Claris FileMaker

SQL 参考

Claris

Claris International Inc.
One Apple Park Way
Cupertino, CA 95014

版本：2022 年 2 月

目录

第 1 章
简介

作为数据库开发人员，您可以在不具备任何 SQL 知识的情况下使用 Claris® FileMaker Pro® 创建数据库解决方案。但是，如果您具备一些 SQL 知识，就可以使用 FileMaker Pro 数据库文件作为 ODBC 或 JDBC 数据源，并与其他使用 ODBC 或 JDBC 的应用程序共享数据。您还可以使用 FileMaker Pro ExecuteSQL 函数从 FileMaker Pro 数据库中表摹本检索数据。

本参考介绍由 Claris FileMaker® 软件支持的 SQL 语句和标准。FileMaker ODBC 和 JDBC 客户端驱动程序支持在本参考中介绍的所有 SQL 语句。FileMaker Pro ExecuteSQL 函数只支持 SELECT 语句。

关于本参考

- 有关在早期 FileMaker Pro 版本上使用 ODBC 和 JDBC 的信息，请参阅产品文档中心。
- 本参考假定您熟悉使用 FileMaker Pro 函数、编码 ODBC 和 JDBC 应用程序和构建 SQL 查询的基础知识。有关上述主题的更多信息，请参阅第三方书籍。

关于 SQL

SQL 又称结构化查询语言，是一种设计用于从关系数据库查询数据的编程语言。用于查询数据库的主语句为 SELECT 语句。

除了查询数据库的语言之外，SQL 提供执行数据操作的语句，这些语句允许您添加、更新和删除数据。

SQL 还提供执行数据定义的语句。利用这些语句可以创建并修改表和索引。

第 2 章，“支持的标准”中介绍了 FileMaker 软件支持的 SQL 语句和标准。

将 FileMaker Pro 数据库用作数据源

将 FileMaker Pro 数据库托管为 ODBC 或JDBC 数据源时，FileMaker 数据可以与兼容 ODBC 和 JDBC 的应用程序共享。应用程序使用 FileMaker 客户端驱动程序连接到 FileMaker 数据源，使用 ODBC 或 JDBC 构建和执行 SQL 查询，并处理从 FileMaker Pro 数据库解决方案检索的数据。

有关如何将 FileMaker 软件用作 ODBC 和 JDBC 应用程序的数据源的更多信息，请参阅《FileMaker ODBC 和 JDBC 指南》。

FileMaker ODBC 和 JDBC 客户端驱动程序支持在本参考中介绍的所有 SQL 语句。

使用 ExecuteSQL 函数

利用 FileMaker Pro ExecuteSQL 函数可从关系图（但与任何定义的关系无关）中命名的表摹本中检索数据。在不创建表联接或表之间的任何关系的情况下，可以从多个表中检索数据。在某些情况下，您可以通过使用 ExecuteSQL 函数来降低关系图的复杂性。

您使用 ExecuteSQL 函数查询的字段不必处于任何布局上，因此您可以使用 ExecuteSQL 函数检索数据而不需要任何布局上下文。由于上下文的独立性，所以在脚本中使用 ExecuteSQL 函数可以提高脚本的可移植性。您可以在可指定计算的任意位置（包括图表和报表）使用 ExecuteSQL 函数。

ExecuteSQL 函数只支持 SELECT 语句，如第 8 页上的“SELECT 语句”。部分所述。

另外，ExecuteSQL 函数仅接受带有大括号 ({}) 的 SQL-92 语法 ISO 日期和时间格式。ExecuteSQL 函数不接受用大括号括起的 ODBC/JDBC 格式日期、时间和时间戳常量。

有关 ExecuteSQL 函数语法和使用的信息，请参阅《FileMaker Pro 帮助》。

第 2 章
支持的标准

使用 FileMaker ODBC 和 JDBC 客户端驱动程序可以从与 ODBC 或 JDBC 兼容的应用程序访问 FileMaker Pro 数据库解决方案。FileMaker Pro 数据库解决方案可以由 FileMaker Pro 或 Claris FileMaker Server® 托管。

- ODBC 客户端驱动程序支持 ODBC 3.0 级别 1。
- JDBC 客户端驱动程序为 JDBC 3.0 规范提供部分支持。
- ODBC 和 JDBC 客户端驱动程序都支持 SQL-92 条目分类一致性，以及一些 SQL-92 中间功能。

支持 Unicode 字符

ODBC 和 JDBC 客户端驱动程序支持 Unicode API。但是，如果要创建使用客户端驱动程序的自定义应用程序，请对字段名、表名称和文件名使用 ASCII（以防使用非 Unicode 查询工具或应用程序）。

说明　要插入和检索 Unicode 数据，使用 `SQL_C_WCHAR`。

SQL 语句

ODBC 和 JDBC 客户端驱动程序提供对以下 SQL 语句的支持：

- SELECT（第 8 页）
- DELETE（第 17 页）
- INSERT（第 17 页）
- UPDATE（第 19 页）
- CREATE TABLE（第 20 页）
- TRUNCATE TABLE（第 22 页）
- ALTER TABLE（第 22 页）
- CREATE INDEX（第 22 页）
- DROP INDEX（第 23 页）

客户端驱动程序还支持映射到 ODBC SQL 和 JDBC SQL 数据类型的 FileMaker 数据类型。有关数据类型转换，请参阅《FileMaker ODBC 和 JDBC 指南》。有关构建 SQL 查询的更多信息，请参阅第三方书籍。

说明　ODBC 和 JDBC 客户端驱动程序不支持 FileMaker Pro 入口。

SELECT 语句

使用 SELECT 语句指定您要请求哪些列。SELECT 语句后跟您要检索的列表达式（与列名称类似，例如 last_name）。表达式可以包括数学运算或字符串操作（例如，SALARY * 1.05）。

SELECT 语句可以使用各种子句：

```
SELECT [DISTINCT] {* | column_expression [[AS] column_alias],...}
FROM table_name [table_alias], ...
[ WHERE expr1 rel_operator expr2 ]
[ GROUP BY {column_expression, ...}]
[ HAVING expr1 rel_operator expr2 ]
[ UNION [ALL] (SELECT...)]
[ ORDER BY {sort_expression [DESC | ASC]}, ...]
[ OFFSET n {ROWS | ROW} ]
FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } {ONLY | WITH TIES } ]
[ FOR UPDATE [OF {column_expression, ...}]]
```

括号中的项目是可选的。

column_alias 可用于为列提供更具描述性的名称，或缩写较长的列名称。

示例

将别名 department 分配给列 dept。

```
SELECT dept AS department FROM emp
```

字段名可以以表名称或表别名为前缀。例如，EMP.LAST_NAME 或 E.LAST_NAME，其中 E 是表 EMP 的别名。

DISTINCT 运算符可以位于第一个列表达式之前。此运算符消除了查询结果中的重复行。

示例

```
SELECT DISTINCT dept FROM emp
```

SQL 子句

ODBC 和 JDBC 客户端驱动程序提供对以下 SQL 子句的支持：

使用此 SQL 子句	目的
FROM（第 9 页）	指示在 SELECT 语句中使用哪些表。
WHERE（第 11 页）	指定要检索的记录必须符合的条件（类似于 FileMaker Pro 的查找请求）。
GROUP BY（第 11 页）	指定返回的值应按其分组的一个或多个字段的名称。此子句用于通过为每组返回一行返回一组聚合值（类似于 FileMaker Pro 的小计）。
HAVING（第 12 页）	为记录组指定条件（例如，仅显示薪金合计大于 $200,000 的部门）。
UNION（第 12 页）	将两个或多个 SELECT 语句的结果合并为一个结果。
ORDER BY（第 13 页）	指示排序记录的方式。
OFFSET（第 13 页）	说明在开始检索行之前要跳过的行数。
FETCH FIRST（第 13 页）	指定要检索的行数。返回不超过指定值的行数，虽然查询生成的行数少于指定的行数时可能会返回更少的行。
FOR UPDATE（第 14 页）	通过 SQL 光标执行定位更新或定位删除。

