**游泳馆会员管理系统**

**设计说明书**

目 录

**一、需求分析** 3

1.概述 3

1.1. 目标用户 3

1.2. 主要功能 3

1.3. 问题描述 3

2. 需求分析 4

2.1. 数据流图 4

2.2. 数据字典 5

**二、概念结构设计**  6

1. 概念模型 6

2. 数据完整性约束 6

**三、物理结构设计**  7

1. 物理模型（PDM） 7

2. 生成SQL Server数据库 7

**四、T-SQL源程序分析和测试结果** 8

1. 存储过程 8

2. 触发器 13

3. 函数 14

**五、小结** 17

**六、参考文献** 18

**详 细 程 序 设 计 书**

**一、需求分析**

**1.概述**

一个会员制的游泳馆希望通过计算机实现会员管理。

本系统设计主要面向操作员。操作员可以对会员进行登记，销售实体会员卡。由操作员读取会员卡卡号并登记存款金额，实现存款的记录。在出场馆时系统或操作员可读取会员卡卡号，并按照卡片类型获取折扣倍数，实现消费的记录。操作员可以为会员办理退卡、挂失等禁用功能，同时交易记录仍会保留在系统内。会员和操作员都可以对存款、消费、禁用记录进行查询。操作员在销售会员卡时，可以自行设置折扣大小。

**1.1目标用户**

（1）操作员（系统用户）

（2）会员

**1.2主要功能**

（1）售卡管理：销售会员卡时要输人卡号、金额、折扣、类型（储值卡、折扣卡）以及持卡人基本资料（姓名、电话、证件号等）等信息。这些信息记录到会员资料中。

（2）会员资料管理：查询和修改会员资料（其中“卡号”和“金额”两项不能修改）。可按照卡号、持卡人姓名查询信息。

（3）存款管理：如果会员卡类别是储值卡，可根据卡号进行预存款操作。

（4）退卡管理：执行退卡操作后该卡不可再使用，但保留该卡以前的消费记录。

（5）会员禁用：用来挂失或是停止会员卡的使用（记录禁用原因），实现对会员卡的安全管理。可以通过解禁功能恢复会员卡的正常使用。

（6）消费管理：记录会员的消费情况，可以查看会员的消费金额、存款金额、结存余额，可以查询任何一段时间内该卡的消费情况，也可以查询所有会员卡汇总记录。

（7）系统管理：包括参数设置、权限设置、更改密码等。

**1.3安全性要求**

保密性：只有操作员通过工号或姓名、口令登录系统，完成售卡、存款、禁用等操作。

完整性：限制修改卡号、金额，保证数据的完整，防止非法篡改记录。

可用性：数据库通过SQL Server软件进行记录日志和备份，保证可用性。

可控性：设置不同用户的访问需求，例如admin是所有权限，其他工号会有限制。

可存活性：通过SQL Server软件可以有效保护数据库被攻击。

**2.需求分析**

**2.1数据流图**

**A.顶层数据流图**

****

图1.2.1.1 游泳馆会员管理系统数据流图-1

**B.一层数据流图**

****

图1.2.1.2 游泳馆会员管理系统数据流图-2

**C.二层数据流图**

****

图1.2.1.3 游泳馆会员管理系统数据流图-3

**2.2数据字典**

1. **会员卡表**

**会员卡表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 卡号 | ID | char(10) | 10 | TRUE | FALSE |
| 参数号 | PID | char(5) | 5 | FALSE | TRUE |
| 姓名 | Name | varchar(20) | 20 | FALSE | FALSE |
| 手机号 | PhoneNumber | char(11) | 11 | FALSE | FALSE |
| 性别 | Sex | char(2) | 2 | FALSE | FALSE |
| 生日 | Birth | date |  | FALSE | FALSE |
| 类型 | Type | char(6) | 6 | FALSE | FALSE |
| 金额 | Balance | numeric(6,2) | 6 | FALSE | FALSE |
| 状态 | Status | bit |  | FALSE | FALSE |

1. **参数表**

**参数表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 参数号 | PID | char(5) | 5 | TRUE | FALSE |
| 参数名 | PName | char(20) | 20 | FALSE | FALSE |
| 折扣 | PValue | float(10) | 10 | FALSE | FALSE |

