武汉纺织大学管理学院电子商务专业

最新数据库管理系统

（2019—2020 学年第二学期）

项目名称：单词速记数据库分析设计与实现

班　　级：电商 11802 班

团 队 名：抽象带篮子

姓　　名：牟伦利(组长)褚四浩陈思琴曹鹏飞

2020　年 11 月 26 日

目录

一. 系统需求分析

1. 信息需求

不会吧不会吧，还有人在背单词系统的数据库是基于 SQL Server 2017 开发的 。其系统的主要需求是完成对用户单词记忆数据的相关管理。系统的主要面对对象是广大需要背单词的英语学习者。系统的信息需求如下：

1.用户录入昵称、密码、性别、邮箱、个人简介等基本信息注册进入系统的身份，用户输入 ID 和密码开始使用系统，初始化选择的学习词典编号和学习计划。

2.用户可以根据自己的学习安排选择词典和学习计划。

3.学习新单词与复习旧单词，初始化今日单词学习数据

4.记忆单词时，正确释义配合三个类似单词的释义让用户做选择

5.根据用户单词的记忆次数得出单词的记忆错误率，给出用户直观可见的学习分析数据

2. 功能需求

表 1 功能需求

功能分类	序号	名称	主要内容	输入数据	输出数据
信息录入	1	用户登录	用户根据通信证进入系统开始使用	用户信息（昵称或 id，密码）	用户信息表
	2	用户注册	用户根据相关的要求制定自己的通行证使用系统	用户信息（昵称，密码，邮箱，性别，个人简介）	用户信息表
制定学习计划	3	选择词典	选择记忆的词典 CET 4、CET6 等等	词典编号	个人学习计划表
	4	制定学习计划	在设定好的多种学习计划中选择自己想要的学习进度（例：每天 35 个单词，用时 100 天。依次类推）	学习计划选项	个人学习计划表
	5	调整学习计划	后期的学习计划更改	学习计划选项	个人学习计划表
记忆单词	6	学习新单词	从词典中选择没有记过的单词加入今日学习单词表中（一定数量）	单词信息（单词 id，单词拼写，单词词性，单词释义）	今日学习数据表
	7	复习旧单词	从之前学习过的单词记录表中抽取一定数量的单词进行复习	单词信息（单词 id，单词拼写，单词词性，单词释义）	今日学习数据表
分析数据	8	单词记忆的日分析和周分析	得出每天单词记忆的学习情况	单词信息+记忆次数+错误率+时间戳	数据分析表

3. 系统相关算法和公式分析

LD 算法就是自然语言处理(NLP)里的“编辑距离"算法。俄国科学家 Levenshtein 提出的，故又叫 Levenshtein Distance （LD 算法），该算法是指指两个字符串之间，由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。一般来说，编辑距离越小，两个串的相似度越大。用于计算单词和单词之间的相似度，在用户进行背单词时生成模糊选项。

实现图解：

	空	x	y	z	a	b
空	0	1	2	3	4	5
a	1					
x	2					
y	3					
z	4					
c	5					

初始化，把数组第1行和第1列初始化为0-5，对应min(i,j) = 0这一条件。表示由空串转换为某一子串所需步骤。为方便起见，(a,y)表示"a"和"xy"的最短距离，下面的表示方法以此类推。

	空	x	y	z	a	b
空	0	1	2	3	4	5
a	1	1	2	3	3	4
x	2					
y	3					
z	4					
c	5					

对a所在的行进行计算，即数组第2行数据。例如：第二个数据(a,y)，由于a≠y，所以(a,x)=min{(a,x)+1,(空,x)+1,(空,y)+1}=2。
第4个数据(a,a)，由于a=a，所以(a,a) = (空,z) = 3

图 1 LD 算法

	空	x	y	z	a	b
空	0	1	2	3	4	5
a	1	1	2	3	3	4
x	2	1	2	3	4	4
y	3	2	1	2	3	4
z	4	3	2	1	2	3
c	5	4	3	2	2	3

接着对x、y、z、c行重复上面的计算，知道填满整个表格，表格最右下角的数值，即表示字符串"axyzc"和"xyzab"的最短编辑距离，这里为3.

