■文档属性

|  |  |
| --- | --- |
| 标题 | Sybase数据库协议分析 |
| 编号 |  |
| 编写日期 | 2018-5-16 |
| 修改日期 |  |
| 作者 |  |
| 产品名称 |  |
| 产品型号 |  |
| 产品版本号 |  |

文档修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本号 | 修订说明 | 修订人 |
| 2018-5-16 | V1.0 | 创建 | 李蔚恒 |
|  |  |  |  |

## 前言

Sybase数据库使用TDS5.0协议进行通信。TDS协议由Sybase开发，后来微软加入，将其作为SQL Server的通信协议，在TDS4.2之后，微软和Sybase分道扬镳，双方开始各自开发自己的TDS协议，下面的表格是TDS协议的版本及其所支持产品：

|  |  |
| --- | --- |
| TDS Version | Supported Products |
| 4.2 | Sybase SQL Server < 10 and Microsoft SQL Server 6.5 |
| 5.0 | Sybase SQL Server >= 10 |
| 7.0 | Microsoft SQL Server 7.0 |
| 7.1 | Microsoft SQL Server 2000 |
| 7.2 | Microsoft SQL Server 2005 |

表格摘自<http://www.freetds.org/tds.html#examples>

本文档分析所使用的产品为SAP Adaptive Server Enterprise(Sybase ASE 16)，其使用的协议为TDS5.0。

TDS 5.0为闭源协议，Sybase只放出了部分文档，最新版为TDS 5.0 Functional Specification Version 3.8，时间为2006年，因此协议分析比较困难。

## 协议概述

TDS协议格式如下：

|  |  |
| --- | --- |
| head(8字节) | stream(n字节) |

对于head部分，我们可以细分为：

|  |  |  |  |
| --- | --- | --- | --- |
| packet type(1) | last packet indicator(1) | packet size(2) | unknown(4) |

packet type：标识数据包所属的类型，对于TDS 5.0，有以下几种已知类型：

|  |  |
| --- | --- |
| value | type |
| 0x02 | Login packet |
| 0x03 | RPC |
| 0x04 | responses from server |
| 0x06 | cancels |
| 0x07 | Used in Bulk Copy |
| 0x0f | query |

last packet indicator：用于标识次包是否是最后一个包,0x00表示还有更多包,0x01表示是最后一个包。

packet size：标识包的整体大小，大端序

unknown：用途不明，总是为0x00

对于stream部分，有两种表示方式：

1. Token stream:数据由token+data组成，token用来标识数据的功能，data则是具体内容，不同的token值，token data的结构不同
2. Tokenless stream:数据头部没有token值，数据之接存放在当前位置。tokenless data stream只用于login类型的包和bulk copy operation(批量复制操作)中

### Token stream

Tokens是在用户数据部分中发送的单字节标识，它们后面会跟随特定的数据。Token数据是固定长度的数据或者变长的数据。固定长度的token数据没有length字段。Length字段的大小被编码在token值中，有五种可能的token length字段类别，这里列出了它们的比特模式编码：

#### Zero Length - 110xxxxx

这种类型的token没有length字段，token后没有数据。

#### Fixed Length - xx11xxxx

这种类型的token后面跟随1,2,4或8字节的数据，没有length字段，因为长度已经被编码到token值中了。第3第4比特总是为1，第5第6比特标识定长数据的长度。

* xx1100xx indicates 1 byte of data.
* xx1101xx indicates 2 bytes of data.
* xx1110xx indicates 4 bytes of data.
* xx1111xx indicates 8 bytes of data.

#### Variable Length - any other pattern

这种类型的token后面跟随length字段，length字段的大小被编码在token值中。

* 1010xxxx indicates 2 bytes of length. (NOTE: KEY token in this group is a “zero-length”token, there is no length field.)
* 1110xxxx indicates 2 bytes of length. (NOTE: ROW, ALTROW, PARAM tokens are in thisgroup, but are “zero-length” tokens. The length field is absent.)
* 1000xxxx indicates 2 bytes of length.
* 001000xx or 011000xx indicates 4 bytes of length.
* 001001xx, 001010xx, 011001xx, or 011010xx indicates 1 byte of length