说明 如果尝试从不含列的表中检索数据，则 SELECT 语句不会返回任何内容。

FROM 子句

FROM 子句指示在 SELECT 语句中使用的表。格式为：

```
FROM table_name [table_alias] [, table_name [table_alias]]
```

table_name 是当前数据库中表的名称。表名称必须以字母字符开头。如果表名称以非字母字符开头，则用双引号将其括起来（带引号的标识符）。

table_alias 可用于为表提供更具描述性的名称，以缩写较长的表名称，或在查询中（例如，在自身连接中）多次包括同一个表。

字段名以字母字符开头。如果字段名以非字母字符开头，则用双引号将其括起来（带引号的标识符）。

示例

名为 _LASTNAME 的字段的 ExecuteSQL 语句为：

```
SELECT "_LASTNAME" from emp
```

字段名可以以表名称或表别名为前缀。

示例

如果表规范为 `FROM employee E`，则可将 `LAST_NAME` 字段表示为 `E.LAST_NAME`。
如果 `SELECT` 语句将表加入到本身中，必须使用表别名。

```
SELECT * FROM employee E, employee F WHERE E.manager_id = F.employee_id
```

等号 (=) 仅包括结果中的匹配行。

如果您要连接多个表，并且要放弃在两个源表中没有对应行的所有行，则可以使用 `INNER JOIN`。

示例

```
SELECT *
   FROM Salespeople INNER JOIN Sales_Data
   ON Salespeople.Salesperson_ID = Sales_Data.Salesperson_ID
```

如果您要连接两个表，但不想放弃第一个表（“左侧”表）的行，可以使用 `LEFT OUTER JOIN`。

示例

```
SELECT *
   FROM Salespeople LEFT OUTER JOIN Sales_Data
   ON Salespeople.Salesperson_ID = Sales_Data.Salesperson_ID
```

“Salespeople” 表的每一行都将显示在连接的表中。

注释

- 当前不支持 `RIGHT OUTER JOIN`。
- 当前不支持 `FULL OUTER JOIN`。

WHERE 子句

WHERE 子句指定要检索的记录必须符合的条件。WHERE 子句包括以下形式的条件：

```
WHERE expr1 rel_operator expr2
```

expr1 和 expr2 可以是字段名称、常量值或表达式。

rel_operator 是连接两个表达式的关系运算符。

示例

检索薪金为 $20,000 或更多的员工的姓名。

```
SELECT last_name,first_name FROM emp WHERE salary >= 20000
```

WHERE 子句还可以使用诸如下列的表达式：

```
WHERE expr1 IS NULL
WHERE NOT expr2
```

说明 如果在 SELECT（投影）列表中使用完全限定的名称，还必须在相关的 WHERE 子句中使用完全限制定的名称。

GROUP BY 子句

GROUP BY 子句指定返回的值应按其分组的一个或多个字段的名称。此子句用于返回一组聚合值。它具有以下格式：

```
GROUP BY 列
```

GROUP BY 子句的范围是 FROM 子句的表达式。因此，columns 指定的列表达式必须来自 FROM 子句中指定的表。列表达式可以是以逗号分隔的数据库表的一个或多个字段的名称。

示例

计算每个部门的薪金总额。

```
SELECT dept_id, SUM (salary) FROM emp GROUP BY dept_id
```

此语句为每个不重复的部门 ID 返回一行。每行中包括部门 ID 和部门中各员工薪金的总和。

HAVING 子句

利用 HAVING 子句可以为记录组指定条件（例如，仅显示薪金合计大于 $200,000 的部门）。它具有以下格式：

```
HAVING expr1 rel_operator expr2
```

expr1 和 expr2 可以是字段名称、常量值或表达式。这些表达式不必匹配 SELECT 子句中的列表达式。

rel_operator 是链接两个表达式的关系运算符。

示例

只返回薪金总额大于 $200,000 的部门：

```
SELECT dept_id, SUM (salary) FROM emp
   GROUP BY dept_id HAVING SUM (salary) > 200000
```

UNION 运算符

UNION 运算符将两个或多个 SELECT 语句的结果合并为一个结果。这一结果是 SELECT 语句返回的所有记录。默认情况下，不返回重复的记录。要返回重复的记录，使用 ALL 关键字 (UNION ALL)。格式为：

```
SELECT statement UNION [ALL] SELECT statement
```

使用 UNION 运算符时，每个 SELECT 语句的选择列表必须具有相同数量的列表达式，以及相同的数据类型，并且必须按相同的顺序指定。

示例

```
SELECT last_name, salary, hire_date FROM emp UNION SELECT name, pay,
birth_date FROM person
```

因为列表达式的数据类型不同（EMP 的 SALARY 具有不同于 RAISES 的 LAST_NAME 的数据类型），所以以下示例无效。本示例中每个 SELECT 语句都有相同数目的列表达式，但这些表达式按数据类型并非处于相同顺序。

示例

```
SELECT last_name, salary FROM emp UNION SELECT salary, last_name FROM raises
```

ORDER BY 子句

ORDER BY 子句指示如何排序记录。如果 SELECT 语句不包含 ORDER BY 子句，记录可能会按任意顺序返回。

格式为：

```
ORDER BY {sort_expression [DESC | ASC]}, ...
```

sort_expression 可以是字段名或要使用的列表达式的位置编号。默认为执行升序 (ASC) 排序。

示例

依次按 last_name 和 first_name 排序。

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY last_name, first_name
```

第二个示例使用位置编号 2 和 3 来获取与明确指定 last_name 和 first_name 的前一个示例相同的顺序。

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY 2,3
```

说明 FileMaker Server 使用 Unicode 二进制排序顺序，它不同于 FileMaker Pro 中或使用默认中性语言排序顺序的语言排序。

OFFSET 和 FETCH FIRST 子句

OFFSET 和 FETCH FIRST 子句用于在结果集中返回从特定起点开始的指定范围的行。通过限制从较大结果集中检索的行数的功能，可以“逐页”浏览数据，并且提高效率。

OFFSET 子句指示在开始返回数据之前要跳过的行数。如果 OFFSET 子句未在 SELECT 语句中使用，则起始行为 0。FETCH FIRST 子句以大于或等于 1 的无符号整数或者百分比的形式指示从 OFFSET 子句中指示的起点开始要返回的行数。如果同时在 SELECT 语句中使用 OFFSET 和 FETCH FIRST，应首先使用 OFFSET 子句。

OFFSET 和 FETCH FIRST 子句在子查询中不受支持。

OFFSET 格式

OFFSET 格式为：

```
OFFSET n {ROWS | ROW} ]
```

n 是无符号整数。如果 n 大于结果集中返回的行数，则不返回任何内容，并且不显示错误消息。

ROWS 与 ROW 相同。

FETCH FIRST 格式

FETCH FIRST 格式为：

```
FETCH FIRST [ n [ PERCENT ] ] { ROWS | ROW } {ONLY | WITH TIES } ]
```

n 是要返回的行数。如果省略 n，则默认值为 1。

n 是大于或等于 1 的无符号整数，除非其后面是 PERCENT。如果 n 后面是 PERCENT，则值是正小数值或者是无符号整数。

ROWS 与 ROW 相同。

WITH TIES 必须与 ORDER BY 子句一起使用。

因为还会返回对等行（根据 ORDER BY 子句相互间无区别的行），所以 WITH TIES 允许返回的行数要多于在 FETCH 计数值中所指定的行数。

示例

返回依次按 last_name 和 first_name 排序的结果集中第 26 行的信息。

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY last_name, first_name
OFFSET 25 ROWS
```

指定您需要只返回 10 行。

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY last_name, first_name
OFFSET 25 ROWS FETCH FIRST 10 ROWS ONLY
```