图 2 LD 算法 2

二. 系统数据字典、业务流程图、数据流程图、ER 图、代码设计

1. ER 图

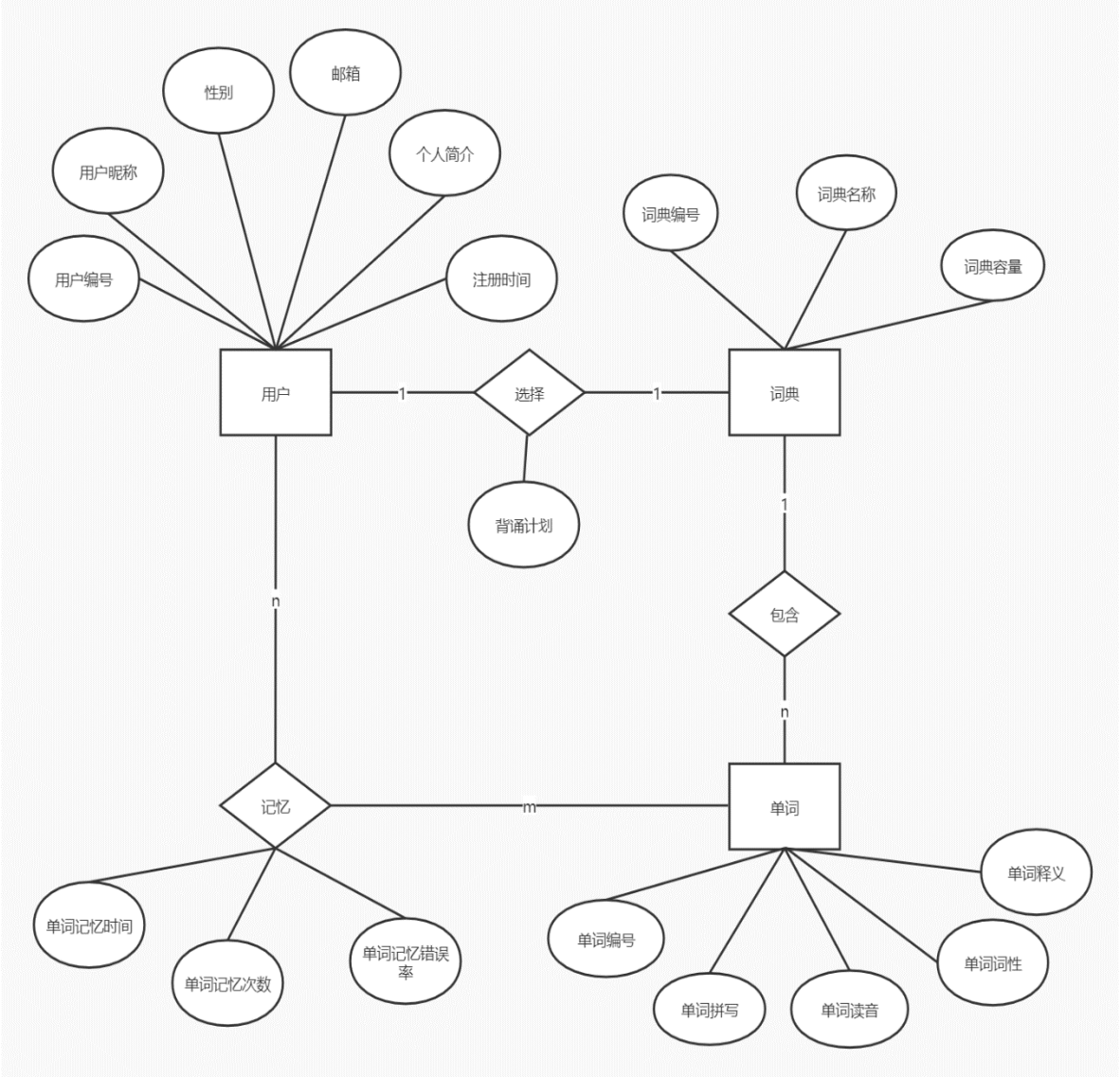
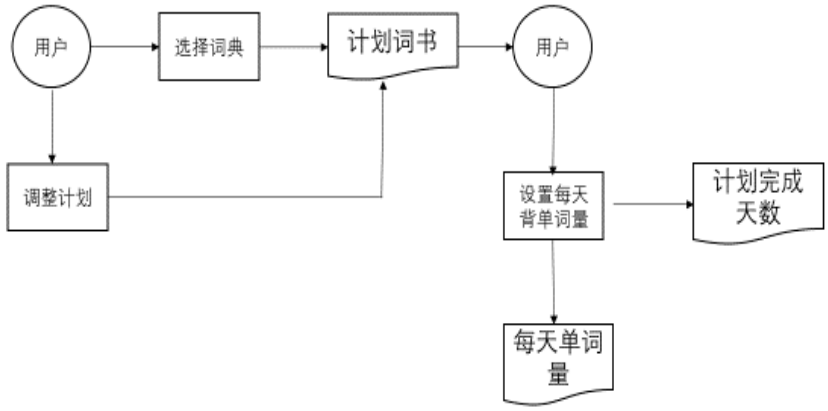


图 3 ER 图

2. 业务流程图



单词速记 DCSJ--计划选择业务流程图

主要活动：用户计划选择，设置每天背单词量相关的信息载体：词书词汇量(计划词书)，单词量/天(每天单词量)，计划完成天数(计划完成天数)(相关信息放在tb_plan表中)
计划选择业务流程图：用户进行计划选择，选择想要开始计划的词书(例CET4,CET6)，用户根据已选词书的词汇量来设置每天的背单词量，系统会根据用户的计划得出完成天数及计划完成日期

图 4 单词速记 DCSJ--计划选择业务流程图

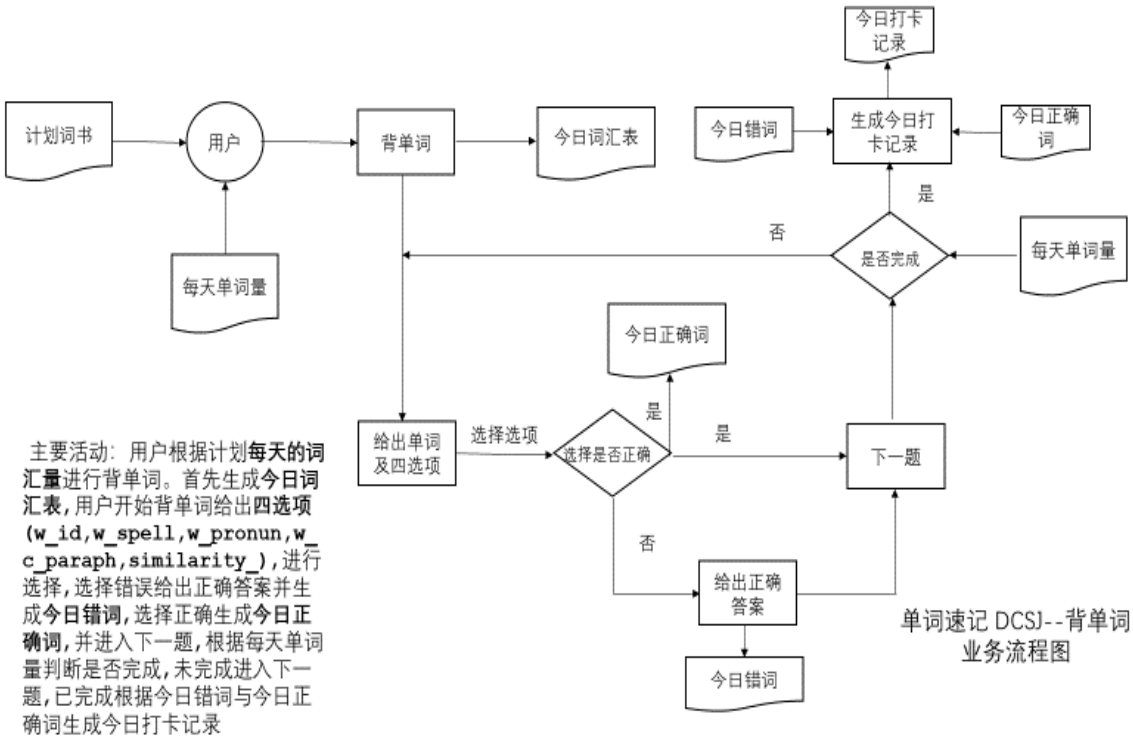
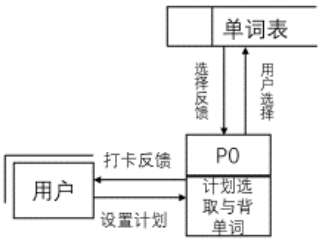


