一、数据库基本概念

A.数据(Data)

能够输入计算机并能被计算机程序识别和处理的信息集合。

B.数据库(Database)

数据库是在数据库管理系统管理和控制之下，存放在存储介质上的数据集合。

**注意：数据库管理系统(DataBase Mangement System) --DBMMS**

C.数据库管理系统(DBMS)

DBMS是数据库系统中对数据进行统一管理和控制的软件系统

(1)数据库定义功能  (Data Definition)

(2)数据库操纵功能  (Data Manpulation)

(3)数据库运行控制功能  (Data Control)

(4)数据通信功能(Data  Communication)

即 为应用程序提供的访问接口

(5)支持存取海量数据(Mass Data)

二、文件管理和数据库管理

A.文件管理阶段

优点 : (i)数据可长期保存

          (ii)能存储大量数据

缺点: (i)数据冗余度(redundancy)、数据一致性(consistency)、完整性(integrity)难以维持

         (ii)数据与程序缺乏高度独立性

B.数据库系统阶段

(i)数据组织结构化

(ii)数据冗余度比较小，易扩充

(iii)具有较高的数据与程序之间的独立性

(iv)统一的数据控制

三、常用的数据库介绍

A.大型数据库

Oracle公司是最早开发关系数据库的厂商之一，其产品支持最广泛的操作系统平台。目前Oracle关系数据库产品的市场占有率名列前茅。

IBM的DB2是第一个具备网上功能的多媒体关系数据库管理系统，支持包括Linux在内的一系列平台。

B.中型数据库

SQLServer是微软开发的数据库产品，主要支持windows平台。

C.小型数据库

mysql是一个小型关系型数据管理系统，开发者为瑞典mysql AB公司，2005年被sun公司收购。开放 源码。

D.基于嵌入式Linux的数据库

<1>.基于嵌入式linux的数据库主要有SQLite,Firbird,Berkeley DB,extremeDB

<2>.Firebird是关系型数据库，功能强大，支持存储过程、SQL兼容等

<3>.SQLite关系型数据库，体积小，支持 ACID（原子性、一致性、独立性及持久性 Atomicity、Consistency、Isolation、Durability）事物。

<4>.Berkeley DB中并米有数据库服务器的概念，它的程序库直接连接到应用程序中

<5>.extremeDB是内存数据库，运行效率高

四、sqlite的介绍

A.sqlite的源代码是C,其源代码完全开放。sqlite第一个Alpha版本诞生于2000年5月。他是一个轻量级的嵌入式数据库。

B.sqlite有一下特性

<1>零配置，无需安装和配置

<2>储存在单一磁盘文件中的一个完整的数据库

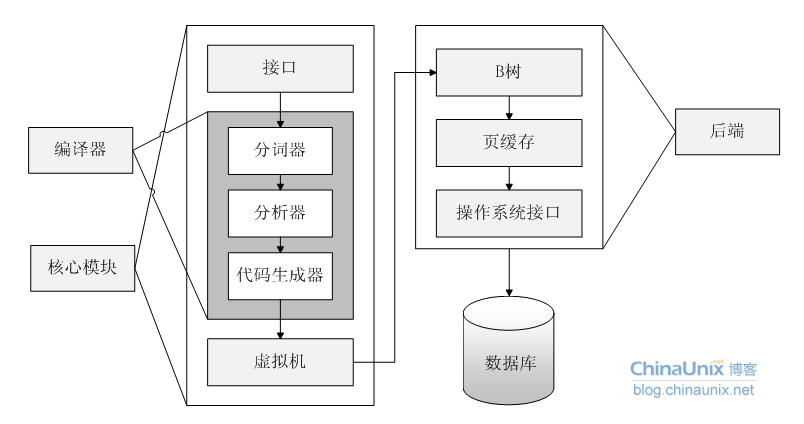
<3>数据库文件可以在不同字节顺序的机器间自由共享

<4>支持数据库大小之2TB

<5>足够小，全部源代码大致3万行C代码，250KB

<6>比目前流行的大多数数据库对数据的操作要快

C.sqlite数据库工作原理

sqlite数据库采用模块化设计，由8个独立的模块构成，这些独立模块又构成了三个主要的子系统，模块将复杂的查询过程分解为细小的工作进行处理。   
  
[](http://blog.chinaunix.net/attachment/201206/11/26833883_133941942021uL.png)

sqlite主要由7个构件子系统(也就是模块)组成，这些模块被分割为两个部分:

前端解析系统和后端引擎。

前端:

前端预处理应用程序传递过来的SQL语句和SQLite命令。对获取的编码分析，优化，并转换 为后端能够执行的SQLite内部字节编码。前端可分为三个模块:

<1>标示分析(Tokenizer)

将输入的SQL语句分成标识符；

<2>语法分析(Parser)

解析器分析通过标识器产生的标识分析语句的结构，并且得到一颗语法树。解析器同时也包含了重构语法树的优化器，因此能够找到一颗产生一个高效的字节编码程序的语法树。

<3>代码生成器(Code Generator)

代码生成器遍历语法树，并且生成一个等价的字节编码程序

前端实现了sqlite3\_prepare API函数

后端:

后端是用来解释字节编码程序的引擎。该引擎做的才是真正的数据库处理工作。后端部分由四个模块组成：

<1>虚拟机(VM)

VM模块是一个内部字节编码语言的解释器。它通过执行字节编码语句来实现SQL语句的工作。它是数据库中数据的最终的操作者。它把数据库看成表和索引的集合，而表和索引则是一系列的元组或者记录。

<2>B/B+树

B/B+树模块把每一个元组集组织进一个一次排好序的树状数据结构中，表和索引被分别置于单独的B+和Ｂ树中。该模块帮助VM进行搜索，插入和删除树中的元组。它也帮助VM创建新的树和删除旧的树

<3>页面调度程序(pager)

页面调度程序模块在原始文件的上层实现了一个面向页面的数据库文件抽象。它管理B/B+树使用的内存内缓存(数据库页的)，另外，他也管理文件的锁定，并用日志来实现事物的ACID属性。

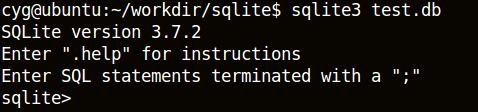
<4>操作系统交界面(system interface)

操作系统界面模块提供了对应于不同本地操作系统的统一交界面

后端实现了sqlite3\_bind\_\*,sqlite3\_setp,sqlite3\_coloumn\_\*,sqlite3\_reset和sqlite3\_finalize API函数。

五、sqlite的一些手动操作语句和命令

A.sqlite常用命令介绍

<1>在终端下运行sqlite3 <\*.db>,出现如下提示符  
  
[](http://blog.chinaunix.net/attachment/201206/11/26833883_13394194684zMW.png)

<\*.db>是要打开的数据库文件。若该文件不存在，则自动创建。

<2>显示所有命令

        sqlite> .help

<3>退出sqlite3

   sqlite>.quit

<4>显示当前打开的数据库文件

sqlite>.database

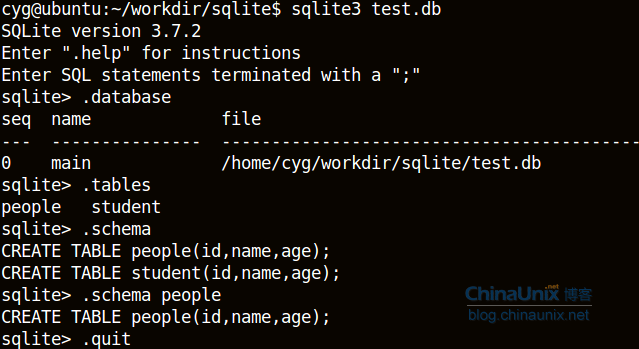
<5>显示数据库中所有表名

sqlite>.tables

<6>查看表的结构

sqlite>.schema  <table\_name>

**注意：这些命令都是以 " .  "开头的。**

操作案例:(我这个数据库文件中已经创建好了表，后面将介绍如何创建和操作表)  
  