返回 10 行及其对等行（根据 ORDER BY 子句相互间无区别的行）。

```
SELECT emp_id, last_name, first_name FROM emp ORDER BY last_name, first_name
OFFSET 25 ROWS FETCH FIRST 10 ROWS WITH TIES
```

FOR UPDATE 子句

FOR UPDATE 子句通过 SQL 光标锁定记录来进行定位更新或定位删除。格式为：

```
FOR UPDATE [OF column_expressions]
```

column_expressions 是您要更新的数据库表中的字段名列表，由逗号分隔。column_expressions 是可选的，可以忽略。

示例

返回员工数据库中 SALARY 字段值大于 $20,000 的所有记录。

```
SELECT * FROM emp WHERE salary > 20000
   FOR UPDATE OF last_name, first_name, salary
```

在获取每个记录时，会对其进行锁定。如果记录已更新或删除，在您提交更改之前会保持锁定。否则，在您获取下一个记录时释放锁定。

示例

使用	示例 SQL
文本常量	`SELECT 'CatDog' FROM Salespeople`
数字常量	`SELECT 999 FROM Salespeople`
日期常量	`SELECT DATE '2021-06-05' FROM Salespeople`
时间常量	`SELECT TIME '02:49:03' FROM Salespeople`
时间戳常量	`SELECT TIMESTAMP '2021-06-05 02:49:03' FROM Salespeople`
文本列	`SELECT Company_Name FROM Sales_Data` `SELECT DISTINCT Company_Name FROM Sales_Data`
数字列	`SELECT Amount FROM Sales_Data` `SELECT DISTINCT Amount FROM Sales_Data`
日期列	`SELECT Date_Sold FROM Sales_Data` `SELECT DISTINCT Date_Sold FROM Sales_Data`
时间列	`SELECT Time_Sold FROM Sales_Data` `SELECT DISTINCT Time_Sold FROM Sales_Data`
时间戳列	`SELECT Timestamp_Sold FROM Sales_Data` `SELECT DISTINCT Timestamp_Sold FROM Sales_Data`
BLOB[a] 列	`SELECT Company_Brochures FROM Sales_Data` `SELECT GETAS(Company_Logo, 'JPEG') FROM Sales_Data`
通配符 *	`SELECT * FROM Salespeople` `SELECT DISTINCT * FROM Salespeople`

a. BLOB 是 FileMaker Pro 数据库文件容器字段。

示例的注释

`column` 是对 FileMaker Pro 数据库文件中字段的引用。（字段可以包含许多非重复值。）

星号 (*) 通配符是“所有内容”的简略表示。对于示例 `SELECT * FROM Salespeople`，结果是 `Salespeople` 表中的所有列。对于示例 `SELECT DISTINCT * FROM Salespeople`，结果是 `Salespeople` 表中的所有唯一行（无重复项）。

- FileMaker 软件没有存储空字符串的数据，因此以下查询始终不返回记录：

  ```
  SELECT * FROM test WHERE c =''
  SELECT * FROM test WHERE c <>''
  ```

- 如果将 `SELECT` 与二进制数据一起使用，则必须使用 `GetAs()` 函数来指定要返回的流。有关更多信息，请参阅“检索容器字段的内容：CAST() 函数和 GetAs() 函数”一节。

检索容器字段的内容：CAST() 函数和 GetAs() 函数

您可以从容器字段中检索文件引用信息、二进制数据或特定文件类型的数据。

- 要从容器字段检索文件引用信息，例如某个文件的文件路径、图片或 QuickTime 电影，可以将 `CAST()` 函数与 `SELECT` 语句一起使用。
- 如果文件数据或 JPEG 二进制数据存在，则具有 `GetAS(field name, 'JPEG')` 的 `SELECT` 语句会检索二进制形式的数据；否则具有字段名称的 `SELECT` 语句会返回 `NULL`。

示例

使用 `CAST()` 函数及 `SELECT` 语句来检索文件引用信息。

```
SELECT CAST(Company_Brochures AS VARCHAR) FROM Sales_Data
```

在本示例中，如果您：

- 使用 FileMaker Pro 将文件插入容器字段，但仅存储文件的引用，`SELECT` 语句会以类型 `SQL_VARCHAR` 检索文件引用信息。
- 使用 FileMaker Pro 将文件的内容插入容器字段，`SELECT` 语句会检索文件的名称。
- 将文件从其他应用程序导入容器字段，`SELECT` 语句会显示 “?”（文件在 FileMaker Pro 中显示为 **Untitled.dat**）。

您可以使用 `SELECT` 语句和 `GetAs()` 函数通过以下方法检索二进制形式的数据：

- 使用 `GetAs()` 函数及 `DEFAULT` 选项时，检索的是容器的默认数据流，不需要明确定义流类型。

  #### 示例

  ```
  SELECT GetAs(Company_Brochures, DEFAULT) FROM Sales_Data
  ```

- 要从容器检索单个流类型，根据在 FileMaker Pro 中将数据插入容器字段的方式，将 `GetAs()` 函数与文件的类型一起使用。

  #### 示例

  如果使用“**插入**”>“**文件**”命令插入数据，则在 `GetAs()` 函数中指定 `'FILE'`。

  ```
  SELECT GetAs(Company_Brochures, 'FILE') FROM Sales_Data
  ```

  #### 示例

  如果使用“**插入**”>“**图片**”命令插入数据，从剪贴板拖放或粘贴，指定在下表中列出的文件类型之一，例如 `'JPEG'`。

  ```
  SELECT GetAs(Company_Logo, 'JPEG') FROM Company_Icons
  ```

文件类型	说明
'GIFf'	图形交换格式
'JPEG'	摄影图像
'TIFF'	数字图像的光栅文件格式
'PDF'	可移植文档格式
'PNGf'	位图图像格式

DELETE 语句

使用 DELETE 语句删除数据库表的记录。DELETE 语句的格式为：

```
DELETE FROM table_name [ WHERE { conditions } ]
```

说明 WHERE 子句确定要删除哪些记录。如果未包括 WHERE 关键字，会删除表中的所有记录（但会保留表）。

示例

从 emp 表删除一条记录。

```
DELETE FROM emp WHERE emp_id = 'E10001'
```

每个 DELETE 语句会删除符合 WHERE 子句中的各个条件的每个记录。在本例中，会删除员工 ID 为 E10001 的每个记录。因为员工 ID 在“员工”表中是唯一的，所以仅删除一个记录。

INSERT 语句

使用 INSERT 语句在数据库表中创建记录。您可以指定：

- 要作为新记录插入的值的列表
- 从要作为一组新记录插入的其他表复制数据的 SELECT 语句

INSERT 语句的格式为：

```
INSERT INTO table_name [(column_name, ...)]VALUES (expr, ...)
```

column_name 是提供列（其值在 VALUES 子句中指定）的名称和顺序的列名称的可选列表。如果省略 column_name，值表达式 (expr) 必须提供在表中定义的所有列的值，并且顺序必须与针对表定义列的顺序相同。column_name 也可指定字段描述，例如 lastDates[4]。

expr 是为新记录的列提供值的表达式的列表。通常，表达式为列的常量值（但也可以是子查询）。必须用一对单引号 (') 将字符串值括起。要在用单引号括起的字符串值中加入单引号，请同时使用两个单引号（例如，'Don''t'）。