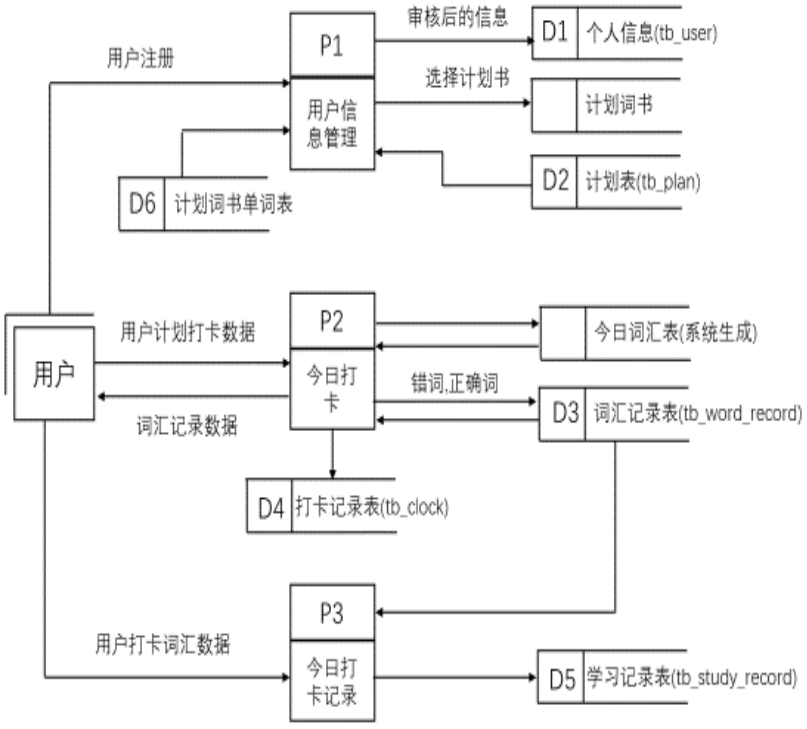
图 5 单词速记 DCSJ--背单词业务流程图

3. 数据流程图



单词速记 DCSJ　0层DFD图

图 6 单词速记 DCSJ　0 层 DFD 图



单词速记 DCSJ　1层DFD图

图 7 单词速记 DCSJ　1 层 DFD 图

4. 系统数据字典

4.1 数据项

表 2 数据项

数据项编号	数据项名称	别名	简述	类型及宽度
I01-01	u_id	用户编号	用户注册时生成的编码	char（10），随机数
I01-02	P_select	词书	用户选择计划时生成的编号	Char(2),(01 02)
I01-03	p_id	用户计划编号	用户选择学习计划生成的编号	char(8)，随机数
I01-04	w_id	单词编号	每个单词根据自己所属词典生成的唯一编号	char(10), 随机数
I02-01	w_r_id	单词学习记录编号	记录用户学过单词的记录编号	char(8)，随机数
I03-01	c_id	打卡记录编号	记录用户完成打卡任务的编号	int，自增主键
I03-02	s_r_id	用户学习记录编号	记录用户每日学习单词情况的编号	char(8)，随机数

4.2 数据结构

表 3 数据结构

数据结构编号	数据结构名称	简述	数据结构组成
DS01-01	设置用户计划	用于标识用户制定的计划	I01-01+ I01-02+ I01-03
DS02-01	词汇记录	用户今日打卡的词汇记录表,包括今日错词以及错词次数	I02-01+ I01-04
DS03-01	学习打卡记录	用户每次打卡将会被记录	I03-01+ I03-02

4.3 数据流

表 4 数据流

数据流编号	数据流名称	简述	数据流来源	数据流去向	数据流组成	数据流量	高峰流量
DF01-01	计划打卡	用户制定计划，设置打卡内容	用户计划词书	计划表	每日计划词以及打卡日期	不定	不定
DF02-01	今日应背词	根据打卡内容，生成今日应背词	今日打卡模块	今日词汇表	用户编号，单词编号，单词拼写，单词发音，单词意思	10000 次/h	25000 次/h
DF02-02	背词记录	今日词汇表中错误的词进行记录	今日词汇表	词汇记录表	单词学习记录编号，用户编号，单词编号，错误次数	10000 次/h	30000 次/h
DF03-01	学习记录	对用户当天的学习词汇进行记录	词汇记录表	今日打卡记录模块	用户学习记录编号，用户编号，学习单词数量，错误单词数量	不定	不定

4.4 处理逻辑定义

表 5 处理逻辑定义

处理逻辑编号	处理逻辑名称	输入数据流	处理方式	输出数据流	处理频率
P02-01	今日词汇表生成	计划打卡	Levenshtein Distance (编辑距离)算法实现	今日应背词	10000 次/h
P02-02	错词统计	今日应背词	将每一个错误词累加	背词记录	10000 次/h
P03-01	打卡记录表生成	背词记录	用户今日背词记录和用户计划背词对比	学习记录	不定

4.5 数据存储

表 6 数据存储

数据存储编号	数据存储名称	简述	数据存储组成	关键字	相关联的处理
D1	个人信息表	存储用户个人信息	u_id,u_nickname, u_password,u_sex u_email,u_intro register_time	u_id	P02-01
D2	计划表	存储用户制定的计划	p_id,u_id,p_select p_daywords,p_total p_leave_total, p_leave_days, update_time	p_id,u_id	P02-01
D3	词汇记录表	存储用户背单词的情况	w_r_id,u_id,w_id wrong_nums,study_date	w_r_id	P02-02
D4	打卡记录表	用户打卡情况	c_id,u_id,clock_time	c_id	P03-01
D5	学习记录表	用户学习记录情况	s_r_id，u_id，s_total， s_wrong，study_date	s_r_id	P03-01
D6	计划词书单词表	用户选择计划单词书	w_id,w_spell,w_pronun w_c_paraph	w_id	P02-01

