

# **华易数据库管理系统 Huayisoft DB Server**

## **技术白皮书**

**九江易龙技术有限公司**

**V6.2.21.03.15**



## 目录

<b>1. HUAYISOFT DB SERVER 概述 .....</b>	<b>7</b>
1.1 HUAYISOFT DB SERVER 是什么.....	8
1.2 HUAYISOFT DB SERVER 的特点.....	9
1.3 HUAYISOFT DB SERVER 语法结构中的符号说明 .....	10
<b>2. HUAYISOFT DB SERVER 的数据类型 .....</b>	<b>11</b>
2.1 基本数据类型 .....	11
2.1.1 CHARACTER 字符数据类型.....	11
2.1.2 二进制数据类型.....	13
2.1.3 位数据类型.....	14
2.1.4 数值数据类型.....	14
2.1.5 布尔数据类型.....	17
2.1.6 日期时间数据类型 .....	18
2.1.7 INTERVAL 时间间隔数据类型.....	19
2.2 数组数据类型 .....	24
2.3 记录数据类型 .....	25
2.4 表数据类型 .....	26
2.5 类数据类型 .....	26
2.6 类型数据类型 .....	27
多项式选择数据类型.....	28
2.6.1 多项式单选数据类型.....	28
2.6.2 多项式多选数据类型.....	28
2.7 方法调用数据类型.....	28
2.8 对象数据类型 .....	29
2.9 抽象数据类型 .....	29
2.10 实用数据类型 .....	29
2.10.1 PCODE 中华人民共和国身份证数据类型.....	30
2.10.2 IP 地址数据类型.....	30
2.10.3 VERSION 版本号数据类型.....	31
2.10.4 MONEY 货币数据类型.....	31
<b>3. HUAYISOFT DB SERVER 的值及其表达式 .....</b>	<b>32</b>



3.1 NULL 值.....	32
3.2 常量.....	32
3.2.1 数字常量.....	33
3.2.2 字符串常量,即包含在一对单引号内的文本:.....	33
3.2.3 二进制常量,即以 X 打头且包含在一对单引号内的文本:.....	34
3.2.4 位常量,即以 B 打头且包含在一对单引号内的文本:.....	34
3.3 表达式.....	34
3.3.1 CAST 表达式.....	34
3.3.2 CASE 表达式.....	34
3.3.3 COALESCE 表达式.....	35
3.3.4 NULLIF 表达式.....	35
3.3.5 静态方法.....	35
3.3.6 new 类对象.....	36
3.3.7 数值表达式.....	36
3.3.8 字符值表达式.....	36
3.3.9 记录构造表达式.....	36
3.3.10 数组构造表达式.....	37
3.3.11 布尔值表达式.....	37
<b>4. HUAYISOFT DB SERVER 数据类型与类对象.....</b>	<b>39</b>
4.1 HUAYISOFT DB SERVER 数据类型与类对象对应表.....	39
4.2 接口 INTERFACEY 与 类对象 CLASS .....	40
<b>5. HUAYISOFT DB SERVER 数据库的查询概述.....</b>	<b>41</b>
5.1 HUAYISOFT DB SERVER 查询的总体说明.....	41
5.2 单表查询.....	43
5.2.1 SELECT 子句.....	43
5.2.2 记录选择 ( WHERE 子句 ) .....	45
5.2.3 对查询结果显示顺序的控制 ( ORDER BY 子句 ) .....	48
5.3 多表查询 ( 连接查询 ) .....	49
5.3.1 等连接查询.....	49
5.3.2 多个匹配字段.....	49
5.3.3 内连接.....	50
5.3.4 外连接.....	50



5.4 统计查询.....	52
5.4.1 集函数.....	52
5.4.2 分组查询 ( GROUP BY 子句 ) .....	54
5.4.3 分组搜索条件 ( HAVING 子句 ) .....	54
5.5 合并查询结果 ( UNION 子查询 ) .....	56
5.6 显示指定范围的记录 ( LIMIT 子句 ) .....	56
<b>6. HUAYISOFT DB SERVER 的数据定义.....</b>	<b>57</b>
6.1 数据库定义.....	57
6.1.1 创建数据库.....	57
6.1.2 删除数据库.....	57
6.1.3 设置当前数据库.....	58
6.2 用户定义.....	58
6.2.1 创建用户.....	58
6.2.2 修改用户.....	58
6.2.3 删除用户.....	59
6.3 模式定义.....	59
6.3.1 创建模式.....	59
6.3.2 删除模式.....	60
6.3.3 设置当前模式.....	60
6.4 表定义.....	60
6.4.1 创建表.....	61
6.4.2 修改表.....	62
6.4.3 删除表.....	62
6.5 抽象数据类型定义.....	62
6.5.1 创建抽象数据类型.....	63
6.5.2 删除指定的自定义抽象数据类型.....	63
6.6 数据域定义.....	63
6.6.1 创建数据域.....	63
6.6.2 删除数据域.....	63
6.7 断言定义.....	63
6.7.1 创建断言.....	64
6.7.2 删除断言.....	64



6.8 索引定义.....	64
6.8.1 创建索引.....	64
6.8.2 删除索引.....	64
6.9 视图定义.....	65
6.9.1 创建视图.....	65
6.9.2 删除指定的视图.....	65
6.10 记录文本定义.....	65
6.10.1 创建记录文本(数据库的导出).....	65
<b>7. HUAYISOFT DB SERVER 数据库记录的插入、修改与删除 .....</b>	<b>67</b>
7.1 插入数据到表中 ( INSERT).....	67
7.1.1 插入单条记录.....	67
7.1.2 插入一个查询的结果.....	68
7.1.3 在指定的行位置插入一记录.....	68
7.1.4 数据库的导入.....	68
7.2 修改表中的数据(UPDATE).....	69
7.2.1 修改满足条件的记录.....	69
7.2.2 修改表中的所有记录.....	69
7.2.3 对指定的某一记录进行修改.....	69
7.3 从表中删除数据 ( DELETE ) .....	69
7.3.1 删除满足条件的所有记录。.....	70
7.3.2 删除表中的所有记录。.....	70
7.3.3 删除指定的某一记录.....	70
<b>8. HUAYISOFT DB SERVER 的安全策略.....</b>	<b>71</b>
8.1 数据控制.....	71
8.1.1 授权语句.....	71
8.1.2 收回权限语句.....	71
<b>9. HUAYISOFT DB SERVER 中的 SET、GET 语句.....</b>	<b>72</b>
9.1 SET 语句.....	72
9.2 GET 语句.....	72
9.2.1 获取用户的缺省菜单.....	72
9.2.2 获取用户的 FTP 信息.....	72



---

9.2.3 获取用户的 HTTP 信息.....	73
9.2.4 获取用户当前目录下的文件信息.....	73
<b>10. HUAYISOFT DB SERVER 的 LOCK、UNLOCK 语句说明.....</b>	<b>74</b>
10.1 人工加锁(直到人工减锁或用户退出).....	74
10.2 人工减锁.....	74
<b>11. HUAYISOFT DB SERVER 中的 SERVER、SERVICE 语句.....</b>	<b>75</b>
<b>12. HUAYISOFT DB SERVER 中的 NEW 语句 .....</b>	<b>78</b>
12.1 NEW 语句.....	78
<b>13. HUAYISOFT DB SERVER 中的 FTP 语句.....</b>	<b>78</b>
13.1 FTP 语句 .....	78
<b>14. HUAYISOFT DB SERVER 中的 EXECUTE 语句.....</b>	<b>79</b>
14.1 执行登录获取用户的应用程序.....	79
14.2 执行批语句 .....	79
14.3 执行服务器端的应用程序的方法 .....	79
<b>15. HUAYISOFT DB SERVER 中的 HELP 语句 .....</b>	<b>80</b>
15.1 HELP 语句.....	80
<b>16. HUAYISOFT DB SERVER 的编程 .....</b>	<b>80</b>
16.1 JDBC 应用编程接口.....	80
16.2 三个超级用户 .....	86
16.3 创建第一个数据库及其管理员用户 .....	86
16.4 HUAYISOFT DB SERVER 的运行环境.....	87
<b>17. HUAYISOFT DB SERVER 的事务处理说明.....</b>	<b>88</b>
<b>附录 A. 样本数据库 .....</b>	<b>89</b>
<b>附录 B. HUAYISOFT DB SERVER 信息模式.....</b>	<b>95</b>



# 1. Huayisoft DB Server 概述

1. 更详细的语法说明请参考：[db\\_bnfs.htm](#)
2. 编程接口类包：[db\\_jdbc.jar](#)
3. 编程接口所涉及的类及相关常用类请参考：[documents/index.html](#)
4. `StartLogin.exe` 为超级用户或管理员用户提供了图形界面的使用操作
5. `server/db.obj` 和 `server.jar` 为华易数据库服务端的安装程序。



## 1.1 Huayisoft DB Server 是什么

Huayisoft DB Server 是一款新型对象化关系型数据库管理系统！

中文名称：华易数据库管理系统（或称数据库组件）！

英文名称：Huayisoft DB Server(原名 HYSQL)

编写语言：Java 语言编写

开发公司：九江易龙技术有限公司（原九江华易软件有限公司 2001-2012、北京华易方圆软件有限公司 2011—2015）

开发时间：1997 年 12 月开始研发

开发宗旨：简化应用开发、提高开发效率

版本来源：完全原始取得（完全独立自主开发）

产品类型：面向对象的关系型数据库

编程接口：JDBC

服务对象：中小型网络应用(B/S 结构)

首先，它是一款关系型数据库，因为.dbf 文件中存放的都是关系型记录，也支持 SQL 标准的核心内容(绝大多数的 SQL 标准语句，几乎完全支持 sql92、部分支持 sql99 标准)；

也是一款对象化数据库，其字段类型、列、记录、表以及其查询或计算结果都是一个 Java 类对象，在对象出现的地方可以随意地调用其方法进行复杂的计算；

也是一款文件数据库，它通过 FTP 命令及 EXECUTE FILE 命令，可以进行多种形式的文件存取，其相当于一款 FTP 文件服务器，但比其更高效，更节约资源；

也是一款微服务构架数据库，( SERVER 及 SERVICE 命令)，微服务功能的实现，可以有效地解决复杂业务应用，提高其开发效率；

也是一款‘多数据库’数据库，在同一台电脑上可以创建任意多个数据库(CATALOG)，每个 CATALOG 拥有独立的用户系统及其权限系统，在一个 CATALOG 上可以部署一个应用系统，也可以部署多个应用系统；一个复杂的应用系统可以部署在多个 CATALOG 上，多个 CATALOG 可以分布在不同的服务器上；从而方便部署分布式应用，也非常适合部署‘非中心化’的应用！

注：.dbf 文件与第三方文件后缀可能相同，在其内容上没有任何相似或关联，只是取 DataBaseFile 之意！





## 1.2 Huayisoft DB Server 的特点

Huayisoft DB Server (华易数据库管理系统)全部采用纯 Java 语言开发,并有机地与 Java 结合在一起,充分利用 Java 先进的面向对象及错误捕捉等技术,并提供了许多有益的功能扩展,为开发安全稳定、高质量高效率的应用提供了坚实的基础:

### 一、数据类型对象化

1. Huayisoft DB Server 的所有数据类型都对应一个 Java 类  
如: 'abc' 等价于 new String('abc')
2. 任何能调用 Huayisoft DB Server 字段数据的地方都可直接调用其对应的方法  
如: 'abc'.toString ()
3. 所有 Huayisoft DB Server 数据对象及其方法执行结果又可作为中间参数组成新的数据  
如: 'abc'.toString ().toString ().toString ()

new Integer ( '123456'.substring(1).substring(2).length() ) ( 结果 3 )

这种自由的组合及直接调用 java 原有丰富的类(方法)可以极大地提高了数据库的整体功能,比如:CHAR 数据类型对应 java.lang.String 类,6.0 版的 String 就 87 个方法(其中 15 个同构方法,72 个一般方法),与传统数据库依赖大量函数及过程来提高整体功能的做法相比,使用面向对象技术不仅简化数据库系统本身,也简化了学习和使用数据库成本,也有利于开发高质量的应用程序。

二、可支持任何复杂的对象数据类型,并极为简单化了用户自定义数据类型的定义过程,不需要象 SQL99 那样繁琐的定义过程。

三、可直接调用其它的 java 类(方法),如可以直接调用 java.lang.Math 中的所有方法。

四、采用对象方法来代替非对象化的存储过程。

五、采用 Java API(JDBC)应用编程接口方式编程。并提供简化了的编程接口:本数据库封装了装载驱动程序、建立联接、创立语句执行等操作,仅提供一个对象化接口,利用本对象的方法,直接执行语句并获得结果对象。在客户机的浏览器中也可直接获得此编程接口。其结果是简化了客户端的应用模块。不支持非对象化的既不是主机语言,也不是数据库语言,即所谓嵌入式 SQL 编程。

六、数据库与应用高度有机结合(独特的 EXECUTE CLASS 调用):一般的数据库与应用程序的关系是应用程序调用数据库,而 Huayisoft DB Server 的 EXECUTE 调用则恰恰相反。这样做的最大好处是可简化服务器端的应用模块,甚至不需要服务器端的应用模块。



在这里,应用模块(Java 方法)就好像是数据库功能的扩展。对应于一种高效的应用模式就是:客户端的应用模块将直接连接到服务器端的数据库管理系统而不是服务器端的应用模块,也不是所谓的应用服务器。

七、编程时,可传递的参数或返回的结果不仅仅是 Huayisoft DB Server 中定义的数据类型对象,也可以是可序列化或符合 Huayisoft DB Server 规范的任何数据对象。

八、当有错误发生时, Huayisoft DB Server 将直接返回错误信息,其中可能包括所运行的类信息。这里没有错误代号(代码)的概念。

九、集 B/S、C/S 结构之优点:客户端的应用模块既可以在浏览器中运行(APPLET),也可以通过一个所谓的引子程序调用服务器端的代码及数据(B/S 的优点)而不在浏览器中运行(APPLICATION);也提供了浏览器版 Huayisoft DB Server 命令解释器,可在浏览器中直接执行各 Huayisoft DB Server 语句。

十、支持**微服务构架**(SERVER 及 SERVICE 命令),与 EXECUTE CLASS 语句有一定的相似性,但部署与调用都更为简单。现在的许多应用有个特点,那就是复杂化与多样性,解决的思想还是模块化思想,其技术方法之一就是微服务构架。

十一、另外,许多应用也需要组织与存取其它类型的文件,比如:菜单等资源文件,提供 FTP 命令可以简化服务器部署及应用开发。

十二、‘多数据库’、多个独立的用户系统及其权限系统,就是为了解决复杂化应用模块化,大型文件碎片化,非中心化的分布式应用提供支持。

总之, Huayisoft DB Server 数据库管理系统通过采用面向对象以及可调用其它的 java 类(方法)等技术措施,打造了一款功能灵活强大、使用又非常简单的对象关系数据库。

### 1.3 Huayisoft DB Server 语法结构中的符号说明

<> 尖括号内的内容是语法定义名称

::= 语法定义符,左边是语法定义名称,右边是其定义

[] 方括号内的内容是可选的(也可不选的),但 ARRAY 后面的方括号则是数组单元的分隔符。

{ } 大括号内的内容是必须选择其中之一

| 多项选择符号,选择其中之一

... 表示其之前的单元是可以重复多个的



## 2. Huayisoft DB Server 的数据类型

数据类型作为数据库最基本的性能, 是简化应用程序开发、提高开发效率最基本的功能部件! Huayisoft DB Server 不仅提供了丰富的(抽象-结构化)数据类型, 同时结合对象技术使数据类型的性能得到极大的扩展!