子查询必须用括号括起。

示例

插入一个表达式列表。

```
INSERT INTO emp (last_name, first_name, emp_id, salary, hire_date)
   VALUES ('Smith', 'John', 'E22345', 27500, DATE '2019-06-05')
```

每个 INSERT 语句将一个记录添加到数据库表。在本例中，一个记录已经被添加到员工数据库表 emp。为五个列指定值。为表中剩余的列分配空白值，表示 Null。

说明 在容器字段中，只能 INSERT 文本，除非您准备参数化语句，并从应用程序中流式传输数据。要使用二进制数据，可以通过用单引号将文件名括起来分配文件名，也可以使用 PutAs() 函数。指定文件名时，从文件扩展名中推导出文件类型：

```
INSERT INTO table_name (container_name) VALUES(? AS 'filename.file extension')
```

不受支持的文件类型将插入为 FILE 类型。

在使用 PutAs() 函数时，指定类型：PutAs(col, 'type')，其中类型值是第 16 页上的“检索容器字段的内容：CAST() 函数和 GetAs() 函数”中所述的受支持的文件类型。

SELECT 语句是为在列名称列表中指定的每个 column_name 值返回值的查询。使用 SELECT 语句（而不是值表达式的列表）允许从一个表中选择一组行，并使用单个 INSERT 语句将其插入其他表。

示例

使用 SELECT 语句插入。

```
INSERT INTO emp1 (first_name, last_name, emp_id, dept, salary)
   SELECT first_name, last_name, emp_id, dept, salary from emp
   WHERE dept = 'D050'
```

在此类型的 INSERT 语句中，要插入的列的数目必须与 SELECT 语句中列的数目匹配。要插入的列的列表必须与 SELECT 语句中各个列对应，就像与其他类型的 INSERT 语句中值表达式的列表相对应一样。例如，插入的第一个列对应所选的第一个列，插入的第二个列对应所选的第二个列，依次类推。

这些对应列的大小和数据必须兼容。SELECT 列表中的每列都应具有在对 INSERT 列表中的相应列定期进行 INSERT/UPDATE 时 ODBC 或 JDBC 客户端驱动程序接受的数据类型。SELECT 列表列中的值的大小大于对应的 INSERT 列表列的大小时，会截断值。

在插入值之前会先执行 SELECT 语句。

UPDATE 语句

使用 UPDATE 语句更改数据库表中的记录。UPDATE 语句的格式为：

```
UPDATE table_name SET column_name = expr, ...[ WHERE { conditions } ]
```

column_name 是要更改其值的列的名称。在一个语句中可以更改多个列。

expr 是列的新值。

通常，表达式为列的常量值（但也可以是子查询）。必须用一对单引号 (') 将字符串值括起。要在用单引号括起的字符串值中加入单引号，请同时使用两个单引号（例如，'Don''t'）。

子查询必须用括号括起。

WHERE 子句是任何有效的子句。它决定了更新哪些记录。

示例

emp 表上的 UPDATE 语句。

```
UPDATE emp SET salary=32000, exempt=1 WHERE emp_id = 'E10001'
```

UPDATE 语句会更改符合 WHERE 子句中的各个条件的每个记录。在本例中，针对员工 ID 为 E10001 的所有员工更改薪金和免税状态。因为员工 ID 在“员工”表中是唯一的，所以仅更新一个记录。

示例

emp 表上带子查询的 UPDATE 语句。

```
UPDATE emp SET salary = (SELECT avg(salary) from emp) WHERE emp_id = 'E10001'
```

在本例中，针对员工 ID 为 E10001 的员工，将薪金更改为公司的平均薪金。

说明 在容器字段中，只能 UPDATE 文本，除非您准备参数化语句，并从应用程序中流式传输数据。要使用二进制数据，可以通过用单引号将文件名括起来分配文件名，也可以使用 PutAs() 函数。指定文件名时，从文件扩展名中推导出文件类型：

```
UPDATE table_name SET (container_name) = ? AS 'filename.file extension'
```

不受支持的文件类型将插入为 FILE 类型。

在使用 PutAs() 函数时，指定类型：PutAs(col, 'type')，其中类型值是第 16 页上的“检索容器字段的内容：CAST() 函数和 GetAs() 函数”中所述的受支持的文件类型。

CREATE TABLE 语句

使用 CREATE TABLE 语句在数据库文件中创建表。CREATE TABLE 语句的格式为：

```
CREATE TABLE table_name ( table_element_list [, table_element_list...])
```

在语句中，指定每列的名称和数据类型。

- table_name 是表格的名称。table_name 的字符数限于 100。切勿事先定义同名的表。表名称必须以字母字符开头。如果表名称以非字母字符开头，则用双引号将其括起来（带引号的标识符）。
- table_element_list 格式为：

  ```
  field_name field_type [[repetitions]]
  [DEFAULT expr] [UNIQUE | NOT NULL | PRIMARY KEY | GLOBAL]
  [EXTERNAL relative_path_string [SECURE | OPEN calc_path_string]]
  ```

  - field_name 是字段的名称。字段名称必须唯一。字段名以字母字符开头。如果字段名以非字母字符开头，则用双引号将其括起来（带引号的标识符）。

    示例

    名为 _LASTNAME 的字段的 CREATE TABLE 语句为：

    ```
    CREATE TABLE "_EMPLOYEE" (ID INT PRIMARY KEY, "_FIRSTNAME" VARCHAR(20),
    "_LASTNAME" VARCHAR(20))
    ```

  - 对于 CREATE TABLE 语句 repetitions，在字段类型后使用方括号的中数字 1 至 32000 指定字段重复。

    示例

    ```
    EMPLOYEE_ID INT[4]
    LASTNAME VARCHAR(20)[4]
    ```

  - field_type 可以是以下任一项：NUMERIC、DECIMAL、INT、DATE、TIME、TIMESTAMP、VARCHAR、CHARACTER VARYING、BLOB、VARBINARY、LONGVARBINARY 或 BINARY VARYING。对于 NUMERIC 和 DECIMAL，您可以指定精度和标度。例如：DECIMAL(10,0)。对于 TIME 和 TIMESTAMP，您可以指定精度。例如：TIMESTAMP(6)。对于 VARCHAR 和 CHARACTER VARYING，您可以指定字符串的长度。

示例

```
VARCHAR(255)
```

- 利用 DEFAULT 关键字可设置列的默认值。对于 expr，可以使用常量值或表达式。允许的表达式为 USER、USERNAME、CURRENT_USER、CURRENT_DATE、CURDATE、CURRENT_TIME、CURTIME、CURRENT_TIMESTAMP、CURTIMESTAMP 和 NULL。
- 将某个列定义为 UNIQUE 会为 FileMaker Pro 数据库文件中的相应字段自动选择“**唯一**”验证选项。
- 将某个列定义为 NOT NULL 会为 FileMaker Pro 数据库文件中的相应字段自动选择“**不为空**”验证选项。在 FileMaker Pro 中“管理数据库”对话框的“**字段**”选项卡中，该字段被标记为“**必需值**”。
- 要将某个列定义为容器字段，请对 field_type 使用 BLOB、VARBINARY 或 BINARY VARYING。
- 要将某个列定义为在外部存储数据的容器字段，使用 EXTERNAL 关键字。relative_path_string 定义在外部存储数据的文件夹，相对于 FileMaker Pro 数据库的位置。在“FileMaker Pro 管理容器”对话框中，必须将此路径指定基本目录。必须为安全存储指定 SECURE，或为开放存储指定 OPEN。如果使用的是开放存储，calc_path_string 是要在其中存储容器对象的 relative_path_string 文件夹内部的文件夹。路径在文件夹名称中使用正斜杠 (/)。