4.6 外部实体定义

表 7 外部实体定义

外部实体编号	外部实体名称	简述	输入数据流	输出数据流
O1	用户	进行选择计划并进行背单词的用户	DF03-01,DF02-02	DF01-01
O2	单词表	供用户选择的词书	DF01-01	用户选择

5. 代码设计

单词编号

010001

前两位是词典的编号，01 是四级，02 是六级

后四位是单词在数据表中的位置

该单词编号是 四级单词的第一个 outdoors

三. 系统主要数据表

```
--用户表
create table tb_user(
  u_id char(10) primary key,-- 用户id,主键
  u_nickname varchar(14) not null,-- 用户昵称
  u_password char(32) not null, -- 密码
  u_sex char(1) not null default '0',-- 用户性别 0男1女
  u_email varchar(20) ,-- 用户邮箱
  u_intro varchar(200),-- 用户个人简介
  register_time date not null default(GETDATE())-- 注册日期
)
```

图 8 tb_user

```
--词典cet4
create table tb_cet4_dictionary(
  w_id char(6) primary key,--单词id,主键 （代码设计来区分不同词典 01xxxx）
  w_spell varchar(50) not null,--单词拼写
  w_pronun nvarchar(50)  not null ,--单词读音
  w_c_paraph nvarchar(255) not null,--单词 词性+释义
)
-- 更新w_id
-- UPDATE tb_cet4_dictionary SET w_id = concat('02000',w_id) where w_id like '_'
```

图 9 tb_cet4_dictionary

```
--词典cet6
create table tb_cet6_dictionary(
  w_id char(6) primary key,--单词id,主键 （代码设计来区分不同词典 02xxxx）
  w_spell varchar(50) not null,--单词拼写
  w_pronun nvarchar(50)  not null ,--单词读音
  w_c_paraph nvarchar(255) not null,--单词 词性+释义
)
```

图 10 tb_cet6_dictionary

```
--单词学习记录表
create table tb_word_record(
  w_r_id char(8) primary key,--单词学习记录id,主键
  u_id varchar(10) not null,--用户id
  w_id varchar(6) not null,--单词id
  wrong_nums int not null default 0,--错误次数
  study_date date not null default(GETDATE()),--学习日期
)
```

图 11 tb_word_record

```
--用户学习记录表（每个学习记录id对应学习的单词以及单词数）
create table tb_study_record(
  s_r_id char(8) primary key,--用户学习记录id
  u_id varchar(10) not null,--用户id
  s_total int not null,--学习总单词数
  s_wrong int not null,--选错次数
  study_date date not null default(GETDATE()),--学习日期
)
```

图 12 tb_study_record

```
--用户计划表
create table tb_plan(
  p_id char(8) primary key,--用户计划id
  u_id varchar(10) not null,--用户id
  p_select char(2) not null,--选择词典(01:cet4  02:cet6 ...)
  p_daywords int not null default(35),--每日学习单词数
  p_leave_total int not null,--学习剩余总单词数
  p_total int not null,--学习总单词数
  p_leave_days int not null,--学习剩余天数
  update_time date not null default(GETDATE()),--调整日期
)
```

图 13 tb_plan

```
--用户打卡表
create table tb_clock(
  c_id int primary key IDENTITY(1,1),--打卡记录id
  u_id varchar(10) not null,--用户id
  clock_time date not null default(GETDATE()),--打卡日期
)
```

图 14 tb_clock

四. 系统视图和临时表

1. 总单词表

把四六级单词合并在一个视图，生成总单词表

```
-- 总单词表合并
CREATE VIEW all_words
AS
SELECT * FROM tb_cet4_dictionary
UNION ALL
SELECT * FROM tb_cet6_dictionary
GO
```

图 15 总单词表

结果 消息

	w_id	w_spell	w_pronun	w_c_paraph
1	010001	outdoors	ˈaʊtˈdɔːz	adv.在户外 在野外 n.露天 野外 名词outdoor的复数形式
2	010002	dial	ˈdaɪəl	n.钟面 拨号盘 刻度盘 针面 转盘 vt.拨 用仪表测量 操作仪表 vi.拨号 用仪表测量 ...
3	010003	strain	strein	vt.拉紧 劳累 过份使用 vi.尽力 n.紧张 拉紧 血统
4	010004	snow	snəu	n.雪 下雪 似雪花的东西 粉状物 vi.下雪 vt.雪一般落下 使雪白 欺骗
5	010005	wolf	wulf	n.狼 残暴的人 v.狼吞虎咽
6	010006	feedback	ˈfiːdbæk	n.回授 反馈 反应 成果
7	010007	umbrella	ʌmˈbrelə	n.伞 雨伞 adj.像伞状分布的 vt.(用伞)遮住
8	010008	explore	iksˈplɔː	vt. & vi.探险 探索 探测 探究 [计算机] 探讨
9	010009	decide	diˈsaid	vt. & vi.决定 决心 解决 作出抉择
10	010010	if	ɪf	conj.假如 如果 是否 即使 n.条件 设想
11	010011	target	ˈtɑːgɪt	n.靶 标的 目标 对象 vt.把...作为目标 瞄准
12	010012	knob	nɔb	n.门把 拉手 旋钮 瘤
13	010013	ratio	ˈreɪʃɪəu	n.比 比率
14	010014	microphone	ˈmaɪkrəfəun	n.话筒 麦克风 扩音器

图 16 总单词表详情

2. 记录单词表

通过把用户记录的单词编号和单词表进行左连接，得到用户已经记录单词的详细信息

```
-- 有记录的单词表
CREATE VIEW hasrecord_words
AS
SELECT u_id, wr.w_id, w_spell, w_pronun, w_c_paraph, wrong_nums, study_date
FROM tb_word_record wr
LEFT OUTER JOIN
all_words aw on
wr.w_id = aw.w_id
```

图 17 用户已经记录的单词

结果 消息

	u_id	w_id	w_spell	w_pronun	w_c_paraph	wrong_nums	study_date
1	1743669115	010181	full	ful	adj. 满的 完全的 充满的 丰富的 adv. 完全地 整整 vt. 使...充满 通过缩水...	1	2020-11-25
2	1743669115	014377	strategy	ˈstrætɪdʒɪ	n.战略 策略	0	2020-11-25

