iOS中的数据存储方式

- Plist（NSArray\NSDictionary）,只能存储数组，字典，但是数组和字典里面不能有自定义对象
- Preference（偏好设置\NSUserDefaults）
- NSCoding（NSKeyedArchiver\NSkeyedUnarchiver）
- SQLite3
- Core Data

SQLite

- 什么是SQLite
- SQLite是一款轻型的嵌入式数据库
- 它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了
- 它的处理速度比Mysql、PostgreSQL这两款著名的数据库都还快

- 什么是数据库
- 数据库（Database）是按照数据结构来组织、存储和管理数据的仓库
- 数据库可以分为2大种类
- ✓ 关系型数据库（主流）
- ✓ 对象型数据库

- 常用关系型数据库
- PC端：Oracle、MySQL、SQL Server、Access、DB2、Sybase
- 嵌入式\移动客户端：SQLite

如何存储数据

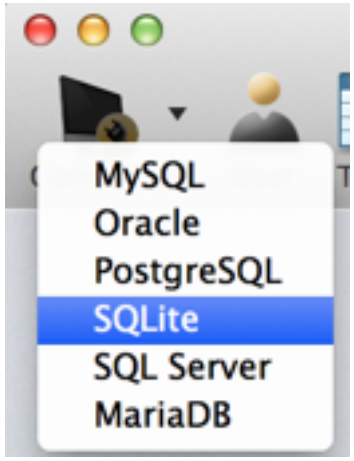
- 数据库是如何存储数据的
- 数据库的存储结构和excel很像，以表（table）为单位

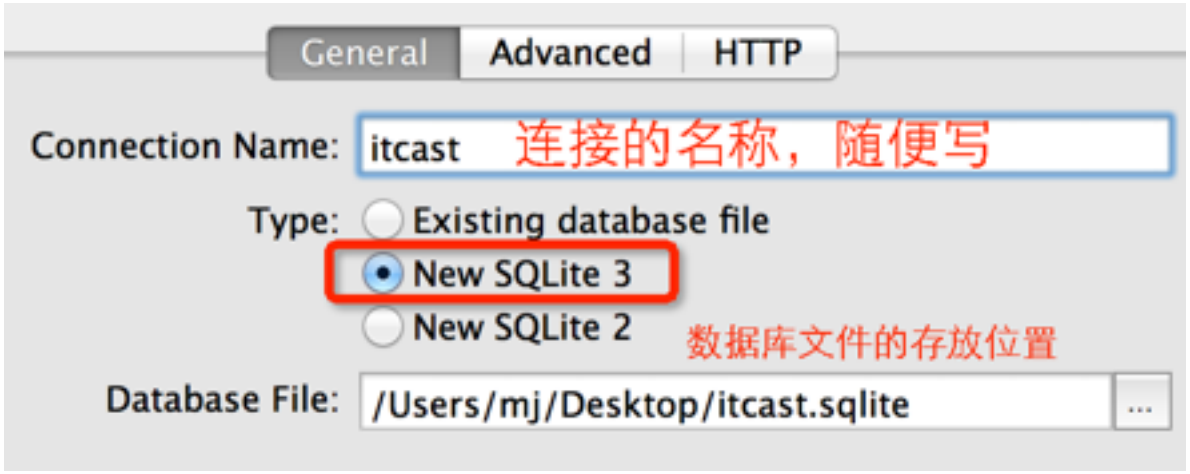
- 数据库存储数据的步骤
- 新建一张表（table）
- 添加多个字段（column，列，属性）
- 添加多行记录（row，每行存放多个字段对应的值）

Navicat

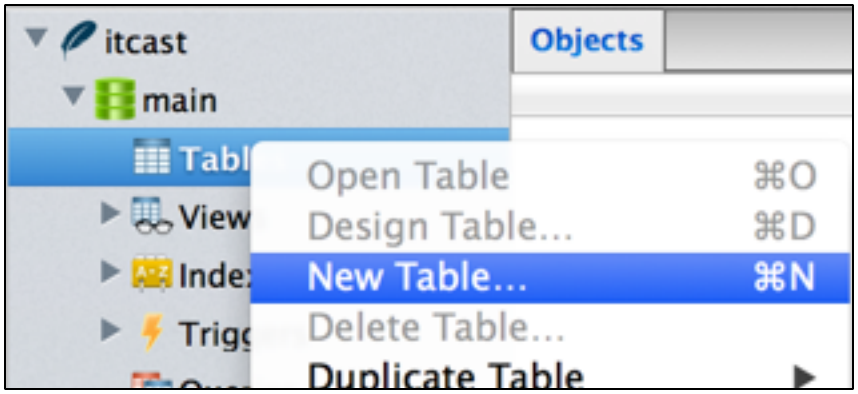
- Navicat是一款著名的数据库管理软件，支持大部分主流数据库（包括SQLite）