语法:

```
<数据类型> ::= <基本数据类型>
                | <数组数据类型>
                | <记录数据类型>
                | <表数据类型>
                | <类数据类型>
                | <类型数据类型>
                | <方法调用数据类型>
                | <对象数据类型>
                | <抽象数据类型>
                | <实用数据类型>
```

### 2.1 基本数据类型

语法:

```
<基本数据类型> ::= <字符数据类型>
                    | <数值数据类型>
                    | <二进制数据类型>
                    | <位数据类型>
                    | <布尔数据类型>
                    | <日期时间数据类型>
                    | <时间间隔数据类型>
```

#### 2.1.1 CHARACTER 字符数据类型

语法:

```
<字符数据类型> ::= <定长字符数据类型>
```



- | <变长字符数据类型>
- | <字符大型对象数据类型>
- | <双字节定长字符数据类型>
- | <双字节变长字符数据类型>
- | <双字节字符大型对象数据类型>

### (1) 定长字符数据类型

语法：

```
<定长字符数据类型> ::= CHAR[(长度)]  
| CHARACTER[(长度)]
```

功能：定义固定长度的字符串。最大长度为 16K，缺省时为 1。

例如：CHAR(99)。

### (2) 变长字符数据类型

语法：

```
<变长字符数据类型> ::= VARCHAR[(长度)]  
| CHARACTER VARYING [(长度)]  
| CHAR VARYING [(长度)]
```

功能：定义变长字符串，最大长度为 16K，缺省时为 1。

例如：VARCHAR(99)。

### (3) 字符大型对象数据类型

语法：

```
<字符大型对象数据类型> ::= CLOB [(长度)]  
| CHAR LARGE OBJECT [(长度)]  
| CHARACTER LARGE OBJECT [(长度)]
```

### (4) 双字节定长字符数据类型

语法：



```
<双字节定长字符数据类型> ::= UNICODE[(长度)]
| NATIONAL CHARACTER [(长度)]
| NATIONAL CHAR[(长度)]
| NCHAR[(长度)]
```

### (5) 双字节变长字符数据类型

语法：

```
<双字节变长字符数据类型> ::= NATIONAL CHARACTER VARYING[(长度)]
| NATIONAL CHAR VARYING [(长度)]
| NCHAR VARYING[(长度)]
```

### (6) 双字节字符大型对象数据类型

语法：

```
<双字节字符大型对象数据类型> ::= NCLOB [(长度)]
| ULOB [(长度)]
| NCHAR LARGE OBJECT [(长度)]
| NATIONAL CHARACTER LARGE OBJECT [(长度)]
```

## 2.1.2 二进制数据类型

语法：

```
<二进制数据类型> ::= <二进制定长数据类型>
| <二进制大型对象数据类型>
```

### (1) 二进制定长数据类型

语法：

```
<二进制定长数据类型> ::= BINARY[(长度)]
| BYTE[(长度)]
```



功能：定义二进制数据类型，长度指字节长度，默认长度为 1。

例如：BINARY(3)，定义了 3 个字节的二进制数据类型。

X'09ADFC99'

X'09 AD FC 99' 是合法的。

## (2) 二进制大型对象数据类型

语法：

```
<二进制大型对象数据类型> ::= BLOB [(长度)]  
| BINARY LARGE OBJECT [(长度)]
```

### 2.1.3 位数据类型

语法：

```
<位数据类型> ::= BIT[(长度)]  
| BIT VARYING [(长度)]
```

功能：定义位数据类型，长度指位长度，默认长度为 1。

例如：BIT(3)，定义了 3 个位长度的位数据类型。

B'101' 是合法的。

### 2.1.4 数值数据类型

语法：

```
<数值数据类型> ::= <精确数值数据类型>  
| <近似数值数据类型>
```

#### 一、精确数值数据类型

语法：

```
<精确数值数据类型> ::= <整数数据类型>  
| <带小数位的精确数值数据类型>
```

```
<整数数据类型> ::= <微整数数据类型>
```



| <小整数数据类型>  
| <整数数据类型>  
| <大整数数据类型>

<带小数位的精确数值数据类型> ::= <NUMERIC 数据类>  
| <DECIMAL 数据类型>

### (1) 微整数数据类型

TINYINT 类型

语法：

<微整数数据类型> ::= TINYINT

功能：定义 1 字节整数数据类型，取值范围为：-128 ~ +127。

### (2) 小整数数据类型

SMALLINT 类型

语法：

<小整数数据类型> ::= SMALLINT

功能：定义 2 字节整数数据类型，取值范围为：-32768 ~ +32767。

### (3) 整数数据类型

INT 类型

语法：

<整数数据类型> ::= INT | INTEGER

功能：定义 4 字节整数数据类型，取值范围为：-2147483648 ~ +2147483647。

### (4) 大整数数据类型

BIGINT 数据类型

语法：

<大整数数据类型> ::= BIGINT | LONG



功能：定义 8 字节整数数据类型，取值范围为：-9223372036854775808 ~ 9223372036854775807。

### (5) 带小数位的精确数值数据类型

#### NUMERIC 数据类型

语法：

```
<NUMERIC 数据类型> ::= NUMERIC [(精度 [,标度 ])]
```

功能：定义带小数位的精确数值类型，精度为数值的总数字位数，取值范围是 1 至 19，缺省是 19。标度为小数点右边的数字位数，取值范围是 0 至精度，标度缺省是 0。

当精度为 1 至 2，标度为 0 时，则与 TINYINT 类型相同；

当精度为 3 至 4，标度为 0 时，则与 SMALLINT 类型相同；

当精度为 5 至 9，标度为 0 时，则与 INTEGER 类型相同；

当精度为 10 至 19，标度为 0 时，则与 BIGINT 类型相同；

例如：NUMERIC(9,3)定义了小数点左边 6 位和小数点右边 3 位，共 9 位的数字，范围在-999999.999 到 999999.999。

#### DECIMAL 数据类型

语法：

```
<DECIMAL 数据类型> ::= DECIMAL[(精度[,标度])]
| DEC[(精度[,标度])]
```

功能：与 NUMERIC 相似。

## 二、 近似数值数据类型

语法：

```
<近似数值数据类型> ::= <浮点数数据类型>
| <单精度浮点数数据类型>
| <双精度浮点数数据类型>
```





### (1) 单精度浮点数数据类型

REAL 类型

语法：

```
<浮点数数据类型> ::= REAL
```

功能：定义 4 字节浮点数数据类型，取值范围 $-3.4E + 38 \sim 3.4E + 38$ 。

### (2) 双精度浮点数数据类型

语法：

```
<双精度浮点数数据类型> ::= DOUBLE PRECISION
```

功能：定义 8 字节浮点数数据类型，取值范围 $-1.7E + 308 \sim 1.7E + 308$ 。

### (1) 单精度浮点数数据类型

语法：

```
<浮点数数据类型> ::= FLOAT[(精度)]
```

功能：当精度为 1 至 7，则与 REAL 类型相同；  
否则与 DOUBLE PRECISION 类型相同；

## 2.1.5 布尔数据类型

语法：

```
<布尔数据类型> ::= { BOOLEAN | BOOL } [ <真值> , <假值> [ , <未知值> ] ]
```

```
<真值> ::= 字符串
```

```
<假值> ::= 字符串
```

```
<未知值> ::= 字符串
```

功能：定义布尔数据类型，默认值是 TURE、FALSE 或 UNKNOWN，在数据库中用 1、0 或 2 进行存储，1 表示 TRUE，0 表示 FALSE，2 或其它值为 UNKNOWN。

例如：



```
CREATE TABLE BOOL_TEST
(
    姓名 CHAR(8),
    性别 BOOL ('男', '女', '不知道'),
    结婚否 BOOL
);

INSERT INTO BOOL_TEST VALUES ('李某', '男', TRUE);
INSERT INTO BOOL_TEST VALUES ('王某', '不知道', UNKNOWN);

SELECT * FROM BOOL_TEST;
```

姓名	性别	结婚否
李某	男	TRUE
王某	不知道	UNKNOWN

## 2.1.6 日期时间数据类型

语法：

```
<日期时间数据类型> ::= <日期数据类型>
                        | <时间数据类型>
                        | <时间戳数据类型>
```

### (1) DATE 日期数据类型

DATE 类型

语法：

```
<日期数据类型> ::= DATE
```

功能：定义日期数据类型，DATE 类型包括年、月、日信息。

年，取值范围：1~32767

月，取值范围：1~12

日，取值范围：1~31

例如：



'2010 年 02 月 25 日'

'2001.01.08'

'2001/01/08'

'2001-01-08'

## ( 2 ) TIME 时间数据类型

TIME 类型

语法：

```
<时间数据类型> ::= TIME [ ( 小数秒精度 ) ]
```

功能：定义时间数据类型，TIME 类型包括时、分、秒信息，定义了一个在'00:00:00'和'23:59:59'之间的有效时间。例如：

'08 时 08 分 08 秒'

'08:08:08'

## ( 3 ) TIMESTAMP 时间戳数据类型

TIMESTAMP 类型

语法：

```
<时间戳数据类型> ::= TIMESTAMP [ ( 小数秒精度 ) ]  
| DATETIME
```

功能：定义日期时间数据类型，TIMESTAMP 类型包括年、月、日、时、分、秒信息，定义了一个在'0001-01-01 00:00:00'和'9999-12-31 23:59:59'之间的有效格里高利日期时间。

## 2.1.7 INTERVAL 时间间隔数据类型

语法：

```
<时间间隔数据类型> ::= <年间隔数据类型>  
| <月间隔数据类型>  
| <年月间隔数据类型>  
| <日间隔数据类型>
```



| <日小时间隔数据类型>  
| <日分钟间隔数据类型>  
| <日秒间隔数据类型>  
| <小时间隔数据类型>  
| <小时分钟间隔数据类型>  
| <小时秒间隔数据类型>  
| <分钟间隔数据类型>  
| <分钟秒间隔数据类型>  
| <秒间隔数据类型>

### ( 1 ) INTERVAL YEAR 年间隔数据类型

语法：

<年间隔数据类型> ::= INTERVAL YEAR [ ( 引导精度 ) ]

功能：定义年间隔数据类型，引导精度规定年的取值范围，其默认值是 2。

例如：INTERVAL YEAR(3)，则取值范围为-999 ~ 999 年。

INTERVAL '99' YEAR 是合法的。

### ( 2 ) INTERVAL MONTH 月间隔数据类型

语法：

<月间隔数据类型> ::= INTERVAL MONTH [ ( 引导精度 ) ]

功能：定义月间隔数据类型，引导精度规定月的取值范围，其默认值是 2。

例如：INTERVAL MONTH(3)，则取值范围为-999 ~ 999 月。

INTERVAL '99' MONTH 是合法的。

### ( 3 ) INTERVAL YEAR TO MONTH 年月间隔数据类型

语法：

<年月间隔数据类型> ::= INTERVAL YEAR [ ( 引导精度 ) ] TO MONTH

功能：定义年月间隔数据类型，引导精度规定年的取值范围，其默认值是 2，月的取值范围为 0 ~ 11。



例如：INTERVAL YEAR(3) TO MONTH，则取值范围为-999 年零 12 月到 999 年零 12 月。INTERVAL '99-09' YEAR TO MONTH 是合法的。

#### ( 4 ) INTERVAL DAY 日间隔数据类型

语法：

```
<日间隔数据类型> ::= INTERVAL DAY [ ( 引导精度 ) ]
```

功能：定义日间隔数据类型，引导精度规定日的取值范围，其默认值是 2。

例如：INTERVAL DAY(3)，则取值范围为-999 ~ 999 日。

INTERVAL '99' DAY 是合法的。

#### ( 5 ) INTERVAL DAY TO HOUR 日小时间隔数据类型

语法：

```
<日小时间隔数据类型> ::= INTERVAL DAY [ ( 引导精度 ) ] TO HOUR
```

功能：定义日小时时间间隔数据类型，引导精度规定日的取值范围，其默认值是 2。小时的取值范围为 0 ~ 23。

例如：INTERVAL DAY(3) TO HOUR，则取值范围为负 999 日零 23 小时到正 999 日零 23 小时。INTERVAL '99 09' DAY TO HOUR 是合法的。

#### ( 6 ) INTERVAL DAY TO MINUTE 日分钟间隔数据类型

语法：

```
<日分钟间隔数据类型> ::= INTERVAL DAY [ ( 引导精度 ) ] TO MINUTE
```

功能：定义日分间隔数据类型，引导精度规定日的取值范围，其默认值是 2。小时的取值范围为 0 ~ 23，分钟的取值范围为 0 ~ 59。

例如：INTERVAL DAY(3) TO MINUTE，则取值范围为负 999 日零 23 小时零 59 分到正 999 日零 23 小时零 59 分。

INTERVAL '099 09:09' DAY TO MINUTE 是合法的。

#### ( 7 ) INTERVAL DAY TO SECOND 日秒间隔数据类型

语法：



<日秒间隔数据类型> ::= INTERVAL DAY [ ( 引导精度 ) ] TO SECOND [ ( 小数秒精度 ) ]

功能：定义日秒间隔数据类型，引导精度规定日的取值范围，其默认值是 2。小时的取值范围为 0 ~ 23，分钟的取值范围为 0 ~ 59，秒整数部分的取值范围为 0 ~ 59。小数秒精度规定秒字段中小数点后面的位数，其取值范围为 0 ~ 6，默认值是 6。

例如：INTERVAL DAY(3) TO SECOND(1)，则取值范围为负 999 日零 23 小时零 59 分零 59.9 秒到正 999 日零 23 小时零 59 分零 59.9 秒。

INTERVAL '099 09:09:09.9' DAY TO SECOND 是合法的。

### ( 8 ) INTERVAL HOUR 小时间隔数据类型

语法：

<小时间隔数据类型> ::= INTERVAL HOUR [ ( 引导精度 ) ]

功能：定义小时间隔数据类型，引导精度规定小时的取值范围，其默认值是 2。

例如：INTERVAL HOUR(3)，则取值范围为负 999 小时到正 999 小时。

INTERVAL '099' HOUR 是合法的。

### ( 9 ) INTERVAL HOUR TO MINUTE 小时分钟间隔数据类型

语法：

<小时分钟间隔数据类型> ::= INTERVAL HOUR [ ( 引导精度 ) ] TO MINUTE

功能：定义小时分钟间隔数据类型，引导精度规定小时的取值范围，其默认值是 2，分钟的取值范围为 0 ~ 59。

例如：INTERVAL HOUR(3) TO MINUTE，则取值范围为负 999 小时零 59 分到正 999 小时零 59 分。

INTERVAL '99:09' HOUR TO MINUTE 是合法的。

### ( 10 ) INTERVAL HOUR TO SECOND 小时秒间隔数据类型

语法：

<小时秒间隔数据类型> ::= INTERVAL HOUR [ ( 引导精度 ) ] TO SECOND [ ( 小数秒精度 ) ]