图 18 详细信息

3. 插入临时表

该临时表作为用户在进行一次打卡任务所生成的单词表，当复习单词时，选错的单词会插入该表

```
-- 创建临时表
CREATE TABLE ##temp_todaywords (
u_id char(10) not null,
w_id char(6) not null,
w_spell varchar(50) not null,
w_pronun nvarchar(50)  not null,
w_c_paraph nvarchar(255) not null,
constraint un_wid unique(w_id)  -- 为w_id 添加唯一索引约束
)
-- 临时表插入数据
GO
```

图 19 进行一次打卡任务所生成的单词表

结果 消息

	u_id	w_id	w_spell	w_pronun	w_c_paraph
1	1743669115	010001	outdoors	ˈaʊtˈdɔːz	adv. 在户外 在野外 n. 露天 野外 名词outdoor的复数形式
2	1743669115	010044	private	ˈpraɪvɪt	adj. 私人的 私下的 隐蔽的 n. 士兵 列兵
3	1743669115	010255	notebook	ˈnəʊtbʊk	n. 笔记本 笔记本电脑 期票簿
4	1743669115	010335	right	raɪt	adj. 正确的 正直的 合适的 垂直的 右面的 正常的 正面的 adv. 对 正好 恰当 径...
5	1743669115	010384	rack	ræk	n. 搁物架 行李架 拷问台 齿轨 vt. 折磨 使痛苦 拷问 vi. 顶风飞行
6	1743669115	010463	write	raɪt	vt. 书写 写 vi. 写
7	1743669115	010499	humour	ˈhjuːmə	n. 幽默 诙谐 幽默感 体液 vi. 纵容 迁就 =humor(美)
8	1743669115	010625	sour	ˈsaʊə	adj. 酸的 酸腐的 刻薄的 脾气坏的 v. 变酸 n. 酸物
9	1743669115	010639	anchor	ˈæŋkə	n. 铁锚 vi. 抛锚 停泊 用锚系住 担任（广播 电视新闻节目）的主持人
10	1743669115	010755	recommend	ˌrekəˈmend	vt. 推荐 介绍 劝告 建议 使成为可取 使受欢迎
11	1743669115	010779	favourable	ˈfeɪvərəbl	adj. 有利的 有用的 良好的 赞成的
12	1743669115	010784	formula	ˈfɔːmjʊlə	n. 公式 式 配方 规则 代乳品 adj. (赛车的)级的 方程式的
13	1743669115	010846	practically	ˈpræktɪkəli	adv. 实际上 几乎 简直 adj. 实际的 几乎
14	1743669115	010855	pie	paɪ	n. (西点)馅饼 派 杂乱 喜鹊 爱说话的人 饼图 vt. 弄乱

图 20 详细信息

五. 系统存储过程、触发器

1. 存储过程

1.1 add_user

用户注册的存储过程(add_user)

```
-- 1.用户注册(添加用户)
IF EXISTS (SELECT name FROM sysobjects WHERE name='add_user' AND type='P')
DROP PROCEDURE add_user
GO
CREATE PROCEDURE add_user
    @u_nickname varchar(14),
    @u_password char(32),
    @u_sex char(1),
    @u_email varchar(20),
    @u_intro varchar(200)
AS
DECLARE @u_id char(10)
SET @u_id = (SELECT cast(floor(rand()*10000000000)as bigint))
INSERT INTO tb_user VALUES(@u_id,@u_nickname,@u_password,@u_sex,@u_email,@u_intro,GETDATE())
GO
-- 测试
EXECUTE add_user 'sakurazz','123','0','1622135447@qq.com','暂无'
SELECT * FROM tb_user
```

图 21 用户注册的存储过程(add_user)

1.2 choose_dictionary

用户选择词典的存储过程（choose_dictionary）

该存储过程用于用户来选择词典，选择四级/六级词典（01/02），同时更新计划表(tb_plan),默认设定计划每天背词数为 50 个。

```
-- 2.选择词典+(未调整前)默认计划：每日50词
IF EXISTS (SELECT name FROM sysobjects WHERE name='choose_dictionary' AND type='P')
DROP PROCEDURE choose_dictionary
GO
CREATE PROCEDURE choose_dictionary
    @u_id char(10),
    @p_select char(2),
    @p_daywords int = 50
AS
DECLARE @p_id char(8), @p_leave_total int, @p_total int, @p_leave_days int
SET @p_id = (SELECT cast(floor(rand()*100000000)as bigint))
SELECT @p_total = COUNT(*) FROM all_words WHERE w_id like @p_select + '%'
SET @p_leave_total = @p_total - (SELECT COUNT(*) FROM tb_word_record WHERE u_id=@u_id AND w_id=@p_select+'%')
SET @p_leave_days = ceiling(cast(@p_leave_total as float)/@p_daywords)
IF EXISTS (SELECT * FROM tb_plan WHERE u_id=@u_id)
    UPDATE tb_plan SET p_select=@p_select, p_daywords=@p_daywords, p_leave_total=@p_leave_total,
    p_total=@p_total, p_leave_days=@p_leave_days, update_time=GETDATE()
    WHERE u_id=@u_id
ELSE
    INSERT INTO tb_plan VALUES(@p_id,@u_id,@p_select,@p_daywords,@p_leave_total,@p_total,@p_leave_days,GETDATE())
GO
-- 测试
EXECUTE choose_dictionary '6592330246','01'
```

图 22 用户选择词典的存储过程（choose_dictionary）

1.3 adjust_plan

用户调整计划的存储过程（adjust_plan）

该存储过程用于用户来调整计划，即调整每日的背词数量，同时刷新计划表（tb_plan）中的剩余天数。

```
-- 3. 调整计划
IF EXISTS (SELECT name FROM sysobjects WHERE name='adjust_plan' AND type='P')
DROP PROCEDURE adjust_plan
GO
CREATE PROCEDURE adjust_plan
    @u_id char(10),
    @p_daywords int
AS
DECLARE @p_leave_total int, @p_leave_days int
SELECT @p_leave_total = p_leave_total FROM tb_plan WHERE u_id = @u_id
SET @p_leave_days = ceiling(cast(@p_leave_total as float)/@p_daywords)
UPDATE tb_plan SET p_daywords=@p_daywords,p_leave_days=@p_leave_days, update_time=GETDATE()
WHERE u_id=@u_id
GO
-- 测试
EXECUTE adjust_plan '6592330246',10
```

