TempMT_Index 使用手册

目录

第1章安装

1.1 环境需求

OS: windows 64 位操作系统（包含服务器版本），
JAVA 运行环境，JDK1.5 或以上
MYSQL 5.0 或以上
Tomcat 7.0 或以上
多核 CPU 硬件环境（仅使用多线程策略时）

1.2 通用环境下安装 TempMT_Index

1.2.1 安装 JDK

从 http://www.oracle.com/technetwork/java/index.html 下载并安装 Java SDK 1.5 以上环境。并且配置好环境变量。

1.2.2 安装 Mysql

- **下载 Mysql**

从 http://www.mysql.com/downloads/ 下载并安装 Mysql。

- **设置 Mysql**

在安装的最后一步，安装程序会弹出对话框询问是否设置 MySql 数据库。此时可以对其进行相关设置。
使用 TCP/IP 网络连接，指定端口号（MySQL 默认为 3306 端口）。如图 1-1 所示。

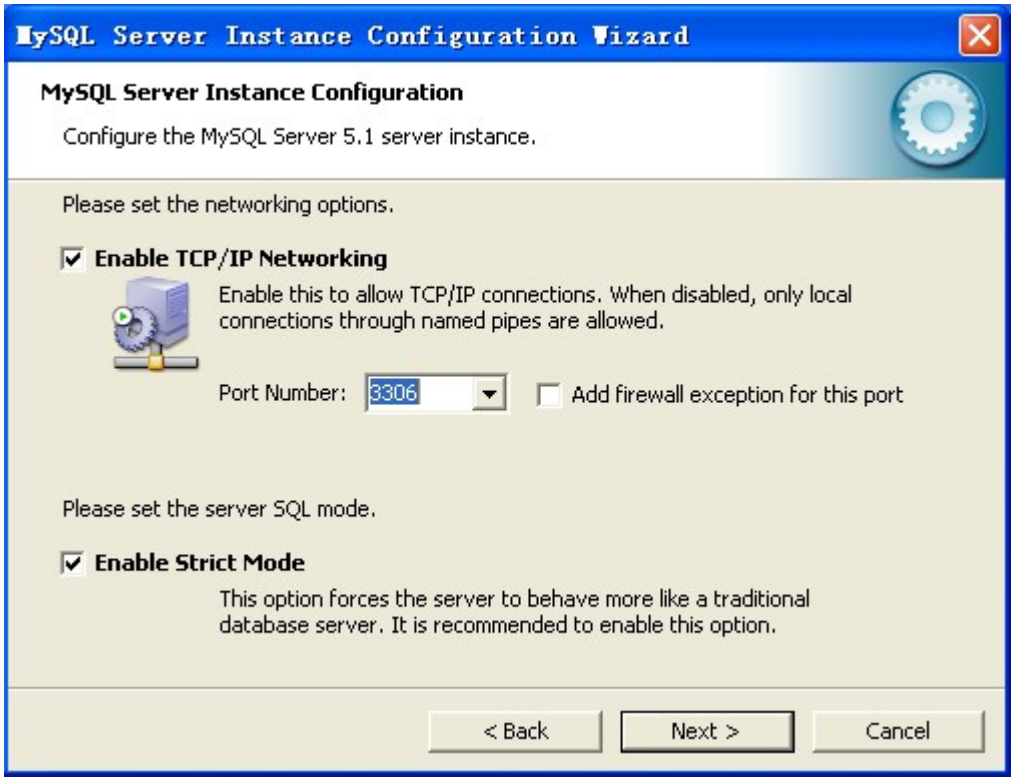

图 1-1 Mysql 服务器的网络设置

使用 UTF-8 作为默认的字符集，以便于存储和使用中文。如图 1-2 所示。

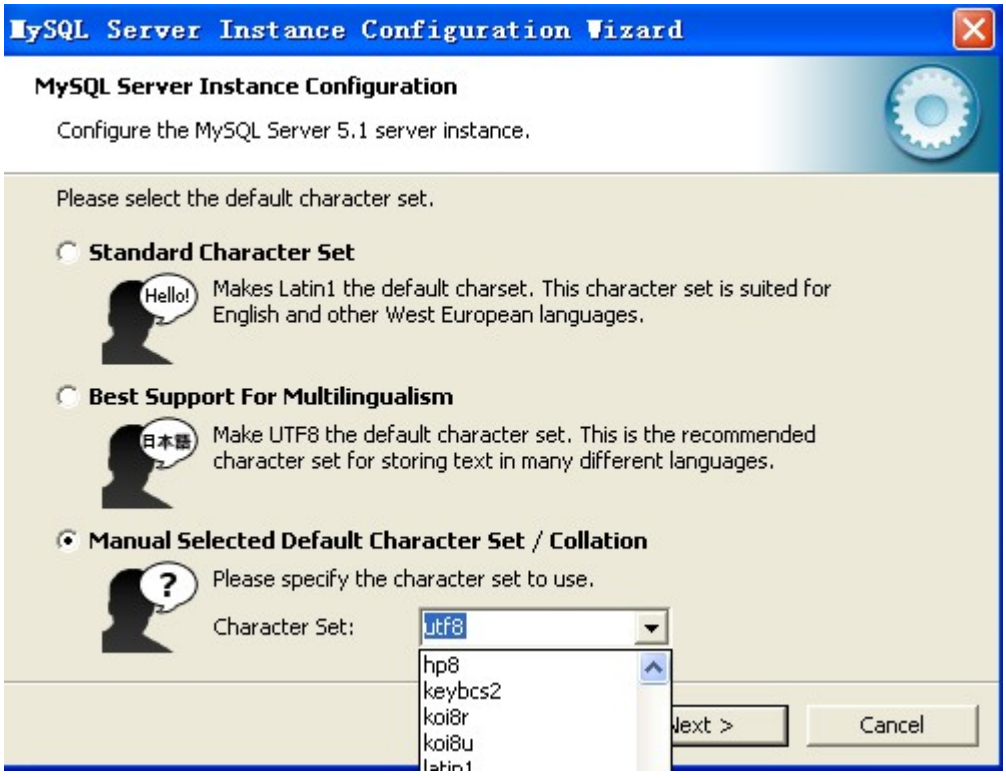

图 1-2 MySql 设置中选择字符集

需要为 root 用户指定密码，默认为空（由数据库管理员完成）。

- **导入实验数据库**

在 cmd 中运行命令 mysql -u root –p，登录命令后端，运行脚本 tmtindex 导入.bat。

1.2.3 安装 Tomcat

从 http://tomcat.apache.org/ 下载，安装 Tomcat 并启动 tomcat 服务

1.2.4 配置 web 系统

找到 tomcat 目录下的 webapp 文件夹，将 TempMT_Index.war 工程文件放入此目录，如图 1-3 所示。

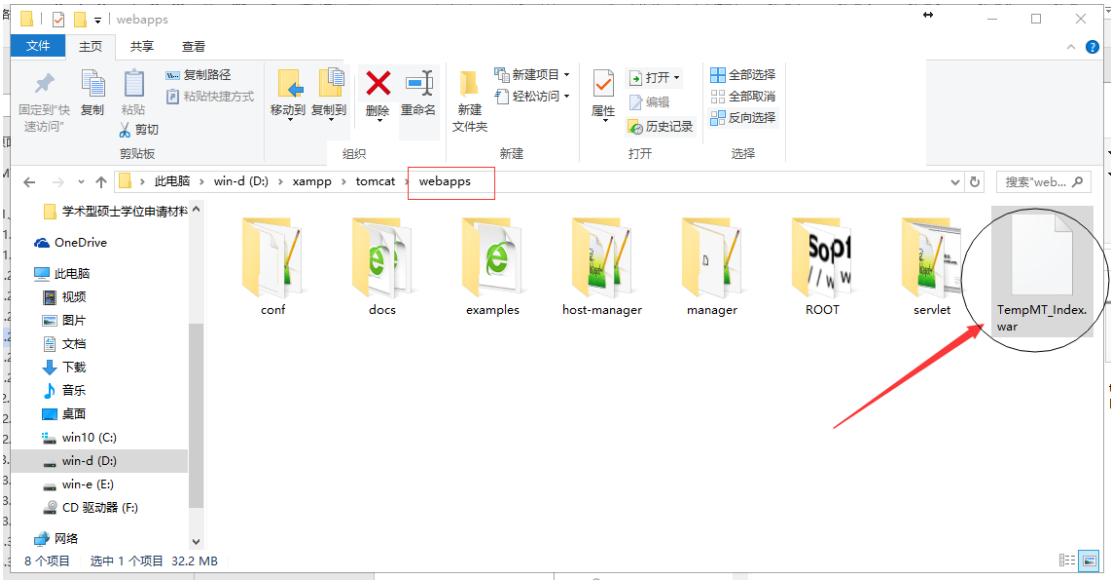

图 1-3 tomcat 的部署目录

重新启动 tomcat 服务，发现有一展开的目录 TempMT_Index，访问浏览器 http://localhost:8080/TempMT_Index/ 即可成功运行，如图所示。