#### 2.1.4 Token列表

|  |  |
| --- | --- |
| Name | Value |
| TDS\_VOID | 0x1f |
| TDS\_PARAMFMT2 | 0x20 |
| TDS\_LANGUAGE | 0x21 |
| TDS\_IMAGE | 0x22 |
| TDS\_ORDERBY2 | 0x22 |
| TDS\_TEXT | 0x23 |
| TDS\_CURDECLARE2 | 0x23 |
| TDS\_BLOB | 0x24 |
| TDS\_VARBINARY | 0x25 |
| TDS\_INTN | 0x26 |
| TDS\_VARCHAR | 0x27 |
| TDS\_BINARY | 0x2d |
| TDS\_CHAR | 0x2f |
| TDS\_INT1 | 0x30 |
| TDS\_BIT | 0x32 |
| TDS\_INT2 | 0x34 |
| TDS\_INT4 | 0x38 |
| TDS\_SHORTDATE | 0x3a |
| TDS\_FLT4 | 0x3b |
| TDS\_MONEY | 0x3c |
| TDS\_DATETIME | 0x3d |
| TDS\_FLT8 | 0x3e |
| TDS\_UINT1 | 0x40 |
| TDS\_UINT2 | 0x41 |
| TDS\_UINT4 | 0x42 |
| TDS\_UINT8 | 0x43 |
| TDS\_UINTN | 0x44 |
| TDS\_DEBUGCMD | 0x60 |
| TDS\_ROWFMT2 | 0x61 |
| TDS\_MSG | 0x65 |
| TDS\_SENSITIVITY | 0x67 |
| TDS\_BOUNDARY | 0x68 |
| TDS\_DECN | 0x6a |
| TDS\_NUMN | 0x6c |
| TDS\_FLTN | 0x6d |
| TDS\_MONEYN | 0x6e |
| TDS\_DATETIMEN | 0x6f |
| TDS\_LOGOUT | 0x71 |
| TDS\_OFFSET | 0x78 |
| TDS\_RETURNSTATUS | 0x79 |
| TDS\_SHORTMONEY | 0x7a |
| TDS\_CURCLOSE | 0x80 |
| TDS\_CURDELETE | 0x81 |
| TDS\_CURFETCH | 0x82 |
| TDS\_CURINFO | 0x83 |
| TDS\_CUROPEN | 0x84 |
| TDS\_CURUPDATE | 0x85 |
| TDS\_CURDECLARE | 0x86 |
| TDS\_EVENTNOTICE | 0xa2 |
| TDS\_DYNAMIC2 | 0xa3 |
| TDS\_TABNAME | 0xa4 |
| TDS\_COLINFO | 0xa5 |
| TDS\_OPTIONCMD | 0xa6 |
| TDS\_ALTNAME | 0xa7 |
| TDS\_ALTFMT | 0xa8 |
| TDS\_ORDERBY | 0xa9 |
| TDS\_LOGINACK | 0xad |
| TDS\_CONTROL | 0xae |
| TDS\_LONGCHAR | 0xaf |
| TDS\_KEY | 0xca |
| TDS\_ROW | 0xd1 |
| TDS\_ALTROW | 0xd3 |
| TDS\_PARAMS | 0xd7 |
| TDS\_LONGBINARY | 0xe1 |
| TDS\_CAPABILITY | 0xe2 |
| TDS\_ENVCHANGE | 0xe3 |
| TDS\_EED | 0xe5 |
| TDS\_DBRPC | 0xe6 |
| TDS\_DYNAMIC | 0xe7 |
| TDS\_PARAMFMT | 0xec |
| TDS\_ROWFMT | 0xee |
| TDS\_DONE | 0xfd |
| TDS\_DONEPROC | 0xfe |
| TDS\_DONEINPROC | 0xff |

### 2.2登录协议

客户端向服务器发送type为0x02的登录包，包的数据部分为tokenless stream，实际为一名为login record的结构体：

#define TDS\_MAXNAME 30

#define TDS\_PKTLEN 6

#define TDS\_PROGNLEN 10

typedef struct loginrec

{

TDS\_BYTE lhostname[TDS\_MAXNAME];

TDS\_BYTE lhostnlen;

TDS\_BYTE lusername[TDS\_MAXNAME];

TDS\_BYTE lusernlen;

TDS\_BYTE lpw[TDS\_MAXNAME];