1. **存款表**

**存款表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 卡号 | ID | char(10) | 10 | TRUE | TRUE |
| 记录号 | RecordID | int |  | TRUE | TRUE |
| 存款金额 | DepositAmount | numeric(6,2) | 6 | FALSE | FALSE |
| 存款时间 | DepositDate | smalldatetime |  | FALSE | FALSE |

1. **操作员表**

**操作员表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 工号 | Employee\_ID | char(5) | 5 | TRUE | FALSE |
| 姓名 | Employee\_Name | varchar(20) | 20 | FALSE | FALSE |
| 口令 | Employee\_Code | char(4) | 4 | FALSE | FALSE |

1. **消费表**

**消费表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 卡号 | ID | char(10) | 10 | TRUE | TRUE |
| 记录号 | RecordID | int |  | TRUE | TRUE |
| 消费金额 | ExpenseAmount | numeric(6,2) | 6 | FALSE | FALSE |
| 消费时间 | ExpenseDate | smalldatetime |  | FALSE | FALSE |

1. **禁用表**

**禁用表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 卡号 | ID | char(10) | 10 | TRUE | TRUE |
| 记录号 | RecordID | int |  | TRUE | TRUE |
| 工号 | Employee\_ID | char(5) | 5 | TRUE | TRUE |
| 禁用原因 | Disable\_Reason | varchar(10) | 10 | FALSE | FALSE |
| 禁用时间 | Disable\_Date | smalldatetime |  | FALSE | FALSE |

1. **记录表**

**记录表的栏的清单**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 长度 | 主要的 | 外来键 |
| 记录号 | RecordID | int |  | TRUE | FALSE |
| 卡号 | ID | char(10) | 10 | FALSE | TRUE |
| 类别 | Category | char(5) | 5 | FALSE | FALSE |

**二、概念结构设计**

**1.概念模型（CDM图）**

****

图2.1.1 游泳馆会员管理系统CDM图

**2.数据完整性约束说明**

1. 会员卡表

|  |  |  |
| --- | --- | --- |
| 类别 | 属性 | 说明 |
| 实体完整性 | 卡号 | 卡号是会员卡表的主属性，不能取空值。 |
| 参照完整性 | 参数号 | 参数号是会员卡表的外码，决定会员卡类型、折扣。 |
| 用户定义完整性 |  | 姓名、手机号、类型、金额等不能取空值；  状态：布尔型；金额：长度6位、精度2位。 |

1. 参数表

|  |  |  |
| --- | --- | --- |
| 类别 | 属性 | 说明 |
| 实体完整性 | 参数号 | 参数号是参数表的主属性，不能取空值。 |
| 参照完整性 | 无 | 参数表的外码为空。 |
| 用户定义完整性 |  | 折扣值使用float型。 |

1. 操作员表

|  |  |  |
| --- | --- | --- |
| 类别 | 属性 | 说明 |
| 实体完整性 | 工号 | 工号是操作员表的主属性，不能取空值。 |
| 参照完整性 | 无 | 操作员表的外码为空。 |
| 用户定义完整性 |  | 姓名、口令不能取空值。 |

1. 记录表

|  |  |  |
| --- | --- | --- |
| 类别 | 属性 | 说明 |
| 实体完整性 | 记录号 | 记录号是记录表的主属性，不能取空值。 |
| 参照完整性 | 卡号 | 卡号是记录表的外码，关联存款、消费、禁用的卡号。 |
| 用户定义完整性 |  | 类别不能为空值，记录号使用int型并可自增长。 |

1. 存款表、消费表、禁用表

三张表没有主属性。有参照完整性，与会员卡表和记录表存在参照关系。记录号和卡号作为参照，完成存款、消费、禁用的记录。在用户定义完整性上，存款时间、消费时间、禁用时间采用datetime类型。金额使用num(6,2)，长度6位，精度2位。

