

下载安装

2017年3月15日 星期三 22:25

1. 下载MySQL.dmg版本，默认安装，记录最后的密码

2017-03-15T14:16:31.083690Z 1 [Note] A temporary password is generated for root@localhost: **t!!B/gZrW2Du**

If you lose this password, please consult the section How to Reset the Root Password in the MySQL reference manual.

2. 下载node.js 默认安装

3. 启动MySQL：点击“设置/mysql”，点击/start server

4. 将mysql加入环境变量PATH

1. 打开终端, 输入: `cd ~`

会进入~文件夹

2. 然后输入: `touch .bash_profile`

回车执行后,

2. 再输入: `open -e .bash_profile`

会在TextEdit中打开这个文件（如果以前没有配置过环境变量，那么这应该是一个空白文档）。如果有内容，请在结束符前输入，如果没有内容，请直接输入如下语句：

```
export PATH=${PATH}:/usr/local/mysql/bin
```

然后，保存，退出TextEdit（一定是退出），关闭终端并退出。

5. 更改编码

(1) `mysql>show variables like 'character%';`

查看

mysql> show variables like 'character%';		下载安装	1. 打开终端, 输入: cd ~
			会进入~文件夹
		默认操作	2. 然后输入: touch .bash_profile
		-D 打开指定数据库 -h 服...	回车执行后,
		常用命令	2. 再输入: open -e .bash_profile
		MySQL关键字、函数名称...	会在TextEdit中打开这个文件（如果以前没有配置过环境变量，那么这应该是一个空白文档）。如果有内容，请在结束符前输入，如果没有内容，请直接输入如下语句：
		数据类型	export PATH=\${PATH}:/usr/local/mysql/bin
		日期型 YEAR YY...	然后，保存，退出TextEdit（一定是退出），关闭终端并退出。
		创建数据库/库	
8 rows in set (0.01 sec)		先声明：黑色内容必须要有...	5. 更改编码

如果value不是utf8，则需要更改

(2) 偏好设置里面 stop mysql

(3) finder，导航栏/前往，输入"/usr/local/mysql/support-files"，将mysql-default.cnf拷贝到桌面上，重命名为my.cnf

(4) 用xcode打开，加入以下内容：

```
[client]
```

default-character-set=utf8

[mysqld]

default-storage-engine=INNODB

character-set-server=utf8

collation-server=utf8_general_ci

(5) 将这个文件拷贝到 "/etc"目录下

(6) 在终端输入 "chmod -r--r--r-- /etc/my.cnf"

(7) 重启stop mysql

简单操作

2017年3月15日 星期三 22:39

-D	打开指定数据库
-h	服务器名称 (IP地址)
-p	密码
-P	端口号,默认是3306
-u	用户名
-V	输出版本信息且退出

登录：

```
macdeMacBook-Air:~ mac$ mysql -u root -p
```

Enter password: (用户登录密码)

退出：

```
mysql> exit
```

```
mysql> quit
```

```
mysql> \q
```

修改提示符

```
mysql> prompt $
```

常用命令

2017年3月16日 星期四 11:17

- MySQL关键字，函数名称---大写
- 数据库名称，表名称，字段名称---小写
- SQL语句以分号结尾

SELECT VERSION();	显示服务器版本
SELECT NOW();	显示当前日期时间
SELECT USER();	显示当前用户
SHOW WARNINGS;	显示warning信息
SHOW CREATE TABLE tb_name;	显示之前创建表的操作信息
show global variables like 'port';	显示端口号

数据类型

2017年3月16日 星期四 12:54

整型

数据类型	存储范围	字节
TINYINT	有符号值: -128 到127 (-2^7 到 2^7-1) 无符号值: 0到255 (0 到 2^8-1)	1
SMALLINT	有符号值: -32768 到32767 (-2^{15} 到 $2^{15}-1$) 无符号值: 0到65535 (0 到 $2^{16}-1$)	2
MEDIUMINT	有符号值: -8388608 到8388607 (-2^{23} 到 $2^{23}-1$) 无符号值: 0到16777215 (0 到 $2^{24}-1$)	3
INT	有符号值: -2147483648 到2147483647 (-2^{31} 到 $2^{31}-1$) 无符号值: 0到4294967295 (0 到 $2^{32}-1$)	4
BIGINT	有符号值: -9223372036854775808 到9223373036854775807 (-2^{63} 到 $2^{63}-1$) 无符号值: 0到18446744073709551615 (0 到 $2^{64}-1$)	8