图 1-4 开启 TempMT_Index 访问地址

1.2.5 卸载 TempMT_Index

将 TempMT_Index 目录和 TempMT_Index.war 文件从 webapps 目录中删除即可。

1.3 集成环境下安装 TempMT_Index

1.3.1 安装 JDK

从 http://www.oracle.com/technetwork/java/index.html 下载并安装 Java SDK 1.5 以上环境。并且配置好环境变量。

1.3.2 安装 xampp 集成工具包

xampp 集成工具包中配备有 mysql 和 tomcat 工具，更加快速和简洁安装使用环境。

- **解压 xampp-win32-5.6.30-0-VC11.7z 集成包**
- **配置 xampp 和打开 xampp 控制面板**
- **开启 apache 服务和 mysql 服务**
- **导入 mysql 环境数据**
- **布置工程文件**
- 找到 tomcat 目录下的 webapp 文件夹，将 TempMT_Index.war 工程文件放入此目录，如图 1-5 所示。

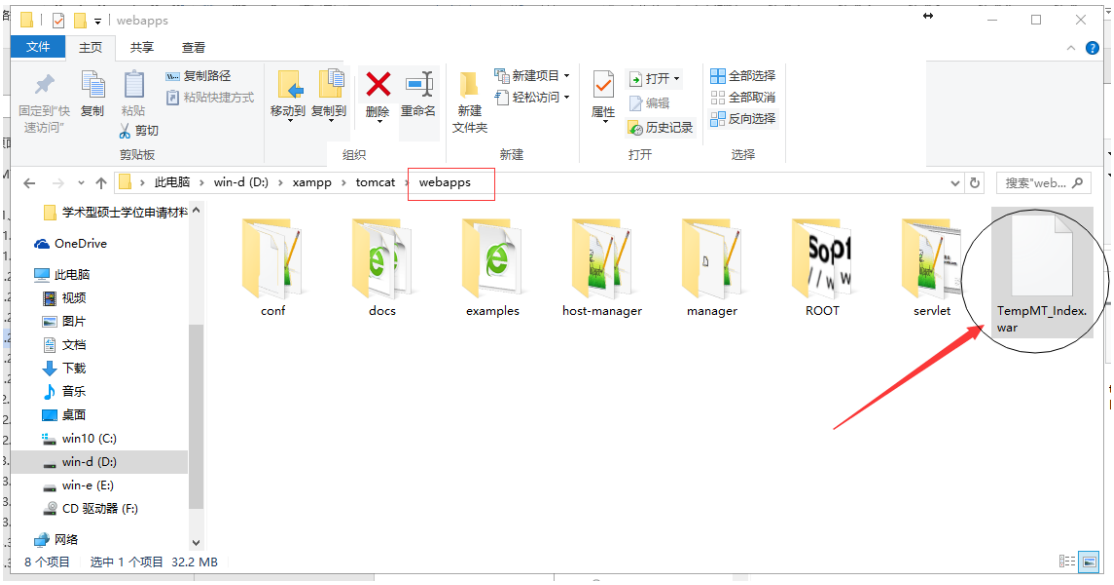

图 1-5 布置工程文件

- **开启 tomcat 服务**

- **访问 web 系统**

重新启动 tomcat 服务，发现有一展开的目录 TempMT_Index，访问浏览器 http://localhost:8080/TempMT_Index/ 即可成功运行，如图所示。

图 1-6 开启 TempMT_Index 访问地址

1.3.3 卸载 TempMT_Index

将 TempMT_Index 目录和 TempMT_Index.war 文件从 webapps 目录中删除即可。

第2章如何使用 TempMT_Index

2.1 客户机访问

在正确部署 TempMT_Index 之后，访问浏览器 http://localhost:8080/TempMT_Index/ 即可开启太。在局域网内部可通过 http://局域网内部 ip 地址:8080/TempMT_Index/访问，在互联网可使用 http://广域网 ip 地址或者域名:8080/TempMT_Index/访问。

图 2-1 TempMT_Index 界面

若要退出平台，关闭浏览器即可。

2.2 软件的界面

TempMT_Index 系统首页，包含系统 logo、标题、导航栏、背景等，其他所有模块如图 2-2 至图 2-9 所示。

图 2-2 主页功能区页面展示图

图 2-3 内置语句选项效果图

图 2-4 索引类型选项效果图

图 2-5 数据库信息展示效果图

查询结果显示区

sid	name	e_mail	course_id	vts_timeDB	vte_timeDB
56	Kyle Gomez	MarvinMurphy@163.net	100063	1990-01-04 13:38:39.0	2019-08-05 08:56:17.0
56	Kyle Gomez	MarvinMurphy@163.net	100063	1985-12-28 21:04:18.0	2018-04-20 16:23:22.0
56	Kyle Gomez	MarvinMurphy@163.net	100063	1990-04-23 15:00:09.0	2016-05-10 02:19:19.0
56	Kyle Gomez	MarvinMurphy@163.net	100063	1984-01-08 18:03:29.0	2021-05-06 09:08:12.0
56	Kyle Gomez	MarvinMurphy@163.net	100063	1993-05-09 20:04:29.0	2022-05-09 06:17:31.0
57	Diana Stevens	BradleyDuncan@msn.com	100075	1995-08-12 05:38:46.0	2024-03-03 11:16:49.0
57	Diana Stevens	BradleyDuncan@msn.com	100005	1993-04-16 01:15:56.0	2025-10-27 14:14:33.0
57	Diana Stevens	BradleyDuncan@msn.com	100005	1986-08-17 19:45:12.0	2021-08-10 17:58:27.0
59	Mike Martinez	ValerieNichols@163.com	100010	1992-04-29 00:08:55.0	2023-02-05 19:44:05.0
60	Sherry Griffin	DerekAdams@msn.com	100087	1991-08-27 04:53:32.0	2019-11-15 18:01:17.0

首页 上一页 **11** 12 13 下一页 尾页

图 2-6 时态选择跨度查询结果图

查询结果显示区

sid	name	e_mail	course_id	vts_timeDB	vte_timeDB
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2005-06-24 03:23:33.0	2024-09-04 15:45:43.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2008-05-16 11:24:36.0	2017-03-29 05:48:35.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2009-03-29 00:23:29.0	2025-10-31 10:05:23.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1998-12-23 11:44:11.0	2025-01-17 04:44:51.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1991-12-17 03:46:44.0	2026-02-28 18:44:29.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1994-01-09 15:35:23.0	2016-05-31 14:54:01.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2005-08-14 04:40:21.0	2023-09-07 20:53:55.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2001-06-23 20:19:03.0	2026-07-31 10:46:15.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2004-08-14 11:59:15.0	2026-06-19 16:32:02.0
64	Ana Barnes	CarolMyers@sogou.com	100070	2012-03-28 09:56:12.0	2016-06-03 18:40:41.0

首页 **1** 2 下一页 尾页

图 2-7 时态选择期间包含查询结果图


Tempdb
帮助与下载
查询
数据库
语言
帮助与下载
我们的团队
2016年1月17日
不带时间查询
功能 ATSQL创建数据库 create table chufang4(id int primary key,name varchar(100) not null,mename varchar(100) not null);...
阅读全文 下载全文
2016年1月17日
基于时间查询
功能 ATSQL创建数据库 create table chufang(id int,name varchar(100),mename varchar(100))as validtime;时态查询 validtime select * from student; validtime select * from ...
阅读全文 下载全文
2016年1月17日
TempDB 2.2使用手册
SQL主要分成四部分：
DDL（数据定义）：用于定义SQL模式、基本表、视图和索引的创建和 撤消操作。CREATE ALTER DROP TRUNCATE COMMENTRENAMEDML（数据操纵）：...
阅读全文 下载全文