**三、物理结构设计**

**1.物理模型（PDM）**

****

图3.1.1 游泳馆会员管理系统PDM图

**2.生成SQL Server数据库**

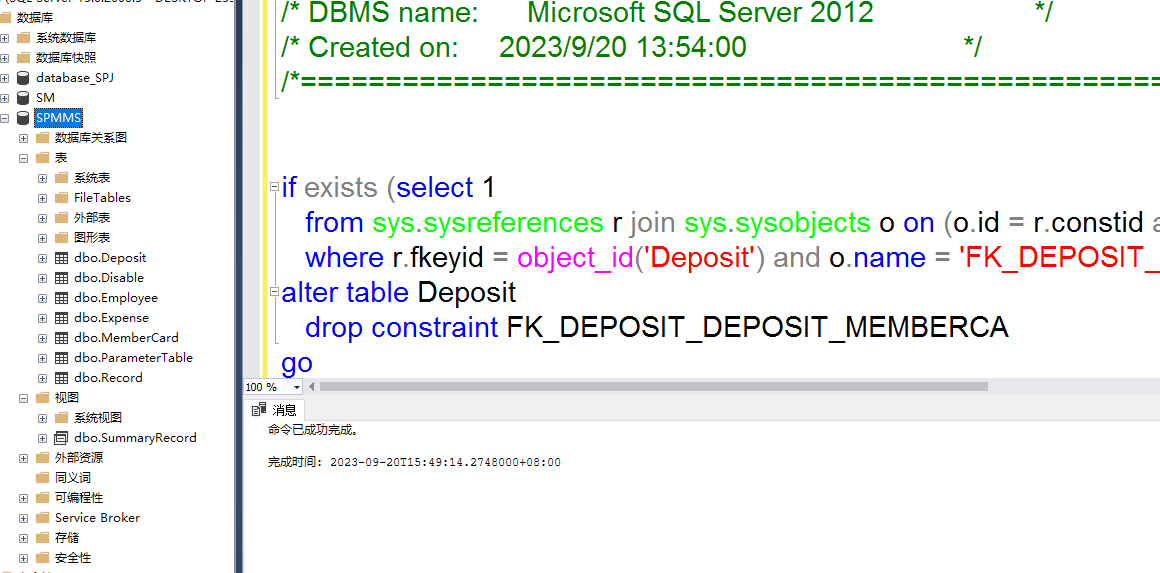


图3.1.2 生成游泳馆会员管理系统数据库

**四、部分T-SQL源程序及分析和测试结果**

**1.存储过程**

**1.1 会员存款的存储过程**

**分析：由于存款会影响多张表，所以需要通过存储过程完成。在输入存款卡号和存款金额后，首先判断该操作是否合法。通过存款金额大小、卡号存在、卡号非禁用等三个方面判断，否则报错退出程序。其次按照逻辑顺序修改各个表的内容。记录表获取流水单号，应用到存款表后，更新会员卡表金额。最后提交事务，完成存款。**

**代码：**

CREATE PROCEDURE Depositing (

@ID CHAR(10),

@Amount NUMERIC(6,2)

)

AS

BEGIN

BEGIN TRANSACTION; -- 开始事务

DECLARE @RecordID INT; -- 声明一个变量来存储流水号

DISABLE TRIGGER CardDataSecurity ON MemberCard;

-- 验证操作是否合法

-- 甲：存款金额必须为正数

if @Amount = 0 OR @Amount = 10000

BEGIN

RAISERROR('存款金额必须大于零，且单次金额需小于一万元。', 16, 1);

ROLLBACK TRANSACTION;

return; -- 退出存储过程

END

-- 乙：会员卡必须存在

if not exists (select ID from MemberCard where ID = @ID)

BEGIN

RAISERROR('无效的会员卡，请检查卡号是否正确。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 丙：会员卡须处于非禁用状态

if (select Status from MemberCard where ID = @ID) = 1

BEGIN

RAISERROR('会员卡被禁用，请联系管理员。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 在记录表创建新行

insert into Record(Category, ID) values

('存款', @ID)

-- 保存新的自增长流水号在存储过程中

SET @RecordID = SCOPE\_IDENTITY();

-- 在存款表创建存款记录，日期为当前系统时间

insert into Deposit(ID, RecordID, DepositAmount, DepositDate) values

(@ID, @RecordID, @Amount, CONVERT(smalldatetime, GETDATE()))

-- 更新会员卡内金额

update MemberCard

set Balance = Balance + @Amount

where ID = @ID;

