碟片被出租出去了，无论是否已经被归还，都在inventory表中

   copy of film in stock则表示手里还可以被出租出去的或者可以被售卖出去的碟片的数量

sakila数据库结构总览

sakila的内容主要可以分为:

1. 表
2. 视图
3. 存储过程
4. 存储函数
5. 触发器

表

1. actor

   该表罗列了演员的信息

   actor表通过film_acotr表连接到film表

   - actor_id:演员id
   - 演员的名
   - 演员的姓
   - last_update，这个元组创造的时间address 表

2. address

   address表包括客户，职工和商店的信息

   address表的主键是customer,staff,store的外键

   - address_id:一个自然主键用来唯一标志地点
   - address:地址的第一行
   - address2:地址的第二行,(可选，也就是可以为空)
   - district:地址所在的地域，可能是一个区，省，或者
   - city_id:city表的外键

- postal_code:TODO
- phone:地址对应的电话号码
- last_update:该行创建的时间或者最后一次更新时间
- location:一个地理上的空间坐标

3. category

   category表罗列了电影的分类目录

   category表可以通过film_category表连接到film表

   字段说明:

   - category_id:目录id，自然主键，用来唯一区分一个目录
   - name:目录的名字
   - last_update:最后一次更新的时间

4. city

   city表罗列了城市

   city表的主键是address表的外键，同时它通过含有country表的主键外联country表

   列说明：

   - city_id:自然主键，城市id，唯一标识表中的城市
   - city:城市名
   - country_id:国家id
   - last_update:该列创建的时间或者最后一次更新的时候

5. country

   国家表。被city表通过外键外联

   - country_id:国家id。为自然主键用来唯一标识表中的每个国家
   - country:国家名
   - last_update:

6. customer:

   包含所有客户的列表

   被payment 和rental表通过外键customer_id关联，

   并且通过外键address_id关联address，通过外键store_id关联store表

- customer_id:客户id。代理主键，唯一标识表中的客户
- store_id:标识客户的"家 商店"的外键。这个商店id虽然被赋予了这个客户，但是这个客户也可以在其他的商店购物。
- first_name:客户的名
- last_name:客户的姓
- email:客户的电子邮箱地址
- address_id:一个用来关联address表的外键，地址id
- active:表明这个客户是否是一个活跃客户的标记量。设置这个量为false来代替显式地删除这个客户。查询时可以用where active=true 条件来忽视这个标记为删除的客户。
- create_date:这个客户被加入到系统中时的时间。这个日期使用一个触发器在插入操作时自动设置
- last_update:该行第一次被创建的时间，或者最近一次更新的时间

7. film

   电影表。

   film表是所有商店里面可能存在的电影的列表。实际还有存货的电影被放到inventory列表中

   电影表外联language表，并且被film_category，film_actor，inventory这三个表外联

   列：

   - film_id:代理主键。用来唯一区分表中的每一个电影
   - title:电影名
   - description:一个关于这个电影的简要描述或者情节总结
   - release_year:这个电影发型的年份
   - language_id:用来指向语言表的外键。标识这个电影的语言
   - original_language_id:一个指向语言表的外键。标识了这个电影原本版本使用的语言。单一个电影被翻译称其他语言的时候使用这个字段
   - rental_duration:这个电影被租借的时间长度，单位为天
   - rental_rate:租界这个电影rental_duration指定的天数需要的花费
   - length:这个电影的时长，单位分钟
   - replacement_cost:电影预期未还或者还时损坏的补偿金额。
   - rating:电影评级。在集合{G,PG,PG-13,R或者NC-17)中
   - specail_features:电影特点。列举了这个dvd的更多属性，这些属性可以是{Trailers,Commentaryies,Deleted Scens,Behind the Screen}中的一个或多个

- last_update:最新更新时间

8. film_actor表

   电影-演员表被用来支持电影和演员之间的多对多联系，对于给定电影的每个演员，都会有一个对应的元组保存在表中

   电影演员表通过外键外联film表和actor表

   - actor_id:用来标识演员的外键
   - film_id:用来标识电影的外键
   - last_update:最新更新时间

9. film_category

   该表用来支持电影和电影分类目录之间的多对多关联。对于每对电影和目录之间的关联，在该表中存在一行与之对应

   电影-分类目录外联film和category表

   列:

   - film_id
   - category_id
   - last_update

10. film_text

    film_text 表包括电影编号，影名以及描述

    - film_id:代理主键唯一标识一个影片文本
    - title:电影名
    - description:简介

    这个表的值永远不能够直接修改，而是通过触发器从film表来获得改变

11. inventory

    库存表。

    库存表给每个商店拥有的每个电影的每张碟片都记录了一行。（包括借出的和没有借出的）。

    库存表外联film表和store表并且被rental表外联。

    - inventory_id:库存号
    - film_id
    - store_id

- last_update

12. language

    语言表。

    语言表展示了电影可能有的所有语言，

    语言表被电影表外联

    - language_id:唯一区分表中每个语言的代理主键
    - name:语言的英文名
    - last_update

13. payment

    支付表记录了客户的每次支付，包括账单金额和租单

    payment表外联customer表，rental表，staff表

    - payment_id:支付号
    - customer_id:客户号
    - staff_id:职工号
    - rental_id:租赁号
    - amount:金额
    - payment_date:支付日期
    - last_update:最新更新时间

14. rental

    租赁表。

    每行含信息:什么人什么时候租借了什么物项什么时候归还以及哪个职工处理的

    行:

    - rental_id:租赁号
    - rental_date:租借日期
    - inventory_id:库存号（或者对应库存中这个物品的编号
    - customer_id:客户号
    - return_date:归还日期，该字段可以为NULL，为NULL表示该租借记录租借的书没有归还。
    - staff_id:处理该次租借的职工号

- last_update:最新更新时间

15. staff

    职工表

    职工表罗列了所有职工成员，包括他们的电子邮箱，注册信息，以及照片。

    职工表外联store表和address表，并且被rental，paymen，store表外联

    - staff_id:职工号。唯一标识职工的代理主键
    - first_name:职工的名
    - last_name:职工姓
    - address_id:外联地址表用的外键
    - picture:一个用来包含雇员信息的图片的BLOB
    - email:电子邮箱
    - store_id:职工的注册商店。虽然这个商店号被赋予了这个员工，但是这个员工可以也可以在其他商店工作。
    - active:用来判断这个雇员是否活跃。如果这个雇员离开了，对应的行不会被删除而是会设置这个字段的属性为false
    - username:用户名，员工用来访问租借系统的用户名
    - password:访问密码
    - last_update:最新更新时间

16. store

    商店表。

    store表罗列了系统中所有商店。所有的库存都与一个特定的店铺绑定到一起，所有客户和员工都和一个"家商店"绑定在一起。

    商店表外联职工表和地址表，并且被职工表，客户表，库存表外联。

    - store_id：商店号。唯一标识商店的代理主键。
    - manager_staff_id：管理者号。一个用来标识这个商店的管理者的外键。
    - address_id:商店的地址号。外键，用来外联商店表
    - last_update:最新更新时间

视图

这个部分会描述sakila样例数据库中包含的视图

1. actor_info

   actor_info视图罗列了各个电影分类下演员和他们参演过的电影的信息

   staff_list视图组织了来自film, actor,category,film_actor,film_category表的数据

2. customer_list

   该视图提供了客户，名，姓以及地址等信息。

   组织了来自客户表，地址表，城市表，国家表的信息

3. film_list

   film_list视图提供了film表的一个格式化视图。并且每个电影给出参演该电影的演员名逗号枚举表达式

   film_list视图组织了来自film,category,film_category，actor以及film_actor表的数据

4. nicer_but_slower_film_list

   nicer_but_slower_film_list视图包含了film表的格式化，包含每个电影的演员列表

   该视图不同于film_llist视图的地方在于演员列表。演员名字的字母大小写是自适应的，因此每个名字的第一个字母都是大写，而不是所有名字都是全部字母都大写的

   正如在视图名中表达的一样，nicer_but_slower_film_list视图进行了更多处理，因此花费比film_list更长的时间去返回信息

   该视图组织了来自film，category,film_category,actor,以及film_actor表的数据

5. sales_by_film_category

   该视图提供了每个电影分类目录下交易总额的列表

   因为一个电影可能被分类进入多个目录中，不可能通过累加所有电影分类目录的交易总额来获取所有电影的交易总额

   sales_by_film_category视图组织了来自category,payment,rental,inventory,film,film_category以及category表的数据

6. sales_by_store

   该视图提供了每个商店的交易总额

   这个视图返回商店地址，管理者名字，以及交易总额。

   该视图组织了来自city,country,payment,rental,inventory,store,address,以及staff表的数据

7. staff_list

   该视图提供了店员列表，包括他们的地址以及赋予的店铺信息（如前文，赋予店员的商店并不一定是店员正在工作的商店）

   该视图组织了来自staff表和address表的信息

存储过程

下面以字符顺序描述sakila数据库中的存储过程

除非另外说明所有列举的参数都是IN类型的参数

(ps，参数类型有:IN,OUT,INOUT)