功能：定义小时秒间隔数据类型，引导精度规定小时的取值范围，其默认值是 2。分钟的取值范围为 0 ~ 59 之间，秒整数部分的取值范围为 0 ~ 59。小数秒精度规定秒字段中小数点后面的位数，其取值范围为 0 ~ 6，默认值是 6。

例如：INTERVAL HOUR(3) TO SECOND(1)，则取值范围为负 999 小时零 59 分零 59.9 秒到正 999 小时零 59 分零 59.9 秒。

INTERVAL '99:09:09.9' HOUR TO SECOND 是合法的。

### ( 11 ) INTERVAL MINUTE 分钟间隔数据类型

语法：

```
<分钟间隔数据类型> ::= INTERVAL MINUTE [ ( 引导精度 ) ]
```

功能：定义分钟间隔数据类型，引导精度规定分钟的取值范围，其默认值是 2。

例如：INTERVAL MINUTE(3)，则取值范围为负 999 分钟到正 999 分钟。

INTERVAL '99' MINUTE 是合法的。

### ( 12 ) INTERVAL MINUTE TO SECOND 分钟秒间隔数据类型

语法：

```
<分钟秒间隔数据类型> ::= INTERVAL MINUTE [ ( 引导精度 ) ]  
TO SECOND [ ( 小数秒精度 ) ]
```

功能：定义分钟秒间隔数据类型，引导精度规定分钟的取值范围，其默认值是 2，秒整数部分的取值范围为 0 ~ 59，小数秒精度规定秒字段中小数点后面的位数，其取值范围为 0 ~ 6，默认值是 6。

例如：INTERVAL MINUTE(3) TO SECOND(1)，则取值范围为负 999 分零 59.9 秒到正 999 分零 59.9 秒。

INTERVAL '99:09.9' MINUTE TO SECOND 是合法的。

### ( 13 ) INTERVAL SECOND 秒间隔数据类型

语法：

```
<秒间隔数据类型> ::= INTERVAL SECOND [ ( 引导精度 [ , 小数秒精度 ] ) ]
```



功能：定义秒间隔数据类型，引导精度规定秒的取值范围，其默认值是 2，小数秒精度规定秒字段中小数点后面的位数，其取值范围为 0~6，默认值是 6。

例如：INTERVAL SECOND(3,1)，则取值范围为负 999.9 秒到正 999.9 秒。

INTERVAL '99.9' SECOND 是合法的。

## 2.2 数组数据类型

### ARRAY 数据类型

语法：

```
<数组数据类型> ::= <数据类型> ARRAY [ 长度 ]
```

注：此处的方括号为语法元素

功能：定义多个同一数据类型的数据。

例如：

INT ARRAY[3] 定义了包含 3 个整型的数组数据类型。 ARRAY[123,123,123] 是合法的。

```
CREATE TABLE ARRAY_TEST
```

```
(  
    编号 INT,  
    坐标点 INT ARRAY [2]  
);
```

```
INSERT INTO ARRAY_TEST VALUES (1,ARRAY[8,9]);
```

```
INSERT INTO ARRAY_TEST VALUES (2,ARRAY[10,11]);
```

```
SELECT 编号, 坐标点 FROM ARRAY_TEST;
```

编号	坐标点
1	[8,9]
2	[10,11]





```
UPDATE ARRAY_TEST SET 坐标点[1]=118 WHERE 坐标点[1]=10;
```

```
SELECT 坐标点[1] FROM ARRAY_TEST;
```

```
坐标点[1]
```

```
8
```

```
118
```

## 2.3 记录数据类型

ROW 记录（结构化）数据类型

语法：

```
<记录数据类型> ::= ROW ( 列名 <数据类型> [ { 列名 <数据类型> } ... ] )
```

功能：定义多个不同数据类型的数据。

例如：ROW (username CHAR(12), password CHAR(8)) 。

ROW('huayisoft', 'www.com') 是合法的。

```
CREATE TABLE ROW_TEST  
(  
    编号 INT,  
    坐标点 ROW(x INT, y INT)  
);
```

```
INSERT INTO ROW_TEST VALUES (1, ROW(8,9));  
INSERT INTO ROW_TEST VALUES (2, ROW(10,11));
```

```
SELECT 编号, 坐标点 FROM ROW_TEST;
```

```
编号 坐标点  
3    (8,9)  
4    (10,11)
```

```
UPDATE ROW_TEST SET 坐标点.x=118 WHERE 坐标点.x=10;
```

```
SELECT 编号,坐标点.x, 坐标点.y FROM ROW_TEST;
```

```
编号    坐标点.x    坐标点.y
```



1	8	9
2	118	11

## 2.4 表数据类型

TABLE 记录（结构化）数据类型

语法：

```
<表数据类型> ::= TABEL ( 列名 <数据类型> [ {, 列名 <数据类型> } ... ] ) [ <行数> ]
```

功能：定义一个表数据类型的数据。

例如：TABEL (username CHAR(12), password CHAR(8)) 。

TABEL ('huayisoft', 'www.com') 是合法的。

```
CREATE TABLE TABLE_TEST  
(  
    编号 INT,  
    坐标点 TABLE (x INT, y INT) 3  
);
```

```
INSERT INTO TABLE_TEST VALUES (1, TABLE{(1,2)(3,4)(5,6)});
```

```
INSERT INTO TABLE_TEST VALUES (2, TABLE{(21,2)(23,4)(25,6)});
```

```
INSERT INTO TABLE_TEST VALUES (3, TABLE{(31,2)(33,4)(35,6)});
```

```
SELECT 编号, 坐标点 FROM TABLE_TEST;
```

```
编号 坐标点  
1,{(1,2)(3,4)(5,6)}  
2,{(21,2)(23,4)(25,6)}  
3,{(31,2)(33,4)(35,6)}
```

## 2.5 类数据类型

CLASS 记录（结构化）数据类型

语法：

```
<类数据类型> ::= <数据类型> CLASS <class name>
```



功能：定义类数据类型的数据。

```
CREATE TABLE IP_VERSION
(
    IPS CHAR (15),
    IP BINARY(4) CLASS db.jdbc.type.IP,
    VERSION BINARY(4) CLASS db.jdbc.type.Version
);

INSERT INTO IP_VERSION VALUES ( '001' , '127.0.0.1','1.2.4_08');
INSERT INTO IP_VERSION VALUES ('002', '127.0.0.2','1.2.4');

UPDATE IP_VERSION SET IP.set ('222.222.222.222') WHERE IP.get(2)='0';
```

## 2.6 类型数据类型

TYPE 记录（结构化）数据类型

语法：

```
<类型数据类型> ::= TYPE <class name> ( 字符串, 字符串 ... )
```

功能：定义类型数据类型的数据。

```
CREATE TABLE CHECKBOX
(
    姓名 CHAR (15),
    民族 TYPE db.jdbc.type.CheckBoxType ( '汉族' , '蒙古族' , '回族' , '藏族' )
);

INSERT INTO CHECKBOX VALUES ( '张三', '汉族');
INSERT INTO CHECKBOX VALUES ( '李四', '藏族');
```



## 多项式选择数据类型

### 2.6.1 多项式单选数据类型

语法：

<多项式单选数据类型> ::= TYPE db.jdbc.type.RadioButtonType(选择项 1, 选择项 2, 选择项 3 [,...选择项 n)

### 2.6.2 多项式多选数据类型

语法：

<多项式多选数据类型> ::= TYPE db.jdbc.type.CheckBoxType(选择项 1, 选择项 2, 选择项 3 [,...选择项 n)

例如：

```
CREATE TABLE CHECKBOX
(
    姓名 CHAR (15),
    民族    TYPE db.jdbc.type.RadioButtonType (汉族,蒙古族,回族,藏族),
    文化程度 TYPE db.jdbc.type.RadioButtonType(小学,中学,中专,大专,本科,研究生,博士,博士后),
    交通工具 TYPE db.jdbc.type.CheckBoxType(自行车,摩托车,三轮车,轿车,小货车,大货车,其它)
);
```

## 2.7 方法调用数据类型

语法：

<方法调用数据类型> ::= METHOD 字段名称.方法名称()

功能：调用对应字段对象中的方法,并显示其对应结果值

例如：

```
CREATE TABLE METHOD_TEST
(
```



```
身份证号 PCODE,  
姓名 CHAR(8),  
性别 METHOD 身份证号.getSex(),  
年龄 METHOD 身份证号.getAge(),  
出生年月 METHOD 身份证号.getYearMonth (),  
办证地 METHOD 身份证号.getBirthPlace ()  
);
```

## 2.8 对象数据类型

语法：

```
<对象数据类型> ::= OBJECT [(长度) ]
```

功能：对象数据类型的列值可以是任何的数据对象,当然也包含了其它字段数据类型的对象(值),其显示值是方法.toString()所显示的内容。

## 2.9 抽象数据类型

语法：

```
<抽象数据类型> ::= 抽象数据类型名称
```

功能：用户可以将以上数据类型扩展为用户自定义的数据类型。通常用户可以使用自己开发的类对象来处理对应的数据。

例如：

```
CREATE TYPE PCODE AS BIGINT CLASS db.jdbc.type.PCode
```

则：

```
CREATE TABLE TYPE_TEST (身份证号 PCODE, 姓名 CHAR(8)) 是合法的。
```

也可删除用户定义类型

```
DROP TYPE PCODE
```

## 2.10 实用数据类型

语法：

```
<实用数据类型> ::= <中华人民共和国身份证数据类型>  
| <IP 地址数据类型>
```



| <版本号数据类型>  
| <货币数据类型>

## 2.10.1 PCODE 中华人民共和国身份证数据类型

PCODE 类型

语法：

```
<中华人民共和国身份证数据类型> ::= PCODE
```

功能：定义有效中华人民共和国身份证号。满足 GB11643-1999<<公民身份号码>>中的规定。

例如：'11010519491231002X' 是合法的。

'110105194912310021' 是不合法的。

```
CREATE TABLE PCODE_TEST (身份证号 PCODE, 姓名 CHAR(8));  
INSERT INTO PCODE_TEST VALUES ('11010519491231002X', '张某');  
INSERT INTO PCODE_TEST VALUES ('440524188001010014', '王某');
```

```
SELECT 身份证号, 姓名,  
       CAST(身份证号.getSex() AS CHAR(2)) AS 性别,  
       CAST(身份证号.getBirthday() AS DATE) AS 出生日期  
FROM PCODE_TEST;
```

身份证号	姓名	性别	出生日期
11010519491231002X	张某	女	1949年12月31日
440524188001010014	王某	男	1880年1月1日

## 2.10.2 IP 地址数据类型

语法：

```
<IP 地址数据类型> ::= IP
```

功能：定义有效的 IP 地址。

例如：'127.0.0.1' 是合法的。



### 2.10.3 VERSION 版本号数据类型

语法：

```
<版本号数据类型> ::= VERSION
```

功能：定义有效的版本号。

例如：'1.6.0' 是合法的。

### 2.10.4 MONEY 货币数据类型

MONEY 数据类型

语法：

```
<货币数据类型> ::= MONEY [( 精度 [, 标度 ] )]
```

功能：该类型用于存储货币数据，货币数据用正、负货币值表示。货币数据存储的是精度为 19，标度为 4 的精确数值。货币数值用小数点将局部的货币单位（如角分）从总体货币单位中分隔出来。例如 2.15 表示 2 元 15 分。功能与 DEC ( 19 , 4 ) 相同。



## 3. Huayisoft DB Server 的值及其表达式

Huayisoft DB Server 支持的表达式有:数值表达式、字符串表达式、记录构造表达式、数组构造表达式、布尔值表达式等。

### 数值及其表达式

语法:

```
<值> ::= <常量>
      | <字段值>
      | <数组单元引用>
      | <记录值字段引用>
      | <集函数>
      | <子查询>
      | <静态方法>
      | <new 类对象>
      | <CAST 表达式>
      | <CASE 表达式>
      | <COALESCE 表达式>
      | <NULLIF 表达式>
      | ( <值> )
```

### 3.1 NULL 值

为了更真实地反应现实世界的对象,SQL 通过 NULL 值来表示数据的不可知或不适用。

例如:一个学生数学考 0 分与他没有参加考试可能要加以区分!

NULL 值就是用来处理这类情况的, NULL 值不是数字数据的 0,也不是字符串的".

它只是个符号,表示不可知或不适用的数据.

### 3.2 常量

语法:

```
<常量> ::= <数字常量>
        | <字符串常量>
        | <二进制常量>
        | <位常量>
```





### 3.2.1 数字常量

语法：

<数字常量> ::= <精确数字常量> | <近似数字常量>

1、精确数字常量(整数和小数常量):

99  
-99.99

所有 HUAYISOFT DB SERVER 数据类型都对应一个类对象,常量也不例外,  
精确数字常量类对象对应规则是:

没有小数位时:

如果值在-2147483648 ~ +2147483647 范围内,则对应 java.lang.Integer  
否则对应 java.lang.Long

有小数位时对应 db.jdbc.Numeric

因此,SELECT (99).toString().length()  
是合法的,且结果是 2

2、近似数字常量(浮点数常量),即带 E 符号的浮点数:

9.0E3  
-99.99E3

近似数字常量类对象对应规则是:

如果值在-3.4E + 38 ~ 3.4E + 38 范围内,则对应 java.lang.Float  
否则对应 java.lang.Double

### 3.2.2 字符串常量,即包含在一对单引号内的文本:

'Hello World!'

'abc.123'

如果字符串中包含单引号,则要求写成两个连续的单引号:

'It isn't ...'

结果: It isn't ...