示例

使用	示例 SQL
文本列	CREATE TABLE T1 (C1 VARCHAR, C2 VARCHAR (50), C3 VARCHAR (1001), C4 VARCHAR (500276))
文本列，NOT NULL	CREATE TABLE T1NN (C1 VARCHAR NOT NULL, C2 VARCHAR (50) NOT NULL, C3 VARCHAR (1001) NOT NULL, C4 VARCHAR (500276) NOT NULL)
数字列	CREATE TABLE T2 (C1 DECIMAL, C2 DECIMAL (10,0), C3 DECIMAL (7539,2), C4 DECIMAL (497925,301))
日期列	CREATE TABLE T3 (C1 DATE, C2 DATE, C3 DATE, C4 DATE)
时间列	CREATE TABLE T4 (C1 TIME, C2 TIME, C3 TIME, C4 TIME)
时间戳列	CREATE TABLE T5 (C1 TIMESTAMP, C2 TIMESTAMP, C3 TIMESTAMP, C4 TIMESTAMP)
用于容器字段的列	CREATE TABLE T6 (C1 BLOB, C2 BLOB, C3 BLOB, C4 BLOB)
用于外部存储容器字段的列	CREATE TABLE T7 (C1 BLOB EXTERNAL 'Files/MyDatabase/' SECURE) CREATE TABLE T8 (C1 BLOB EXTERNAL 'Files/MyDatabase/' OPEN 'Objects')

TRUNCATE TABLE 语句

使用 TRUNCATE TABLE 语句可以快速删除指定表中的所有记录，从而清空表中的所有数据。

```
TRUNCATE TABLE table_name
```

不能在 TRUNCATE TABLE 语句中指定 WHERE 子句。TRUNCATE TABLE 语句可删除所有记录。

仅删除 table_name 指定的表中的记录。任何相关表中的记录均不会受到影响。

TRUNCATE TABLE 语句必须能够锁定表中的所有记录，才能删除记录数据。只要表中的任何记录由其他用户锁定，FileMaker 软件便会返回错误代码 301（“记录正在被其他用户使用”）。

ALTER TABLE 语句

使用 ALTER TABLE 语句更改数据库文件中现有表的结构。每个语句只能修改一个列。ALTER TABLE 语句的格式为：

```
ALTER TABLE table_name ADD [COLUMN] column_definition
ALTER TABLE table_name DROP [COLUMN] unqualified_column_name
ALTER TABLE table_name ALTER [COLUMN] column_definition SET DEFAULT expr
ALTER TABLE table_name ALTER [COLUMN] column_definition DROP DEFAULT
```

使用 ALTER TABLE 语句之前，必须了解表的结构以及您要如何对其进行修改。

示例

目的	示例 SQL
添加列	ALTER TABLE Salespeople ADD C1 VARCHAR
删除列	ALTER TABLE Salespeople DROP C1
设置列的默认值	ALTER TABLE Salespeople ALTER Company SET DEFAULT 'Claris'
删除列的默认值	ALTER TABLE Salespeople ALTER Company DROP DEFAULT

说明 SET DEFAULT 和 DROP DEFAULT 不影响表中的现有行，但会更改随后添加到表的行的默认值。

CREATE INDEX 语句

使用 CREATE INDEX 语句在数据库文件中加速搜索。CREATE INDEX 语句的格式为：

```
CREATE INDEX ON table_name.column_name
CREATE INDEX ON table_name (column_name)
```

单列支持 CREATE INDEX（不支持多列索引）。在与容器字段类型、合计字段、在 FileMaker Pro 数据库库文件中具有全局存储选项的字段或未存储计算字段的字段相应对的列上不允许索引。

自动创建文本列的索引会为 FileMaker Pro 数据库文件中的相应字段选择“**索引**”中的“**最小**”作为“存储选项”。自动创建非文本列（或设定为日文文本格式的列）的索引会为 FileMaker Pro 数据库文件中的相应字段选择“**索引**”中的“**全部**”作为“存储选项”。

自动创建任何列的索引会为 FileMaker Pro 数据库文件中的相应字段选择“**索引**”中的“**根据需要自动创建索引**”作为“存储选项”。

FileMaker 软件根据需要自动创建索引。使用 CREATE INDEX 会导致立即构建索引，而不是根据需要。

示例

```
CREATE INDEX ON Salespeople.Salesperson_ID
```

DROP INDEX 语句

使用 DROP INDEX 语句从数据库文件删除索引。DROP INDEX 语句的格式为：

```
DROP INDEX ON table_name.column_name
DROP INDEX ON table_name (column_name)
```

在数据库文件过大或者不经常在查询中使用索引的情况下，您可以删除索引。

如果您的查询出现性能下降，并且您处理的是具有许多索引文本字段并且极大的 FileMaker Pro 数据库文件，请考虑从一些字段中删除索引。另外，请考虑从在 SELECT 语句中很少使用的字段中删除索引。

自动删除任何列的索引会为 FileMaker Pro 数据库文件中的相应字段选择“**无**”作为“存储选项”，并清除“**索引**”中的“**根据需要自动创建索引**”。

不支持 PREVENT INDEX CREATION 属性。

示例

```
DROP INDEX ON Salespeople.Salesperson_ID
```

SQL 表达式

在 SELECT 语句的 WHERE、HAVING 和 ORDER BY 子句中使用表达式来构建详细和复杂的数据库查询。有效的表达式元素为：

- 字段名称
- 常量
- 指数/科学记数法
- 数值运算符
- 字符运算符
- 日期运算符
- 关系运算符
- 逻辑运算符
- 函数

字段名称

最常见的表达式是简单的字段名称，例如 calc 或 Sales_Data.Invoice_ID。

常量

常量是不改变的值。例如，在表达式 PRICE * 1.05 中，值 1.05 是常量。可以将值 30 分配给常量 Number_Of_Days_In_June。

必须用一对单引号 (') 将字符常量括起。要在用单引号括起的字符常量中加入单引号，请同时使用两个单引号（例如，'Don''t'）。

对于 ODBC 和 JDBC 应用程序，FileMaker 软件接受用大括号 ({}) 括起的 ODBC/JDBC 格式日期、时间和时间戳常量。

示例

- `{D '2019-06-05'}`
- `{T '14:35:10'}`
- `{TS '2019-06-05 14:35:10'}`

FileMaker 软件允许类型说明符（`D`、`T`、`TS`）为大写或小写。您可以在类型说明符之后使用任意数目的空格，甚至省略空格。

FileMaker 软件还接受没有大括号的 SQL-92 语法 ISO 日期和时间格式。

示例

- `DATE 'YYYY-MM-DD'`
- `TIME 'HH:MM:SS'`
- `TIMESTAMP 'YYYY-MM-DD HH:MM:SS'`

FileMaker Pro ExecuteSQL 函数只接受 没有大括号的 SQL-92 语法 ISO 日期和时间格式。

常量	可接受的语法（示例）
文本	`'Paris'`
数字	`1.05`

日期	`DATE '2019-06-05'`
	`{ D '2019-06-05' }`
	`{06/05/2019}`
	`{06/05/19}`
	注意 不支持对 ODBC/JDBC 格式或 SQL-92 格式使用 2 位数年份语法。