慕课网

浮点型

数据类型	存储范围
FLOAT[(M,D)]	-3.402823466E+38到-1.175494351E-38、0和1.175494351E-38到3.402823466E+38。 M是数字总位数，D是小数点后面的位数。如果M和D被省略，根据硬件允许的限制来保存值。单精度浮点数精确到大约7位小数位。
DOUBLE[(M,D)]	-1.7976931348623157E+308到-2.2250738585072014E-308、0和2.2250738585072014E-308到1.7976931348623157E+308。

慕课网

字符型

列类型	存储需求
CHAR(M)	M个字节, $0 \leq M \leq 255$
VARCHAR(M)	L+1个字节, 其中 $L \leq M$ 且 $0 \leq M \leq 65535$
TINYTEXT	L+1个字节, 其中 $L < 2^8$
TEXT	L+2个字节, 其中 $L < 2^{16}$
MEDIUMTEXT	L+3个字节, 其中 $L < 2^{24}$
LONGTEXT	L+4个字节, 其中 $L < 2^{32}$
ENUM('value1','value2',...)	1或2个字节, 取决于枚举值的个数(最多65,535个值)
SET('value1','value2',...)	1、2、3、4或者8个字节, 取决于set成员的数目(最多64个成员)

慕课网

日期型

YEAR	YYYY	1byte
TIME	HH:MM:SS	3bytes
DATE	YYYY:MM:DD	4bytes
TIMESTAMP	YYYY:MM:DD:HH:MM:SS	4bytes
DATETIME	YYYY:MM:DD:HH:MM:SS	8bytes

创建数据库/表

2017年3月16日 星期四 11:24

先声明：黑色内容必须要有，绿色内容是可有可无的，蓝色是用户自定义的内容，“|”表示选择

■ 数据库的创建和删除

//创建数据库

```
CREATE DATABASE|SCHEMA IF NOT EXISTS db_name CHARACTER SET  
charset_name ;
```

//进入某个数据库

```
USE db_name;
```

//查看当前服务器下的数据库列表

```
SHOW DATABASES;
```

//显示数据库创建的时候使用的编码方式

```
SHOW CREATE DATABASE db_name;
```

//修改数据库编码方式

```
ALTER DATABASE db_name CHARACTER SET charset_name;
```

```
mysql>ALTER DATABASE singer2 CHARACTER SET utf8;  
Query OK, 1 row affected (0.00 sec)
```

//删除数据库

```
DROP DATABASE IF NOT EXISTS db_name ;
```

数据表的操作

2017年3月16日 星期四 18:53

//创建数据表

CREATE TABLE IF NOT EXISTS table_name(col1 col1_def, col2 col2_def,...);

```
mysql> CREATE TABLE singer(  
-> singerName VARCHAR(20),  
-> age TINYINT,  
-> sex CHAR(8),  
-> company VARCHAR(20)  
-> );  
Query OK, 0 rows affected (0.05 sec)
```

//查看数据表列表

SHOW TABLES FROM db_name **LIKE** 'pattern' | **WHERE** expr ;

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_music |  
+-----+  
| singer          |  
+-----+  
1 row in set (0.00 sec)
```



```
mysql> SHOW TABLES FROM mysql;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv     |
| db               |
| engine_cost      |
| event            |
| func             |
| general_log      |
| gtid_executed    |
| help_category    |
| help_keyword     |
| help_relation    |
| help_topic       |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| plugin           |
| proc             |
| procs_priv       |
| proxies_priv     |
| server_cost      |
| servers          |
| slave_master_info |
| slow_log         |
| tables_priv      |
| time_zone        |
| time_zone_leap_second |
| time_zone_name   |
| time_zone_transition |
| time_zone_transition_type |
| user             |
+-----+
31 rows in set (0.00 sec)
```

//查看数据表的结构

SHOW COLUMNS FROM `table_name` ;

```
mysql> SHOW COLUMNS FROM singer;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| singerName | varchar(20)   | YES  |     | NULL    |       |
| age        | tinyint(4)    | YES  |     | NULL    |       |
| sex        | char(8)       | YES  |     | NULL    |       |
| company    | varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

//查看索引表

SHOW INDEXES FROM `tb_name` ;

```
mysql> SHOW INDEXES FROM test\G;
***** 1. row *****
      Table: test
      Non_unique: 0
      Key_name: PRIMARY
      Seq_in_index: 1
      Column_name: id
      Collation: A
      Cardinality: 0
      Sub_part: NULL
      Packed: NULL
      Null:
      Index_type: BTREE
      Comment:
      Index_comment:
1 row in set (0.00 sec)

ERROR:
No query specified
```