1. film_in_stock

   描述

   > 该存储过程能获得指定商店的指定电影的垫片可以出租的数量

   参数:

   - p_film_id:待检查的电影的id，来自film表的film_id字段
   - p_store_id:待检查的商店的id，来自store表的store_id字段
   - p_film_count:返回类型的参数，符合要求的库存记录的数量

   返回值:

   > 这个存储过程提供了一个符合要求的库存记录id的表，并且通过p_film_count参数
   > 返回了哪个表的列数

   使用样例

```
mysql> CALL film_in_stock(1,1,@count);
+--------------+
| inventory_id |
+--------------+
|            1 |
|            2 |
|            3 |
|            4 |
+--------------+
4 rows in set (0.01 sec)
Query OK, 1 row affected (0.01 sec)
mysql> SELECT @count;
+--------+
| @count |
+--------+
|      4 |
+--------+
1 row in set (0.00 sec)
```

2. film_not_in_stock存储过程

   描述:

   > 该存储记录能够判断指定商店是否有指定电影的可以出租或者
   > 出售的任何碟片

   参数:

   - p_film_id:指定电影的id,参照film表的film_id字段
   - p_store_id:指定商店的id,参照store表的store_id字段
   - p_film_count:可返回参数，用来返回不能够用来出租i或者购买的碟片的数量

   返回值

   > 这个存储过程产生一个所有不能够用来盈利的电影库存id的表，
   > 并且通过p_film_count参数返回表的行数，也就是指定商店拥有的
   > 指定电影的不能够用来盈利的碟片的数量(比如被出租出去的碟片)

   用例

```
mysql> CALL film_not_in_stock(2,2,@count);
+--------------+
| inventory_id |
+--------------+
|            9 |
+--------------+
1 row in set (0.01 sec)
Query OK, 1 row affected (0.01 sec)
mysql> SELECT @count;
+--------+
| @count |
+--------+
|      1 |
+--------+
1 row in set (0.00 sec)
```

3. rewards_report

   描述

   > 该存储过程产生近期优质客户的列表

   参数

- min_monthly_purchases:

  优质客户最近一个月的最少消费次数

- min_dollar_amount_purchases:

  优质客户最近一个月消费的最少金额

- count_rewardees:返回类型的参数，用来返回达到标准的优质客户数量。

返回值

```
该存储过程提供一个所有符合参数要求的客户的列表，列和
customer表一样。同时通过count_rewardees参数返回符合
要求的客户数量
```

用例

```
mysql> CALL rewards_report(7,20.00,@count);
...
| 598         | 1        | WADE        | DELVALLE     |
WADE.DELVALLE@sakilacustomer.org         | 604        | 1      | 2006-02-24
10:48:30 | 2006-02-15 04:57:20 |
| 599         | 2        | AUSTIN      | CINTRON      |
AUSTIN.CINTRON@sakilacustomer.org        | 605        | 1      | 2006-02-24
10:48:30 | 2006-02-15 04:57:20 |
...
42 rows in set (0.11 sec)
Query OK, 0 rows affected (0.67 sec)
mysql> SELECT @count;
+--------+
| @count |
+--------+
|     42 |
+--------+
1 row in set (0.00 sec)
```

存储函数

这节描述了sakila数据库中用到的存储函数

1. get_customer_balance

   该函数返回指定客户账号的近期消费金额

   参数:

   1. p_customer_id:指定客户id的参数，参照customer表的customer_id字段
   2. p_effective_date:用来统计金额的截至日期，超过这个日期的所有交易不被统计

返回值

```
返回这个客户账号下的消费金额
```

用例:

```
select get_customer_balance(298,now());
```

2. inventory_held_by_customer

该函数返回借走了指定库存碟片的客户的id

参数:

1. p_inventory_id:给碟片分配的库存id

返回值

```
这个函数返回最近租走指定库存碟片的客户的id，如果碟片没有被
租走的话，返回NULL.(如果没有没有没有对应该id的碟片记录的话
，也返回NULL)
```

用例

```
SELECT inventory_held_by_customer(8);
+------------------------------+
| inventory_held_by_customer(8) |
+------------------------------+
|                         NULL |
+------------------------------+
1 row in set (0.00 sec)
mysql> SELECT inventory_held_by_customer(9);
+------------------------------+
| inventory_held_by_customer(9) |
+------------------------------+
|                          366 |
+------------------------------+
1 row in set (0.00 sec)
```