-- 提交事务

ENABLE TRIGGER CardDataSecurity ON MemberCard;

COMMIT TRANSACTION;

END;

**测试数据：**EXEC Depositing 'A230900001' , '7.5'

**测试结果：**

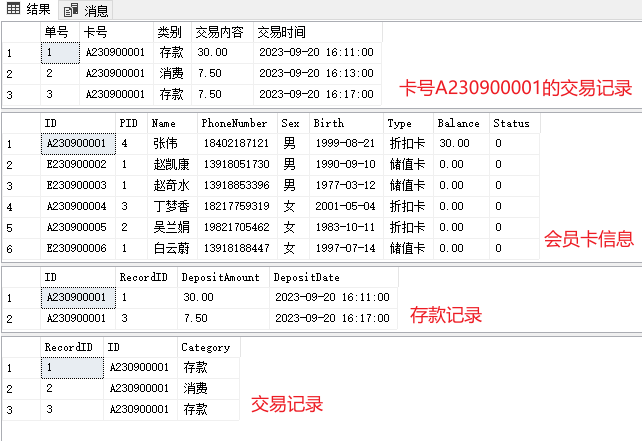


图4.1.1 会员存款7.5元后各表的变化

**1.2 会员消费的存储过程**

**分析：由于消费会影响多张表，所以需要通过存储过程完成。在输入消费卡号和消费金额后，首先判断该操作是否合法。通过消费金额大小、卡号存在、卡号非禁用等三个方面判断，否则报错退出程序。其中还需要到参数表中获取折扣倍数。其次按照逻辑顺序修改各个表的内容。记录表获取流水单号，应用到消费表后，更新会员卡表金额。最后提交事务，完成消费。**

**代码：**

CREATE PROCEDURE Expensing (

@ID CHAR(10),

@Amount NUMERIC(6,2)

)

AS

BEGIN

BEGIN TRANSACTION; -- 开始事务

DECLARE @RecordID INT; -- 声明一个变量来存储流水号

DECLARE @PID CHAR(5);

DECLARE @PValue float(10);

DECLARE @balance NUMERIC(6,2);

DISABLE TRIGGER CardDataSecurity ON MemberCard;

-- 甲：会员卡必须存在

if not exists (select ID from MemberCard where ID = @ID)

BEGIN

RAISERROR('无效的会员卡，请检查卡号是否正确。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 乙：会员卡须处于非禁用状态

if (select Status from MemberCard where ID = @ID) = 1

BEGIN

RAISERROR('会员卡被禁用，请联系管理员。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

SET @PID = (select pid from MemberCard where ID = @ID)

SET @PValue = (select pvalue from ParameterTable where pid=@pid)

SET @balance = (select balance from MemberCard where ID = @ID)

-- 验证操作是否合法

-- 丙：消费金额必须为正数或小于一万或消费完余额大于0

if @Amount = 0 OR @Amount = 10000 OR (@balance - @Amount\*@PValue)<0

BEGIN

RAISERROR('消费金额不符合要求或余额不足。', 16, 1);

ROLLBACK TRANSACTION;

return; -- 退出存储过程

END

-- 在记录表创建新行

insert into Record(Category, ID) values

('消费', @ID)

-- 保存新的自增长流水号在存储过程中

SET @RecordID = SCOPE\_IDENTITY();

-- 在消费表创建消费记录，消费金额需要乘以折扣(无折扣则为1)，日期为当前系统时间

insert into Expense(ID, RecordID, ExpenseAmount, ExpenseDate) values

(@ID, @RecordID, @Amount\*@PValue, CONVERT(smalldatetime, GETDATE()))

-- 更新会员卡内金额

update MemberCard

set Balance = Balance - @Amount\*@PValue

where ID = @ID;

ENABLE TRIGGER CardDataSecurity ON MemberCard;

-- 提交事务

COMMIT TRANSACTION;

RETURN;

END;