TDS\_BYTE lpwnlen;

TDS\_BYTE lhostproc[TDS\_MAXNAME];

TDS\_BYTE lhplen;

TDS\_BYTE lint2;

TDS\_BYTE lint4;

TDS\_BYTE lchar;

TDS\_BYTE lflt;

TDS\_BYTE ldate;

TDS\_BYTE lusedb;

TDS\_BYTE ldmpld;

TDS\_BYTE linterfacespare;

TDS\_BYTE ltype;

TDS\_BYTE lbufsize[TDS\_NETBUF];

TDS\_BYTE lspare[3];

TDS\_BYTE lappname[TDS\_MAXNAME];

TDS\_BYTE lappnlen;

TDS\_BYTE lservname[TDS\_MAXNAME];

TDS\_BYTE lservnlen;

TDS\_BYTE lrempw[TDS\_RPLEN];

TDS\_BYTE lrempwlen;

TDS\_BYTE ltds[TDS\_VERSIZE];

TDS\_BYTE lprogname[TDS\_PROGNLEN];

TDS\_BYTE lprognlen;

TDS\_BYTE lprogvers[TDS\_VERSIZE];

TDS\_BYTE lnoshort;

TDS\_BYTE lflt4;

TDS\_BYTE ldate4;

TDS\_BYTE llanguage[TDS\_MAXNAME];

TDS\_BYTE llanglen;

TDS\_BYTE lsetlang;

/\*

\*\* The following 13 bytes were used by 1.0 secure servers. Actually 2 bytes in

\*\* the middle are unused. Since we do not support logins to 1.0 secure servers,

\*\* we can re-use these 13 bytes.

\*\* However, non-secure servers, check if the first 2 bytes are non-zero. If they

\*\* are non-zero, they assume that the user want's to login a secure server and

\*\* reject the login.

\*/

TDS\_BYTE loldsecure[TDS\_OLDSECURE];

TDS\_BYTE lseclogin;

TDS\_BYTE lsecbulk;

/\*

\*\* The following 2 fields were added in specification revision 3.2 to support High

\*\* Availability failover. The lhalogin byte and the 6 lhasessionid bytes were taken from

\*\* the lsecspare bytes, The TDS\_SECURE value was reduced from 9 to 2 accordingly.

\*/

TDS\_BYTE lhalogin;

TDS\_BYTE lhasessionid[TDS\_HA];

TDS\_BYTE lsecspare[TDS\_SECURE];

TDS\_BYTE lcharset[TDS\_MAXNAME];

TDS\_BYTE lcharsetlen;

TDS\_BYTE lsetcharset;

TDS\_BYTE lpacketsize[TDS\_PKTLEN];

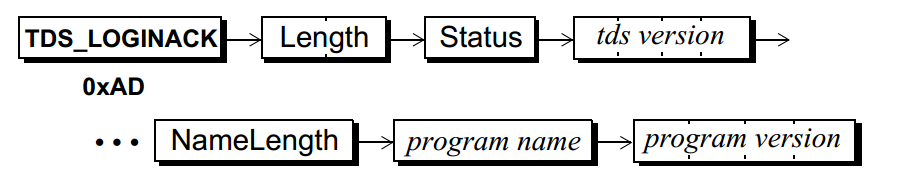
TDS\_BYTE lpacketsizelen;

TDS\_BYTE ldummy[4];

} LOGINREC;

lusername字段为客户端的用户名，lpw为客户端的密码，这两个字段被用于认证。

服务器会向客户端返回包含有token类型为TDS\_LOGINACK(值0xAD)的数据流的数据包来告知客户端登录是否成功。一个类型为TDS\_LOGINACK的token stream结构为：



Status字段用于标识登录的结果：

* TDS\_LOG\_SUCCEED – 5

登录请求成功

* TDS\_LOG\_FAIL – 6

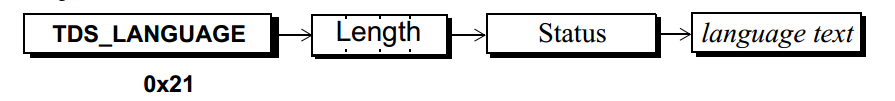
登录请求失败

* TDS\_LOG\_NEGOTIATE – 7

服务器要求客户端在登录之前完成一个协商，登录协商使用TDS\_MSG类型的token stream来完成

### 2.3命令过程

客户端向服务器发送命令请求报文的头部的type字段的值为0x0f，stream部分是类型为TDS\_LANGUAGE封装的token数据流，此token的值为0x21，结构如下所示：