字符串常量的类对象是 java.lang.String

因此,SELECT 'It isn't ...'.indexOf ('')

结果: 6



### 3.2.3 二进制常量,即以 X 打头且包含在一对单引号内的文本:

```
X' 44 55 66'
```

二进制常量的类对象是 db.jdbc.Binary

因此,SELECT X'44 55 66' .readByte ()

结果: 68

### 3.2.4 位常量,即以 B 打头且包含在一对单引号内的文本:

```
B' 1100'
```

位常量的类对象是 db.jdbc.Bits

因此,SELECT B'1100' .get (0)

结果: true

## 3.3 表达式

### 3.3.1 CAST 表达式

语法:

```
< CAST 表达式> ::= CAST ( { <值表达式> | NULL } AS { <数据类型> | 数据域名 } )
```

```
SELECT CAST (8 AS CHAR(5)) || 'a'
```

结果: 8a

### 3.3.2 CASE 表达式

语法:

```
< CASE 表达式> ::= CASE WHEN <搜索条件> THEN <RESULT 子句>
```

```
[ <ELSE 子句> ] END
```

```
| CASE <测试值> <简单 WHEN 子句> ... [ <ELSE 子句> ]
```

```
<简单 WHEN 子句> ::= WHEN <值表达式> THEN <RESULT 子句>
```

```
<ELSE 子句> ::= ELSE <RESULT 子句>
```

```
<RESULT 子句> ::= <值表达式> | NULL
```

```
<测试值> ::= <值表达式>
```



```
SELECT CASE WHEN 2>1 THEN 8 ELSE 9 END
```

结果: 8

```
SELECT CASE WHEN 2>3 THEN 8 ELSE 9 END
```

结果: 9

```
SELECT CASE 2 WHEN 1 THEN 8 ELSE 9 END
```

结果: 9

```
SELECT CASE 2 WHEN 2 THEN 8 ELSE 9 END
```

结果: 8

### 3.3.3 COALESCE 表达式

语法:

```
< COALESCE 表达式 > ::= COALESCE ( <值表达式> {, <值表达式> } ... )
```

```
SELECT COALESCE (8,9)
```

结果: 8

```
SELECT COALESCE (8,0,9)
```

结果: 8

### 3.3.4 NULLIF 表达式

语法:

```
< NULLIF 表达式 > ::= NULLIF ( <值表达式> , <值表达式> )
```

```
SELECT NULLIF (8, 9)
```

结果: 8

### 3.3.5 静态方法

静态方法是可以直接调用的方法:

```
SELECT Math::abs (-99.99)
```

结果: 99.99

```
SELECT Math::min (9, 10)
```

结果: 9



```
SELECT Math::cos (0.1 as double)
```

结果: 0.9950041651292624

注意：类名与方法名之间是用双冒号 (::) 分隔的，这与 Java 调用不同，主要是与字段方法调用加以区分。

### 3.3.6 new 类对象

由于 Huayisoft DB Server 数据类型都对应一个类对象，因此可以使用 new 生成一个 Huayisoft DB Server 数据：

```
SELECT new db.jdbc.Date ( '2009 年 09 月 09 日' )
```

结果: 2009 年 09 月 09 日

```
SELECT new db.jdbc.interval.Year (2009)
```

结果: 2009

```
SELECT new db.jdbc.type.PCode ( '11010519491231002X' ).getSex ()
```

结果: 女

### 3.3.7 数值表达式

语法：

```
<数值表达式> ::= <数值项> [ { { + | - } <数值项> }... ]
```

```
<数值项> ::= <数值子项> [ { { * | / | % } <数值子项> }... ]
```

```
<数值子项> ::= [ { + | - } ] <值>
```

### 3.3.8 字符值表达式

语法：

```
<字符值表达式> ::= <字符值> [ { || <字符值> }... ]
```

```
SELECT '1'||'+||'2||'='||'3'
```

结果: 1+2=3

### 3.3.9 记录构造表达式

语法：

```
<记录构造表达式> ::= ROW ( <值表达式> [ { , <值表达式> }... ] )
```



### 3.3.10 数组构造表达式

语法：

<数组构造表达式> ::= <数组构造> [ { || <数组构造> } ... ]

<数组构造> ::= ARRAY [ <值表达式> [ { , <值表达式> } ... ] ]

### 3.3.11 布尔值表达式

语法：

<布尔表达式> ::= <布尔项> [ { OR <布尔项> } ... ]

<布尔项> ::= <布尔子项> [ { AND <布尔子项> } ... ]

<布尔子项> ::= [ NOT ] { <谓词> | ( <布尔表达式> ) }  
[ IS [ NOT ] { TRUE | FALSE | UNKNOWN } ]

<谓词> ::= <比较谓词>  
| < BETWEEN 谓词>  
| < LIKE 谓词>  
| < EXISTS 谓词>  
| < IN 谓词>  
| < UNIQUE 谓词>  
| < NULL 谓词>  
| < MATCH 谓词>  
| < OVERLAPS 谓词>  
| < QUANTIFIED 比较谓词>

<比较谓词> ::= <行值构造> <比较符> <行值构造>

<比较符> ::= =  
| >  
| <  
| >=  
| <=  
| <>

< BETWEEN 谓词> ::= <行值构造> [ NOT ] BETWEEN <行值构造> AND <行值构造>



< LIKE 谓词> ::= <字符值表达式> [ NOT ] LIKE <字符值表达式> [ ESCAPE 字符 ]

< EXISTS 谓词> ::= EXISTS <子查询>

< IN 谓词> ::= <行值构造> [ NOT ] IN { <子查询> | ( <值表达式> [, <值表达式> ]... ) }

< UNIQUE 谓词> ::= UNIQUE <子查询>

< NULL 谓词> ::= <行值构造> IS [ NOT ] NULL

< MATCH 谓词> ::= <行值构造> MATCH [ UNIQUE ] [ PARTIAL | FULL ] <子查询>

< OVERLAPS 谓词> ::= <行值构造> OVERLAPS <行值构造>

< QUANTIFIED 比较谓词> ::= <行值构造> <比较符> { ALL | { SOME | ANY } <子查询>



## 4. Huayisoft DB Server 数据类型与类对象

所有的 HUAYISOFT DB SERVER 数据类型都对应一个 Java 类对象：

### 4.1 Huayisoft DB Server 数据类型与类对象对应表

数据类型	类对象
CHAR	java.lang.String
VARCHAR	java.lang.String
CLOB	db.jdbc.type.Clob
NATIONAL CHARACTER	java.lang.String
NATIONAL CHARACTER VARYING	java.lang.String
NCLOB	db.jdbc.type.NClob
BINARY	db.jdbc.Binary
BLOB	db.jdbc.type.Blob
BIT	db.jdbc.Bits
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
INTEGER	java.lang.Integer
BIGINT	java.lang.Long
NUMERIC	db.jdbc.Mumeric
DECIMAL	db.jdbc.Mumeric
REAL	java.lang.Float
DOUBLE PRECISION	java.lang.Double
BOOL	java.lang.Boolean
DATE	db.jdbc.Date
TIME	db.jdbc.Time
TIMESTAMP	db.jdbc.Timestamp
INTERVAL YEAR	db.jdbc.interval.Year
INTERVAL MONTH	db.jdbc.interval.Month
INTERVAL YEAR TO MONTH	db.jdbc.interval.YearMonth
INTERVAL DAY	db.jdbc.interval.Day
INTERVAL DAY TO HOUR	db.jdbc.interval.DayHour
INTERVAL DAY TO MINUTE	db.jdbc.interval.DayMinute
INTERVAL DAY TO SECOND	db.jdbc.interval.DaySecond
INTERVAL HOUR	db.jdbc.interval.Hour
数据类型	类对象



INTERVAL HOUR TO MINUTE	db.jdbc.interval.HourMinute
INTERVAL HOUR TO SECOND	db.jdbc.interval.HourSecond
INTERVAL MINUTE	db.jdbc.interval.Minute
INTERVAL MINUTE TO SECOND	db.jdbc.interval.MinuteSecond
INTERVAL SECOND	db.jdbc.interval.Second
ARRAY	db.jdbc.type.Array
ROW	db.jdbc.Record
TABLE	db.jdbc.RecordSet
PCODE	db.jdbc.type.PCode
IP	db.jdbc.type.IP
VERSION	db.jdbc.type.Version
MONEY	db.jdbc.type.Money

注：接口 `java.io.Serializable` 及类 `java.lang.Byte`、`java.lang.Short`、`java.lang.Integer`、`java.lang.Long`、`java.lang.Float`、`java.lang.Double`、`java.lang.String`、`java.lang.Boolean` 请参考有关 java 语言书。

## 4.2 接口 Interfacey 与 类对象 Class

请参考：<documents/index.html>。







<表表达式> ::= <FROM 子句>

| <WHERE 子句>

| <GROUP BY 子句>

| <HAVING 子句>

| <LIMIT 子句>

<FROM 子句> ::= FROM <表参考> [,表参考] ...

<表参考> ::= <主表> | <连接表>

<主表> ::= 表名 [[ AS ] 表别名 [( <列名列表> )]]

<WHERE 子句> ::= WHERE <搜索条件>

<GROUP BY 子句> ::= GROUP BY 列名 | <值表达式> [,列名 | <值表达式> ] ...

<HAVING 子句> ::= HAVING <搜索条件>

<LIMIT 子句> ::= LIMIT [起始行] 行数

<行值构造单元> ::= NULL | DEFAULT | <值表达式>

<直接表> ::= TABLE 表名

<连接表> ::= <交叉连接表>

| <连接表>

| <自然连接表>

| <合并连接表>

<交叉连接表> ::= <表参考> CROSS JOIN <主表>

<连接表> ::= <表参考> [<连接类型>] JOIN <表参考> <连接说明>

<自然连接表> ::= <表参考> NATURAL [ <连接类型> ] JOIN <主表>

<合并连接表> ::= <表参考> UNION JOIN <主表>

<连接类型> ::= INNER | <外连接类型> [ OUTER]



<外连接类型> ::= LEFT | RIGHT | FULL

<连接说明> ::= ON <搜索条件> | USING ( <列名列表> )

<ORDER BY 子句> ::= ORDER BY  
                   { 列名 | 列号 } [ ASC | DESC ]  
                   [ { 列名 | 列号 } [ ASC | DESC ] ] ...  
                   [ BY DICTIONARY ]

<INTO 子句> ::= INTO { TABLE 临时表名 | TEXT 文本文件名 }

<搜索条件> ::= <布尔表达式>

注意:

<表值构造>及<文本记录>请参考<插入记录语句>

## 5.2 单表查询

### 5.2.1 SELECT 子句

选择项可以是下列之一：

- (a) FROM 子句所指定的表中的列名。
- (b) 常量。
- (c) 表达式。

#### 1. 查看指定字段的数据

列出每个销售点的编号、城市名称、所属区域、经理、销售定额、销售额

```
SELECT 编号,城市名称,所属区域,经理,销售定额,销售额
FROM 销售点
```

编号	城市名称	所属区域	经理	销售定额	销售额
1	北京	华北	108	300000	186042
2	南昌	华中	104	575000	692637
3	合肥	华中	105	800000	735042
4	武汉	华中	101	350000	367911
5	天津	华北	108	725000	835915



## 2. 选择所有列 (SELECT \*) \* : 表示 FROM 表中的全部字段

列出每个销售点所有信息

```
SELECT *  
FROM 销售点
```

编号	城市名称	所属区域	经理	销售定额	销售额
1	北京	华北	108	300000	186042
2	南昌	华中	104	575000	692637
3	合肥	华中	105	800000	735042
4	武汉	华中	101	350000	367911
5	天津	华北	108	725000	835915

## 3. 常量可以直接成为选择项

列出每个销售点的城市名称、销售定额

```
SELECT 城市名称, '的销售定额是',销售定额  
FROM 销售点
```

城市名称	的销售定额是	销售定额
北京	的销售定额是	300000
南昌	的销售定额是	575000
合肥	的销售定额是	800000
武汉	的销售定额是	350000
天津	的销售定额是	725000

## 4. 去重(DISTINCT), 即查询结果中没有相同的列。

列出每个销售点经理的编号

```
SELECT DISTINCT 经理  
FROM 销售点
```

```
经理  
-----  
108  
104
```



105

101

### 5. 字段计算

列出每个销售点的城市名称及销售额与销售定额之差

```
SELECT 城市名称, 销售额 - 销售定额  
FROM 销售点
```

城市名称 销售额-销售定额

```
-----  
北京      -113958  
南昌      117637  
合肥      -64958  
武汉      17911  
天津      110915
```

### 6. 给选择项另取一个更富于意义的名称

列出每个销售点经理的编号

```
SELECT DISTINCT 经理 AS 经理编号  
FROM 销售点
```

经理编号

```
-----  
108  
104  
105  
101
```

## 5.2.2 记录选择 ( WHERE 子句 )

如上述所示，缺省 WHERE 子句，表示永真条件，即选择 FROM 表中所有的记录。

### 1. 比较测试

操作符说明如下：

操作符	说明
-----	----



=	等于
<>	不等于
>	大于
>=	大于等于
<	小于
<=	小于等于

列出北京的经理编号

```
SELECT 经理
      FROM 销售点
      WHERE 城市名称='北京'
```

经理

-----

108

## 2. 范围测试 ( BETWEEN )

列出金额在 2 万至 3 万之内的订单

```
SELECT 编号, 金额
      FROM 订单
      WHERE 金额 BETWEEN 20000 AND 29999.99
```

编号 金额

-----

8236 22500

8187 27500

8242 22500

以上 BETWEEN 完全等价于：

金额 >= 20000 AND 金额 <=29999.99

## 3. 组成员测试([NOT] IN)

列出在北京、天津工作的销售员



```
SELECT 编号,姓名
      FROM 销售员
      WHERE 销售点 IN (1,5)
```

编号 姓名

-----

103 杨刚

108 李立发

110 蔡晓

以上 IN 完全等价于：

销售点=1 OR 销售点=5

#### 4. 模式匹配词 ( [NOT] LIKE )

查询所有姓'柯' 销售员

```
SELECT 编号,姓名
      FROM 销售员
      WHERE 姓名 LIKE '柯%'
```

编号 姓名

-----

101 柯长富

这里百分符 ( % ) 表示任何 0 个或多个字符的字符串，下划线 ( \_ ) 表示任何单个字符。

#### 5. 空值测试 ( IS [NOT] NULL )

查询还没有分配到销售点的销售员

```
SELECT 编号,姓名
      FROM 销售员
      WHERE 销售点 IS NULL
```

编号 姓名

-----

107 夏大宝

注：“销售点 = NULL” 是非法的。



## 6. 复合搜索条件 ( AND , OR , NOT )

WHERE 中可以包含多个搜索条件。搜索条件之间用 AND 或者 OR 连接。

列出在北京或天津工作的销售员

```
SELECT 编号,姓名
      FROM 销售员
      WHERE 销售点=1 OR 销售点=5
```

编号	姓名
103	杨刚
109	李立发
110	蔡晓

## 5.2.3 对查询结果显示顺序的控制 ( ORDER BY 子句 )

对查询结果按一个或多个列进行排序.排序列必需在查询结果中

列出每个销售点的城市名称、销售额,并按销售额从大到小排序

```
SELECT 城市名称, 销售额
      FROM 销售点
      ORDER BY 销售额 DESC
```

城市名称	销售额
天津	835915
合肥	735042
南昌	692637
武汉	367911
北京	186042

其中关键字 DESC 表明按销售额降序排列,如果是升序,关键字 ASC 可以省略。排序列可以不止一个,按其排列的先后顺序决定其主序列。





## 5.3 多表查询（连接查询）

两个或两个以上表的查询称为多表查询。

### 5.3.1 等连接查询

列出销售点及其经理的姓名

```
SELECT 城市名称, 姓名
      FROM 销售点, 销售员
      WHERE 经理=销售员.编号
```

城市名称	姓名
北京	李立发
南昌	张民山
合肥	罗明贵
武汉	柯长富
天津	李立发

这里要查询销售点经理的姓名，必须涉及到两个表：销售点表及销售员表，连接条件是销售点表中经理代号必须与销售员表的编号相同。

注意：其中 FROM 子句（列出查询数据所在的表）中的表与表之间要用逗号‘，’ 隔开，连接条件需要用表名（销售员）作为前缀以便区分同名字段（编号）。

在 FROM 子句中可以给表另取一个更便于使用的名称，上述语句可以写成：

```
SELECT 城市名称, 姓名
      FROM 销售点, 销售员 AS B
      WHERE 经理=B.编号
```

### 5.3.2 多个匹配字段

列出销售点及其经理的姓名

```
SELECT 编号, 产品名称, 金额
      FROM 订单 A, 产品 B
      WHERE A.类别=B.类别
            AND A.号=B.代号
```



编号	产品名称	金额
8161	笔记本电脑	31500
8212	冲电器 B	3745
8189	化妆品 A	1458
.		
.		
.		