[](http://blog.chinaunix.net/attachment/201206/11/26833883_13394195092zuR.png)

B.sqlite常用语句

**注意：每条语句都必须以";"结尾。**

<1>创建新表

**sqlite>create  table  <table\_name>  (f1  type1, f2  type2,…);**

        例如:

**create table  people(id,name,age);**

<2>删除表

**sqlite>drop  table  <table\_name>**

        例如:

**drop table people;**

<3>向表中添加新记录

**sqlite>insert  into  <table\_name>  values (value1, value2,…);**

        例如：

**insert into people  values(1,'A',10);**

**insert into people  values(2,'B',13);**

**insert into people  values(3,'C',9);**

**insert into people  values(4,'C',15);**

**insert into people  values(5,NULL,NULL);**

**注意： 字符串要用单引号括起来。**

<4>查询表中所有记录

**sqlite>select  \*  from  <table\_name>;**

        例如 :

**select   \*   from  people;**

<4>按指定条件查询表中记录

**sqlite>select  \*  from  <table\_name>  where  <expression>；**

        例如:

        在表中搜索名字是A的项所有信息

**select  \*  from  people  where  name='A';**

         在表中搜索年龄>=10并且<=15的项的所有信息

**select  \*  from   people  where age>=10  and  age<=15;**

        在表中搜索名字是C的项，显示其name和age

**select name,age from people where name='C';**

        显示表中的前2项所有信息

**select  \*  from  people  limit  2;**

        显示以年龄排序表中的信息

**select \* from people  order by age;**

<6>按指定条件删除表中记录

**sqlite>delete  from  <table\_name>  where  <expression>**

        例如:

        删除表中名字是'C'的项

**delete from pople  where name='C';**

<7>更新表中记录

**sqlite>update  <table\_name>  set  <f1=value1>, <f2=value2>…   where  <expression>;**

        例如:

        将表中年龄是15并且ID是4项，名字改为CYG

**update  people set  name='cyg'  where  id=4 and  age=15;**

<8>在表中添加字段

        sqlite>alter table <table> add column <field> ;

        例如:

        在people表中添加一个addr字段

        alter table  people add column addr;

**注意：(来自网络)**

**今天在做数据库升级时，碰到要对原来数据库中一张表的一个字段名进行修改，但是用：**

**alter table tablename rename column oldColumnName to newColumnName;**

**始终不成功，后面查阅相关信息：**

**SQLite supports a limited subset of ALTER TABLE. The ALTER TABLE command in SQLite allows the user to rename a table or to add a new column to an existing table. It is not possible to rename a column, remove a column, or add or remove constraints from a table.**

**sqlite支持一个更改表内容的有限子集，就是说在sqlite更改表的命令中，只允许用户重命名表名或者增加多一个列到一个的表中。而重命名一个字段名和删除一个字段、或者增加和删除系统规定的参数这些操作是不可能的。**

**解决办法:**

**例如：在上面的操作过程中，我们在people表中新添加了一个字段addr,要删除这个字段，直接用sqlite的语句时无法完成的。**

**我们可以这样干:**

**A.将people表重命名为temp;**

**B.重新创建people表;**

**C.将temp表中的相应字段内容复制到people表中。**

**D.删除temp表**

**操作如下:**

**A.alter table people rename to temp;**

**B.create table people(id,name,age);**

**C.insert  into  people  select  id,name,age  from temp;**

六、sqlite的一些常用API

<1>sqlite里最常用到的是sqlite3 \*类型。从数据库打开开始，sqlite就要为这个类型准备好内存，直到数据库关闭，整个过程都需要用到这个类型。当数据库打开时开始，这个类型的变量就代表了你要操作的数据库，即句柄。