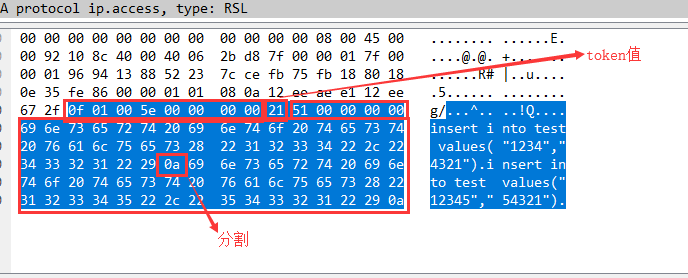


Length字段占4字节，标识剩余token数据的字节数，小端序。

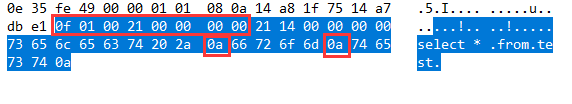
Status占一字节，是一个位掩码，唯一有定义的值是0x01，表明该命令是参数化的，后面跟随有PARAMFMT/PARAM token数据。

Language text是此token封装的命令，一个language text可以存放多个sql命令，用“\n”分割，亲测一个sql命令也可以在Language text中以\n”分割的形式存在。

**“\n”分割多个sql命令：**



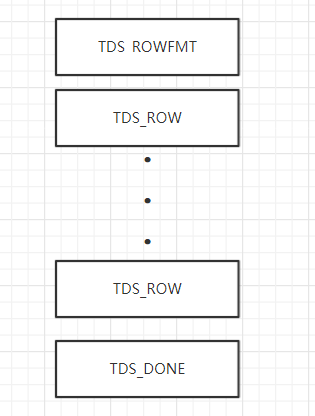
**“\n”分割一条sql语句：**



服务器对客户端的命令做出响应时，包头的type值变成了0x04，数据包的结构变为如下的形式：

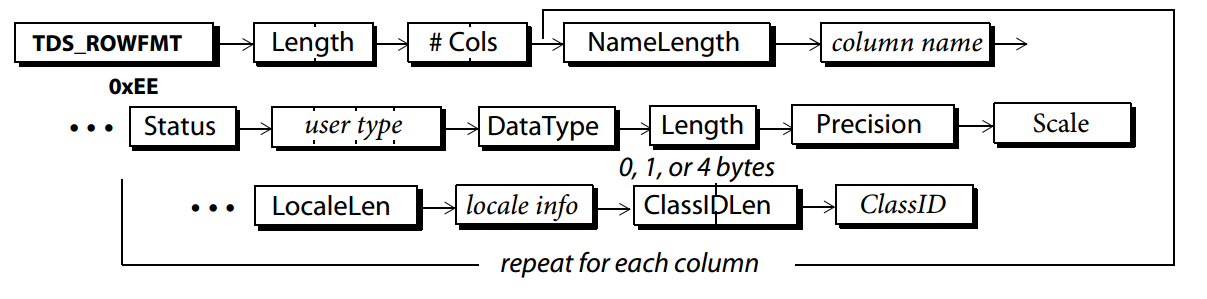
|  |  |  |  |
| --- | --- | --- | --- |
| 数据包头 | Token1 | ... | tokenn |

不同的命令返回不同的token，在最后会以TDS\_DONE作为结尾，对于查询语句返回的结果，token的排列具有如下的特征：



返回结果以TDS\_ROWFMT数据开始，中间为n个TDS\_ROW数据，最后以TDS\_DONE表示结尾。

TDS\_ROWFMT用于描述每一列数据的数据类型、长度和状态，具体结构如下所示：



Length字段为剩余数据的字节数，2字节，小端序

Cols字段描述查询返回的结果一共有几列，2字节，小端序

后面的字段为一个整体，用于描述每一列数据的各种属性，重复的次数为cols的值：

NameLength字段描述列名的长度

Column name为列名

status用于描述数据的任何非数据类型特征

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| TDS\_PARAM\_RETURN | 0x01 | 这是一个返回参数。 它就像通过引用传递的参数 |
| TDS\_PARAM\_COLUMNSTATUS | 0x08 | 此参数在其相应的TDS\_PARAMtoken中将具有columnstatus字段，请注意，当TDS\_DATA\_COLUMNSTATUS功能位关闭时，该位将被认为协议错误 |
| TDS\_PARAM\_NULLALLOWED | 0x20 | 这个参数可以为空 |