时间	`TIME '14:35:10'`
	`{ T '14:35:10' }`
	`{14:35:10}`

时间戳	`TIMESTAMP '2019-06-05 14:35:10'`
	`{ TS '2019-06-05 14:35:10'}`
	`{06/05/2019 14:35:10}`
	`{06/05/19 14:35:10}`
	请确保未针对使用此 2 位数年份语法的字段，选择“**严格数据类型：4 位年份日期**”作为 FileMaker Pro 数据库文件的验证选项。
	注意 不支持对 ODBC/JDBC 格式或 SQL-92 格式使用 2 位数年份语法。

输入日期和时间值时，请匹配数据库文件区域设置的格式。例如，如果数据库是在意大利语系统上创建的，则使用意大利语日期和时间格式。

指数/科学记数法

可以使用科学记数法表示数字。

示例

```
SELECT column1 / 3.4E+7 FROM table1 WHERE calc < 3.4E-6 * column2
```

数值运算符

可以在数字表达式中包括以下运算符：+、-、*、/ 和 ^ 或 **（乘方）。

可以在数值表达式的前面加上一元加号 (+) 或一元减号 (-)。

字符运算符

您可以将多个字符连接起来。在下表中，`last_name` 为 `'JONES '`，first_name 为 `'ROBERT '`。

运算符	串联	示例	结果
+	保留末尾的空格字符	first_name + last_name	'ROBERT JONES '
-	将末尾的空格字符移到最后	first_name - last_name	'ROBERTJONES '

日期运算符

您可以修改日期。在下表中，`hire_date` 为 `DATE '2019-01-30'`。

运算符	对日期的影响	示例	结果
+	将天数添加到日期	`hire_date` + 5	DATE '2019-02-04'
-	查找两个日期之间的天数	`hire_date - DATE '2019-01-01'`	`29`
	从日期中减去天数	`hire_date - 10`	DATE '2019-01-20'

示例

```
SELECT Date_Sold, Date_Sold + 30 AS agg FROM Sales_Data
SELECT Date_Sold, Date_Sold - 30 AS agg FROM Sales_Data
```

关系运算符

运算符	含义
=	等于
<>	不等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
LIKE	匹配模式
NOT LIKE	不匹配模式
IS NULL	等于 Null
IS NOT NULL	不等于 Null
BETWEEN	下限和上限之间值的范围
IN	一组指定值的成员或子查询的成员
NOT IN	并非一组指定值的成员或子查询的成员
EXISTS	如果子查询返回至少一个记录，则为“True”
ANY	将某个值与子查询返回的每个值进行比较（运算符前面必须是 =、<>、>、>=、< 或 <=）；=Any 相当于 In
ALL	将某个值与子查询返回的每个值进行比较（运算符前面必须是 =、<>、>、>=、< 或 <=）

示例

```
SELECT Sales_Data.Invoice_ID FROM Sales_Data
   WHERE Sales_Data.Salesperson_ID = 'SP-1'
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Invoice_ID <> 125
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount > 3000
SELECT Sales_Data.Time_Sold FROM Sales_Data
   WHERE Sales_Data.Time_Sold < '12:00:00'
SELECT Sales_Data.Company_Name FROM Sales_Data
   WHERE Sales_Data.Company_Name LIKE '%University'
SELECT Sales_Data.Company_Name FROM Sales_Data
   WHERE Sales_Data.Company_Name NOT LIKE '%University'
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount IS NULL
SELECT Sales_Data.Amount FROM Sales_Data WHERE Sales_Data.Amount IS NOT NULL
SELECT Sales_Data.Invoice_ID FROM Sales_Data
   WHERE Sales_Data.Invoice_ID BETWEEN 1 AND 10
SELECT COUNT(Sales_Data.Invoice_ID) AS agg
   FROM Sales_Data WHERE Sales_Data.INVOICE_ID IN (50,250,100)
SELECT COUNT(Sales_Data.Invoice_ID) AS agg
   FROM Sales_Data WHERE Sales_Data.INVOICE_ID NOT IN (50,250,100)
SELECT COUNT(Sales_Data.Invoice_ID) AS agg FROM Sales_Data
   WHERE Sales_Data.INVOICE_ID NOT IN (SELECT Sales_Data.Invoice_ID
   FROM Sales_Data WHERE Sales_Data.Salesperson_ID = 'SP-4')
SELECT *
   FROM Sales_Data WHERE EXISTS (SELECT Sales_Data.Amount
   FROM Sales_Data WHERE Sales_Data.Salesperson_ID IS NOT NULL)
SELECT *
   FROM Sales_Data WHERE Sales_Data.Amount = ANY (SELECT Sales_Data.Amount
   FROM Sales_Data WHERE Sales_Data.Salesperson_ID = 'SP-1')
SELECT *
   FROM Sales_Data WHERE Sales_Data.Amount = ALL (SELECT Sales_Data.Amount
   FROM Sales_Data WHERE Sales_Data.Salesperson_ID IS NULL)
```

逻辑运算符

可以合并一个或多个条件。条件必须由 AND 或 OR 相关联，例如：

```
salary = 40000 AND exempt = 1
```

逻辑 NOT 运算符用于反转含义，例如：

```
NOT (salary = 40000 AND exempt = 1)
```

示例

```
SELECT * FROM Sales_Data WHERE Sales_Data.Company_Name
   NOT LIKE '%University' AND Sales_Data.Amount > 3000
SELECT * FROM Sales_Data WHERE (Sales_Data.Company_Name
   LIKE '%University' OR Sales_Data.Amount > 3000)
   AND Sales_Data.Salesperson_ID = 'SP-1'
```

运算符优先级

因为表达式变得更复杂，所以执行表达式的顺序变得很重要。此表显示运算符的执行顺序。首先执行第一行的运算符，依次类推。从左至右执行表达式中同一行的运算符。

优先级	运算符
1	Unary '-'、Unary '+'
2	^, **
3	*, /
4	+, -
5	=、<>、<、<=、>、>=、Like、Not Like、Is Null、Is Not Null、Between、In、Exists、Any、All
6	Not
7	AND
8	OR

示例

```
WHERE salary > 40000 OR hire_date > (DATE '2008-01-30') AND dept =
'D101'
```

因为首先执行 AND，此查询会检索从 2008 年 1 月 30 日后录用的部门 D101 的员工，以及收入超过 $40,000 的每个员工（无论所属部门或录用日期如何）。

要强制按不同的顺序执行子句，使用括号将要首先执行的条件括起。

```
WHERE (salary > 40000 OR hire_date > DATE '2008-01-30') AND dept =
'D101'
```

此示例检索部门 D101 收入超过 $40,000 或自 2008 年 1 月 30 日后录用的员工。

SQL 函数

Claris 为 FileMaker 平台提供 SQL 标准的实现，并且支持可用于表达式的许多功能。有些函数返回字符串，有些返回数值，有些返回日期，有些返回值，取决于函数参数所满足的条件。

聚合函数

聚合函数返回一组记录的单个值。您可以将聚合函数用作 `SELECT` 语句的一部分，与字段名（例如，`AVG(SALARY)`）一起使用，或结合列表达式（例如，`AVG(SALARY * 1.07)`）一起使用。