**测试数据：**EXEC Depositing 'A230900001' , '10'

**测试结果：**



图4.1.2 会员张伟消费原价10元后各表的变化

**1.3 会员禁用的存储过程**

**分析：由于禁用会影响多张表，所以需要通过存储过程完成。在输入禁用卡号、禁用原因、操作员工号（默认001 号admin管理员）后，首先判断该操作是否合法。通过卡号存在、卡片状态等两个方面判断，否则报错退出程序。接着确认禁用原因是否符合要求，仅限于“挂失”、“停用”、“其他”等三种。然后按照逻辑顺序修改各个表的内容。记录表获取流水单号，应用到禁用表后，更新会员卡表状态。最后提交事务，完成禁用。**

**代码：**

CREATE PROCEDURE Disabling (

@ID CHAR(10),

@Disable\_Reason CHAR(50),

@Employee\_ID CHAR(5) = '001'

)

AS

BEGIN

BEGIN TRANSACTION; -- 开始事务

DECLARE @RecordID INT; -- 声明一个变量来存储流水号

-- 验证操作是否合法

-- 甲：会员卡必须存在

if not exists (select ID from MemberCard where ID = @ID)

BEGIN

RAISERROR('无效的会员卡，请检查卡号是否正确。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 乙：会员卡须处于非禁用状态

if (select Status from MemberCard where ID = @ID) = 1

BEGIN

RAISERROR('会员卡已被禁用，无需继续操作。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 丙：原因只能'挂失' '停用' '其他'

if @Disable\_Reason != '挂失' AND @Disable\_Reason != '停用'

BEGIN

SET @Disable\_Reason = '其他'

END

-- 丁：防止外部恶意修改

-- 改进之处：可以要求输入口令，或者在前端实现

if not exists (select Employee\_ID from Employee where Employee\_ID = @Employee\_ID)

BEGIN

RAISERROR('工号不存在，无法执行操作。', 16, 1);

ROLLBACK TRANSACTION;

RETURN; -- 退出存储过程

END

-- 在记录表创建新行

insert into Record(Category, ID) values

('禁用', @ID)

-- 保存新的自增长流水号在存储过程中

SET @RecordID = SCOPE\_IDENTITY();

-- 在禁用表创建禁用记录，日期为当前系统时间

insert into Disable(ID, Employee\_ID, Recordid, Disable\_Reason, Disable\_Date) values

(@ID, @Employee\_ID, @RecordID, @Disable\_Reason, CONVERT(smalldatetime, GETDATE()))

-- 更新会员卡内状态

update MemberCard

set status = 1

where ID = @ID;

-- 提交事务

COMMIT TRANSACTION;

RETURN;

END;

**测试数据：**EXEC Disabling 'A230900001' , '挂失'

**测试结果：**

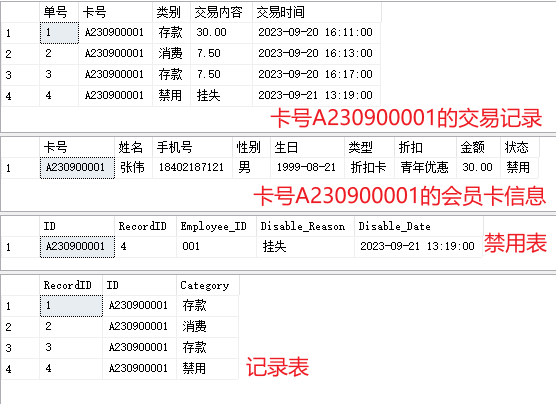


图4.1.3 会员张伟挂失后各表的变化

**1.4 会员解禁的存储过程**

**分析：由于解禁需要实现更新操作，所以需要通过存储过程完成。在输入解禁卡号后，首先获取卡片当前的状态。接着判断状态是否处于禁用。然后更新会员卡信息，实现解禁。最后提交事务，完成解禁。由于解禁并不影响多张表数据，当前不做记录。如果在后续有需求可以进行功能扩充。**

**代码：**

CREATE PROCEDURE CardResuming (

@ID char(10)

)

AS

BEGIN

--获取当前卡片状态

DECLARE @Status varchar(6)

SET @Status = (

select Status from MemberCard where ID=@ID

)

IF @Status=1

BEGIN

update MemberCard set Status=0 where ID=@ID

print @ID + '已解禁，可正常使用。'