- 利用Navicat建立数据库连接



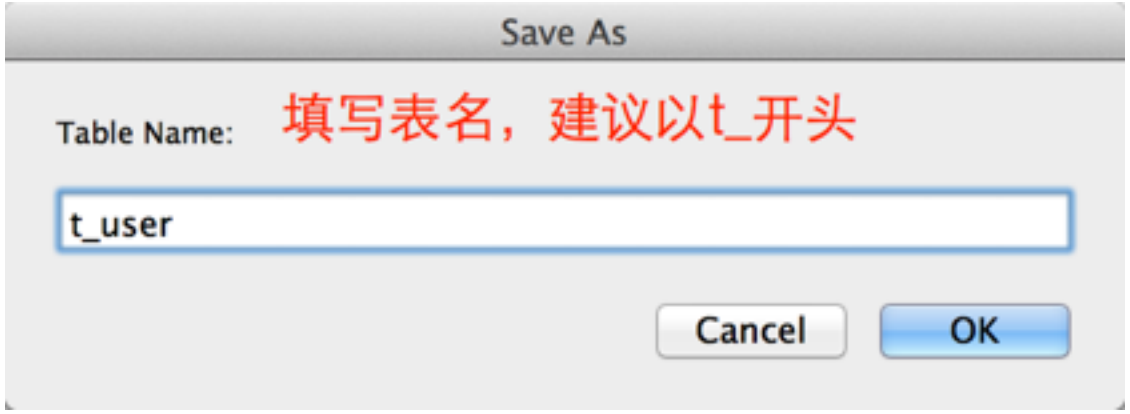


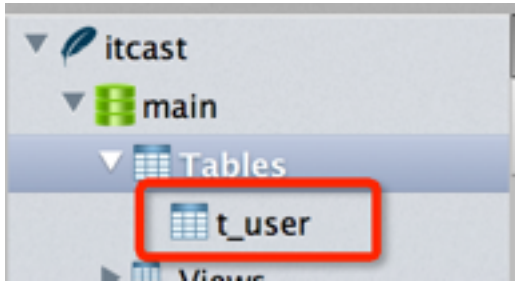
建表


itcast
Objects
main
Tabl
Open Table ⌘O
Design Table... ⌘D
New Table... ⌘N
Delete Table...
Duplicate Table

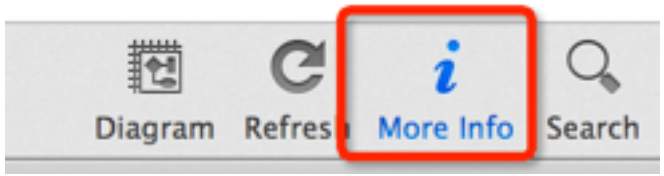

Fields
Indexes
Foreign Keys
Uniques
Name
Type
Length
Decimals
Not Null
Key
id
INTEGER
0
0
1
name
text
0
0
填写表字段