//查找记录

SELECT expr,... FROM table_name;

```
mysql> SELECT * FROM singer;
+-----+-----+-----+-----+
| singerName | age | sex   | company |
+-----+-----+-----+-----+
| luoxin     | 23  | female | WHU     |
| LinJunJie  | NULL | man   | NULL    |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

//自动编号AUTO_INCREMENT，默认起始为1，增量为1

AUTO_INCREMENT必须与主键PRIMARY KEY组合使用

主键必须是NOT NULL

```
mysql> CREATE TABLE music(
  -> id SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  -> song VARCHAR(30) NOT NULL,
  -> singer VARCHAR(20),
  -> album VARCHAR(30)
  -> );
Query OK, 0 rows affected (0.02 sec)
```

Field	Type	Null	Key	Default	Extra
id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
song	varchar(30)	NO		NULL	
singer	varchar(20)	YES		NULL	
album	varchar(30)	YES		NULL	

id	song	singer	album
1	beat it	NULL	NULL
2	As long as you love me	NULL	NULL

//添加单列

```
ALTER TABLE tb_name ADD COLUMN col_name column_definition
FIRST|AFTER col_name;
```

//添加多列

```
ALTER TABLE tb_name ADD COLUMN (col_name column_definition);
```

//删除表

```
DROP TABLE IF EXISTS tb_name;
```

//删除列

```
ALTER TABLE tb_name DROP col_name;
```

//修改表名

- ①. ALTER TABLE tb_name RENAME TO|AS new_tb_name ;
- ②. RENAME TABLE tb_name TO new_tb_name, tb_name2 TO new_tb_name2...;

//修改列定义

```
ALTER TABLE tb_name MODIFY COLUMN col_name col_definition
FIRST|AFTER col_name ;
```

//修改列定义及名称

```
ALTER TABLE tb_name CHANGE COLUMN old_col_name new_col_name
col_definition FIRST|AFTER col_name ;
```

约束

2017年3月17日 10:33

约束分为表级约束、列级约束

1. 主键约束PRIMARY KEY

不能为空值，一张表仅有一个主键

//添加主键约束

```
ALTER TABLE tb_name ADD CONSTRAINT symbol PRIMARY KEY index_type  
(index_col_name);
```

symbol是约束的名称，index_type是索引类型

//删除主键约束

```
ALTER TABLE tb_name DROP PRIMARY KEY;
```

2. 唯一约束UNIQUE KEY

可以为空值，一张表可以有多个唯一约束

//添加唯一约束

```
ALTER TABLE tb_name ADD CONSTRAINT symbol UNIQUE INDEX|KEY index_name  
index_type (index_col_name1,index_col_name2...);
```

//删除唯一约束

```
ALTER TABLE tb_name DROP INDEX|KEY key_name;
```

3. 外键约束FOREIGN KEY

a. 实现一对一，一对多的关系

b. 存储引擎必须是INNODB

c. 外键列和参照列必须有相似的数据类型，数字长度、是否有符号位必须相同；而字符的长度则可以不同

d. 外键列和参照列必须创建索引

e. 外键所在表为子表，参照列所在表为父表

// 在tb_child中定义外键约束:

```
FOREIGN KEY(key1) REFERENCES tb_father(primKey);
```

//添加外键约束

```
ALTER TABLE tb_name ADD CONSTRAINT symbol FOREIGN KEY index_name  
(index_col_name1,index_col_name2...) REFERENCES tb_father(primKey);
```

//删除外键约束

```
ALTER TABLE tb_name DROP FOREIGN KEY key_name;
```

```
mysql> SHOW CREATE TABLE test;
+-----+
| Table | Create Table
+-----+
| test  | CREATE TABLE `test` (
  `name` char(1) NOT NULL,
  `id` smallint(6) NOT NULL DEFAULT '1',
  `age` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `name` (`name`),
  CONSTRAINT `test_ibfk_1` FOREIGN KEY (`name`) REFERENCES `singerlist` (`name`
)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+

1 row in set (0.00 sec)
mysql> ALTER TABLE test DROP FOREIGN KEY test_ibfk_1;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> SHOW CREATE TABLE test;
+-----+
| Table | Create Table
+-----+
| test  | CREATE TABLE `test` (
  `name` char(1) NOT NULL,
  `id` smallint(6) NOT NULL DEFAULT '1',
  `age` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+

1 row in set (0.00 sec)
```

4. 默认约束DEFAULT

//添加默认约束

```
ALTER TABLE tb_name ALTER COLUMN col_name SET DEFAULT value;
```

//删除默认约束

```
ALTER TABLE tb_name ALTER COLUMN col_name DROP DEFAULT;
```

5. 非空约束NOT NULL

记录的操作

2017年3月17日 15:10

//添加记录

INSERT `table_name`(`column1,column2...`) VALUES(`val1, val2...`);