<2>int   sqlite3\_open(char \*path,sqlite3   \*\*db);

功能：打开sqlite数据库

path:数据库文件路径(如果不存在，则创建)

db:指向sqlite句柄的指针

返回值:如果是SQLITE\_OK则表示操作正常。相关的返回值sqlite定义了一些宏。具体这些宏的含义可以参考sqlite3.h文件。

<3>int   sqlite3\_close(sqlite3  \*db);

功能：关闭sqlite数据库

放回值：成功返回0，失败返回错误码

<4>const char \*sqlite3\_errmsg(sqlite3   \*db);

返回值：返回错误信息

案例:

1. #include <sqlite3.h>
2. #include <errno.h>
3. #include <stdlib.h>
4. #include <stdio.h>
5. int main(int argc,char \*argv[])
6. {
7. sqlite3 \*\*db;
8. int result;
9. if(argc < 2)
10. {
11. fprintf(stderr,"usage : %s argv[1].\n",argv[0]);
12. exit(EXIT\_FAILURE);
13. }
14. result = sqlite3\_open(argv[1],&db);
15. if(result != SQLITE\_OK)
16. {
17. printf("Fail to sqlite3 open %s : %s.\n",argv[1],sqlite3\_errmsg(db));
18. exit(EXIT\_FAILURE);
19. }
20. result = sqlite3\_close(db);
21. if(result != 0)
22. {
23. printf("Fail to sqlite3 close %s : %s.\n",argv[1],sqlite3\_errmsg(db));
24. exit(EXIT\_FAILURE);
25. }
27. exit(EXIT\_SUCCESS);
28. }

<5>执行一条sql语句

typedef  int (\*sqlite3\_callback)(void \*,int ,char \*\*,char \*\*);

int  sqlite3\_exec

(

    sqlite3   \*db ,

    const  char \*sql ,

    sqlite3\_callback  callback   ,

    void  \*arg,char  \*\*errmsg

);

a.第一个参数是前面sqlite3\_open函数得到的指针。

b.第二个参数const  char   \*sql是一条sql语句，以\0结尾。

c.第三个参数callbak是回调，当这条语句执行之后,sqlite3会去调用你提供的这个函数。

d.第四个参数void \*是你提供的指针，你可以传递任何一个指针参数到这里，这个参数最终会传到回调函数里面，如果不需要传递指针给回调函数，可以填NULL。

e.第5个参数char  \*\*errmsg是错误信息。注意是指针的指针。sqlite3里面有很多固定的错误信息。执行sqlite3\_exec之后，执行失败是可以查阅这个指针;

例如:

char \*errmsg;

调用sqlite3\_exec,失败后printf("%s\n",errmsg)得到一串字符串信息，这串信息高诉你错在什么地方。sqlite3\_exec函数通过修改你传入的指针的指针，把你提供的指针指向错误提示信息，这样sqlite3\_exec函数外面就可以通过这个char \*errmsg得到具体错误提示

说明:通常,sqlite3\_callback和它后面的void \*这两个位置都可以填NULL。填NULL表示你不需要回调。比如你做insert操作，做delete操作，就没有必要使用回调。而当你做select时，就要使用回调，因为sqlite3把数据查出来，得通过回调告诉你查出了什么数据。