注：必须使用 AND 操作符连接多个连接条件。

### 5.3.3 内连接

INNER JOIN 表明查询结果仅包含两个表中相匹配的行。

列出销售点及其经理的姓名

```
SELECT 城市名称, 姓名
FROM 销售点 INNER JOIN 销售员
ON 经理=销售员.编号
```

城市名称	姓名
北京	李立发
南昌	张民山
合肥	罗明贵
武汉	柯长富
天津	李立发

从这个查询结果中可以看出与上述‘5.3.1 等连接查询’中的结果是一样的；但是，这里使用 JOIN 操作明确地指明要连接的二个表，并把它们的连接条件放在 ON 子句中，

连接条件 JOIN,查询条件 WHERE 是分开的。

### 5.3.4 外连接

LEFT JOIN 表明查询结果中包含左表中的所有行以及与右表相匹配的行。如果与右表中不匹配时，将用空值来代替。

RIGHT JOIN 表明查询结果中包含右表中的所有行以及与左表相匹配的行。如果与左表中不匹配



时，将用空值来代替。建议把 RIGHT JOIN 语句转化为 LEFT JOIN 语句来使用。

FULL JOIN 表明查询结果中包含两个表中相匹配的和 unmatched 的所有行。

列出销售员及其他的城市名称

```
SELECT 编号,姓名, 年龄,城市名称
FROM 销售员 LEFT JOIN 销售点
ON 销售员.销售点 =销售点.编号
```

编号	姓名	城市名称
101	柯长富	武汉
102	王美丽	南昌
103	杨刚	天津
104	张民山	南昌
105	罗明贵	合肥
106	刘应东	合肥
107	夏大宝	NULL
108	李立发	天津
109	吴小虎	合肥
110	蔡晓	北京

列出销售员及其他的城市名称

```
SELECT 编号,姓名, 年龄,城市名称
FROM 销售点 RIGHT JOIN 销售员
ON 销售点.编号 = 销售员.销售点
```

编号	姓名	城市名称
101	柯长富	武汉
102	王美丽	南昌
103	杨刚	天津
104	张民山	南昌
105	罗明贵	合肥
106	刘应东	合肥
107	夏大宝	NULL
108	李立发	天津
109	吴小虎	合肥
110	蔡晓	北京



以上左连接与右连接其结果是一样的，也就是说，只要主表次表关系不变，书写成左连接还是右连接是没有关系的。

## 5.4 统计查询

统计查询是通过集函数以及 GROUP BY 分组子句来完成的。

### 5.4.1 集函数

HUAYISOFT DB SERVER 支持标准集函数：即求和 (SUM)、最大值 (MAX)、最小值(MIN)、平均值 (AVG)、记录数(COUNT)。语法如下：

语法：

```
<集函数> ::= COUNT (*)
          | AVG (<选择项>)
          | COUNT (<选择项>)
          | MIN (<选择项>)
          | MAX (<选择项>)
          | SUM (<选择项>)
```

#### 1. 求数值型字段的平均值(AVG)

求茶叶的平均价格

```
SELECT AVG (价格)
FROM 产品
WHERE 类别= '茶叶'
```

AVG (价格)

-----

352.33

#### 2. 求选择项中的最小值和最大值 ( MIN 和 MAX )。

求茶叶的最小价格和最大价格

```
SELECT MIN (价格),MAX(价格)
FROM 产品
WHERE 类别= '茶叶'
```



MIN (价格) MAX(价格)

-----  
180            652

### 3. 求一列数据的和(SUM)

求所有销售员的销售总额

```
SELECT SUM (销售额)
FROM 销售员
WHERE 类别 = '茶叶'
```

SUM (销售额)

-----  
2893532

### 4. 求一列数据的和(COUNT)

COUNT (\*) 统计查询结果中的记录数。

COUNT (<选择项>),统计一列中选择项的个数。

求销售员的人数

```
SELECT COUNT (*)
FROM 销售员
```

COUNT (\*)

-----  
10

求已编号的销售员人数

```
SELECT COUNT (编号)
FROM 销售员
```

COUNT (编号)

-----  
10

求已分配销售点的销售员人数



```
SELECT COUNT (销售点)
FROM 销售员
```

```
COUNT (销售点)
```

```
-----
```

```
9
```

## 5.4.2 分组查询 ( GROUP BY 子句 )

可以对一个或多个列进行分组。

通常与集函数配合使用

由于分组最终是要为每组计算出一个结果元组 ( 而且每组也只能有一个元组 ), 而聚集函数正是实现这一目的的, 因此集函数大量地被使用在分组场合。

计算销售员订单的平均金额

```
SELECT 销售员,AVG (金额)
FROM 销售员
GROUP BY 销售员
```

销售员	AVG (金额)
-----	-----
101	7865.40
102	3552.50
103	5694
104	16479
106	8876
107	11566
108	8376
109	1350
110	11477.33

注意 SELECT 目标列之间的一致性

能够从每组元组中计算出结果元组的, 除了集函数外, 还可以是分组列, 其它的任何内容都不能成为结果元组的组成部分。也就是说分组的计算结果必须是描述组的属性。

## 5.4.3 分组搜索条件 ( HAVING 子句 )

HAVING 条件必须是刻划分组的属性或其表达式, 或者是集函数。

计算销售员订单的平均金额大于 10000 的平均金额



```
SELECT 销售员,AVG (金额)
FROM 销售员
GROUP BY 销售员
HAVING AVG (金额) > 10000
```

销售员	AVG (金额)
104	16479
107	11566
110	11477.33

#### 分清 WHERE 与 HAVING 的作用

WHERE 条件是对原始表的元组的限制条件，而 HAVING 条件是对分组后的中间表的元组的限制，所以一般情况下二者是不可混淆也不可交换的。但有一个特例，既 HAVING 条件是单纯的作用于分组属性上的条件，这时可以将其转化为 WHERE 条件以提高执行效率。

计算销售员 101 订单的平均金额

```
SELECT 销售员,AVG (金额)
FROM 销售员
GROUP BY 销售员
HAVING 销售员=101
```

销售员	AVG (金额)
101	7865.40

该查询等价于:

```
SELECT 销售员,AVG (金额)
FROM 销售员
WHERE 销售员=101
GROUP BY 销售员
```

销售员	AVG (金额)
101	7865.40



## 5.5 合并查询结果 ( UNION 子查询 )

查询结果包含主查询结果以及 UNION 子查询的查询结果. 可以包含多个 UNION 子句。  
列出单价超出 2000 或在一个订单中订购了超过 30000 元所有产品

```
SELECT 类别, 代号
      FROM 订单
      WHERE 金额>30000
UNION
SELECT DISTINCT 类别, 代号
      FROM 产品
      WHERE 价格>2000
```

类别	代号
-----	-----
电脑	A02
电脑	A03
玉器	D02
手机	B01
手机	B02

UNION 子句遵循以下规则：  
主查询以及各 UNION 子查询必须包含相同的结果列。  
各结果列的类型及宽度必须相同。

## 5.6 显示指定范围的记录 ( LIMIT 子句 )

列出头 3 个销售员的姓名

```
SELECT 姓名
      FROM 销售员
      LIMIT 1,3
```

姓名
-----
柯长富
王美丽
杨刚





## 6. Huayisoft DB Server 的数据定义

数据定义语言(Data Definition Language,DDL)用以完成数据库中各种对象的定义。包括数据库、用户、模式、基本表、索引、视图等。HUAYISOFT DB SERVER 使用 CREATE、ALTER、DROP 语句来对数据库结构进行动态地定义、修改与删除。

语法：

```
<数据定义> ::= <数据库定义>
                | <用户定义>
                | <模式定义>
                | <表定义>
                | <抽象数据类型定义>
                | <数据域定义>
                | <断言定义>
                | <索引定义>
                | <视图定义>
                | <记录文本定义>
```

### 6.1 数据库定义

Huayisoft DB Server 数据库(catalog)是数据库模式 ( schema ) 的集合。

语法：

```
<数据库定义> ::= <创建数据库语句>
                | <删除数据库语句>
```

#### 6.1.1 创建数据库

语法：

```
<创建数据库语句> ::= CREATE { CATALOG | DATABASE } [IF NOT EXISTS] 数据库名 路径名
```

HUAYISOFT DB SERVER 中的一个数据库对应一个路径。如果要创建的数据库已经存在，将产生一个警告，使用“IF NOT EXISTS”将不会产生一个警告。

#### 6.1.2 删除数据库

语法：

```
<删除数据库语句> ::= DROP { CATALOG | DATABASE } [IF EXISTS] 数据库名
```



如果删除的数据库名不存在，将出现一个警告，使用“IF EXISTS”将不会出现警告。

### 6.1.3 设置当前数据库

语法：

```
<设置当前数据库语句> ::= SET DATABASE | CATALOG 数据库名
```

## 6.2 用户定义

语法：

```
<用户定义> ::= <创建用户语句>  
                | <修改用户语句>  
                | <删除用户语句>
```

### 6.2.1 创建用户

语法：

```
<创建用户语句> ::= CREATE USER [IF NOT EXISTS] 用户名 密码  
                [ <修改用户子句> ]
```

如果要创建的用户已经存在，将产生一个警告，使用“IF NOT EXISTS”将不会产生一个警告。

Huayisoft DB Server 华易数据库管理系统安全策略：

### 6.2.2 修改用户

语法：

```
<修改用户语句> ::= ALTER USER 用户名 <修改用户子句>
```

```
<修改用户子句> ::= <修改用户值> [,<修改用户值>] ...
```

```
<修改用户值> ::= <用户列名> [ 新值 ]
```

```
<用户列名> ::= NAME  
            | USERNAME  
            | PASSWROD  
            | SCHEMA  
            | BOOT  
            | MENU
```



```
| STARTDATE  
| ENDDATE  
| SERVICE
```

其中：NAME 为用户的真实姓名；  
USERNAME 为用户名称；  
PASSWROD 为用户密码；  
SCHEMA 为缺省的模式名称；  
BOOT 为用户登录类名称；  
MENU 为用户菜单名称；  
STARTDATE 为用户账号使用期的起始日期；  
ENDDATE 为用户账号使用期的结束日期；  
SERVICE 是否为用户提供某服务（如：FTP）标记；

### 6.2.3 删除用户

语法：

```
<删除用户语句> ::= DROP USER [IF EXISTS] 用户名
```

如果删除的用户名不存在，将出现一个警告，使用“IF EXISTS”将不会出现警告。

## 6.3 模式定义

Huayisoft DB Server 模式(schema)是表(table)、视图(view)、权限(privilege)等数据库对象的集合。

语法：

```
<模式定义> ::= <创建模式语句>  
| <删除模式语句>
```

### 6.3.1 创建模式

语法：

```
<创建模式语句> ::= CREATE SCHEMA [IF NOT EXISTS]  
    { 模式名 [AUTHORIZATION 用户名] | [模式名] AUTHORIZATION 用户名 }  
    [ <模式单元> ] ...
```

```
<模式单元> ::= <表定义>  
| <视图定义>  
| <数据域定义>  
| <断言定义>
```



```
| <触发器定义>  
| <抽象数据类型定义>  
| <授权语句>
```

如果要创建的已经模式存在，将产生一个警告，使用“IF NOT EXISTS”将不会产生一个警告。

使用 CREATE SCHEMA 语句可以方便地创建有引用循环(利用主键字/外键字对,两个或多个表可以互相引用)的表

如:

```
CREATE SCHEMA REF_CYCLE  
CREATE TABLE PRIMARY_TABLE  
    (CODE INTEGER NOT NULL PRIMARY KEY,  
     ID INTEGER REFERENCES FOREIGN)  
  
CREATE TABLE FOREIGN_TABLE  
    (ID INTEGER NOT NULL PRIMARY KEY,  
     CODE INTEGER REFERENCES PRIMARY_TABLE)
```

## 6.3.2 删除模式

语法：

```
<删除模式语句> ::= DROP SCHEMA [IF EXISTS] 模式名
```

如果删除的模式名不存在，将出现一个警告，使用“IF EXISTS”将不会出现警告。

## 6.3.3 设置当前模式

语法：

```
<设置当前数据库语句> ::= SET SCHEMA 模式名
```

## 6.4 表定义

语法：

```
<表定义> ::= <创建表语句>  
           | <修改表语句>  
           | <删除表语句>
```

注意: Huayisoft DB Server 中的.dbf 文件与其它数据库的.dbf 文件除了在文件后缀名相同之外，





<列名列表> ::= 列名 [, 列名 ] ...

<约束属性> ::= [ NOT ] DEFERRABLE { INITIALLY DEFERRED | INITIALLY IMMEDIATE }

如果要创建的基本表已经存在，将产生一个警告，使用“IF NOT EXISTS”将不会产生一个警告。

## 6.4.2 修改表

语法：

<修改表语句> ::= ALTER TABLE 表名 <修改表操作>

<修改表操作> ::= ADD [ COLUMN ] <列定义>  
| INSERT 列插入位置 <列定义>  
| ALTER [ COLUMN ] 列名 SET DEFAULT 缺省值  
| ALTER [ COLUMN ] 列名 DROP DEFAULT  
| DROP [ COLUMN ] 列名  
| ADD <表约束定义>  
| DROP CONSTRAINT 约束名称

以上分别为：

- 加入一列
- 在指定列位置插入一列
- 设置列缺省值
- 删除列缺省值
- 删除指定的列
- 加入表约束
- 删除表约束

## 6.4.3 删除表

语法：

<删除表语句> ::= DROP TABLE [IF EXISTS] 表名 [,表名] ...