图 23 用户调整计划的存储过程（adjust_plan）

1.4 get_today_words

生成今日词汇的存储过程（get_today_words）

该存储过程用于生成用户每日须背的词汇表。若用户此前无此词典的学习记录，则今日词汇的组成仅为今日新学，即直接随机从词典中抽取指定数量的单词出来；若用户有此词典的学习记录，则今日词汇的组成为须复习（日期距今最近一天学习单词的 1/3）+今日新学。

```
-- 4.生成今日词汇表
IF EXISTS (SELECT name FROM sysobjects WHERE name='get_today_words')
DROP PROCEDURE get_today_words
GO
CREATE PROCEDURE get_today_words
    @u_id char(10)
AS
DECLARE @p_daywords int, @p_select char(2), @old_words int
SELECT @p_daywords = p_daywords, @p_select = p_select FROM tb_plan WHERE u_id=@u_id
SELECT @old_words = count(*)
    FROM hasrecord_words
    WHERE u_id=@u_id AND w_id like @p_select + '%' AND study_date =
    (SELECT min(study_date) FROM hasrecord_words WHERE study_date<=getdate())
IF EXISTS (SELECT * FROM tb_word_record WHERE u_id=@u_id)
-- 有单词学习记录 则单词组成为 需复习单词+今日新学
    BEGIN
        SELECT * FROM (SELECT TOP (@old_words/3) u_id,w_id,w_spell,w_pronun,w_c_paraph
        FROM hasrecord_words
        WHERE u_id=@u_id AND w_id like @p_select + '%' AND study_date =
            (SELECT min(study_date) FROM hasrecord_words WHERE study_date<=getdate())
        GROUP BY u_id,w_id,w_spell,w_pronun,w_c_paraph,wrong_nums) a
      UNION ALL
        SELECT * FROM (SELECT TOP (@p_daywords) u_id=@u_id,w_id,w_spell,w_pronun,w_c_paraph
        FROM all_words  WHERE w_id like @p_select + '%' AND w_id NOT IN (SELECT w_id FROM tb_word_record WHERE u_id = @u_id)
        order by newid()) b
        ORDER BY w_id
    END
ELSE
-- 无单词学习记录 则单词组成仅仅为今日新学
    SELECT TOP (@p_daywords) u_id=@u_id,w_id,w_spell,w_pronun,w_c_paraph FROM all_words WHERE w_id like @p_select + '%' ORDER BY NewID()
GO
-- 测试
EXECUTE get_today_words '6592330246'
```

图 24 生成今日词汇的存储过程（get_today_words）

另外，此存储过程生成的今日词汇表会存储到临时表##temp_todaywords 中，便于用户一天内多次访问。

```
-- 存入临时表
    -- 创建临时表
    CREATE TABLE ##temp_todaywords (
    u_id char(10) not null,
    w_id char(6) not null,
    w_spell varchar(50) not null,
    w_pronun nvarchar(50)  not null,
    w_c_paraph nvarchar(255) not null,
    CONSTRAINT un_wid unique(w_id)
    )
    -- 临时表插入数据
    INSERT INTO ##temp_todaywords EXECUTE get_today_words '6592330246'
    -- 查询临时表
    SELECT * FROM ##temp_todaywords
```

图 25 临时表##temp_todaywords

结果 消息

	u_id	w_id	w_spell	w_pronun	w_c_paraph
1	6592330246	010165	collision	kə'liʒən	n.碰撞 冲突
2	6592330246	010716	literature	'litərətʃə	n.文学 文献
3	6592330246	011039	interfe...	.intə'...	n.干涉 ...
4	6592330246	011064	ability	ə'biliti	n. 能力...
5	6592330246	011270	hasty	'heisti	adj.急...
6	6592330246	011456	approxi...	ə'prɔk...	adj.近...
7	6592330246	011630	large	lɑ:dʒ	adj.大...
8	6592330246	011989	African	'æfrikən	adj. 非...
9	6592330246	012273	debate	di'beit	n. &vi....
10	6592330246	012392	microco...	'maik...	n.微型计...
11	6592330246	012821	despair	di'spɛə	n.绝望 ...
12	6592330246	012822	reform	ri'fɔ:rm	n.改革 ...
13	6592330246	013561	quote	kwəut	vt.引用...

查询已成功执行。

图 26 临时表##temp_todaywords 值

1.5 get_options

得到单词四个选项的存储过程(get_options)

该存储过程用于生成每个单词对应的中文选项。其过程运用了 Levenshtein Distance (编辑距离)算法，通过此算法来找到与当前单词拼写字符串相似度最高的 4 个单词选项。

```
-- 5.得到当前词的4个中文选项
IF EXISTS (SELECT name FROM sysobjects WHERE name='get_options')
DROP PROCEDURE get_options
GO
CREATE PROCEDURE get_options
    @w_id char(6)
AS
DECLARE @w_spell varchar(50), @dictionary char(2)
SELECT @w_spell = w_spell FROM all_words WHERE w_id=@w_id
SELECT @dictionary=LEFT(@w_id,2)
SELECT TOP 4 w_id,w_spell,w_pronun,w_c_paraph,(SELECT dbo.edit_distance_within(@w_spell,all_words.w_spell,10))
AS similarity FROM all_words WHERE w_id like @dictionary+'%'
AND ((SELECT dbo.edit_distance_within(@w_spell,all_words.w_spell,10)) BETWEEN 0 AND 9) ORDER BY similarity
GO
-- 测试
EXECUTE get_options '025000'
```

图 27 得到单词四个选项的存储过程(get_options)

	w_id	w_spell	w_pronun	w_c_paraph	similarity
1	025000	audience	'ɔ:diəns	n.听众观众读者拥护者倾听正式会见	0
2	020263	absence	'æbsəns	n.缺席不在场缺乏	3
3	021634	alliance	ə' laiəns	n. 联盟,联合	3
4	021976	science	'saiəns	n.科学科学研究	3

查询已成功执行。

图 28 单词四个选项

1.6 select_option

用户单击选项的存储过程(select_option)