图 2-8 帮助与下载

图 2-9 团队信息

第3章TempMT_Index 的 atsql 命令举例说明

本示例是基于一个有四个数据表格的示例数据库。数据库的 ER 图如图 3.1 所示。

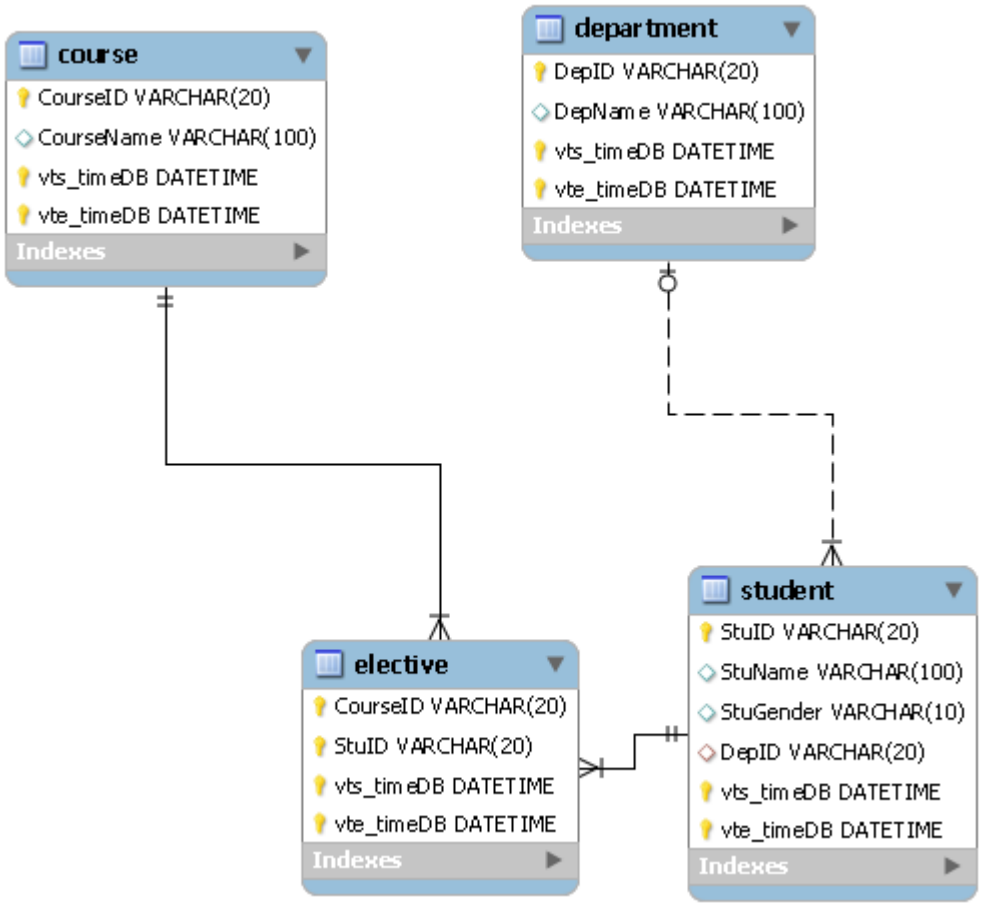

图 3.1 示例数据库 ER 图

值得注意的是，由于目前没有一个通用的结构来表示我们的时态数据库中的表格。因此我们现在示例的 ER 图（图 4.1）是由后台关系型数据库中的表格的角度来说明的。需要说明的是，这个三个时态表格中的主键是为用户指定主键联合两个有效时间列构成的联合主键。

3.1 创建时态表格

由于四个时态表格的创建是具有相似性，因此，本节仅给出 temp 表的创建语句并进行分析。

A、ATSQL 语句

```
CREATE TABLE temp
(id INT,
name VARCHAR(255),
email VARCHAR (255),
course_id VARCHAR (20)
)AS VALIDTIME;
```

在此语句中，我们指定了 temp 表格的表结构，并指定创建的是一张时态表。

B、转换后的标准 SQL 语句。

```
CREATE TABLE temp
(id INT,
name VARCHAR(255),
email VARCHAR (255),
course_id VARCHAR (20),
vts_timeDB datetime,
vte_timeDB datetime
);
```

从在转换之后的标准 sql 语句可以看到，这个时态表格的创建反映在后台关系数据库中会增加两个时间列用以存储有效时间。

3.2 删除时态表

删除某一特定的时态表和我们的 SQL 操作是一致的。只需要使用“drop table”关键字进行删除即可。

3.3 插入记录

A 时态数据插入语句

```
VALIDTIME PERIOD [ DATE '2014-09-01' - DATE '2017-07-01' ] INSERT INTO temp VALUES
( '67' ,'Chi Xuehui','qazcxh@163.com','100070') ;
```

B 转换后的标准 SQL 语句

```
INSERT INTO temp VALUES ('67','Chi Xuehui', 'qazcxh@163.com',
'2014-09-01', '2017-07-01');
```

C 执行该插入语句前后数据库表格的相关部分如图 3-1 和图 3-2 所示。

+ 选项

id	name	email	course_id	vts_timeDB	vte_timeDB
67	Chi Xuehui	qazcxh@163.com	100070	2014-09-01 00:00:00	2014-12-01 00:00:00
67	Chi Xuehui	qazcxh@163.com	100070	2010-09-01 00:00:00	2014-07-01 00:00:00

图 3-1 时态数据插入前

+ 选项

id	name	email	course_id	vts_timeDB	vte_timeDB
67	Chi Xuehui	qazcxh@163.com	100070	2014-09-01 00:00:00	2014-12-01 00:00:00
67	Chi Xuehui	qazcxh@163.com	100070	2010-09-01 00:00:00	2014-07-01 00:00:00
67	Chi Xuehui	qazcxh@163.com	100070	2014-09-01 00:00:00	2017-07-01 00:00:00

图 3-2 时态插入后

D 用例解释

此用例展示的是最为普通的时态插入数据。运行该语句后，后台数据库中生成一条对应插入元组的记录。

3.4 查询记录

3.4.1 期间包含查询

A 查询语句

```
VALIDTIME PERIOD [ DATE '2013-1-1' - DATE '2014-7-1' ] SELECT * FROM student WHERE course_id = 100070;
```

B 转后的标准 SQL 语句

```
SELECT * FROM student WHERE course_id = 100070 AND vts_timeDB <= '2013-1-1' AND '2014-7-1' <= vte_timeDB;
```

C 执行该查询语句后的结果集如图 3-3 所示。

+ 选项

sid	name	e_mail	course_id	vts_timeDB ▲ 1	vte_timeDB
64	Ana Barnes	CarolMyers@sogou.com	100070	1986-12-23 01:28:51	2019-05-26 02:01:48
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1991-12-17 03:46:44	2026-02-28 18:44:29
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1994-01-09 15:35:23	2016-05-31 14:54:01
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1998-12-23 11:44:11	2025-01-17 04:44:51
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2001-06-23 20:19:03	2026-07-31 10:46:15
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2004-08-14 11:59:15	2026-06-19 16:32:02
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2005-06-24 03:23:33	2024-09-04 15:45:43
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2005-08-14 04:40:21	2023-09-07 20:53:55
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2008-05-16 11:24:36	2017-03-29 05:48:35
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2009-03-29 00:23:29	2025-10-31 10:05:23
64	Ana Barnes	CarolMyers@sogou.com	100070	2012-03-28 09:56:12	2016-06-03 18:40:41