```
mysql> INSERT singer VALUES('luoxin',23,'female','WHU');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> INSERT singer(singerName,sex) VALUES('LinJunJie','man');  
Query OK, 1 row affected (0.00 sec)
```

//修改记录

UPDATE `tb_name` SET `col_name` WHERE 要更新的记录

//删除记录

DELETE FROM `tb_name` WHERE 要删除的记录

创建

2017年3月14日 13:37

1. 用对象资源管理器创建

- ①. 右击 ‘数据库’ > ‘新建数据库’
- ②. 输入数据库名称

数据库名称(N):						
所有者(O):	<默认值>					
<input checked="" type="checkbox"/> 使用全文索引(I)						
数据库文件(F):						
逻辑名称	文件类型	文件组	初始大小(MB)	自动增长/最大大小	路径	文件名
	行数据	PRIMARY	5	增量为 1 MB，增长无限制	...	E:\SQL server2012\MSSQL11.MSSQLSERVER\MSSQL\DATA\
_log	日志	不适用	2	增量为 10%，增长无限制	...	E:\SQL server2012\MSSQL11.MSSQLSERVER\MSSQL\DATA\

2. 用SQL命令创建

name：逻辑文件名，符合标识符的命名规则，在修改数据库文件时要利用它指定要修改的数据库文件。

filename：数据库文件要保存的路径及文件名。

size：初始数据库文件的大小。

maxsize：数据库文件的最大值。

filegrowth：数据库文件的自动增长率，可以是百分比，也可以是具体的值。

SQL 语句不区分大小写，每一项的分隔符是“逗号”，并且最后一项没有逗号。

正确输入后，按下键盘上的“F5”键或“执行”按钮，就可以执行该SQL 语句，创建指定数据库文件位置的数据库。

查看修改数据库属性

2017年3月14日 13:57

! 不可以修改数据库文件的文件类型，文件所在的文件组，路径及文件名。

1. 增加文件

// 添加日志文件

```
alter database MUSIC
add log file
(
    name = ChineseMusic_log,
    filename = 'E:\SQL server2012\MSSQL11.MSSQLSERVER\MSSQL\DATA
\ChineseMusic_log.mdf',
    filegrowth = 1%
)
```

// 添加数据文件

```
alter database MUSIC
add file
(
    name = ChineseMusic_data,
    filename = 'E:\SQL server2012\MSSQL11.MSSQLSERVER\MSSQL\DATA
\ChineseMusic_data.mdf',
    size = 4MB
)
```

2. 修改文件

```
alter database MUSIC
modify file
(
    name = ChineseMusic_log,
    size = 5MB,
    maxsize = 50MB
)
```

3. 删除文件

```
alter database MUSIC
remove ChineseMusic_data
```


4. 更改数据库名

```
alter database 原数据库名  
modify name = 新数据库名
```

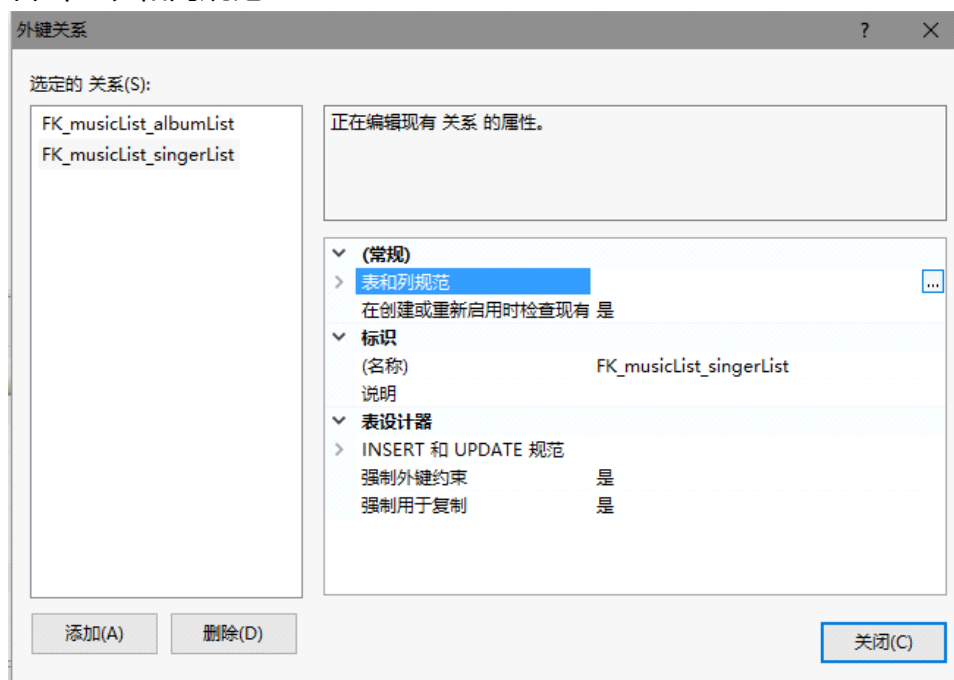
5. 数据库删除