该存储过程用于用户单击选项时一系列处理的存储过程。用户选择时，触发触发器tr_recite， 同时更新单词记录表（tb_word_record），选错时错误次数（wrong_nums）+1。

```
-- 6.用户单击选项（正确/错误及其处理）
IF EXISTS (SELECT name FROM sysobjects WHERE name='select_option')
DROP PROCEDURE select_option
GO
CREATE PROCEDURE select_option
    @u_id char(10),
    @w_id char(6), -- 当前词汇id
    @select_id char(6) -- 选择选项的词汇id
AS
DECLARE @w_r_id char(10), @wrong_nums int
SET @w_r_id = (SELECT cast(floor(rand()*10000000)as bigint))
IF EXISTS (SELECT * FROM tb_word_record WHERE u_id=@u_id AND w_id=@w_id)
    BEGIN
    SELECT @wrong_nums=wrong_nums FROM tb_word_record WHERE u_id=@u_id AND w_id=@w_id
    IF @w_id != @select_id
        -- 选错
        SET @wrong_nums = @wrong_nums+1
    UPDATE tb_word_record SET wrong_nums=@wrong_nums WHERE u_id=@u_id AND w_id=@w_id
    END
ELSE
    BEGIN
    SET @wrong_nums=0
    IF @w_id != @select_id
        -- 选错
        SET @wrong_nums = @wrong_nums+1
    INSERT INTO tb_word_record VALUES(@w_r_id,@u_id,@w_id,@wrong_nums,GETDATE())
    END
GO
-- 测试
EXECUTE select_option '6592330246','014377','014377'
```

图 29 用户单击选项的存储过程(select_option)

1.7 update_study_record

触发器 tr_recite 触发的存储过程(update_study_record)

该存储过程用于用户背单词时学习记录表(tb_study_record)的更新以及计划表(tb_plan)的更新。用户选对时，删除今日词汇表(临时表##temp_todaywords)中的一行该单词数据；用户选错时，会重新插入该单词数据至今日词汇表。同时更新/插入学习记录表和计划表的数据。

```
-- 7.用户背单词触发更新调用的存储过程
IF EXISTS (SELECT name FROM sysobjects WHERE name='update_study_record')
DROP PROCEDURE update_study_record
GO
CREATE PROCEDURE update_study_record
    @u_id char(10), -- word_record表更新/插入行的用户id
    @w_id char(6), -- word_record表更新/插入行的词汇id
    @isRight int-- 是否选对 0错 1对
AS
    DECLARE @s_r_id char(10),@w_spell VARCHAR(50),@w_pronun nvarchar(50),
                @w_c_paraph nvarchar(255),@s_wrong int,@s_total int,
                @p_leave_total int,@p_leave_days int,@p_daywords int,@cur_date date
    SET @s_r_id = (SELECT cast(floor(rand()*10000000)as bigint))
    SET @cur_date = GETDATE()
    SELECT @w_spell = w_spell,@w_pronun = w_pronun,@w_c_paraph =w_c_paraph
    FROM all_words WHERE w_id = @w_id
    SELECT @p_leave_total =p_leave_total,@p_leave_days =p_leave_days,@p_daywords=p_daywords FROM tb_plan WHERE u_id=@u_id
-- 从临时表中删除该行 当日学习单词-1
DELETE FROM ##temp_todaywords WHERE u_id =@u_id AND w_id = @w_id
IF EXISTS (SELECT * FROM tb_study_record WHERE u_id ='6592330246' AND study_date =@cur_date) -- 当日有学习记录，则为更新记录
    IF @isRight = 0
        -- 选错
        BEGIN
            SELECT @s_wrong =count(*) FROM tb_word_record WHERE u_id=@u_id AND wrong_nums > 0 AND study_date = @cur_date
            -- 重新插入临时表尾部 错词重新学习 当日单词+1
            INSERT INTO ##temp_todaywords VALUES(@u_id,@w_id,@w_spell,@w_pronun,@w_c_paraph)
            UPDATE tb_study_record SET s_wrong=@s_wrong WHERE u_id=@u_id
        END
    ELSE
        -- 选对
        BEGIN
            -- plan表中剩余单词总数减1
            SET @p_leave_total= @p_leave_total -1
            -- 更新tb_study_record
            SELECT @s_total = s_total FROM tb_study_record WHERE u_id =@u_id AND study_date =@cur_date
            UPDATE tb_study_record SET s_total=@s_total + 1
        END
ELSE
-- 当日无学习记录，则为插入记录
    IF @isRight = 0
    -- 选错
        BEGIN
            -- 插入临时表
            INSERT INTO ##temp_todaywords VALUES(@u_id,@w_id,@w_spell,@w_pronun,@w_c_paraph)
            -- 插入study_record
            SET @s_total =0
            INSERT INTO tb_study_record VALUES(@s_r_id,@u_id,@s_total,@s_wrong,@cur_date)
        END
    ELSE
        -- 选对
        BEGIN
            -- plan表中剩余单词总数减1
            SET @p_leave_total= @p_leave_total -1
            -- 插入到tb_study_record
            SELECT @s_wrong =s_wrong FROM tb_study_record WHERE u_id =@u_id
            SET @s_wrong=0
            SET @s_total=1
            INSERT INTO tb_study_record VALUES(@s_r_id,@u_id,@s_total,@s_wrong,@cur_date)
        END
-- 更新plan表剩余天数
SET @p_leave_days = ceiling(cast(@p_leave_total as float)/@p_daywords)
UPDATE tb_plan SET p_leave_total=@p_leave_total,p_leave_days=@p_leave_days, update_time=@cur_date
```

图 30 触发器 tr_recite 触发的存储过程(update_study_record)