您可以在列表达式的前面加上 `DISTINCT` 运算符，以清除重复值。

示例

```
COUNT (DISTINCT last_name)
```

在本示例中，仅对唯一的姓氏值进行计数。

聚合函数	返回
SUM	数字字段表达式中值的总计。例如，SUM(SALARY) 返回所有薪金字段值的总和。
AVG	数字字段表达式中值的平均值。例如，AVG(SALARY) 返回所有薪金字段值的平均值。
COUNT	字段表达式中值的数目。例如，COUNT(NAME) 返回名称值的数目。将 COUNT 与字段名一起使用时，COUNT 返回非 null 字段值的数目。特例为 COUNT(*)，它返回集合中记录的数目，包括值为 null 的记录。
MAX	字段表达式中的最大值。例如，MAX(SALARY) 返回最大的薪金字段值。
MIN	字段表达式中的最小值。例如，MIN(SALARY) 返回最小的薪金字段值。

示例

```
SELECT SUM (Sales_Data.Amount) AS agg FROM Sales_Data
SELECT AVG (Sales_Data.Amount) AS agg FROM Sales_Data
SELECT COUNT (Sales_Data.Amount) AS agg FROM Sales_Data
SELECT MAX (Sales_Data.Amount) AS agg FROM Sales_Data
   WHERE Sales_Data.Amount < 3000
SELECT MIN (Sales_Data.Amount) AS agg FROM Sales_Data
   WHERE Sales_Data.Amount > 3000
```

不能将聚合函数作为其他函数的参数。否则，FileMaker 软件将返回错误代码 8309（“不支持包含聚合函数的表达式”）。例如，以下语句无效，因为聚合函数 SUM 不能用作 ROUND 函数的参数：

示例

```
SELECT ROUND(SUM(Salary), 0) FROM Payroll
```

但是，聚合函数可以使用将数值作为参数返回的函数。以下语句有效。

示例

```
SELECT SUM(ROUND(Salary, 0)) FROM Payroll
```

返回字符串的函数

返回字符串的函数	说明	示例
CHR	将 ASCII 代码转换为一个字符的字符串	CHR(67) 返回 **C**
CURRENT_USER	返回在连接时指定的登录 ID	
DAYNAME	返回与指定日期对应的是星期几	
RTRIM	删除字符串末尾的空格	RTRIM('ABC ') 返回 'ABC'
TRIM	删除字符串的前导和末尾空格	TRIM(' ABC ') 返回 'ABC'
LTRIM	删除字符串的前导空格	LTRIM(' ABC') 返回 'ABC'
UPPER	将字符串的每个字母更改为大写	UPPER('Allen') 返回 'ALLEN'
LOWER	将字符串的每个字母更改为小写	LOWER('Allen') 返回 'allen'
LEFT	返回字符串最左侧的字符	LEFT('Mattson',3) 返回 'Mat'

返回字符串的函数	说明	示例
MONTHNAME	返回日历月份的名称	
RIGHT	返回字符串最右侧的字符	RIGHT('Mattson',4) 返回 'tson'
SUBSTR SUBSTRING	返回字符串的子字符串，含字符串的参数，要提取的第一个字符和要提取的字符数（可选）	SUBSTR('Conrad',2,3) 返回 'onr' SUBSTR('Conrad',2) 返回 'onrad'
SPACE	生成空格字符串	SPACE(5) 返回 ' '
STRVAL	将任何类型的值转换为字符串	STRVAL('Woltman') 返回 'Woltman' STRVAL(5 * 3) 返回 '15' STRVAL(4 = 5) 返回 'False' STRVAL(DATE '2019-12-25') 返回 '2019-12-25'
TIME TIMEVAL	以字符串的形式返回当天的时间	在晚上 9:49，TIME() 返回 **21:49:00**
USERNAME USER	返回在连接时指定的登录 ID	

说明 TIME() 函数已弃用。请改为使用 SQL 标准 CURRENT_TIME。

示例

```
SELECT CHR(67) + SPACE(1) + CHR(70) FROM Salespeople

SELECT RTRIM(' ' + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT TRIM(SPACE(1) + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT LTRIM(' ' + Salespeople.Salesperson_ID) AS agg FROM Salespeople

SELECT UPPER(Salespeople.Salesperson) AS agg FROM Salespeople

SELECT LOWER(Salespeople.Salesperson) AS agg FROM Salespeople

SELECT LEFT(Salespeople.Salesperson, 5) AS agg FROM Salespeople

SELECT RIGHT(Salespeople.Salesperson, 7) AS agg FROM Salespeople

SELECT SUBSTR(Salespeople.Salesperson_ID, 2, 2) +
SUBSTR(Salespeople.Salesperson_ID, 4, 2) AS agg FROM Salespeople

SELECT SUBSTR(Salespeople.Salesperson_ID, 2) +
SUBSTR(Salespeople.Salesperson_ID, 4) AS agg FROM Salespeople

SELECT SPACE(2) + Salespeople.Salesperson_ID AS Salesperson_ID FROM
Salespeople

SELECT STRVAL('60506') AS agg FROM Sales_Data WHERE
Sales_Data.Invoice = 1
```

返回数字的函数

返回数字的函数	说明	示例
ABS	返回数值表达式的绝对值	
ATAN	用以弧度表示的角度返回参数的反正切值	
ATAN2	用以弧度表示的角度返回 x 和 y 坐标的反正切值	
CEIL CEILING	返回大于或等于参数的最小整数值	
DEG DEGREES	返回参数的角度值，以弧度表示的角度	
DAY	返回日期中日的部分	DAY(DATE '2019-01-30') 返回 **30**
DAYOFWEEK	返回日期表达式的星期几 (1-7)	DAYOFWEEK(DATE '2004-05-01') 返回 **7**
MOD	两个数相除，返回相除的余数	MOD(10,3) 返回 **1**
EXP	返回一个值，该值是自然对数的底数 (e) 由参数指定的幂	
FLOOR	返回小于或等于参数的最大整数值	
HOUR	返回值的小时部分	
INT	返回数值的整数部分	INT(6.4321) 返回 **6**
LENGTH	返回字符串的长度	LENGTH('ABC') 返回 **3**
MONTH	返回日期的月份部分	MONTH(DATE '2019-01-30') 返回 **1**
LN	返回参数的自然对数	
LOG	返回参数的常用对数	
MAX	返回两个数值中的较大值	MAX(66,89) 返回 **89**
MIN	返回两个数值中的较小值	MIN(66,89) 返回 **66**
MINUTE	返回值的分钟部分	
NUMVAL	将字符串转换为数值。如果字符串不是有效的数字，则函数失败。	NUMVAL('123') 返回 **123**
PI	返回数学常数圆周率 pi 的常量值	
RADIANS	返回以角度表示的参数的弧度数	
ROUND	将数字四舍五入	ROUND(123.456,0) 返回 **123** ROUND(123.456,2) 返回 **123.46** ROUND(123.456,-2) 返回 **100**
SECOND	返回值的秒部分	
SIGN	参数符号的标记：-1 代表负数，0 代表 0，1 代表正数	
SIN	返回参数的正弦	
SQRT	返回参数的平方根	
TAN	返回参数的正切	
YEAR	返回日期的年份部分	YEAR(DATE '2019-01-30') 返回 **2019**

返回日期的函数

返回日期的函数	说明	示例
CURDATE CURRENT_DATE	返回今天的日期	
CURTIME CURRENT_TIME	返回当前时间	
CURTIMESTAMP CURRENT_TIMESTAMP	返回当前系统时间值	
TIMESTAMPVAL	将字符串转换为时间戳	TIMESTAMPVAL('2019-01-30 14:00:00') 返回其时间戳值
DATE TODAY	返回今天的日期	如果今天是 2019/11/21，DATE() 返回 **2019-11-21**
DATEVAL	将字符串转换为日期	DATEVAL('2019-01-30') 返回 **2019-01-30**