```
drop database 数据库名
```

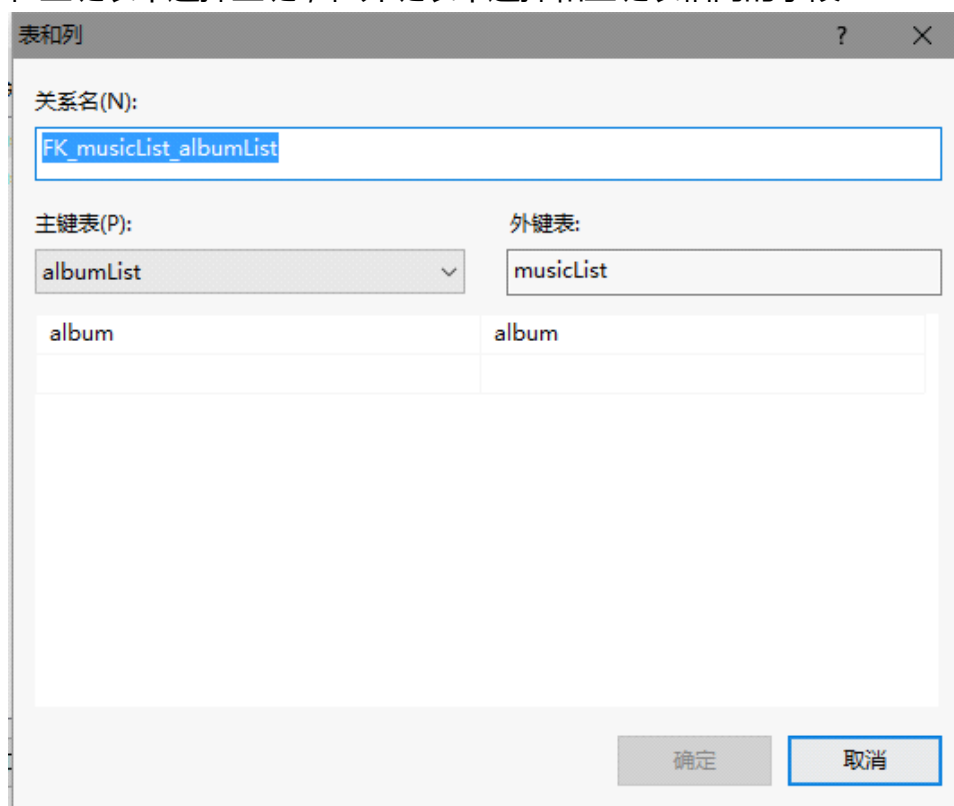
外键

2017年3月14日 17:18

1. 保证其中要建立外键关系的列与主键表中的数据类型完全一致
2. 打开要建立外键的表，右击某字段>‘关系’>‘添加’
3. 右击‘表和列规范’



4. 在主键表中选择主键，在外键表中选择和主键表相同的字段



索引

2017年3月14日 17:18

//在MUSIC数据库中，根据“musicList”表中字段“album”创建索引文件
“IX_album”

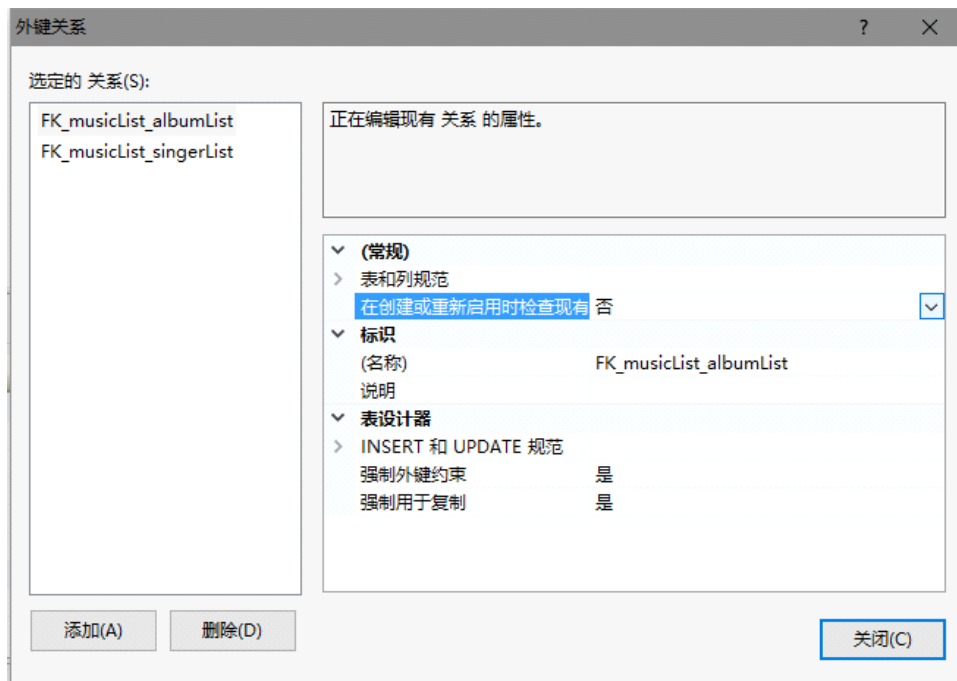
```
use MUSIC
```