**返回值：成功返回0，失败返回错误码**

exec的回调

typedef  int (\*sqlite3\_callback)(void  \*para,int   n\_column,char \*\*column\_value,char  \*\*column\_name);

a.通过para参数，你可以传入一些特殊的指针(比如类指针、结构指针)，然后在这里面强制转换成对应的类型(这里面是void  \*类型,必须强制转换成你的类型才可用)。然后操作这些数据。

b.n\_column是这一条记录有多少个字段(即这条记录有多少列)。

c.char  \*\*column\_value是个关键值，查出来的数据都保存在这里，它实际上可以看做是一个一维的指针数组，每个元素都是一个char \*值，是一个字段的内容(用字符串来表示，以\0结尾)。

d.char  \*\*column\_nam 跟column\_value是对应的，表示这个字段的字段名称 。

**//sqlite3的回调函数，sqlite每查到一条记录，就调用一次这个回调**

**int LoadMyInfo(void  \*para,int  n\_column,char  \*\*column\_value,char  \*\* column\_name)**

**{**

**//para是你在sqlite3\_exec里传入的void  \* 参数**

**int     i;**

**printf("记录包含%d个字段\n",n\_column);**

**for(i = 0;i < n\_column;i ++)**

**{**

**printf("字段名 : %s  <->  字段值 : %s.\n",column\_name[i],column\_value[i]);**

**}**

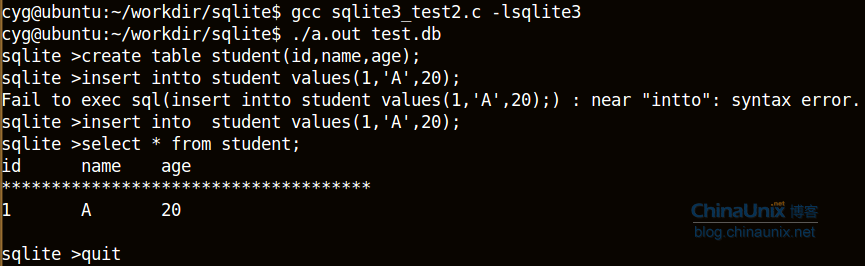
**printf("--------------------------\n");**

**return 0;**

**}**

案例分析:

1. #include <sqlite3.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #define MAX 100
6. typedef int (\*sqlite3\_callbak)(void \*,int ,char \*\*,char \*\*);
7. int ShowMyInfo(void \*arg,int n\_column,char \*\*column\_value,char \*\*column\_name)
8. {
9. int i = 0;
10. for(i = 0;i < n\_column;i ++)
11. {
12. printf("%s\t",column\_name[i]);
13. }
14. printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
15. for(i = 0;i < n\_column;i ++)
16. {
17. printf("%s\t",column\_value[i]);
18. }
19. printf("\n\n");
20. return 0;
21. }
22. int exec\_sql\_string(char \*sql\_string,sqlite3 \*db)
23. {
24. char \*errmsg;
25. if(sqlite3\_exec(db,sql\_string,ShowMyInfo,NULL,&errmsg) != 0)
26. {
27. fprintf(stderr,"Fail to exec sql(%s) : %s.\n",sql\_string,errmsg);
28. return -1;
29. }
30. return 0;
31. }
32. int main(int argc,char \*argv[])
33. {
34. sqlite3 \*db;
35. int result;
36. char sql\_buf[MAX];
37. if(argc < 2)
38. {
39. fprintf(stderr,"usage : %s argv[1].\n",argv[0]);
40. exit(EXIT\_FAILURE);
41. }
42. result = sqlite3\_open(argv[1],&db);
43. if(result != SQLITE\_OK)
44. {
45. fprintf(stderr,"Fail to sqlite3\_open %s : %s.\n",argv[1],sqlite3\_errmsg(db));
46. exit(EXIT\_FAILURE);
47. }
48. while(1)
49. {
50. printf("sqlite >");
51. fgets(sql\_buf,sizeof(sql\_buf),stdin);
52. sql\_buf[strlen(sql\_buf) - 1] = '\0';
54. if(strncmp(sql\_buf,"quit",4) == 0)
55. break;
56. exec\_sql\_string(sql\_buf,db);
57. }
58. result = sqlite3\_close(db);
59. if(result != 0)
60. {
61. fprintf(stderr,"Fail to sqlite3\_open %s : %s.\n",argv[1],sqlite3\_errmsg(db));
62. exit(EXIT\_FAILURE);
63. }
64. exit(EXIT\_SUCCESS);
65. }