如果删除的基本表不存在，将出现一个警告，使用“IF EXISTS”将不会出现警告。

## 6.5 抽象数据类型定义

语法：

<抽象数据类型定义> ::= <创建抽象数据类型语句>



| <删除抽象数据类型语句>

## 6.5.1 创建抽象数据类型

语法：

<创建抽象数据类型语句> ::= CREATE TYPE 类型名 AS <数据类型> CLASS 类对象名

## 6.5.2 删除指定的自定义抽象数据类型

语法：

<删除抽象数据类型语句> ::= DROP TYPE [ IF EXISTS ] 类型名

# 6.6 数据域定义

语法：

<数据域定义> ::= <创建数据域语句>  
| <删除数据域语句>

## 6.6.1 创建数据域

语法：

<创建数据域语句> ::= CREATE DOMAIN 域名  
AS ] <数据类型> [DEFAULT 值] <域约束定义>

<域约束定义> ::= [CONSTRAINT 约束名] <检查约束定义> [ <约束属性> ]

## 6.6.2 删除数据域

语法：

<删除数据域语句> ::= DROP DOMAIN [ IF EXISTS ] 域名

# 6.7 断言定义

语法：

<断言定义> ::= <创建断言语句>



| <删除断言语句>

## 6.7.1 创建断言

语法： <创建断言语句> ::= CREATE ASSERTION 断言名 <检查约束定义> [ <约束属性> ]

## 6.7.2 删除断言

语法：

<删除断言语句> ::= DROP ASSERTION [ IF EXISTS ] 断言名

# 6.8 索引定义

语法：

<索引定义> ::= <创建索引语句>  
| <删除索引语句>

## 6.8.1 创建索引

语法：

<创建索引语句> ::= CREATE [UNIQUE] INDEX [IF NOT EXISTS] 索引名  
ON 表名 ( 列名 [DESC | ASC] [,列名 [DESC | ASC] ] ... )

其中关键字 DESC 表明按降序排列，如果是升序，关键字 ASC 可以省略。

- (1) 索引可以建立在若干个字段上，称为组合索引。对于组合索引必须把最重要的字段加在最前面。
- (2) UNIQUE 选项，指出索引是唯一索引，即在索引字段上的值不能重复，唯一索引用来维护字段的唯一性。
- (3) 一个表上可以建多个索引。

## 6.8.2 删除索引

语法：

<删除索引语句> ::= DROP INDEX [IF EXISTS] 索引名 ON 表名





## 6.9 视图定义

语法：

```
<视图定义> ::= <创建视图语句>
                | <删除视图语句>
```

### 6.9.1 创建视图

语法：

```
<创建视图语句> ::= CREATE VIEW [IF NOT EXISTS] 视图名 [ ( <列名列表> ) ] AS 查询语句
```

### 6.9.2 删除指定的视图

语法：

```
<删除视图语句> ::= DROP VIEW [IF EXISTS] 视图名
```

## 6.10 记录文本定义

语法：

```
<记录文本定义> ::= <创建记录文本语句>
```

### 6.10.1 创建记录文本(数据库的导出)

语法：

```
<创建记录文本语句> ::= CREATE <文本记录> AS <查询>
```

```
<文本记录> ::= TEXT 文本文件名 [ <字段分隔符> ] [ <记录分隔符> ]
```

```
<字段分隔符> ::= FIELDS TERMINATED BY '字符串'
```

```
<记录分隔符> ::= LINES TERMINATED BY '字符串'
```

Huayisoft DB Server 数据库的导出是导入的逆过程，是指把 HUAYISOFT DB SERVER 数据库表中的记录转换成一定格式化的.txt 文本文件。即把查询结果的每一条记录转换成对应文本存入文本文件中，同时，在记录之间存入指定的记录分隔符，字段之间存入指定的字段分隔符。

缺省时(CREATE TEXT 文本文件名 AS <查询>),记录分隔符是回车换行'\r\n'，字段分隔符是。



最常见的转换是一条记录转换成文本文件中的一行(DOS,Macintosh,UNIX 分别以回车换行'\r\n',回车'\r',换行'\n'为行结束符),字段之间用英文逗号','为隔开。



## 7. Huayisoft DB Server 数据库记录的插入、修改与删除

数据库记录的操作主要包括插入、修改和删除。

### 7.1 插入数据到表中 (INSERT)

语法：

```
<插入记录语句> ::= INSERT INTO 表名
                    { [( <列名列表> )] <源数据子句> | DEFAULT VALUES }
                    [ AT 行位置 ]

<源数据子句> ::= <表值构造>
                | <查询>
                | EXECUTE 语句
                | URL_ENCODE 子句
                | <文本记录>

<表值构造> ::= VALUES <行值构造> [, <行值构造> ] ...

<行值构造> ::= <行值构造单元>
                | ( <行值构造单元> [, <行值构造单元> ] ... )
                | <子查询>

<文本记录> ::= TEXT 文本文件名 [ <字段分隔符> ] [ <记录分隔符> ]

<字段分隔符> ::= FIELDS TERMINATED BY '字符串'

<记录分隔符> ::= LINES TERMINATED BY '字符串'
```

#### 7.1.1 插入单条记录

例：新聘用了一个销售员（欧阳娜）：

```
INSERT INTO 销售员 (编号,姓名,年龄,销售点,聘用日期,经理,销售定额,销售额)
VALUES (128, '欧阳娜', 26,1, '2008.08.09',108 ,200000,0)
```



当 VALUES 子句中的常量个数与表的字段数相等，而且常量的先后次序与表中各字段的先后次序一致，则表的字段名列表可以省略：

```
INSERT INTO 销售员
VALUES (128, '欧阳娜',26, 1, '2008.08.09',108 ,200000,0)
```

## 7.1.2 插入一个查询的结果

例：把销售员表中的编号、姓名、销售点、聘用日期等信息复制到员工信息表中：

```
INSERT INTO 员工信息 (编号,姓名,销售点,聘用日期)
SELECT 编号,姓名,销售点,聘用日期
FROM 销售员
```

## 7.1.3 在指定的行位置插入一记录

例：把新设立一个销售点（深圳）放在表的第 3 行中：

```
INSERT INTO 销售点
VALUES (88, '深圳', '华南',108 ,200000,0)
AT 3
```

## 7.1.4 数据库的导入

Huayisoft DB Server 数据库的导入是把一定格式化的.txt 文本文件转换成一条条记录插入到指定的 Huayisoft DB Server 数据库表中。即把文本文件中的文本内容用记录分隔符分解成一条条记录,且每条记录用字段分隔符分解成一个个字段,无条件地插入到表中;分解成的字段应与字段名列表——对。

例：开始时，把含有产品信息的文本文件中的内容装载到产品表中：

```
INSERT INTO 产品 TEXT 产品.txt
LINES TERMINATED BY '\r\n'
FIELDS TERMINATED BY ','
```

注意:当上述语句应缺省时(INSERT INTO 表名 FROM 文本文件名),即说明:

字段名列表为表中的所有字段的以次排列,文本文件中的每一行作为一条记录(以"\r"或"\n"或"\r\n"为行结束标志),且字段之间应以英文逗号','为分隔符,文本文件中的字段顺序应与表名中的字段顺序相一致。



## 7.2 修改表中的数据(UPDATE)

语法：

```
<删除记录语句> ::= UPDATE 表名  
                SET 字段=表达式 [,字段=表达式]...  
                [ WHERE 条件 | AT 行位置 ]
```

### 7.2.1 修改满足条件的记录

即将指定表中满足条件的记录按要求进行修改。所以 WHERE 条件注意不能省略，否则将对全表作设定的修改。

例：将销售员王美丽的销售定额提高 10%：

```
UPDATE 销售员  
    SET 销售定额=销售定额*1.1  
    WHERE 姓名='王美丽'
```

### 7.2.2 修改表中的所有记录

例：将销售点的销售定额提高 10%：

```
UPDATE 销售点  
    SET 销售定额=销售定额*1.1
```

### 7.2.3 对指定的某一记录进行修改

例：将销售员王美丽的经理改为李立发( 编号=108 )

```
UPDATE 销售员  
    SET 经理 = 108  
    AT 2
```

## 7.3 从表中删除数据 ( DELETE )

语法：

```
<删除记录语句> ::= DELETE FROM 表名 [ <删除条件> ]
```

```
<删除条件> ::= WHERE <搜索条件>
```



| AT 行位置 [ BY ORDER ]

### 7.3.1 删除满足条件的所有记录。

例：新聘用的销售员（欧阳娜）辞职了：

```
DELETE FROM 销售员  
WHERE 编号=128
```

### 7.3.2 删除表中的所有记录。

例：删除所有订单：

```
DELETE FROM 订单
```

### 7.3.3 删除指定的某一记录

例：撤销临时成立的销售点（深圳）：

```
DELETE FROM 销售点  
AT 3
```



## 8. Huayisoft DB Server 的安全策略

### 8.1 数据控制

控制用户对数据的存取权限。

#### 8.1.1 授权语句

语法：

```
<授权语句> ::= GRANT 特权 ON 表名 TO 用户[,用户] ...  
                [ WITH GRANT OPTION]
```

其中：

‘特权’ 可为 SELECT、INSERT、UPDATE、DELETE、ALL PRIVILEGES。

‘用户’ 可以是 PUBLIC,表示为所有用户；

(a) 当特权也为 GRANT 时说明授权者把 GRANT 也作为一种权限授予别的用户,被授权的用户此时可以将获得的权限再授予别的用户；WITH GRANT OPTION 子句不推荐使用；

(b) 授权者在此表名上必需拥有 GRANT 特权；

(c) 授权者在此表名上必需拥有将要授出的‘特权’；

#### 8.1.2 收回权限语句

语法：

```
<收回权限语句> ::= REVOKE [GRANT OPTION FOR] 特权 ON 表名 FROM 用户
```

(a) 收回权限者在此表名上必需拥有 REVOKE 特权；

(b) 收回权限者在此表名上必需拥有将要收回的‘特权’；



## 9. Huayisoft DB Server 中的 SET、GET 语句

### 9.1 SET 语句

Huayisoft DB Server 中的 SET 语句：

语法：

```
<SET 语句> ::= SET CATALOG 数据库名
                | SET SCHEMA 模式名
                | SET USER 用户名
                | SET CATALOG_PARENT_PATH 路径名
```

### 9.2 GET 语句

Huayisoft DB Server 中的 GET 语句是为了简化所对应的应用而特设的。

语法：

```
<GET 语句> ::= GET MENU
                | GET FTP HOME
                | GET FTP ACCOUNT
                | GET FTP WRITE ACCOUNT
                | GET HTTP HOME
                | GET HTTP ACCOUNT
                | GET FILE PATH
                | GET FILE NAME
```

#### 9.2.1 获取用户的缺省菜单

GET MENU

#### 9.2.2 获取用户的 FTP 信息

检测用户是否有 FTP 账号

GET FTP ACCOUNT

获取用户的 FTP 的路径

GET FTP HOME





检测用户是否有 FTP 写 ( 修改 ) 账号

GET FTP WRITE ACCOUNT

### 9.2.3 获取用户的 HTTP 信息

检测用户是否有 HTTP 账号

GET HTTP ACCOUNT

获取用户的 HTTP 的路径

GET HTTP HOME

### 9.2.4 获取用户当前目录下的文件信息

获取用户当前目录下的所有目录名称

GET FILE PATH

获取用户当前目录下的所有文件名称

GET FILE NAME



# 10. Huayisoft DB Server 的 LOCK、UNLOCK 语句说明

## 10.1 人工加锁(直到人工减锁或用户退出)

语法:

```
<人工加锁语句> ::= LOCK TABLES 表名 <锁> [,表名 <锁>] ...
```

```
<锁> ::= READ [ READ_LOCAL ] | [ LOW_PRIORITY ] | WRITE
```

对表的读写操作进行加锁；当写操作锁加载成功时,任何其他用户都不能对该文件进行存取；而当读操作锁加载成功时,其他用户也可以对该表进行读操作；即：多个读取操作可以共享资源,而写操作将独占资源。

返回对象

结果说明:失败时抛出一错误信息

## 10.2 人工减锁

语法:

```
<人工减锁语句> ::= UNLOCK TABLES
```

释放所有的读写锁。

返回对象:空

结果说明:



## 11. Huayisoft DB Server 中的 SERVER、SERVICE 语句

SERVER 语法:

```
<SERVER 语句> ::= SERVER 服务名称 [ ALIAS 别名 ] 类名 [( 参数 1, 参数 2 ... )]
```

注：当同构方法中参数有 `common.data.BaseLoader` 或 `db.jdbc.DBAdapter` 时，则它应是最后的一个参数！

而且在执行 SERVER 语句时不用包含这两个参数。

SERVICE 语法:

```
<SERVICE 语句> ::= SERVICEC 服务名称 [ 方法名称 ] [( 参数 1, 参数 2 ... )]
```

注：当方法名为 `service` 时，则可以省去 ‘方法名称’ ！

示例 1：一个提供口令的微服务类：

```
Package pass;
```

```
public class Service {  
  
    private static final String SERVICE_NAME = "pass_service";  
  
    public synchronized Object service () {  
        System.out.println (SERVICE_NAME+" service() .....");  
        return service (6);  
    }  
  
    public synchronized Object service (int length)  
    {  
        return Factory.word (length);  
    }  
  
    public synchronized Object service (int length, int type)  
    {  
        return Factory.word (length, type);  
    }  
}
```



使用下列语句激活该服务：

```
SERVER pass_service pass.Service
```

之后就可以通过 S E R V I C E 语句调用方法服务：

- 1 . 产生长度为 6 的口令： SERVICE pass\_service;
- 2 . 产生长度为 8 的口令： SERVICE pass\_service (8);

示例 2：一个提供身份证号中出生地信息的类：

```
package pcode6;
```

```
import java.io.InputStream;
```

```
import common.TreeNode;
```

```
import common.data.BaseLoader;
```

```
import db.jdbc.DBAdapter;
```

```
public class Service
```

```
{
```

```
    private static final String SERVICE_NAME = "pcode6_service";
```

```
    private static final String filename = "PCode6.txt";
```

```
    private TreeNode pcode6_node = null;
```

```
    public Service (BaseLoader loader) throws Exception
```

```
    {
```

```
        InputStream is = loader.getResourceAsStream (filename);
```

```
        pcode6_node = Node.parse (is);
```

```
        pcode6_node.open_all ();
```

```
    }
```

```
    public Service (DBAdapter db)
```

```
    {
```

```
        System.out.print ("SERVER "+SERVICE_NAME +" START .....9");
```



```
try {
    InputStream is = (InputStream) db.execute ("GET RESOURCE " + filename + '\');
    pcode6_node = Node.parse (is);
    pcode6_node.open_all ();

    System.out.println (" OK!");
} catch (Exception e) {
    System.out.println (" FAIL!");
    e.printStackTrace();
//    throw e;
}
}

//10199,身份证号地址码汇总,util.tree.Call, SERVICE pcode6_service
public synchronized TreeNode service () {
    return pcode6_node;
}

public synchronized String service (String code6)
{
    return pcode6_node.find (code6).getFullname();
}
}
```

使用下列语句激活该服务：

```
SERVER pcode6_service pcode6.Service
```

之后就可以通过 S E R V I C E 语句调用方法服务:

1. 身份证号地址码汇总： SERVICE pcode6\_service;
2. 某身份证号的前 6 位： SERVICE pcode6\_service ( '130527' );  
返回结果为：[中国/河北省/邢台市/南和县](#)



## 12. Huayisoft DB Server 中的 NEW 语句

### 12.1 new 语句

语法:

```
<new 语句> ::= new <类名称> (参数)
```

示例 :

```
NEW java.lang.String( '123' );
```

结果 : 123

## 13. Huayisoft DB Server 中的 FTP 语句

### 13.1 FTP 语句

语法:

```
<FTP 语句> ::= FTP <ftp 命令> [ ftp 参数 ]
```

ftp 命令 :

LIST、NLST、RETR、STOR、STOU、APPE、REST、RNFR、MVTO、DELE、RMD、MKD、FRSH、MDTM、SIZE

示例 :

假设有一句柄为 : D B A d a p t e r db , 路径名为 : huayi , 文件名为 : readme.txt , 则 :

- 1 . 文件名列表 : (String[])db.execute ("FTP NLST huayi");
- 2 . 详细文件名列表: (String[])db.execute ("FTP LIST huayi");
- 3 . 读取文件 : (InputStream)db.execute ("FTP RETR readme.txt");
- 4 . 存放文件 : (OutputStream)db.execute ("FTP STOR readme.txt ");

更多功能请参考已封装文件类 : db.jdbc.RemoteFile

另外请参考 : common.LocaleFile 及 common.FileNode



## 14. Huayisoft DB Server 中的 EXECUTE 语句

语法:

```
<EXECUTE 语句> ::= EXECUTE LOGIN
    | EXECUTE BATCH SQL FROM 文本文件名
    | EXECUTE BATCH SQL STATEMENT <语句> [; <语句> ] ...
    | EXECUTE CLASS [SHARE] 类名 [AS 别名] [方法名 [字符串参数] ]
```

### 14.1 执行登录获取用户的应用程序

```
EXECUTE LOGIN
```

### 14.2 执行批语句

执行文本文件中的批语句

```
EXECUTE BATCH SQL FROM SQL.txt
```

执行批语句

```
EXECUTE BATCH SQL STATEMENT
    SET CATALOG AAA;
    SET SCHEMA SAMPLE;
```

### 14.3 执行服务器端的应用程序的方法

```
EXECUTE CLASS hy.app.pay.server.Server getBank
```

- 注：1. 类名为不带.class 后缀的全称类名,如:hy.app.pay.server.Server;  
2. 带 SHARE 时将指示服务器把执行的应用程序放到共享集中,以便提高速度.