user type：这是参数的用户定义的数据类型。 它是一个

有符号的四字节整数。

DataType：这是数据的数据类型。 它是一个单字节的无符号整数。 固定的标准长度（1,2,4或8字节）的数据类型由单个数据类型字节表示，并且后面没有长度参数。 当前不支持text和image数据类型作为参数数据类型。DataType是一个单字节无符号整数。

Length：这是DataType的最大长度（以字节为单位）。 它是一个单字节无符号整数或四字节有符号整数。 长度的大小取决于数据类型。 如果前面的DataType是标准长度的固定长度数据类型，例如int1，int2，datetime等，则没有Length参数

Precision：这是与数字和小数数据类型相关的精度。 只有在数据流中，如果参数是数字或十进制数据类型

Scale：这是与数字和小数数据类型相关的比例。 只有在数据流中，如果参数是数字或小数数据类型。

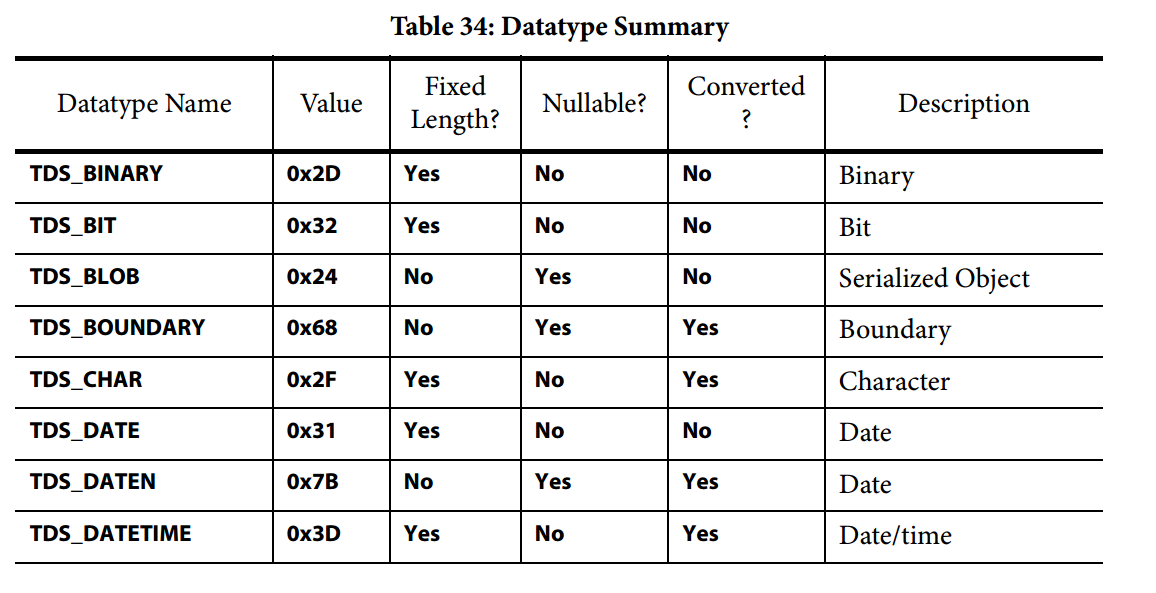
LocaleLen：这是随后的本地化信息的长度（如果有的话）。 它是一个单字节的无符号整数，可以是0.如果长度为0，则不会出现任何本地化信息