END

END

**测试数据：**EXEC CardResuming 'A230900001'

**测试结果：**

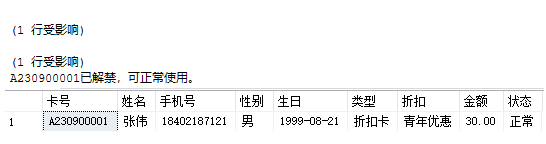


图4.1.4 会员张伟可正常使用

**2.触发器**

**2.1 卡号、金额不可修改**

**分析：由于需要确保卡号和金额的安全性，防止不经授权的修改，所以需要通过触发器完成阻断。当更新会员卡表时，判断是否更新卡号或金额，如果是则阻止更新，报错回滚事务。反之，由于Inserted 表中的行是触发器表中新行的副本，可以借此更新其他列。**

**代码：**

CREATE TRIGGER CardDataSecurity ON MemberCard

--使用inserted表

INSTEAD OF UPDATE

AS

BEGIN

--当更新卡号或金额时进入判断

IF UPDATE(ID) OR UPDATE(Balance)

BEGIN

--回滚事务

RAISERROR('不能修改卡号或是金额。', 16, 1);

ROLLBACK TRANSACTION

END

--当更新其他项时

ELSE

BEGIN

--复制inserted表内容实现更新

UPDATE MemberCard

SET Name=i.Name,

PhoneNumber=i.PhoneNumber,

Sex=i.Sex,

Birth=i.Birth,

Type = i.Type,

Status = i.Status

FROM inserted as i

WHERE MemberCard.id=i.id

END

END

**测试数据：**update membercard set id='A230900011' where phonenumber='18402187121'

**测试结果：**

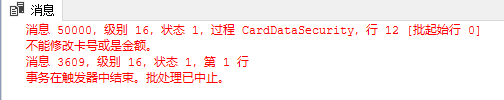


图4.2.1 会员张伟尝试更改ID

**2.2 存款记录、消费记录不可修改**

**分析：由于需要确保存款、消费记录完整性，防止不经授权的修改，所以需要通过触发器完成阻断。当更新或删除存款、消费记录时，触发器会阻止这一行为，报错并回滚事务。如果后续有禁用等记录不可修改的需求，也可以修改部分值实现。**

**代码：**

/\*禁止更改存款记录\*/

CREATE TRIGGER ProhibitChangingDepositRecords ON Deposit

FOR DELETE, UPDATE

AS

BEGIN

--回滚事务

RAISERROR('不能删除或更新存款记录。', 16, 1);

ROLLBACK TRANSACTION

END

/\*禁止更改消费记录\*/

CREATE TRIGGER ProhibitChangingExpenseRecords ON Expense

FOR DELETE, UPDATE

AS

BEGIN

--回滚事务

RAISERROR('不能删除或更新消费记录。', 16, 1);

ROLLBACK TRANSACTION

END

**测试数据：**delete from Expense where RecordID=2

**测试结果：**

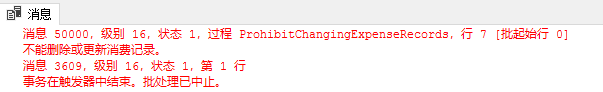


图4.2.2 尝试删除流水号为2的消费记录

**3.函数**

**3.1 通过卡号或持卡人姓名查询信息**

**分析：函数接受卡号或姓名关键字作为输入，并返回一个表格。它使用精准查询卡号和模糊查询姓名，结果表包括卡号、姓名、手机号、性别、生日、类型、折扣、金额和状态列。其中布尔型的状态列转换为文字“正常”或是“禁用”。**

**代码：**

CREATE FUNCTION SearchMember (

@SearchTerm varchar(50)

)

RETURNS TABLE

AS

RETURN(

select

ID as 卡号,

Name as 姓名,

PhoneNumber as 手机号,

Sex as 性别,

Birth as 生日,

Type as 类型,

PName as 折扣,

Balance as 金额,

CASE

WHEN Status=0 THEN '正常'

WHEN Status=1 THEN '禁用'

END as 状态

from MemberCard

join ParameterTable on MemberCard.PID = ParameterTable.PID

where id=@SearchTerm or name like '%'+@SearchTerm+'%'

)

**测试数据：**

select \* from SearchMember('A230900001')

select \* from SearchMember('赵')