## 15. Huayisoft DB Server 中的 HELP 语句

### 15.1 HELP 语句

语法:

```
<HELP 语句> ::= HELP { [ 语句名称 ] | [ BNF 名称 ] }
```

示例:

```
dbp:> help;
```

```
dbp:> help insert;
```

```
dbp:> help <set statement>;
```

## 16. Huayisoft DB Server 的编程

Huayisoft DB Server 是用纯 Java 开发的,提供了 JDBC 编程接口。

### 16.1 JDBC 应用编程接口

**示例 1** ( 下列内容可使用任何文本编辑器进行编辑, 编辑好后保存到 DB1.java 文件中 ):

```
/**
```

```
@(#)DB1.java 1.2 2010.10.18
```

假设 db\_jdbc.jar 及 DB1.java 文件都在 c:\sample 路径中,则可使用下列命令进行编译:

```
C:\sample>javac -cp db_jdbc.jar DB1.java
```

编译好后, 则可使用下列命令执行:

```
C:\sample>java -cp db_jdbc.jar;c:\sample DB1
```

注: 后面的 c:\sample 不能省去, 即使是在此路径中执行也不行, 即:

```
C:\sample>java -cp db.jdbc.jar DB1
```

```
则会出现错误: 提示没有发现类 DB1
```

```
**/
```





```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

import db.jdbc.Driver;

public class DB1
{
    public static void main (String args [])
    {
        String driverName = "db.jdbc.Driver";
        String url = "jdbc://localhost/AAA";
        String username = "test";
        String password = "test";
        String sql = "SELECT * FROM SAMPLE.OFFICES";

        try {
            Driver driver = (Driver)Class.forName (driverName).newInstance();
            DriverManager.registerDriver (driver);
            Connection con = DriverManager.getConnection (url, username, password);
            Statement statement = con.createStatement();

            java.sql.ResultSet set = statement.executeQuery (sql);

            set.beforeFirst();

            while (set.next()) {

                System.out.println (set.getString(1));

            }

            statement.close();
            con.close ();
        } catch(Exception e) {
            e.printStackTrace ();
            System.out.println (e.getMessage());
        }
    }
}
```



```
        System.exit(-1);
    }
}
}
```

**示例 2** ( 下列内容可使用任何文本编辑器进行编辑，编辑好后保存到 DB2.java 文件中 ):

```
/**
 * @(#)DB2.java 1.2 2010.10.18
 * 编译、执行方法同上！
```

在本示例中，获得了 Huayisoft DB Server 应用编程接口 DBAdapter 后，保持并利用这个应用编程接口，就可随时执行 Huayisoft DB Server 语句，并返回需要的数据对象。具体请参考：

documents/index.html 中的 db.jdbc.DBAdapter

```
**/
```

```
import db.jdbc.DBAdapter;
import db.jdbc.ResultSet;
import db.jdbc.User;

public class DB2
{
    public static void main (String args [])
    {
        String host = "localhost";

        String base = "AAA";

        String username = "test";
        String password = "test";

        String sql = "SELECT * FROM SAMPLE.CUSTOMERS";

        try {
            User user = User.parse (host,base,username,password);
            DBAdapter db = DBAdapter.getInstance (user);
```



```
        ResultSet set = db.executeQuery (sql);

        set.beforeFirst();
        while (set.next()) {
            System.out.println (set.getString(1));
        }
    } catch(Exception e) {
        e.printStackTrace ();
    }
}
}
```

### 示例 3 :

```
/**
 * @(#)DB3.java 1.2 2010.10.18
 * 编译、执行方法同上！
```

在本示例中,在 ob 中封装了 Huayisoft DB Server 应用编程接口,可以很方便地执行 Huayisoft DB Server 语句,并返回需要的数据对象。具体请参考：

documents/index.html 中的 db.jdbc.ob

```
**/
```

```
import db.jdbc.ob;
import db.jdbc.ResultSet;
import db.jdbc.User;

public class DB3
{
    public static void main (String args [])
    {
        String host = "localhost";
        String base = "AAA";
        String username = "test";
        String password = "test";
```



```
String sql = "SELECT * FROM SAMPLE.CUSTOMERS";

try {
    User user = User.parse (host,base,username,password);

    ob.register(user);
    ResultSet set = ob.executeQuery (sql);

    set.beforeFirst();
    while (set.next()) {
        System.out.println (set.getString(1));
    }
} catch(Exception e) {
    e.printStackTrace ();
}
}
```

#### 示例 4 :

```
/**
 * @(#)DB4.java 1.2 2016.11.18
 * 编译、执行方法同上！
 */

import db.jdbc.ob;
import db.jdbc.ResultSet;

public class DB4
{
    public static void main (String args [])
    {
        String url = "dbp://test:test@localhost/AAA";
```



```
String sql = "SELECT * FROM SAMPLE.CUSTOMERS";

try {
    ob.register(url);
    ResultSet set = ob.executeQuery(sql);

    set.beforeFirst();
    while (set.next()) {
        System.out.println (set.getString(1));
    }
} catch(Exception e) {
    e.printStackTrace ();
}
}
```

#### 示例 5 :

```
/**
 * @(#)DB5.java 1.2 2016.11.18
 * 编译、执行方法同上！
 */

import db.jdbc.ob;
import db.jdbc.ResultSet;

public class DB5
{
    public static void main (String args [])
    {
        String url = "dbp://test:test@localhost/AAA";
        String sql = "SELECT * FROM SAMPLE.CUSTOMERS";
    }
}
```



```
try {
    ResultSet set = (ResultSet)ob.login_execute (url+'?' +sql);

    set.beforeFirst();
    while (set.next()) {
        System.out.println (set.getString(1));
    }
} catch(Exception e) {
    e.printStackTrace ();
}
}
```

## 16.2 三个超级用户

Huayisoft DB Server 随安装产生了三个超级用户(其信息存放在' db/db\_envron.cert' 文件中), 它们的数据库名都为: \*, 用户名分别为: ROOT\_SCHEMA、INFORMATION\_SCHEMA、DATA\_SCHEMA, 它们将分别对应电脑的根目录、当前目录及当前目录下的 data 目录。

不仅这三个超级用户, 下述的管理员及一般用户的口令(密码)都采用不可逆的形式进行加密存放! 强烈建议分别登录后使用下列语句修改用户名和密码(是的, 用户名和密码都将被修改。因此, 要妥善保存好! ):

```
ALTER ADMINISTRATOR 用户名 '密码'
```

## 16.3 创建第一个数据库及其管理员用户

必须采用上述三个超级用户之一登陆(编程接口或 StartLogin.exe)后, 执行下列命令:

```
CREATE CATALOG IF NOT EXISTS AAA;
SET CATALOG AAA;
CREATE USER IF NOT EXISTS manager '123ABxyz' ;
```



以上创建了一个数据库名为 AAA 的数据库，且其管理员为 manager，创建其它用户则最好用 manager 来登录并创建它们。可以创建多个 CATALOG，每个 CATALOG 都拥有一个独立的用户系统及权限系统。

## 16.4 Huayisoft DB Server 的运行环境

软件环境：Huayisoft DB Server 采用纯 Java 开发，跨平台，可运行于 Windows、Linux、Unix 等等操作系统。

硬件环境：Huayisoft DB Server 可运行于 CPU 为 486、内存为 16MB 以上微机。



## 17. Huayisoft DB Server 的事务处理说明

在传统关系数据库中，事务处理是一个非常重要的概念，也是一个非常复杂的问题，以致于即使是非常有名的关系数据库产品，如果开发人员操作设置不当或简单地完全依赖数据库系统，或许并不能完全有效地解决所有读写冲突问题！

Huayisoft DB Server 在传统事务处理方法上，只是提供了一些简单的事务处理，如：原子读写操作，简单锁，等等。更多的处理方法则由开发者根据具体情况来解决！

这里提供一些建议或方法以供参考：

大化小，复杂化简单（模块化），把两个或两个以上用户对同一表写操作变成一用户对一表写操作！  
如：大型表文件小型化，用户操作分散化：

示例：一个有 300 多个单位的区县的行政工资统发系统。

a),一个简单的设计就是只创建一个表，用一个单位 ID 字段来区分各行政单位，再用时间字段来区分各个月的工资，其余的统统交给数据库来处理，这种设计方法，特别在互联网架构下，可能存在很大的隐患！

b),但如果按月按单位分别创建多个对应表，每个单位再指定特定人员才可录入、修改权限（写操作）（现实情况也往往不许多人修改），就不会产生读写冲突问题了！这样做相对于按时间按单位最大化地对行记录进行了隔离，也有效的把问题控制在最小范围了（一个行政单位内）！也可以任意地读取已形成的以往任何工资表数据！

如果在一个地级市内部署，则通过多个 CATALOG 来解决！

Huayisoft DB Server 不仅可以通过 CREATE TABLE 来创建一个数据表，也可以存取一个数据表，甚至可以复制一个数据表。总之，可以很容易很方便地生成一个数据表，对数据表的个数需要的话可以不作任何限制！

创建一个对应的类，并在相应方法上使用同步 synchronized 字，串行化地解决读写冲突问题，如‘写偏差’问题。

具体部署请查看：SERVER、SERVICE 或 EXECUTE CLASS 等语句！





## 附录 A. 样本数据库

本样本数据库共有 5 张表：

客户 表中每个客户对应一条记录。

定单 表中每个客户订单对应一条记录。

产品 表中每种产品对应一条记录。

销售点 表中每个销售地点对应一条记录。

销售员 表中每名销售员对应一条记录。

这 5 张表可以使用下列语句创建：

```
SET CATALOG AAA;
```

```
CREATE SCHEMA IF NOT EXISTS SAMPLE
```

```
CREATE TABLE IF NOT EXISTS 产品
```

```
(  
    类别 CHAR (6) NOT NULL,  
    代号 CHAR (5) NOT NULL,  
    产品名称 VARCHAR (20) NOT NULL,  
    数量 INTEGER NOT NULL,  
    价格 MONEY NOT NULL,  
    PRIMARY KEY ( 类别,代号)  
)
```

```
CREATE TABLE IF NOT EXISTS 销售点
```

```
(  
    编号 INTEGER NOT NULL PRIMARY KEY,  
    城市名称 VARCHAR(32) NOT NULL,  
    所属区域 VARCHAR(16) NOT NULL,  
    经理 INTEGER REFERENCES 销售员(编号) ON DELETE SET NULL,  
    销售定额 MONEY,
```



销售额 MONEY NOT NULL

)

CREATE TABLE IF NOT EXISTS 销售员

(

编号 INTEGER NOT NULL PRIMARY KEY,

姓名 VARCHAR (8) NOT NULL,

年龄 INTEGER,

销售点 INTEGER REFERENCES 销售点(编号) ON DELETE SET NULL,

聘用日期 DATE NOT NULL,

经理 INTEGER REFERENCES 销售员(编号) ON DELETE SET NULL,

销售定额 MONEY,

销售额 MONEY NOT NULL

)

CREATE TABLE IF NOT EXISTS 客户

(

编号 INTEGER NOT NULL PRIMARY KEY,

单位名称 VARCHAR(32) NOT NULL,

销售员 INTEGER REFERENCES 销售员(编号) ON DELETE SET NULL,

信用额度 MONEY

)

CREATE TABLE IF NOT EXISTS 定单

(

编号 INTEGER NOT NULL PRIMARY KEY,

日期 DATE NOT NULL,

客户 INTEGER NOT NULL REFERENCES 客户(编号) ON DELETE CASCADE,

销售员 INTEGER NOT NULL REFERENCES 销售员(编号) ON DELETE SET NULL,

类别 CHAR (6) NOT NULL,

代号 CHAR (5) NOT NULL,

数量 INTEGER NOT NULL,



```
金额 MONEY NOT NULL,  
CONSTRAINT 产品 FOREIGN KEY (类别,代号) REFERENCES 产品(类别,代号) ON DELETE  
RESTRICT  
);
```

这 5 张表的初始内容如下：

客户表：

编号	单位名称	销售员	信用额度
201	九江华易软件有限公司	109	50000
202	北京首创科技有限公司	106	65000
203	南昌安达日化有限公司	101	50000
204	合肥运捷电子有限公司	103	40000
205	北京紫华国际集团	107	35000
206	天津易大软件有限公司	106	20000
207	武汉光达科技有限公司	104	65000
208	武汉硅科电子厂	108	50000
209	合肥南方进出口有限公司	109	45000
210	北京北方进出口有限公司	103	20000
211	南昌兄弟纺织厂	110	40000
212	武汉兄弟纺织厂	102	55000
213	九江科辉五金厂	104	35000
214	南昌三路硅电子厂	101	30000
215	南昌三好商场	103	50000
216	天津长安古玩大市场	103	65000
217	天津六都文化有限公司	102	25000
218	深圳华南电子集团	108	60000
219	成都民生系统工程集团	105	20000
220	南昌昌盛商务有限公司	109	25000
221	新疆华西葡萄加工集团	106	45000

订单表：



编号	日期	客户	销售员	类别	代号	数量	金额
8161	2009.12.17	213	104	电脑	A02	7	31500
8212	2010.01.11	201	101	手机	B03	35	3745
8189	2010.01.03	207	104	化妆品	F02	6	1458
8251	2010.02.10	218	108	衣服	E01	4	1420
8168	2009.10.12	202	106	手机	B04	34	3978
8236	2010.01.30	205	107	手机	B06	9	22500
8245	2010.02.02	208	106	电脑	A03	10	45000
8163	2009.12.17	203	107	手机	B04	28	3276
8213	2010.01.14	218	108	茶叶	C02	1	652
8258	2010.02.23	212	102	化妆品	F01	10	1480
8197	2010.01.08	211	110	茶叶	C02	1	652
8183	2009.12.27	203	101	手机	B04	6	702
8224	2010.01.20	210	108	衣服	E01	20	7100
8262	2010.02.24	211	110	化妆品	F02	10	2430
8179	2009.10.12	210	103	手机	B01	6	15000
8227	2010.01.22	203	101	手机	B06	54	4104
8207	2010.01.08	208	108	玉器	D04	3	2925
8269	2010.03.02	220	110	玉器	D02	22	31350
8234	2010.01.29	205	107	电脑	A01	8	632
8192	2009.11.04	218	108	手机	B06	10	760
8175	2009.10.12	201	109	电脑	A04	6	2100
8255	2010.02.15	212	106	手机	B07	6	150
8248	2010.02.10	215	103	玉器	D01	2	3750
8193	2009.01.04	216	103	电脑	A01	24	1896
8265	2010.02.27	216	103	衣服	E01	6	2130
8203	2010.01.25	212	102	玉器	D01	3	5625
8249	2010.02.10	218	108	衣服	E01	2	776
8187	2009.12.31	203	101	手机	B02	11	27500
8257	2010.02.18	201	109	手机	B07	24	600
8242	2010.02.02	219	106	电脑	A03	5	22500