运行结果:  
  
[](http://blog.chinaunix.net/attachment/201206/11/26833883_13394196567v7c.png)

<6>不使用回调函数执行SQL语句

int   sqlite3\_get\_table

(

    sqlite3   \*db,

    const char  \*sql,

    char  \*\*\*resultp,

    int  \*nrow,

    int  \*ncolumn,

    char   \*\*errmsg

);

功能：执行sql操作

db  :   数据库句柄

sql  :  sql语句

resultp  : 用来指向sql执行结果的指针

nrow  :  满足条件的记录的数目

**ncolumn  :  每条记录包含的字段数目**

**注意：**

**从第0索引到第ncolumn-1索引都是字段的名称**

**从第ncolumn索引开始，后面都是字段的值**

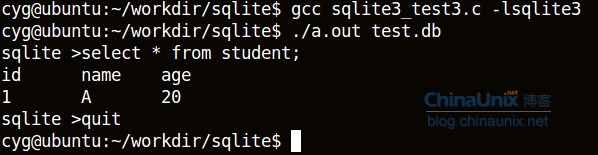
errmsg  :  错误信息指针的地址

返回值：成功返回0，失败返回错误码

实例:

1. #include <sqlite3.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #define MAX 100
6. int exec\_sql\_string(char \*sql\_string,sqlite3 \*db)
7. {
8. char \*errmsg,\*\*dbResult;
9. int nRow,nColumn;
10. int result,i,j,index;
11. result = sqlite3\_get\_table(db,sql\_string,&dbResult,&nRow,&nColumn,&errmsg);
12. if(0 != result){
13. fprintf(stderr,"Fail to exec sql(%s) : %s.\n",sql\_string,errmsg);
14. return -1;
15. }
17. //字段名字
18. for(j = 0;j < nColumn;j ++)
19. {
20. printf("%s\t",dbResult[j]);
21. }
22. printf("\n");
23. index = nColumn;//从它开始是字段对应的值
24. for(i = 0;i < nRow;i ++)//查询到总共记录个数
25. {
26. for(j = 0;j < nColumn;j ++)
27. {
28. printf("%s\t",dbResult[index]);
29. index ++;
30. }
31. printf("\n");
32. }
34. //释放查询结果所分配的内存
35. sqlite3\_free\_table(dbResult);
36. return 0;
37. }
38. int main(int argc,char \*argv[])
39. {
40. sqlite3 \*db;
41. int result;
42. char sql\_buf[MAX];
43. if(argc < 2)
44. {
45. fprintf(stderr,"usage : %s argv[1].\n",argv[0]);
46. exit(EXIT\_FAILURE);
47. }
48. result = sqlite3\_open(argv[1],&db);
49. if(result != SQLITE\_OK)
50. {
51. fprintf(stderr,"Fail to sqlite3\_open %s : %s.\n",argv[1],sqlite3\_errmsg(db));
52. exit(EXIT\_FAILURE);
53. }
54. while(1)
55. {
56. printf("sqlite >");
57. fgets(sql\_buf,sizeof(sql\_buf),stdin);
58. sql\_buf[strlen(sql\_buf) - 1] = '\0';
59. if(strncmp(sql\_buf,"quit",4) == 0)
60. break;
61. exec\_sql\_string(sql\_buf,db);
62. }
63. result = sqlite3\_close(db);
64. if(result != 0)
65. {
66. fprintf(stderr,"Fail to sqlite3\_open %s : %s.\n",argv[1],sqlite3\_errmsg(db));
67. exit(EXIT\_FAILURE);
68. }
69. exit(EXIT\_SUCCESS);
70. }

运行结果:

[](http://blog.chinaunix.net/attachment/201206/11/26833883_1339419750SFkg.png)