```
create index IX_album on musicList(album desc)
```

数据库关系图

2017年3月14日 17:26

解决方法：在含有外键的表中右击外键，选择‘关系’，设置‘在创建或重新应用是检查现有’为否



```
create view MUSIC_dia
```

视图

2017年3月14日 17:34

```
create view view_MUSIC
```

游标

2017年3月14日 19:08

约束

2017年3月15日 10:06

- 默认约束：满足定值
- check约束： 满足某种关系表达式，例如 “grade>0 and grade<=100”
- 主键约束

1. 创建主键约束

Alter table 表名

Add Constraint PrimaryKey Primary Key (主键字段名)

PrimaryKey是用户自定义的主键完整性约束的名称

2. 删除主键约束

Alter table 表名

Drop Constraint PrimaryKey

关系型数据库

2017年9月19日 21:23

实体+联系

行：描述实体的实例

列：描述实体的特征/属性

Codd十二法则

- 1) 信息法则。信息表现为贮存在单元中的数据，正如前面所讨论过的，将VIN作为一个单个的列使用，违反了这条规则。
- 2) 授权存取法则。每一个数据项必须通过一个“表名+行主键+列名”的组合形式访问。例如，如果你能用数组或指针访问一个列，就违反这条规则。
- 3) 必须以一致的方式使用空值。如果由于缺少数字值，空值（Null）被当作0来处理，或者由于缺少字符值而被当作一个空格处理，那么它就违反了这条规则。空值仅仅是指缺少数据而且没有任何数值。如果缺少的数据需要值，软件提供商通常提供使用缺省值的能力满足这一目的。
- 4) 一个活跃的、在线数据字典应作为关系型表被存储，并且该字典应该可以通过常规的数据存取语言访问。如果数据字典的任何部分贮存在操作系统文件里，就违反了这条规则。
- 5) 除了可能的低级存取例程外，数据存取语言必须提供所有的存取方式，并且是存取的仅有方式。如果你能通过一个实用程序而不是一个SQL接口来存取支持一个表的文件，就有可能违反了本规则。参见规则12。
- 6) 所有能被更新的视图应当是可更新的。例如，如果你能将三个表连结起来，作为一个视图的基础，但却不能更新这个视图，则违反本规则。
- 7) 必须有集合级的插入、更新和删除。目前，大多数RDBMS提供商都在某种程度上提供了这种能力。
- 8) 物理数据的独立性。应用不能依赖于物理结构，如果一个支持某表的文件从一张盘移动到其他盘上或重新命名，不应该对应用产生影响。
- 9) 逻辑数据的独立性。应用不应依赖于逻辑结构。如果一个表必须被分成两个部分，那么应该提供一个视图，以把两段连接在一起，以便不会对应用产生影响。
- 10) 完整性的独立性。完整性规则应该贮存在数据字典中。主键约束、外键约束、检查约束、触发器等等都应该贮存在数据字典中。
- 11) 分布独立性。一个数据库即使被分布，也应该能继续工作。这是规则8的一个扩展，一个数据库不仅能在一个系统（本地地）分布，也能在通过系统的网络（远程地）分布。
- 12) 非破坏性法则。如果允许低级存取，一定不能绕过安全性或完整性规则，这些规则是常规的数据存取语言所遵守的

基本概念总览

2017年9月19日 9:20