图 3-3 期间包含查询结果

D 用例解析

只是一个时态查询的例子。在默认的情况下，时态查询将执行包含查询的语义，也就是将两

个时间列整合成一个有效时间来对待的时态查询。

3.4.2 时态全表查询

A 查询语句

```
VALIDTIME SELECT * FROM student;
```

B 转换后的标准 SQL 语句

```
SELECT * FROM student ;
```

C 执行语句之后的查询结果集如图 3-4 所示。

查询结果显示区

sid	name	e_mail	course_id	vts_timeDB	vte_timeDB
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2005-06-24 03:23:33.0	2024-09-04 15:45:43.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2008-05-16 11:24:36.0	2017-03-29 05:48:35.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2019-07-21 21:07:33.0	2023-11-19 05:38:43.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2026-11-15 15:39:09.0	2026-12-17 23:39:23.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2026-06-15 05:33:20.0	2026-12-07 09:02:22.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2009-03-29 00:23:29.0	2025-10-31 10:05:23.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2020-06-11 09:19:24.0	2021-02-06 22:58:33.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1998-12-23 11:44:11.0	2025-01-17 04:44:51.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1991-12-17 03:46:44.0	2026-02-28 18:44:29.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1994-01-09 15:35:23.0	2016-05-31 14:54:01.0

首页 1 2 3 4 5 6 7 8 9 10 下一页 尾页

图 3-4 时态全表查结果

D 用例解析

此例子展示了全表时态查询的结果，便于输出全部记录。

3.4.3 快照查询

A 查询语句

```
VALIDTIME SNAPSHOT SELECT * FROM student;
```

B 转换后的标准 SQL 语句

```
SELECT * FROM student where
vts_timeDB <= now()
and now() <= vte_timeDB ;
```

C 执行该查询语句后的结果集如图 3-4 所示

查询结果显示区

sid	name	e_mail	course_id
0	Cheryl Wagner	AnnaJordan@21cn.com	100070
44	Dale Morales	TheresaMcdonald@hotmail.com	100033
38	Michelle Willis	AlanWillis@126.com	100082
44	Dale Morales	TheresaMcdonald@hotmail.com	100053
7	Brittany Rogers	BeatriceJenkins@sina.com	100024
38	Michelle Willis	AlanWillis@126.com	100077
22	Carolyn Rose	LarryCunningham@sina.com	100062
7	Brittany Rogers	BeatriceJenkins@sina.com	100027
35	Dale Nichols	BonnieGibson@sina.com	100009
56	Kyle Gomez	MarvinMurphy@163.net	100063

首页 1 2 3 4 5 6 7 8 9 10 下一页 尾页

图 3-5 快照查询结果

D 用例解析

从转换后的 SQL 语句中可以看出，TempMT_Index 对待在时态表格上的标准 SQL 查询，会默认地将当前有效的元组组织成为结果集输出。所以，在转换后的 SQL 语句中，TempMT_Index 增加了判断元组的有效时间区间是否包含 NOW 的条件。该查询语句输出的结果集只包含了当前有效的元组。

3.4.4 跨度查询

A 查询语句

```
VALIDTIME INTERVAL year > 24
SELECT * FROM student;
```

B 转换之后的 SQL 语句

```
SELECT * FROM student where
```

timestampdiff (year , vts_timeDB , vte_timeDB) > 24 ;

C 执行该查询语句后的结果集如所示。

查询结果显示区

sid	name	e_mail	course_id	vts_timeDB	vte_timeDB
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1998-12-23 11:44:11.0	2025-01-17 04:44:51.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	1991-12-17 03:46:44.0	2026-02-28 18:44:29.0
0	Cheryl Wagner	AnnaJordan@21cn.com	100070	2001-06-23 20:19:03.0	2026-07-31 10:46:15.0
1	Dan Elliott	LeonardWillis@sohu.com	100089	1986-05-28 18:10:51.0	2017-10-17 03:13:06.0
2	Marion Ross	DanielleFoster@263.net	100055	1995-08-18 14:50:30.0	2024-10-01 12:26:50.0
3	Hazel Anderson	TeresaTurner@@mail.com	100090	1987-05-04 14:33:35.0	2016-10-23 10:18:52.0
3	Hazel Anderson	TeresaTurner@@mail.com	100090	1988-03-22 05:21:27.0	2014-12-12 20:20:17.0
3	Hazel Anderson	TeresaTurner@@mail.com	100090	1993-11-12 09:14:35.0	2026-04-15 01:44:03.0
4	Russell Wagner	MatthewAdams@163.com	100058	1987-04-21 21:19:20.0	2024-06-14 08:32:37.0
4	Russell Wagner	MatthewAdams@163.com	100058	1986-11-27 10:00:37.0	2015-08-07 07:51:07.0