1.8 edit_distance_within

get_options 存储过程调用的算法函数(edit_distance_within)

```
-- Levenshtein Distance (编辑距离)算法实现
/* Levenshtein Distance 算法，又叫 Edit Distance 算法，
是指两个字符串之间，由一个转成另一个所需的最少编辑操作次数。
许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。
一般来说，编辑距离越小，两个串的相似度越大。
*/
/*
**@params s和t为比较字符串,d为限定最大编辑操作次数
* @return 最少编辑操作次数(-1即为超过最大编辑操作次数)
*/
CREATE FUNCTION edit_distance_within(@s nvarchar(4000), @t nvarchar(4000), @d int)
RETURNS int
AS
BEGIN
  DECLARE @sl int, @tl int, @i int, @j int, @sc nchar, @c int, @cl int,
    @cv0 nvarchar(4000), @cv1 nvarchar(4000), @cmin int
  SELECT @sl = LEN(@s), @tl = LEN(@t), @cv1 = '', @j = 1, @i = 1, @c = 0
  WHILE @j <= @tl
    SELECT @cv1 = @cv1 + NCHAR(@j), @j = @j + 1
  WHILE @i <= @sl
  BEGIN
    SELECT @sc = SUBSTRING(@s, @i, 1), @cl = @i, @c = @i, @cv0 = '', @j = 1, @cmin = 4000
    WHILE @j <= @tl
    BEGIN
      SET @c = @c + 1
      SET @cl = @cl - CASE WHEN @sc = SUBSTRING(@t, @j, 1) THEN 1 ELSE 0 END
      IF @c > @cl SET @c = @cl
      SET @cl = UNICODE(SUBSTRING(@cv1, @j, 1)) + 1
      IF @c > @cl SET @c = @cl
      IF @c < @cmin SET @cmin = @c
      SELECT @cv0 = @cv0 + NCHAR(@c), @j = @j + 1
    END
    IF @cmin > @d BREAK
    SELECT @cv1 = @cv0, @i = @i + 1
  END
  RETURN CASE WHEN @cmin <= @d AND @c <= @d THEN @c ELSE -1 END
END
GO
-- 测试
SELECT dbo.edit_distance_within('surface','sdsadas',10)
```

图 31　get_options 存储过程调用的算法函数(edit_distance_within)

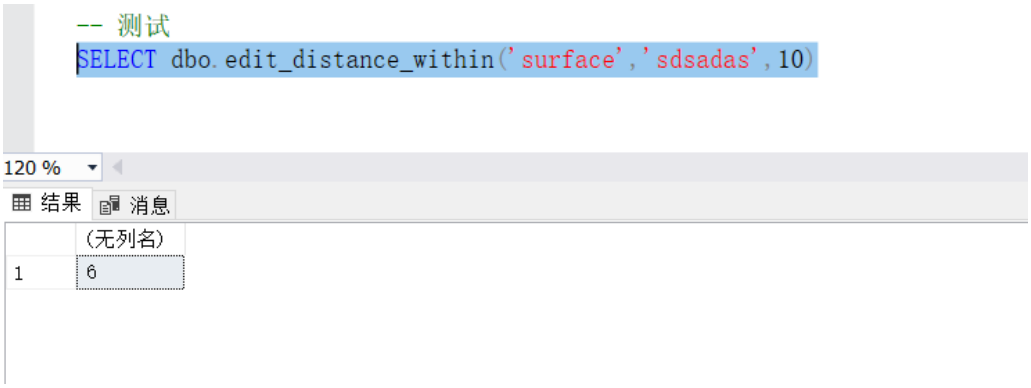


图 32 测试结果:

2. 触发器

2.1 tr_recite

用户背词的触发器(tr_recite)
该触发器用于用户背词时的处理，调用 update_study_record 存储过程。

```
-- 1.用户背词触发更新(选择选项)
-- 监听word_record表
-- 余下过程写入存储过程update_study_record中
-- 选错了 更新今日单词临时表+study_record表
-- 选对了   更新今日单词临时表(删去该单词，减少一行记录)+study_record表+plan表
-- study_record表更新时须记：初次为插入
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'tr_recite' AND type = 'TR')
DROP TRIGGER tr_recite
GO
CREATE TRIGGER tr_recite ON tb_word_record
AFTER INSERT,UPDATE
AS
DECLARE @u_id char(10), -- 插入/更新用户id
                @w_id char(6), -- 插入/更新当前词汇id
                @before_wrong_nums int,-- 插入/更新前的wrong_num
                @after_wrong_nums int,-- 插入/更新后的wrong_num
                @isRight int-- 是否选对 0错 1对
SELECT @u_id =u_id, @w_id =w_id, @after_wrong_nums=wrong_nums FROM INSERTED
SELECT @before_wrong_nums=wrong_nums FROM DELETED
IF (@before_wrong_nums = @after_wrong_nums OR @after_wrong_nums =0)
-- wrong_nums未发生改变或after_wrong_nums为0,即为选对了
    SET @isRight = 1
ELSE
-- 选错了
    SET @isRight = 0
EXECUTE update_study_record @u_id,@w_id,@isRight
```

图 33 用户背词的触发器(tr_recite)

2.2 tr_clock

用户自动打卡的触发器(tr_clock)

该触发器用于用户背完今日词汇时的处理，自动生成打卡记录。

```
-- 2.当日计划词汇背完自动触发打卡(生成打卡记录)
-- 监听study_record表
-- 插入clock表
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'tr_clock' AND type = 'TR')
DROP TRIGGER tr_clock
GO
CREATE TRIGGER tr_clock ON tb_study_record
AFTER UPDATE
AS
DECLARE @u_id char(10), -- 插入/更新用户id
                @s_total int, -- 插入/更新当日学习单词数
                @p_daywords int -- 用户计划每天学习单词数
SELECT @u_id =u_id, @s_total =s_total FROM INSERTED
SELECT @p_daywords= p_daywords FROM tb_plan WHERE u_id=@u_id
IF @s_total = @p_daywords
    INSERT INTO tb_clock(u_id,clock_time)VALUES(@u_id,GETDATE())
```

图 34 用户自动打卡的触发器(tr_clock)

六. 总结和反思

首先，在存储今日词汇的操作上，我们采用了临时表来存储今日词汇，弊端很明显。例如临时表在窗口会话关闭时会自动删除，但是只是为了测试开发的话，还是可以实现功能的完整性的。但如果项目推向落地，采用高级语言 JAVA 来落实的话，我们可能会用到 NoSql（Redis 缓存数据库）来解决这个问题。

再者，可能我们的 sql 代码有点冗余，在一些处理上可能不够简洁明了，代码不够精炼，但这仍需要时间的积累。我相信随着时间的推移，我们一定能写出更加精简、更加高效的代码来解决更复杂的问题。至此，感谢老师这半年的细心教导，有不足之处敬请斧正。