```
CREATE TABLE tablename
(
    key1 type1,
    key2 type2,
    ...
    //约束定义
    PRIMARY KEY (key1)
    FOREIGN KEY (key2) REFERENCES table2 (key1)
    ...
)
```

索引

1. 索引是排序的
2. 索引可以提高数据查询的速度
3. 索引占据一定磁盘空间
4. 减慢了数据插入和删除的速度（因为每次都要更新索引）

表关联：两张表通过字段关联起来

（完整性）约束：保证了数据的准确性和一致性，即数据的accuracy和可靠reliability

- 主键：用于唯一标识一行数据（一条记录），即不同行数据的主键值不能重复, "PRIMARY KEY (keyname)"
【在Oracle中，主键必须添加非空约束】
- 复合主键：联合多个字段作为主键 "PRIMARY KEY(keyname1, keyname2)"
- 外键：关联到其他表主键的字段 FOREIGN KEY(keyname) REFERENCES table2 (keyname2)
- 非空约束：在类型定义后面加上"NOT NULL" 默认值：在类型定义后面加上 "DEFAULT **"

数据类型

2017年9月25日 10:59

■ 字符型

- CHAR (<=255Byte)
- VARCHAR
- NCHAR, 支持多字节和Unicode字符
- NVARCHAR

■ 数字型

- BIT
- TINYINT (0~255)
- SMALLINT (-32768~32767)
- INT (-2147483648~2147483647)
- REAL (4比特的浮点数)
- DOUBLE
- FLOAT
- DECIMAL(n, m) : n表示总长度 , m表示小数长度

■ 日期型

- DATE (XXXX-XX-XX)

```
VALUES ('1994-01-25');
```

- TIME

表

2017年9月25日 10:35

永久表	CREATE TABLE	
全局临时表	CREATE GLOBAL TEMPORARY TABLE	只有在SQL会话期间存在
局部临时表	CREATE LOCAL TEMPORARY TABLE	只有在SQL模块内才能访问，存留时间仅在会话期间

■ 表操作

1. 创建表

```
CREATE TABLE student(  
  student_ID CHAR(4),  
  name CHAR(20),  
  height DOUBLE,  
  weight DOUBLE,  
  birthday DATE,  
  sex CHAR(10),  
  gpa DECIMAL(4,3),  
  rank char(10)  
);
```

2. 删除表

```
DROP TABLE student;
```

3. 修改表名

```
ALTER TABLE studenttable  
  RENAME TO student;
```

■ 列操作

1. 增加新列

```
ALTER TABLE student  
  ADD birthplace VARCHAR(20);
```

2. 删除列

```
ALTER TABLE student  
  DROP birthplace;
```

3. 修改列名

```
ALTER TABLE student  
  CHANGE COLUMN gpa GPA DECIMAL(4,3);
```

4. 修改列字段信息

```
ALTER TABLE student  
    MODIFY student_ID CHAR(4) NOT NULL
```

5. 选取某列/某几列数据

```
SELECT TNO, TNAME FROM teacher;
```

? 去重处理 DISTINCT

```
SELECT DISTINCT TNAME FROM teacher;
```

行操作

1. 插入数据

```
INSERT INTO student  
VALUES ('01', 'luoxin', 158, 44.5, '1994-01-25', 'female', 3.68, '9/80');
```

2. 拷贝插入多行数据

```
INSERT INTO myfriend(fname, sex)  
SELECT name, sex FROM student;
```

```
INSERT INTO myfriend  
VALUES ('lianghuan', 'female', 30),  
       ('yaomeihua', 'female', 56);
```

! 含有外键的表在插入数据时要注意，外键值必须在其目标表中存在

索引

2017年9月19日 19:23

索引是一个单独的、物理的数据库结构，是数据库的一个表中所包含的值的列表

- 簇索引（只能有一个）：对表的所有数据按列进行重新排序，以便与索引的排序相匹配
基于簇索引的表占用最小的磁盘空间，列值在进行查询时效率更高，不再需要ORDER BY语句
- 非簇索引（默认）：索引页存储了关键字的值和行定位器，没有改变数据存放的物理位置

适合场景：

1. 待检索的字段的数据包含很多重复值、空值
2. 大的数据表，且检索结果< 数据总数25%以内
3. WHERE子句所在处、
4. 先装入数据，后建索引
5. 进行大量更新时，先销毁索引，待更新完毕后再创建索引
6. 尽量把表和索引存放在不同磁盘上

只允许对固定长度字符变量创建索引

❖ UNIQUE (DISTINCT) 唯一索引：不允许表中不同行在索引列上取相同值，若已有相同值存在，则系统给出相关信息，不建此索引

主键是一种约束，唯一索引是一种索引，两者在本质上是不同的。

主键创建后一定包含一个唯一性索引，唯一性索引并不一定就是主键。

唯一性索引列允许空值，而主键列不允许为空值。

主键列在创建时，已经默认为空值 + 唯一索引了。

主键可以被其他表引用为外键，而唯一索引不能。

一个表最多只能创建一个主键，但可以创建多个唯一索引。

主键更适合那些不容易更改的唯一标识，如自动递增列、身份证号等。

❖ ASC/DESC：索引表中的值的排序次序（默认是升序ASC）

1. 创建索引