首页 1 2 3 4 5 6 7 8 9 10 下一页 尾页

图 3-6 跨度查询结果

D 用例解析

从转换后的 SQL 语句看出，跨度查询将两个有效时间属性进行了相应的计算后，做的处理。

3.4.5 其他查询

投影查询、连接查询、带索引的包含查询等等。

3.4.6 多线程查询实例

本系统实现了多种方案的包含查询的选项，进行多线程查询的实例展示。

①输入 ATSQL 格式的时态语句，如图 3-7 所示，或者选取预定时态语句，如图 3-8 所示。

②在索引类型框中选择多线程方式，这里采用的是最优的划分策略方案-多线程的前部交叉策略，如图 3-9 所示。

图 3-7 输入语句

图 3-8 选择预定语句

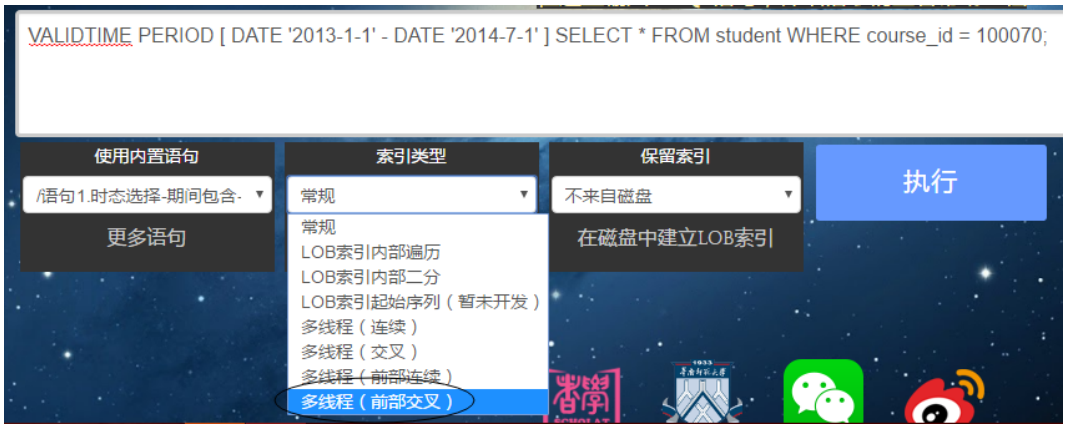

图 3-9 选择多线程策略

③点击执行按钮，如图 3-10 所示。

④执行查询逻辑，得到返回的提示信息，由于 student 表中的数据量为 1000 条，这里的开销小于 1ms，如图 3-11 所示。

图 3-10 点击执行按钮

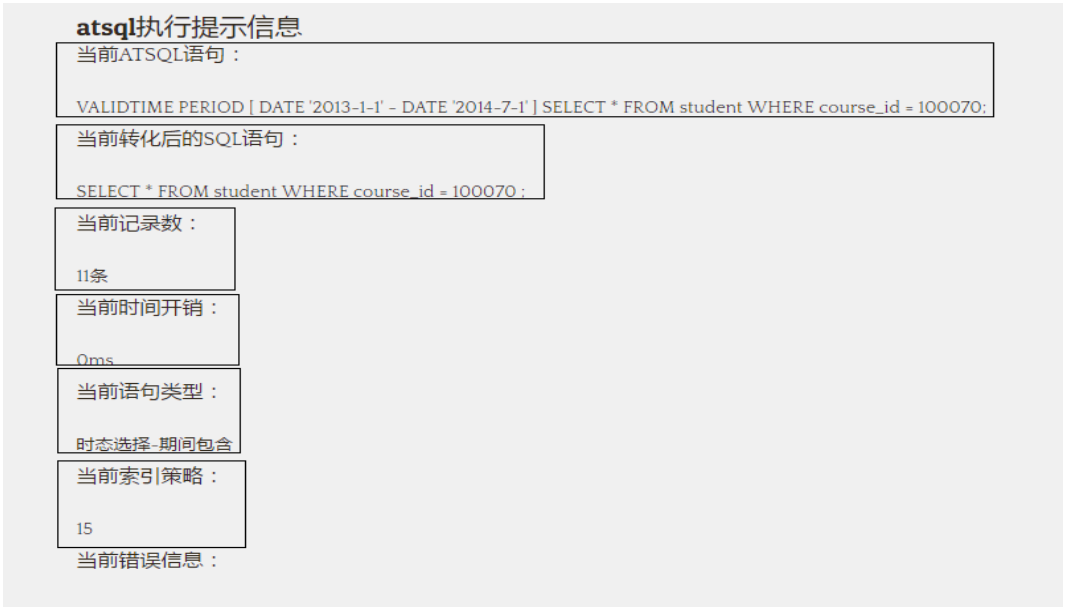

图 3-11 提示信息

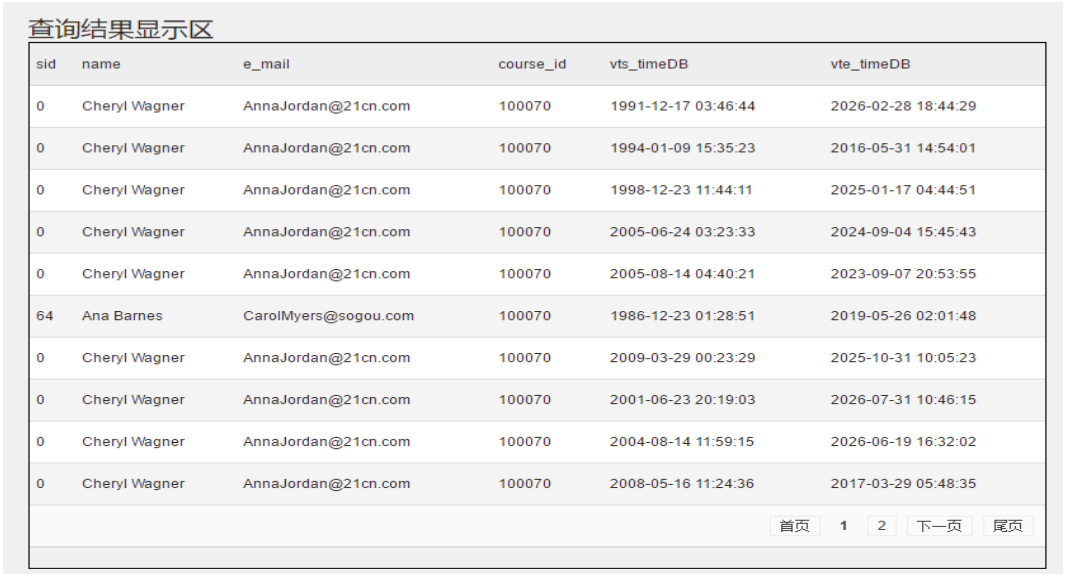

图 3-12 多线程查询结果

⑤查看返回结果，以表格的形式显示，支持可翻页浏览，如图 3-12 所示。

第4章TempMT_Index 编程接口设计

通过这些编程接口，使用者可以通过程序交互的方式使用 TempMT_Index 提供的时态处理能力。

4.1 典型模块的设计与实现

Web 系统采用逻辑业务、控制、界面显示分离的方式组织，便于系统算法业务方面的修改，也便于系统功能扩展与移植等。如图 4-1 所示。

4.1.1 时态索引相关算法模块

包含有时态索引构建算法模块，采用时态索引的期间包含查询算法模块，采用多线程优化的期间包含查询算法模块（含四种分支策略）。

4.1.2 时态数据平台相关算法模块

时态投影查询算法模块，快照查询算法模块，时态连接查询算法模块，时态选择跨度查询算法模块，ATSQL2 中间件模块，时态数据元组模型，提示信息模块。

4.1.3 期间选择策略控制模块

采用二进制标志位方式，设置参数，树的结点有两个孩子从左至右分别为 0，1，三个孩子时从左至右为 00，01，10。对于包含查询，共 8 种策略采用 4 位二进制表示。可知一次性采用 SQL 查的策略标志为 0000，高位为 1 时是采用了时态索引，高位为 10 采用单线程查，其中分支内部遍历标志为 1000，分支内部二分法为 1001，分支内部起始序列法为 1010，高位为 11 采用多线程策略，其中连续策略为 1100，交叉策略为 1101，前部连续策略为 1110，前部交叉策略为 1111。如图 4-2 所示。