说明 DATE() 函数已弃用。请改为使用 SQL 标准 CURRENT_DATE。

条件函数

条件函数	说明	示例
CASE WHEN	**简单 CASE 格式** 将 input_exp 的值与 value_exp 参数的值进行比较来确定结果。 CASE input_exp {WHEN value_exp THEN result...}[ELSE result] END	SELECT Invoice_ID, CASE Company_Name WHEN 'Exports UK' THEN 'Exports UK Found' WHEN 'Home Furniture Suppliers' THEN 'Home Furniture Suppliers Found' ELSE 'Neither Exports UK nor Home Furniture Suppliers' END, Salesperson_ID FROM Sales_Data
	搜索的 CASE 格式 基于 WHEN 表达式指定的条件是否为 true，返回结果。 CASE {WHEN boolean_exp THEN result...}[ELSE result] END	SELECT Invoice_ID, Amount, CASE WHEN Amount > 3000 THEN 'Above 3000' WHEN Amount < 1000 THEN 'Below 3000' ELSE 'Between 1000 and 3000' END, Salesperson_ID FROM Sales_Data
COALESCE	返回不为 NULL 的第一个值	SELECT Salesperson_ID, COALESCE(Sales_Manager, Salesperson) FROM Salespeople
NULLIF	将两个值进行比较，如果两个值相等则返回 NULL；否则，返回第一个值	SELECT Invoice_ID, NULLIF(Amount, -1), Salesperson_ID FROM Sales_Data

FileMaker 系统对象

FileMaker Pro 数据库文件包含以下您可以使用 SQL 查询访问的系统对象。

FileMaker 系统表

每个 FileMaker Pro 数据库文件都包含以下系统表：FileMaker_Tables、FileMaker_Fields 和 FileMaker_BaseTableFields。对于 ODBC 应用程序，这些表包含在目录函数 SQLTables 返回的信息中。对于 JDBC 应用程序，这些表包含在 DatabaseMetaData 方法 getTables 返回的信息中。这些表还可以用在 ExecuteSQL 函数中。

FileMaker_Tables

FileMaker_Tables 表包含有关 FileMaker Pro 文件中定义的数据库表的信息。

在 FileMaker_Tables 表中，关系图中的每个表摹本对应一行，另外还包括以下各列：

- TableName - 表摹本的名称。
- TableId - 表摹本的唯一 ID。
- BaseTableName - 创建表摹本时所用的基本表的名称。
- BaseFileName - 包含基本表的数据库文件的 FileMaker Pro 文件名。
- ModCount - 已提交的、对该表的定义所做更改的总次数。

示例

```
SELECT TableName FROM FileMaker_Tables WHERE TableName LIKE 'Sales%'
```

FileMaker_Fields 表

FileMaker_Fields 表包含有关 FileMaker Pro 文件中为所有表摹本定义的字段的信息。

FileMaker_Fields 表包含以下各列：

- TableName - 包含字段的表的名称。
- FieldName - 字段的名称。
- FieldType - 字段的 SQL 数据类型。
- FieldId - 字段的唯一 ID。
- FieldClass - 以下三个值之一：Summary，适用于合计字段；Calculated，适用于计算结果；或 Normal。
- FieldReps - 字段的重复数。
- ModCount - 已提交的、对该表的定义所做更改的总次数。

示例

```
SELECT * FROM FileMaker_Fields WHERE TableName='Sales'
```

FileMaker_BaseTableFields 表

引入于 FileMaker 平台 19.4.1 版本，FileMaker_BaseTableFields 表包含有关 FileMaker Pro 文件中仅为源(或基础)表定义的字段的信息。

FileMaker_BaseTableFields 表包含以下各列：

- BaseTableName - 包含字段的该基础表名称。
- FieldName - 字段的名称。
- FieldType - 字段的 SQL 数据类型。
- FieldId - 字段的唯一 ID。
- FieldClass - 以下三个值之一：Summary，适用于合计字段；Calculated，适用于计算结果；或 Normal。
- FieldReps - 字段的重复数。
- ModCount - 已提交的、对该基础表的定义所做更改的总次数。

示例

```
SELECT * FROM FileMaker_BaseFields WHERE BaseTableName='Sales'
```

FileMaker 系统列

FileMaker 软件将系统列（字段）添加到 FileMaker Pro 文件中定义的所有表中的所有行（记录）。对于 ODBC 应用程序，这些列包含在目录函数 SQLSpecialColumns 返回的信息中。对于 JDBC 应用程序，这些列包含在 DatabaseMetaData 方法 getVersionColumns 返回的信息中。这些列还可以用在 ExecuteSQL 函数中。

ROWID 列

`ROWID` 系统列包含记录的唯一 ID 号。这与 FileMaker Pro Get (记录 ID) 函数返回的值相同。

ROWMODID 列

`ROWMODID` 系统列包含已提交的、对当前记录所做更改的总次数。这与 FileMaker Pro Get (记录修改次数) 函数返回的值相同。

示例

```
SELECT ROWID, ROWMODID FROM MyTable WHERE ROWMODID > 3
```

预留的 SQL 关键字

本部分列出了预留关键字，这些关键字不应用作列、表、别名或其他用户定义对象的名称。如果您收到语法错误，这些错误可能是由于使用了以下预留关键字之一导致的。如果要使用以下关键字之一，需要使用引号防止将其视为关键字。

示例

使用 DEC 关键字作为数据元素名称。

```
create table t ("dec" numeric)
```

ABSOLUTE
ACTION
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BEGIN
BETWEEN
BINARY
BIT
BIT_LENGTH
BLOB
BOOLEAN
BOTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHARACTER
CHARACTER_LENGTH
CHAR_LENGTH
CHECK
CHR
CLOSE
COALESCE
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CROSS
CURDATE
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
CURTIME
CURTIMESTAMP
DATE
DATEVAL
DAY
DAYNAME
DAYOFWEEK
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFAULT
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DISCONNECT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END

END_EXEC
ESCAPE
EVERY
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR
FOREIGN
FOUND
FROM
FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IMMEDIATE
IN
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION
JOIN
KEY
LANGUAGE
LAST
LEADING
LEFT
LENGTH
LEVEL
LIKE
LOCAL
LONGVARBINARY
LOWER
LTRIM
MATCH
MAX
MIN
MINUTE
MODULE
MONTH
MONTHNAME
NAMES
NATIONAL
NATURAL
NCHAR
NEXT
NO
NOT
NULL
NULLIF
NUMERIC
NUMVAL
OCTET_LENGTH
OF
OFFSET
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PAD
PART
PARTIAL
PERCENT
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
READ
REAL
REFERENCES
RELATIVE
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROUND
ROW
ROWID
ROWS
RTRIM
SCHEMA

SCROLL
SECOND
SECTION
SELECT
SESSION
SESSION_USER
SET
SIZE
SMALLINT
SOME
SPACE
SQL
SQLCODE
SQLERROR
SQLSTATE
STRVAL
SUBSTRING
SUM
SYSTEM_USER
TABLE
TEMPORARY
THEN
TIES
TIME
TIMESTAMP
TIMESTAMPVAL
TIMEVAL
TIMEZONE_HOUR
TIMEZONE_MINUTE
TO
TODAY
TRAILING
TRANSACTION
TRANSLATE
TRANSLATION
TRIM
TRUE
TRUNCATE
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USER
USERNAME
USING
VALUE
VALUES
VARBINARY
VARCHAR
VARYING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
WRITE
YEAR
ZONE

索引

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z