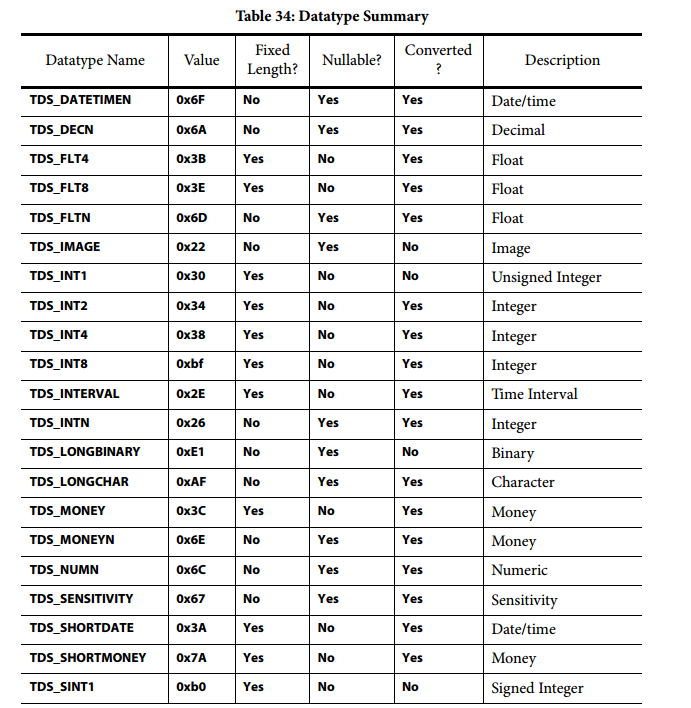
locale info：这是参数的本地化信息。 它是字符串，其长度由LocaleLen给出。

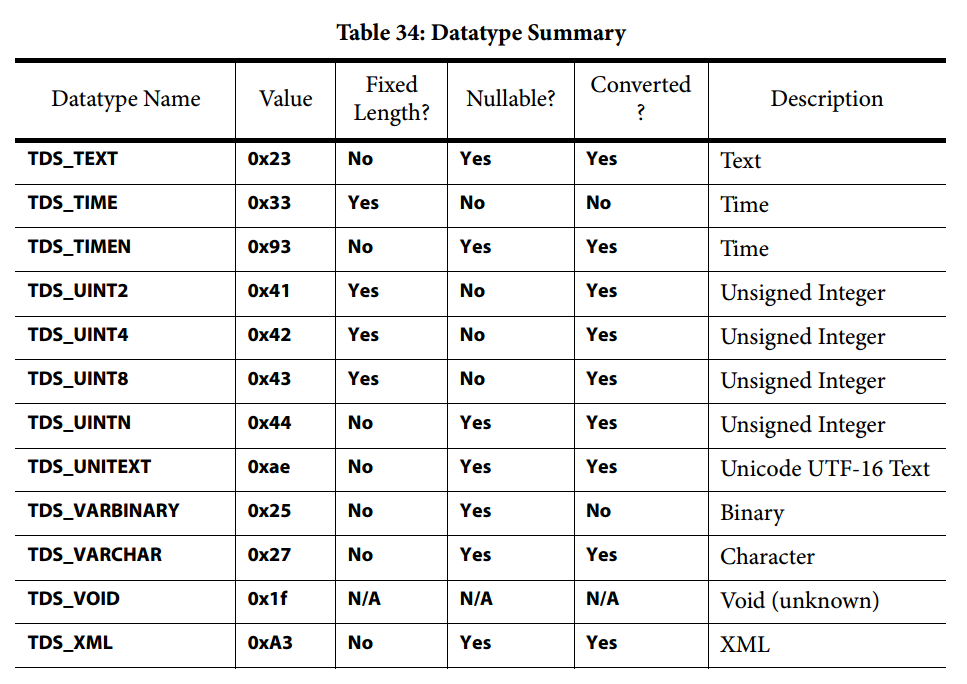
ClassIDLen：这是ClassID的2字节长度值，如果有的话，如下所示。 此长度字段仅在DataType为TDS\_BLOB时才存在。

ClassID：这是BLOB类型的类标识信息。 其以字节为单位的长度由前面的ClassIDLen值给出。 如果ClassIDLen由于不是TDS\_BLOB数据格式而丢失，或者ClassIDLen为0，则该字段不存在。