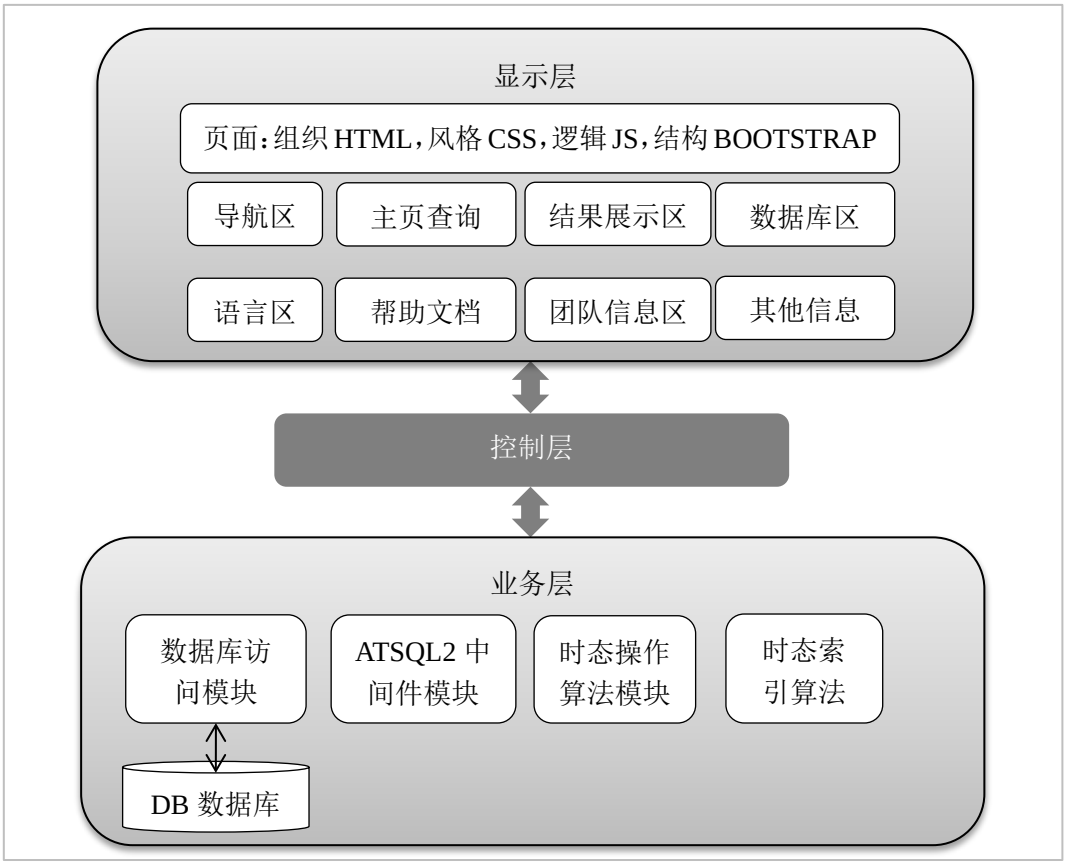

图 4-1 系统模块图

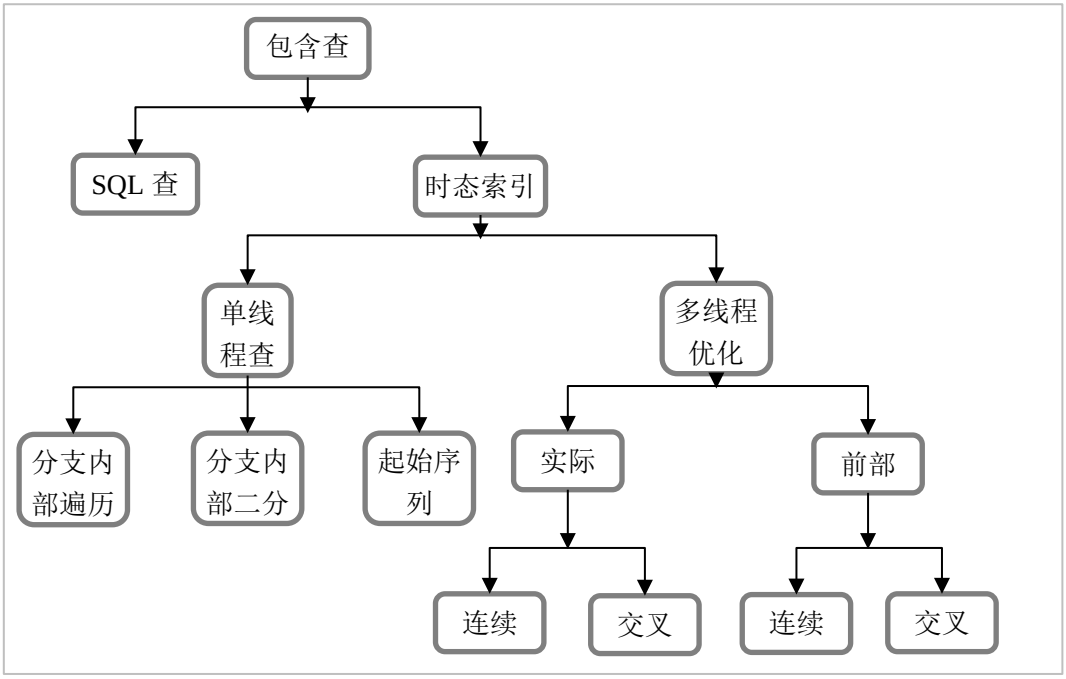

图 4-2 期间选择策略控制图

4.1.4 磁盘文件索引模块

为了便于索引能够及时方便的读取，在磁盘内以文件格式存放已经建立好的索引。系统中索

引来自于磁盘文件，将上层索引采用单一文件以一定的格式存储，对每一个线序分支的数据都建立单独的文件进行存放。当进行查询时先检索上层结点的单一文件，对于线序分支内部需要进一步检索的结果，再读取对应的单独数据文件，由于文件独立存放和查询的部分性，可以减少访问磁盘的 IO 次数，减少查询时间开销，进而增加了查询效率，如图 4-3 所示。

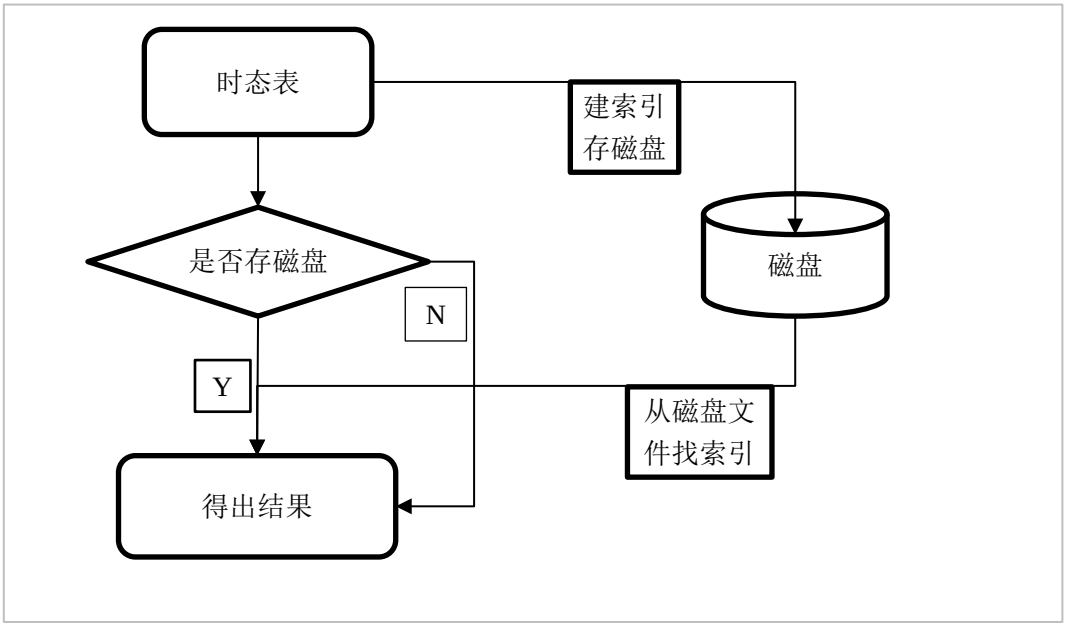

图 4-3 磁盘存放时态索引模块

4.1.5 底层数据库连接模块

采用 JDBC 包，底层数据库为 Mysql。通过 JAVA 的类加载器读取 mysql.properties 中的连接信息等，通过 Class.forName(driver)加载驱动，获取 Connection 连接，调用其 PreparedStatement 加载 SQL 语句，并进行 query，update 等方法的封装，其中 update 方法支持事务性，对异常信息做好抛出处理，并对异常与语句执行完毕后合理调用 close()方法，形成单例模式的数据访问接口 IDAO，统一对所有连接数据库的入口。

4.1.6 单页页面显示模块

为便于展示方便，系统采用单页下拉页面，便于更好的交互学习，也便于操作者便捷使用。页面内容主要分为八个区块：导航区，主页查询区，结果展示区，数据库区，语言区，帮助文档区，团队信息区，其他信息区。

导航区：系统中的展示页面的导航，主要包括 logo，系统名称，可导航至主页查询区，数据库区，语言区，帮助文档区，团队信息区等。起到单页面的页面定位作用。

主页查询区：系统中最重要的部分，包含有 logo，系统名称，查询输入框，查询策略选项，内置 ATSQL 语句块，磁盘存放索引选项等，还包括有外部链接和分享链接，便于系统共享。是系统中最核心的操作区域。

结果展示区：系统中结果回显模块调用的页面，包括提示信息与查询结果信息的展示。

数据库区：查看当前数据库的状态信息。

语言区：语言模块，和一句话模块增强系统的生动性，提供优质内容。

帮助文档区：系统中对于了解系统的重要资料部分。包含与时态索引相关的文献论文，ATSQL2 语句的介绍，时态数据平台的其他系统下载，实例数据库下载，本系统的部署文档等。便于系统共享交流。

团队信息区：介绍本系统相关理论和相关实践工作等的人员信息的情况。
其他信息区：系统的描述信息与页面相关的信息。

4.1.7 结果回显模块

结果回显模块调用包括提示信息和查询结果的回显。提示信息对于 DDL，DQL，DML 等语句执行都会有回显，而查询结果只针对 DQL 的执行后的结果展示，采用 table 表格分页的形式。当 html 提交 ATSQL2 语句后，通过调用 AJAX 异步更新模块发送给指定的语句 Servlet， 语句 Servlet 接收请求和请求的参数，将过程交给业务逻辑层的服务处理，服务根据参数调用对应的 DQL，DDL，DML 中间件和对应的业务查询功能得到结果和提示信息，该结果和提示信息又返回给语句 Servlet，语句 Servlet 将结果返回给 AJAX 的回应模块，通过 JS 采用分页的格式，将数据推送到 html 页面回显结果区域。执行流程也可如图 4-4 所示。

