**存储过程及触发器**

1. **存储过程**
2. 用户注册的存储过程(add\_user)

|  |
| --- |
| 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 'abc','123','0','1622135447@qq.com','暂无' |

1. 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。

|  |
| --- |
| 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' |

1. 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。

|  |
| --- |
| 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 |

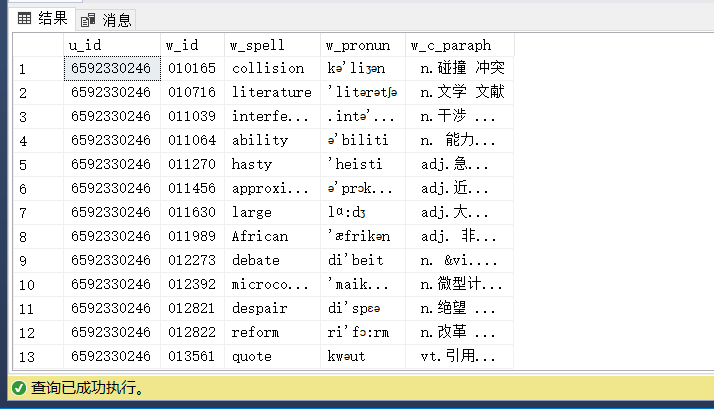
1. 生成今日词汇的存储过程（get\_today\_words）

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。

|  |
| --- |
| 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' |

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。

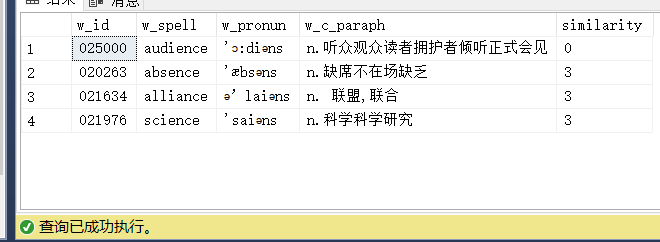
|  |
| --- |
| -- 存入临时表  -- 创建临时表  CREATE TABLE ##temp\_todaywords (  u\_id char(10) not null,  w\_id char(6) not null,  w\_spell varchar(50) not null,  w\_pronun nvarchar(50) not null,  w\_c\_paraph nvarchar(255) not null,  CONSTRAINT un\_wid unique(w\_id)  )  -- 临时表插入数据  INSERT INTO ##temp\_todaywords EXECUTE get\_today\_words '6592330246'  -- 查询临时表  SELECT \* FROM ##temp\_todaywords  -- 删除临时表  if OBJECT\_ID('tempdb..##temp\_todaywords') is not null  drop table ##temp\_todaywords |



1. 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。

|  |
| --- |
| 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' |



1. 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。

|  |
| --- |
| 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' |

1. 触发器tr\_recite触发的存储过程(update\_study\_record)

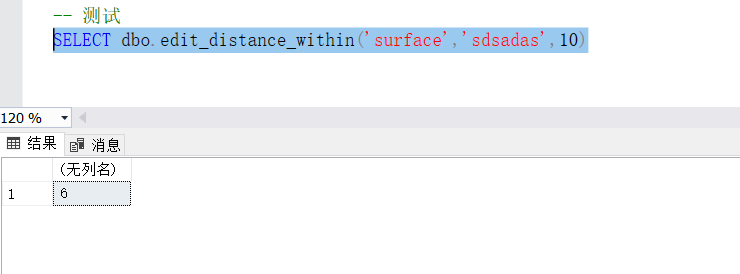
该存储过程用于用户背单词时学习记录表(tb\_study\_record)的更新以及计划表(tb\_plan)的更新。用户选对时，删除今日词汇表(临时表##temp\_todaywords)中的一行该单词数据；用户选错时，会重新插入该单词数据至今日词汇表。同时更新/插入学习记录表和计划表的数据。

|  |
| --- |
| 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 |

1. get\_options存储过程调用的算法函数(edit\_distance\_within)

|  |
| --- |
| /\*  \*\*@param s和t为比较字符串  \* @param 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, @c1 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), @c1 = @i, @c = @i, @cv0 = '', @j = 1, @cmin = 4000  WHILE @j <= @tl  BEGIN  SET @c = @c + 1  SET @c1 = @c1 - CASE WHEN @sc = SUBSTRING(@t, @j, 1) THEN 1 ELSE 0 END  IF @c > @c1 SET @c = @c1  SET @c1 = UNICODE(SUBSTRING(@cv1, @j, 1)) + 1  IF @c > @c1 SET @c = @c1  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) |

测试结果:



1. **触发器**

1.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。

|  |
| --- |
| 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 |

2.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。

|  |
| --- |
| 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()) |