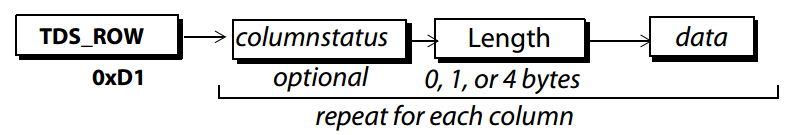
Datatype列表：







TDS\_ROW用于存放查询结果的每一行数据，结构如下：



每行数据的每一列会上图所示的循环部分的结构，此结构重复的次数为查询结果的行数。

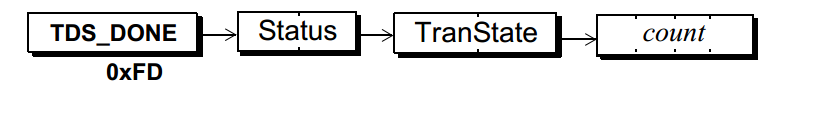
Columnstatus：只有满足以下两个条件时才会出现此字节：第一个是客户端和服务器已通过TDS\_DATA\_COLUMNSTATUS和TDS\_DATA\_NOCOLUMNSTATUS功能位进行协商，以便服务器可以将columnstatus字节发送给客户端。 其次，此列的ROWFMT token具有设置的TDS\_ROW\_COLUMNSTATUS状态位，一般情况下，此字段不出现。

Length：这是以下数据的实际值，而不是最大数据长度（以字节为单位）。 如果以下列数据是标准长度的固定长度数据类型，例如，整数，浮点数，日期时间，则不存在长度参数。 长度可以是一个字节，无符号整数，无符号，二进制整数或带符号的四字节整数。 长度的大小取决于数据的数据类型。

Data: 这是列数据的实际数据。 其长度（如果变量）由前面的Length参数表示。 它采用客户端请求登录记录中指定的格式。 服务器始终进行任何翻译，以便客户端以预期的格式接收数据。

每一行数据由一个TDS\_ROW表示，TDS\_ROW重复的次数为返回结果的行数，在末尾会有TDS\_DONE表示结束。

TDS\_DONE指示命令的完成状态，结果如下：



Status:两字节整数，小端，标识命令完成状态，可能取值:

TDS\_DONE\_FINAL - 0x0000 这是最后一个命令的最终结果。它表明命令已成功完成。

TDS\_DONE\_MORE - 0x0001 此状态指示当前命令具有更多的结果。

TDS\_DONE\_ERROR - 0x0002 表示当前命令发生错误

TDS\_DONE\_INXACT - 0x0004 当前请求中正在进行事务处理

TDS\_DONE\_PROC - 0x0008 TDS\_DONE来自一个存储过程的结果

TDS\_DONE\_COUNT - 0x0010 此状态指示计数参数有效。此位用于区分空计数字段和计数字段的值为0

TDS\_DONE\_ATTN - 0x0020 此TDS\_DONE是确认一个注意命令

TDS\_DONE\_EVENT - 0x0040 TDS\_DONE作为事件通知的一部分而生成

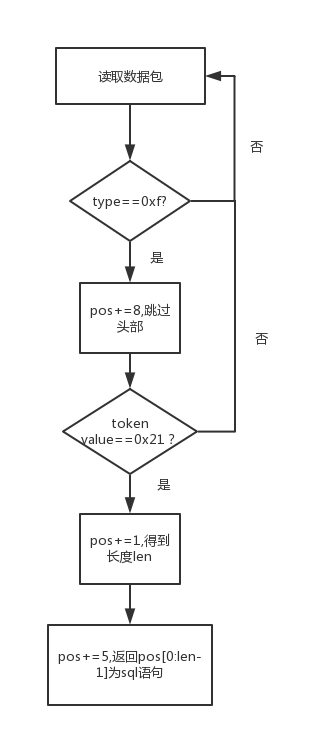
TranState：两字节整数，它指示此连接上的事务的当前状态

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| **TDS\_NOT\_IN\_TRAN** | 0 | 当前不在事务中 |
| **TDS\_TRAN\_SUCCEED** | 1 | 请求引起的事务成功完成. |
| **TDS\_TRAN\_PROGRESS** | 2 | 在这个对话框中，事务仍在进行中. |
| **TDS\_STMT\_ABORT** | 3 | 请求导致语句中止发生 |
| **TDS\_TRAN\_ABORT** | 4 | 请求导致事务中止 |

Count：4字节整数，如果Status设置了，count用来指示当前命令影响的行数

### 2.4从客户端请求报文提取sql语句

根据之前的分析，我们可以得到一个sql提取的流程图：



### 2.5从服务器响应重组查询结果