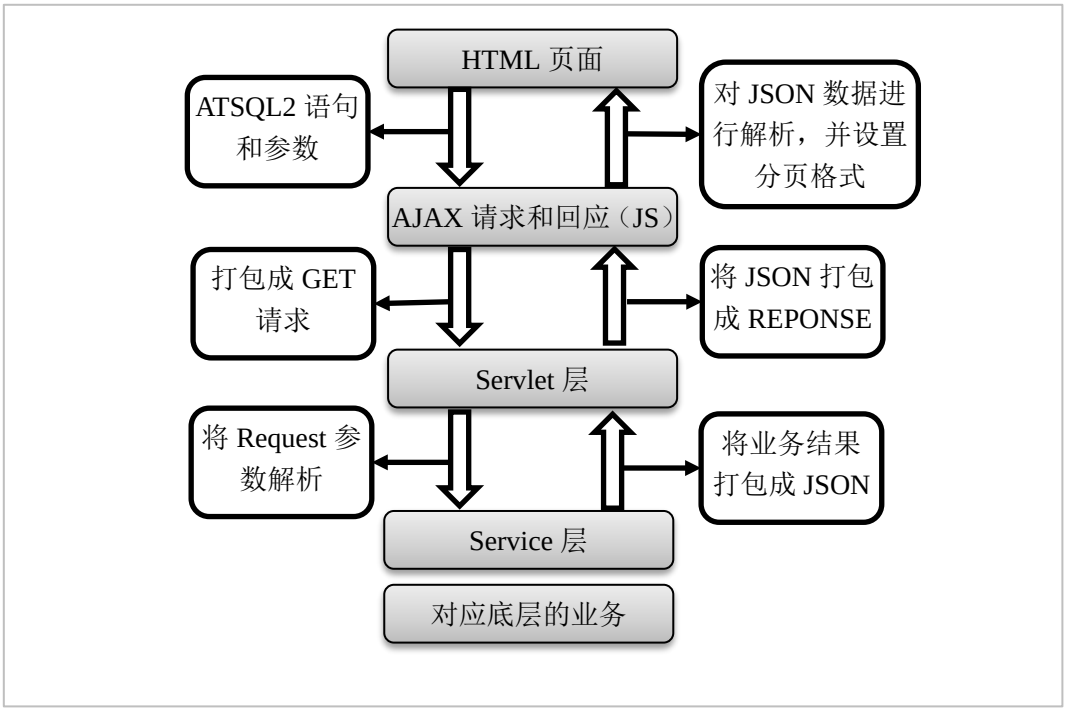

图 4-4 结果回显提示模块流程图

4.2 详细设计

数据库访问层 DAO：dao 包，配置文件目录：conf/。
业务层：service 包，所有的业务逻辑。
控制层：web.action 包，WebRoot/WEB-INF/web.xml。
视图层：在 WebRoot/目录下。
日志信息：log/。
实验仿真：实验逻辑：lab，实验数据：lab/。
数据库访问层：IDAO：定义了 Dao 层可进行的操作，可扩展到多数据库平台。MysqlDAO：IDAO 的 Mysql 数据库实现。DAOFactory：getInstance()方法根据传入的 Dao 名，创建一个实现了 IDao 接口的实例。如图 4-6 所示。
业务层设计包含 DDL,DQL,DML 的中间件，如图 4-5 所示。包含时态索引算法设计（含多线程），如图 4-7 所示。包含数据本体与时间标签，如图 4-8 所示。