**测试结果：**



图4.3.1 分别通过卡号、姓名查询

**3.2 所有会员卡的汇总记录（视图）**

**分析：为了显示存款、消费和禁用等不同交易类型的记录，可以通过LEFT JOIN来连接四张表。同时归并不同表的交易内容、交易时间。最终列出单号、卡号、交易类型、交易内容以及交易时间，可以轻松查看所有会员卡的汇总记录。**

**代码：**

SELECT

a.RecordID AS 单号,

a.ID AS 卡号,

a.Category AS 类别,

CASE

WHEN a.Category = '存款' THEN CAST(b.DepositAmount AS varchar(10))

WHEN a.Category = '消费' THEN CAST(c.ExpenseAmount AS varchar(10))

WHEN a.Category = '禁用' THEN CAST(d.Disable\_Reason AS varchar(10))

END AS 交易内容,

CASE

WHEN a.Category = '存款' THEN b.DepositDate

WHEN a.Category = '消费' THEN c.ExpenseDate

WHEN a.Category = '禁用' THEN d.Disable\_Date

END AS 交易时间

FROM Record AS a

LEFT JOIN Deposit AS b ON a.RecordID = b.RecordID

LEFT JOIN Expense AS c ON a.RecordID = c.RecordID

LEFT JOIN Disable AS d ON a.RecordID = d.RecordID

**测试数据：**SELECT \* FROM SummaryRecord;

**测试结果：**



图4.3.2 所有会员卡的汇总记录

**3.3 查询指定会员交易信息**

**分析：函数接收卡号和可选的交易时间范围，返回符合条件的交易记录表。它使用会员卡汇总记录的视图，匹配卡号和时间范围，默认起始时间是一个月前，结束时间是当前时间。可以方便地查询指定会员交易信息。**

**代码：**

--创建函数 输入卡号 起始时间 结束时间

CREATE FUNCTION GetTransactionHistory (

@ID varchar(10),

@StartTime datetime = null,

@EndTime datetime = null

)

--返回一张表 来自视图中符合要求的ID 起始时间默认1个月前 结束时间默认当前时间

RETURNS TABLE

AS

RETURN(select \* from SummaryRecord where (卡号 = @ID) and (交易时间 between ISNULL (@StartTime, DATEADD(MONTH, -1, GETDATE())) and ISNULL(@EndTime, GETDATE())))

**测试数据：**

select \* from GetTransactionHistory('A23090000100', null, null)

select \* from GetTransactionHistory('A23090000100', null, '2023-09-21')

select \* from GetTransactionHistory('A23090000100', '2023-09-21', null)

**测试结果：**



图4.3.3 查询指定会员交易信息

**五、小结**

本实验完成了游泳馆会员管理系统的数据库设计、数据库实施、数据库完善（T-SQL代码编写）。基本实现了用户需求，完成了数据库系统管理。在数据库设计阶段，了解后客户需求，寻找目标用户并完成主要功能设计。提出数据库安全性要求后，开始绘制数据流图。从顶层出发，由浅入深地不断向下分解。设计数据字典，绘制CDM图，完成数据完整性约束。通过PowerDesigner生成PDM图后，进一步优化后生成数据库代码。通过编写T-SQL触发器、存储过程、函数完善数据库系统。

在数据库设计阶段画了较长的时间，为了保证未来数据库可持续使用，需要确保每一项的完整性、安全性。在编写T-SQL语句过程中，将多表的内容实现有机统一，自动实现。存款、消费、禁用与记录表实现统一，不存在相同单号。在不断地试错中，完成了编写。本系统也预留了很多空间，方便数据库功能的扩展以及有效维护。

第一次从零开始设计数据库，考虑了很多因素，经历了多次调整、重新绘制。最终得以实现游泳馆会员管理系统。

**参考文献**

[1]杨海霞.《数据库实验指导（第2版）》.人民邮电出版社.2013

[2]《游泳馆会员管理系统课程设计报告书》.百度文库.

<https://wenku.baidu.com/view/ed8276e2846fb84ae45c3b3567ec102de3bddfbc.html>

[3]《基于Java+Swing+mysql游泳馆会员管理系统》.CSDN.

<https://blog.csdn.net/Timi2019/article/details/128371809>