## 销售员表：

编号	姓名	年龄	销售点	聘用日期	经理	销售定额	销售额
101	柯长富	37	4	2008.02.12	105	350000	367911
102	王美丽	31	2	2009.10.12	104	300000	392725
103	杨刚	48	5	2006.12.10	108	350000	474050
104	张民山	52	2	2008.07.14	NULL	275000	299912
105	罗明贵	33	3	2007.05.19	104	200000	142594
106	刘应东	45	3	2006.10.20	105	300000	305673
107	夏大宝	41	NULL	2010.01.13	106	NULL	75985
108	李立发	62	5	2009.10.12	104	350000	361865
109	吴小虎	29	3	2007.03.01	105	275000	286775
110	蔡晓	49	1	2008.11.14	108	300000	186042

## 销售点表：

编号	城市名称	所属区域	经理	销售定额	销售额
1	北京	华北	108	300000	186042
2	南昌	华中	104	575000	692637
3	合肥	华中	105	800000	735042
4	武汉	华中	101	350000	367911
5	天津	华北	108	725000	835915

## 产品表：

类别	代号	产品名称	数量	价格
电脑	A01	键盘	210	79
电脑	A02	笔记本电脑	12	4500
电脑	A03	笔记本电脑	12	4500
电脑	A04	硬盘	14	350
手机	B01	手机 B	28	2500
手机	B02	手机 A	25	2750
手机	B03	充电器 B	207	107
手机	B04	充电器 A	139	117
手机	B05	充电器 B	277	55



---

手机	B06	冲电器 A	167	76
手机	B07	手机套	37	25
茶叶	C01	绿茶	0	180
茶叶	C02	龙井	3	652
茶叶	C03	红茶	78	225
玉器	D01	白玉龙摆件	9	1875
玉器	D02	白玉小马	5	1425
玉器	D03	白玉小环	24	250
玉器	D04	白玉小猴	28	975
玉器	D05	白玉小扣	223	54
玉器	D06	白玉小猪	32	475
衣服	E01	男上衣	38	355
衣服	E02	裙子	203	134
衣服	E03	男裤	37	177
化妆品	F01	化妆品 B	115	148
化妆品	F02	化妆品 A	15	243



## 附录 B. Huayisoft DB Server 信息模式

Huayisoft DB Server 通过下列视图来达到自我描述的：

### B.1 超级用户模式 ( 路径 ) : db

#### B.1.1 CATALOGS 视图

每个目录在 CATALOG 视图中有一条记录：

字段名	数据类型	说明
CATALOG_NAME	VARCHAR (32)	目录名
CATALOG_PATH	VARCHAR (128)	目录名对应的路径名
CLASS_ID	INTEGER	目录的默认执行类
ATTRIBUTE	BINARY(16)	目录的属性字
USER_COUNT	INTEGER	目录的最大用户数
SCHEMA_COUNT	INTEGER	目录的最大模式数
TABLE_COUNT	INTEGER	目录的最大表数
DISK_SIZE_M	BIGINT	目录的最大磁盘空间(M)
BALANCE	MONEY	目录的余额数
START_DATE	DATE	目录的起始使用日期
END_DATE	DATE	目录的结束使用日期
OWNER	INTEGER	目录的拥有者
DEFINER	INTEGER	目录的定义者
CHECK	INTEGER	目录的检验和

#### B.1.2 CLASSES 视图

每个 CALSSES 类定义在视图中有一条记录：

字段名	数据类型	说明
CLASS_ID	INTEGER	类编号
CLASS_NAME	VARCHAR (32)	类名称
DEFINER	INTEGER	类的定义者
CHECK_ID	INTEGER	检验和

#### B.1.3 db\_envron.cert 文件

**此文件存放了 Huayisoft DB Server 中的三个超级用户信息 ( 如密码信息等 ) , 因此 , 覆盖此文件**



## 将改变三个超级用户的信息！

### B.1.4 db\_sys.txt 文件

此文件存放 Huayisoft DB Server 系统的一些重要参数：

参数名	值	说明
log	true	是否记录 log 信息
log_cmd	true	是否记录命令信息，前提：log=true
autostart	false	是否在读取 CATALOGS 时，启动每个 CATALOG
make_dir_db	true	是否在创建 CATALOG 时创建一个 db 路径
make_dir_log	true	是否在创建 CATALOG 时创建一个 log 路径 <b>注 1</b>
make_dir_http	true	是否在创建 CATALOG 时创建一个 http 路径 <b>注 2</b>
make_dir_web	false	是否在创建 CATALOG 时创建一个 web 路径
login_timeout	6000	设置登陆时超时时间（秒）
timeout	3999	设置超时时间（秒）
restrict_time	180000	设置登陆失败 3 次后的等待时间（秒），3 分钟
maxconnect	800000	设置最大并发连接数
encoding	GB2312	
catalog_parent_path	data\	设置 CATALOG 存放路径

**注 1**：如果想 log 或 log\_cmd 设置为 true，则此参数必须设置为 true;

**注 2**：Huayisoft DB Server 是可以与 Huayisoft HTTP Server 高度集成的，如每个 CATALOG 同时可以是一个虚拟网站，且 CATALOG 名将自动成为一个对应的二级域名！

### B.1.5 db\_servers.txt 文件

主要用来部署或记录微服务,其中, ID 号为 0 的微服务将在启动各个 CATALOGS 之前启动,且各个 CATALOGS 中的用户在许可下都可以通过 SERVICE 语句来调用！

### B.1.6 db\_threads.txt 文件

存放线程的有关信息。

### B.1.7 resources 路径

存放各个 CATALOGS 共享的有关资源文件,各个 CATALOGS 中的用户在许可下都可以通过 GET RESOURCE 语句来调用！





### B.1.8 services/bin 路径

存放各个 CATALOGS 共享的供 SERVER 或 EXECUTE CLASS 调用的类包，各个 CATALOGS 中的用户在许可下都可以通过 SERVICE 或 EXECUTE CLASS 语句来调用！

## B.2 各个 CATALOG 模式 ( 路径 ): DSCHEMA

### B.2.1 ASSERTIONS 视图

各条声明在 ASSERTIONS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	声明的模式 ID
CONSTRAINT_ID	INT	声明 ID
CONSTRAINT_NAME	VARCHAR(18)	声明名
IS_DEFERRABLE	BOOLEAN	声明是否是可延迟的 (YES/NO)
INITIALLY	BOOLEAN	声明是否延迟 (YES/NO)

### B.2.2 CHECK\_CONSTRAINTS 视图

每条检验约束(检验约束、域检验约束或声明定义)在 CHECK\_CONSTRAINTS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID
CHECK_CLAUSE	CHAR(256)	约束的 SQL 搜索条件的文本定义
CHECK_TYPE	TINYINT	约束类型

### B.2.3 CLASSES 视图

每个 CALSSES 类定义在视图中有一条记录：

字段名	数据类型	说明
CLASS_ID	INTEGER	类编号
CLASS_NAME	VARCHAR(32)	类名称
DEFINER	INTEGER	类的定义者
CHECK_ID	INTEGER	检验和

### B.2.4 CONSTRAINT\_TABLE\_USAGE 视图



每个约束表在 CONSTRAIN\_TABLE\_USAGE 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID
SCHEMA_ID	INT	表的模式 ID
TABEL_ID	INT	该约束的表 ID

### B.2.5 DOMAINS 视图

各个域在 DIMAINS 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	域的模式 ID
DOMAIN_ID	TINYINT	域 ID
DOMAIN_NAME	VARCHAR(18)	域名
DATA_TYPE	SMALLINT	域的基本的 SQL2 数据类型
LENGTH	INTERGER > 0	可变长度字符类型按字符计算的最大长度
SCALE	TINYINT	基于数字数据类型的域的大小
CHAR_SET_ID	TINYINT	域的字符集名
COLLATION_ID	TINYINT	域的校正名
DOMAIN_DEFAULT	VARCHAR(32)	该域默认值的文本表示

### B.2.6 DOMAINS\_COLUMN\_USAGE 视图

使用域的各个字段在 DOMAINS\_COLUMN\_USAGE 视图中有一条记录：

字段名	数据类型	说明
DOMAIN_SCHEMA	INT	域的模式 ID
DOMAIN_ID	INT	域 ID
SCHEMA_ID	INT	字段定义的模式 ID
TABLE_ID	INT	字段的表 ID
COLUMN_NAME	CHAR(18)	字段名

### B.2.7 DOMAINS\_CONSTRAINTS 视图

域上的各个域约束在 DOMAINS CONSTRAINTS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID



CONSTRAINT_NAME	CHAR(18)	约束名
IS_DEFERRABLE	BOOLEAN	
DEFERRED	BOOLEAN	
DOMAIN_SCHEMA	INT	域的模式名
DOMAIN_ID	INT	域名

### B.2.8 INDEXES 视图

每个索引在 INDEXES 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	索引的模式 ID
TABLE_ID	INT	索引表 ID
IS_UNIQUE	BOOLEAN	
INDEX_NAME	CHAR(18)	索引名
SORT_SPEC_LIST	CHAR(36)	索引表达式

### B.2.9 REFERENTIAL\_CONSTRAINTS 视图

每条引用约束在 REFERENTIAL\_CONSTRAINTS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID
REF_COLUMNS	CHAR(120)	被引用的列名
UNIQUE_SCHEMA	INT	被引用约束的模式 ID
UNIQUE_ID	INT	被引用约束 ID
MATCH_OPTION	TINYINT	部分外键字匹配的类型 (NONE/PARTIAL/FULL)
UPDATE_RULE	TINYINT	更新引用约束的规则(CASCADE/SET NULL/SET DEFAULT/NO ACTION)
DELETE_RULE	TINYINT	删除引用约束的规则(CASCADE/SET NULL/SET DEFAULT/NO ACTION)

### B.2.10 SCHEMATA 视图

每个模式在 SCHEMATA 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	模式 ID
SCHEMA_NAME	VARCHAR (32)	模式名



SCHEMA_OWNER	INT	模式的拥有者
SCHAME_DEFINER	INT	模式的定义者

### B.2.11 TABLES 视图

每张表在 TABLES 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	表的模式 ID
TABLE_ID	INT	表 ID
TABLE_NAME	VARCHAR(32)	表名
TABLE_DEFINER	INT	模式的定义者
TABLE_TYPE	VARCHAR(1)	表类型

### B.2.12 TABLE\_CONSTRAINTS 视图

每条表约束在 TABLE\_CONSTRAINTS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID
CONSTRAINT_NAME	VARCHAR(32)	约束名
IS_DEFERRABLE	BOOLEAN	约束是否是可延迟的 (YES/NO)
DEFERRED	BOOLEAN	约束是否延迟(YES/NO)
SCHEMA_ID	INT	表的模式 ID
TABEL_ID	INT	该约束的表 ID
CONSTRAINT_TYPE	TINYINT	约束类型 (UNIQUE/FRIMARY KEY/FOREIGN KEY/CHECK)

### B.2.13 TABLE\_PRIVILEGES 视图

授予用户的表上的各种权限在 TABLE\_PRIVILEGES 视图中有一条记录：

字段名	数据类型	说明
GRANTOR	INT	授权的用户
GRANTEE	INT	被授权的用户
SCHEMA_ID	INT	表的模式 ID
TABLE_ID	INT	表 ID
PRIVILEGE	INT	权限(SELECT/INSERT/DELETE/UPDATE/REFERENCES)
CHECK_ID	INT	检验和



### B.2.14 TRIGGERS 视图

每个触发器在 TRIGGERS 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	触发器的模式 ID
TABLE_ID	INT	触发器的表 ID
TRIGGER_NAME	CHAR(12)	触发器名
time	TINYINT	触发时间 BEFORE/AFTER
event	TINYINT	触发事件 INSERT   DELETE   UPDATE
columns	BINARY(8)	触发列
OLD	TINYINT	旧值类型(ROW/TABLE)
OLD_NAME	CHAR(10)	旧值名
NEW	TINYINT	新值类型(ROW/TABLE)
NEW_NAME	CHAR(10)	新值名
each	TINYINT	每个 ROW   STATEMENT
when	CHAR(128)	触发条件
sql	CHAR(1024)	触发器表达式

### B.2.15 TYPES 视图

每个自定义的数据类型都在 TYPES 视图中有一条字段记录，如：

数据类型名	数据类型	说明
PCODE	BIGINT CLASS db.jdbc.type.PCode	身份证号
IP	BINARY(4) CLASS db.jdbc.type.IP	IP 地址
VERSION	BINARY(4) CLASS db.jdbc.type.Version	版本数据类型

### B.2.16 UNIQUE\_CONSTRAINTS 视图

每条唯一性约束在 UNIQUE\_CONSTRAINTS 视图中有一条记录：

字段名	数据类型	说明
CONSTRAINT_SCHEMA	INT	约束的模式 ID
CONSTRAINT_ID	INT	约束 ID
COLUMN_LIST	CHAR(128)	列名

### B.2.17 USERS 视图

每个用户在 USERS 视图中有一条记录：



字段名	数据类型	说明
USER_ID	INTEGER	用户编号
NAME	VARCHAR (8)	用户的真实名
USERNAME	VARCHAR (12)	用户名
PASSWORD	BINARY(20)	用户密码
SCHEMA_ID	INTEGER	用户的默认模式
CLASS_ID	INTEGER	用户的默认执行类
SERVER	BINARY(4)	用户的相关服务设置
MENUS	VARCHAR(12)	用户的菜单名
STARTDATE	DATE	用户的起始使用日期
ENDDATE	DATE	用户的结束使用日期
DEFINER	INTEGER	用户的定义者

### B.2.18 VIEWS 视图

每张视图在 VIEWS 视图中有一条记录：

字段名	数据类型	说明
SCHEMA_ID	INT	视图的模式 ID
TABLE_ID	INT	视图 ID
VIEW_DEFINITION	VARCHAR(1024)	视图的 SQL SELECT 语句文本
CHECK_OPTION	VARCHAR(8)	视图的检验选项 (CASCADED/LOCAL/NONE)
IS_UPDATABLE	BOOLEAN	视图是否可以更新(YES/NO)

### B.2.19 profile.txt

主要用来部署本 CATALOG 微服务,本 CATALOG 中的用户在许可下都可以通过 SERVICE 语句来调用！

### B.2.20 resources 路径

存放本 CATALOG 的有关资源文件，本 CATALOG 中的用户在许可下都可以通过 GET RESOURCE 语句来调用！

### B.2.21 services/bin 路径

存放本 CATALOG 的供 SERVER 或 EXECUTE CLASS 调用的类包，本 CATALOG 中的用户在许可下都可以通过 SERVICE 或 EXECUTE CLASS 语句来调用！