视图层设计，为页面展示，包括 index.jsp，WEB-INF/MainFrame.html，js/，css/，img/，help/，font/， doc/。

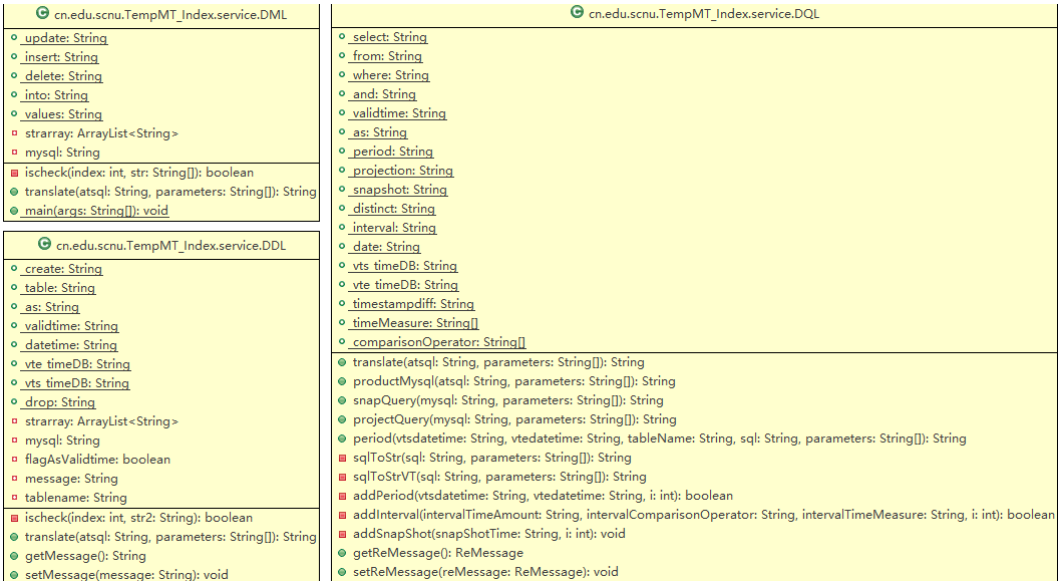

图 4-5 时态语句中间件结构图

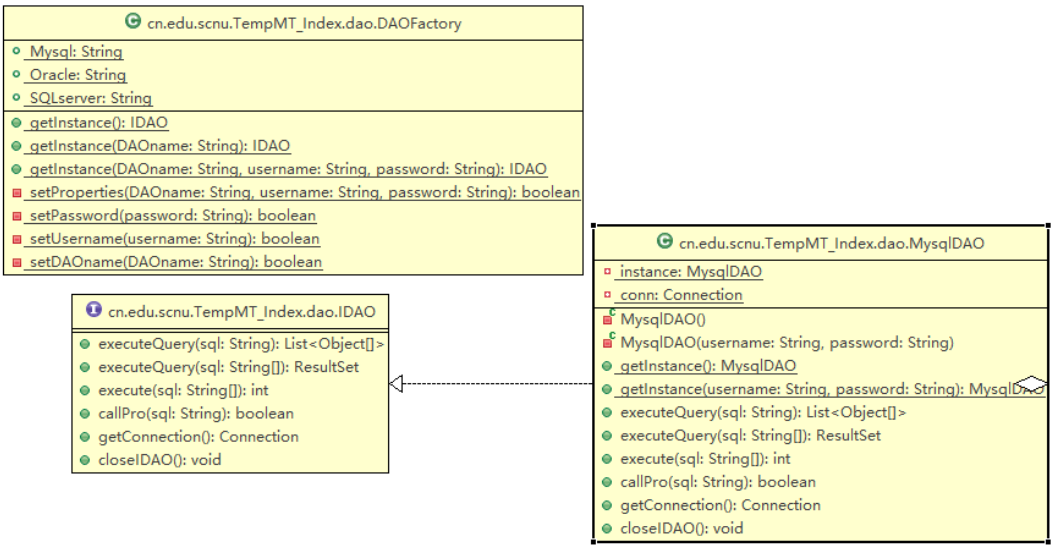

图 4-6 数据访问层及结构图

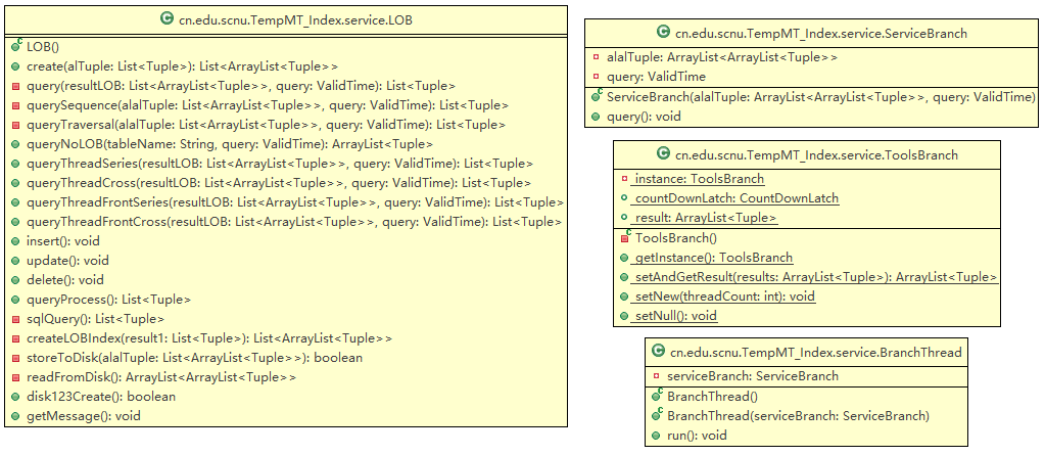

图 4-7 时态索引算法图

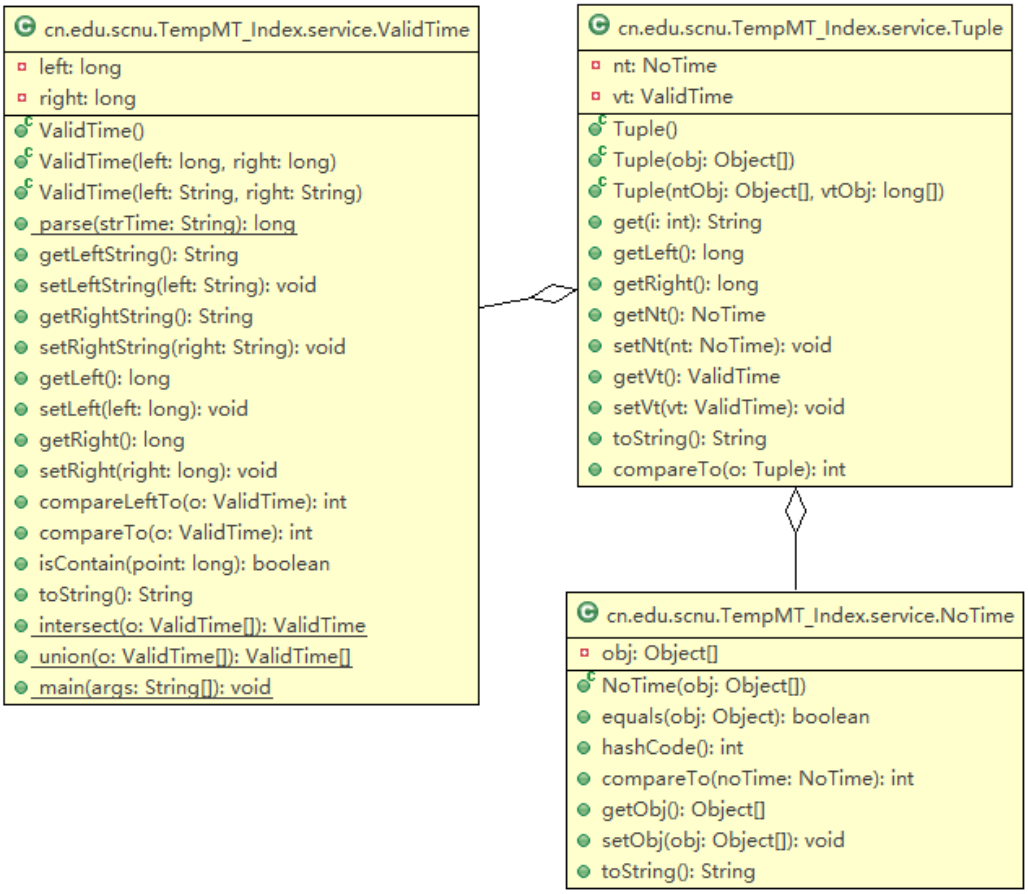

图 4-8 数据本体与时间标签

第5章附录一　ATSQL2 的 BNF 定义

ATSQL2 为本文采用的时态查询原型，以下为 TempMT_Index 中实现的 ATSQL2 变型语言的 BNF 语法定义：

0. 语句组成

Statement　: = (query | ddl | dml | control) ';'

1．查询语句

timeFlag　::= ['nonsequenced'] 'validtime' [identifier | interval]
coal　::= '(' 'period' ')'
query　::= [timeFlag] queryExp
queryExp　::= queryTerm { ('union' | 'except') queryTerm }
queryTerm　::= queryFactor { 'intersect' queryFactor }
queryFactor　::= '(' query ')' [coal] | sfw
sfw　::= 'select' selectItemList
'from' tableRefList
['where' condExp]
['group' 'by' groupByList]
['having' condExp]
selectItemList　::= '*' | selectItem { ',' selectItem }
selectItem　::= scalarExp [alias]
tableRefList　::= tableRef { ',' tableRef }
tableRef　::= '(' query ')' [coal] alias ['(' colList ')'] |
identifier [coal] [alias]
alias　::= ['as'] identifier
condExp　::= condTerm { 'or' condTerm }
condTerm　::= condFactor { 'and' condFactor }
condFactor　::= ['not'] simpleCondFactor
simpleCondFactor::= '(' condExp ')'　|
'exists' '(' query ')'
|
constScalarExp commonOp constScalarExp　|
constScalarExp commonOp ('all' | 'any' | 'some') '(' query ')'
|
constScalarExp ['not'] 'between' constScalarExp 'and' constScalarExp　|
scalarExp ['not'] 'in' '(' query ')'　|
tempScalarExp timeOp tempScalarExp　|
tempScalarExp timeOp ('all' | 'any' | 'some') '(' query ')'
|
eventTerm ['not'] 'between' eventTerm 'and'　eventTerm
|

condOp ::= commonOp | timeOp
commonOp ::= '<' | '>' | '<=' | '>=' | '<>' | '='
timeOp ::= 'before' | 'contains' | 'overlaps' | 'meets' | ‘starts’ | ‘finishes’ | 'equals'

groupByList ::= colRef { ',' colRef }
scalarExp ::= constScalarExp | tempScalarExp
constScalarExp ::= term { ('+' | '-') term }
term ::= factor { ('*' | '/') factor }
factor ::= [('+' | '-')] simpleFactor
simpleFactor ::= colRef | const | '(' constScalarExp ')' | 'abs' '(' constScalarExp ')'
colRef ::= identifier ['.' identifier]
const ::= integer | float | ''' string '''

tempScalarExp ::= interval | eventTerm | span { ('+' | '-') span } |
colRef '-' event |
event '-' event

eventTerm ::= event { ('+' | '-') span } |
colRef {('+' | '-') span}

interval ::= 'validtime' '(' identifier ')' |
'period' intervalExp |
'period' '(' eventTerm ',' eventTerm ')'
intervalExp ::= '[' time '-' time ')'
time ::= timeDBDate | eventExp
event ::= ('begin' | 'end') '(' interval ')' |
('first' | 'last') '(' eventTerm ',' eventTerm ')' |
eventExp
eventExp ::= 'now' |
'beginning' |
'forever' |
'date' dateString |
'date' timeDBDate |
'timestamp' timestampString
dateString ::= ''' YYYY '-' MM '-' DD '''
timestampString ::= ''' YYYY '-' MM '-' DD ' ' HH ':' MM ':' SS '''
timeDBDate ::= ''' YYYY ['/' MM ['/' DD ['~' HH [':' MM [':' SS]]]]] '''
span ::= 'interval' spanExp
spanExp ::= integer qualifier { integer qualifier }
qualifier ::= 'year' | 'month' | 'day' | 'hour' | 'minute' | 'second'

2．数据定义语句

ddl ::= ddlTable | ddlView | dropTable | dropView

```
ddlTable        ::= 'create' 'table' identifier ( tableDef | ddlQuery )
ddlView         ::= 'create' 'view' identifier ddlQuery
tableDef        ::= '(' colDefList ')' [ 'as' 'validtime' ]
ddlQuery        ::= [ '(' colList ')' ] 'as' query
colDefList      ::= colDef { ',' (colDef | tableConstraint) }
colDef          ::= identifier dataType [ columnConstraint ]
columnConstraint::= primKeyCol | refIntegrity | checkConstraint
tableConstraint      ::= ['constraint' identifier] (primKeyTab | foreignKey | checkConstraint)
primKeyCol      ::= 'primary' 'key'
primKeyTab      ::= 'primary' 'key' '(' colList ')'
refIntegrity         ::= 'references' identifier '(' identifier ')'
foreignKey           ::= 'foreign' 'key' '(' colList ')' 'references' identifier '(' colList ')'
checkConstraint      ::= 'check' '(' condExp ')'
colList         ::= col { ',' col }
col             ::= identifier
dataType        ::= 'integer'                       |
                    'float'                         |
                    'char' typeLength               |
                    'varchar' typeLength            |
                    'interval'                      |
                    'datetime'
typeLength           ::= '(' integer ')'
dropTable            ::= 'drop' 'table' identifier
dropView        ::= 'drop' 'view' identifier
```

3．数据表操作语句

```
dml             ::= insert | delete
insert               ::=insertByValues | insertByQuery
insertByValues      ::= ['validtime' 'period' intervalExp] 'insert' 'into' identifier 'values' '(' valList ')'
insertByQuery       ::=[ timeFlag] 'insert' 'into' identifier queryExp
delete          ::= [ 'validtime' 'period' intervalExp ]    'delete' 'from' identifier [ 'where' condExp ]
valList         ::= val { ',' val }
val             ::= integer | float | ''' string ''' | intervalExp | dateString | timestampString |
timeDBDate
```

4．控制语句

```
control         ::= 'commit' | 'rollback'
```