3. inventory_in_stock

该函数用来判断指定的库存碟片(这里指商店拥有的碟片)是否没有被租出去

参数

1. p_inventory_id:要检查的库存记录的库存id

返回值

```
函数返回true表明库存id标识的碟片还没有租出去
函数返回flase表明该碟片已经被租出去
```

用例:

```
mysql> SELECT inventory_in_stock(9);
+-----------------------+
| inventory_in_stock(9) |
+-----------------------+
|                     0 |
+-----------------------+
1 row in set (0.00 sec)
mysql> SELECT inventory_in_stock(8);
+-----------------------+
| inventory_in_stock(8) |
+-----------------------+
|                     1 |
+-----------------------+
1 row in set (0.00 sec)
```

补充说明:

```
#使用下面代码阅读创建该函数的语句
show create function inventory_in_stock;
#思考下如果使用一个不在inventory表中的inventory_id作为参数
#会发生什么?返回的结果是什么
```

结果返回了1(代表true),但是实际上这个库存记录并不存在，这个true并不说明该id对应一个可以被出租的碟片。换句话说，这个函数并不安全，没有检查这个id是否是合法的inventory_id.

触发器

该节描述了Sakila样例数据库中的触发器

1. customer_create_date

   该触发器在把元组插入customer表时，设置元组的create_date字段值为当前时间

2. payment_date

   payment_date触发器在插入payment表的时候设置插入元组的payment_date属性为当前时间

3. rental_date

   插入rental表的时候设置插入元组的payment_date属性为当前时间

4. ins_film

   在插入film表的时候在film_text表中插入对应的行

5. upd_film

   更新film表的时候更新film_text对应的列

6. del_film

   删除film表中元组的时候删除film_text表中对应的元组

使用示例

1. rent a dvd:租借碟片

```
#判断id为10的碟片是否被租出去
select inventory_in_stock(10);
#往租表rental中插入一个
#在现在客户10在员工1处理下租用库存碟片10的记录
insert into rental(rental_date,inventory_id,customer_id,staff_id)
values (now(),10,3,1);
#创建变量并分别赋予最新插入id的值以及最近，客户3租借碟片的金额
set @rentID=last_insert_id(),
@balance=get_customer_balance(3,now());
#查看@rentID,@balance的值
select @rentID,@balance;
#往支付表中加入该次支付信息
insert into payment (customer_id,staff_id,rental_id,amount,payment_date)
values (3,1,@rentID,@balance,NOW());
```

2. return a dvd:归还碟片

   为了归还碟片，我们需要更新rental表，设置碟片对应租借记录的归还时间。为了做这个，我们首先需要找到要归还碟片绑定的inventory_id找到对应的租借记录id。通过这种方式归还，可能有必要检查下客户的时间余额并且可能要处理一次交易，比如逾期赔偿

```
#从表中选择一个之前我们租碟片的记录的id放到@rentalId变量中
select rental_id form rental
where inventory_id=10
and customer_id=2 and return_date=NULL
into @rentID;
#查看获取租借记录id的结果
select @rentID;
#更新rental表的归还时间为现在
update rental set return_date=now()
where rental_id=@rentID;
```

3. Find Overdue DVDs:查找逾期碟片

```
 SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
           address.phone, film.title
           FROM rental INNER JOIN customer ON rental.customer_id =
customer.customer_id
           INNER JOIN address ON customer.address_id = address.address_id
           INNER JOIN inventory ON rental.inventory_id =
inventory.inventory_id
           INNER JOIN film ON inventory.film_id = film.film_id
           WHERE rental.return_date IS NULL
           AND rental_date + INTERVAL film.rental_duration DAY <
CURRENT_DATE()
           ORDER BY title
           LIMIT 5;
```

重要问题

Sakila样例数据库的设计中假设指定商店的员工成员只从那个商店把碟片租给顾客，而不从其他商店。

这个假设表现了rental,inventory,staff,以及store四张表有着循环依赖关系。一个客户仅仅有一个商店的信息，但是员工不能使用同样的约束。

如果一个员工从其他的商店出租物品，在rental表中的数据就会散失一致性。

解决这个问题的方式留给读者思考，下面是一些可能的方式:

- 添加一个store_id列到rental表中并且同样设置该列为外键来保证不仅仅customer_id以及inventory_id，inventory表中的staff_id列同样对应相同的商店
- 给rental表添加插入(insert)触发器和更新(update)触发器

声明

略

Sakila样例数据库的开源许可

略

给作者的建议

略

Sakila历史版本

略