```
CREATE INDEX rank_Index ON student(rank);
```

```
CREATE INDEX WeightHeight_Index ON student(weight, height);
```

```
i GPA_Index (GPA)
i rank_Index (rank)
i WeightHeight_Index (weig
```

2. 使用索引

```
SELECT GPA FROM student;
```

3. 销毁索引

```
ALTER TABLE department  
DROP INDEX CollegeLeader_index;
```

约束

2017年9月19日 21:28

■ 创建/增加主键

```
ALTER TABLE student  
    ADD PRIMARY KEY (student_ID);
```

```
CREATE TABLE T_boss(  
    ID VARCHAR(20),  
    name VARCHAR(20) NOT NULL ,  
    age SMALLINT,  
    sex CHAR(5),  
    salary FLOAT,  
    department VARCHAR(20),  
    PRIMARY KEY (ID),  
    FOREIGN KEY (department) REFERENCES T_department(FId)  
);
```

■ 创建/增加外键

```
ALTER TABLE T_employee  
    ADD FOREIGN KEY (Fboss) REFERENCES T_boss(ID);
```

视图

2017年9月19日 19:36

视图时从一个或多个表中导出来的表，这些数据列/行源于其所引用的表。

视图时一张“虚表”。

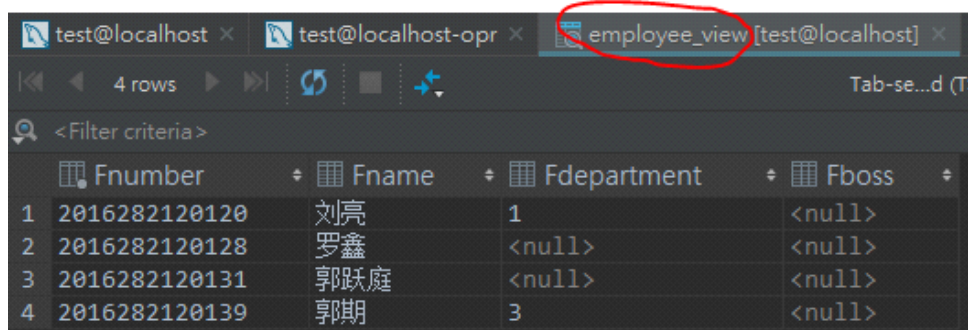
优点：

1. 简化操作
2. 定制数据（让不同的用户以不同的方式看到不同的数据集）
3. 合并分隔的数据
4. 安全性（通过视图，用户只能查看和修改他们所看到的数据）

缺点：

1. 性能：在执行查询操作时，如果查询对象引用了视图，那么DBMS还要执行视图的查询
2. 更新限制：只能更新基于单个表的视图，视图不能使用聚集函数、计算得到的列或SELECT DISTINCT语句

```
CREATE VIEW employee_view AS SELECT * FROM T_employee;
```



	Fnumber	Fname	Fdepartment	Fboss
1	2016282120120	刘亮	1	<null>
2	2016282120128	罗鑫	<null>	<null>
3	2016282120131	郭跃庭	<null>	<null>
4	2016282120139	郭期	3	<null>

简单查询

2017年9月19日 21:28

```
SELECT [DISTINCT | ALL] select_list  
FROM table_name1, table_name2...  
[WHERE search_condition]  
[GROUP BY group_by_expression]  
[HAVING search_condition]  
[ORDER BY order_expression [ASC|DESC] ] 排序
```

DISTINCT: 删除查询结果中相同的行

ALL : 返回查询结果所有行

ASC : 升序

DESC : 降序

排序

2017年9月28日 19:31

ORDER BY colname;

基本概念

2017年9月19日 21:43

触发器是一种特殊的存储过程，它在表的数据发生变化时作用，可以维护数据的完整性

触发器以独立的对象存储

触发器由事件驱动运行

触发器不能接受参数

INSERT
DELETE
UPDATE