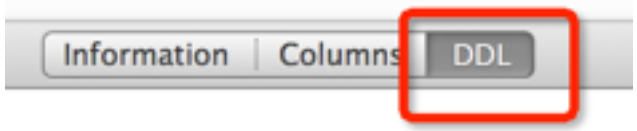
建表


Save As
Table Name:
填写表名，建议以t_开头
t_user
Cancel
OK

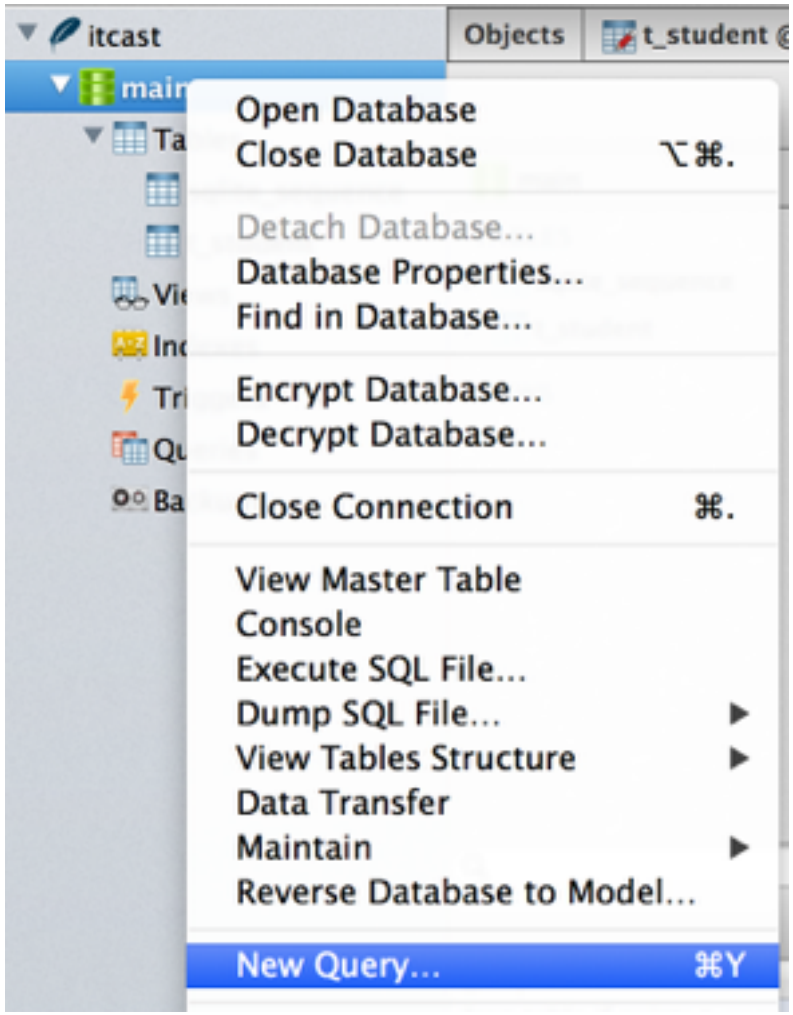

itcast
main
Tables
t_user

查看DDL





执行SQL语句


itcast
Objects
t_student
Open Database
Close Database ⌥⌘.
Detach Database...
Database Properties...
Find in Database...
Encrypt Database...
Decrypt Database...
Close Connection ⌘.
View Master Table
Console
Execute SQL File...
Dump SQL File...
View Tables Structure
Data Transfer
Maintain
Reverse Database to Model...
New Query... ⌘Y

课堂练习

- 新建1张学生表，包含字段
- id： 学号（唯一标识，要求自动增长）
- name： 姓名
- age： 年龄
- score： 分数

- 录入下列数据到数据库中

id	name	age	score
1	***jack***	***19***	***95.5***
2	rose	18	100
3	***jim***	***20***	***98***

SQL语句

- 如何在程序运行过程中操作数据库中的数据
- 那得先学会使用SQL语句

- 什么是SQL
- SQL（structured query language）： 结构化查询语言
- SQL是一种对关系型数据库中的数据进行定义和操作的语言
- SQL语言简洁，语法简单，好学好用

- 什么是SQL语句
- 使用SQL语言编写出来的句子\代码，就是SQL语句
- 在程序运行过程中，要想操作（增删改查，CRUD）数据库中的数据，必须使用SQL语句

SQL语句

- SQL语句的特点
  - 不区分大小写（比如数据库认为user和UsEr是一样的）
  - 每条语句都必须以分号；结尾

- SQL中的常用关键字有
  - select、insert、update、delete、from、create、where、desc、order、by、group、table、alter、view、index等等

- 数据库中不可以使用关键字来命名表、字段

SQL语句的种类

- 数据定义语句（DDL：Data Definition Language）
  - 包括create和drop等操作
  - 在数据库中创建新表或删除表（create table或 drop table）

- 数据操作语句（DML：Data Manipulation Language）
  - 包括insert、update、delete等操作
  - 上面的3种操作分别用于添加、修改、删除表中的数据

- 数据查询语句（DQL：Data Query Language）
  - 可以用于查询获得表中的数据
  - 关键字select是DQL（也是所有SQL）用得最多的操作
  - 其他DQL常用的关键字有where，order by，group by和having

创表

- 格式

p create table 表名 (字段名1 字段类型1, 字段名2 字段类型2, …) ;

p create table if not exists 表名 (字段名1 字段类型1, 字段名2 字段类型2, …) ;

- 示例

p create table t_student (id integer, name text, age inetger, score real) ;

字段类型

- SQLite将数据划分为以下几种存储类型：
- integer：整型值
- real：浮点值
- text：文本字符串
- blob：二进制数据（比如文件）

- 实际上SQLite是无类型的
- 就算声明为integer类型，还是能存储字符串文本（主键除外）
- 建表时声明啥类型或者不声明类型都可以，也就意味着创表语句可以这么写：
- ✓ create table t_student(name, age);

- 为了保持良好的编程规范、方便程序员之间的交流，编写建表语句的时候最好加上每个字段的具体类型

删表

- 格式
  - drop table 表名 ;
  - drop table if exists 表名 ;

- 示例
  - drop table t_student ;

插入数据（insert）

- 格式

p insert into 表名 (字段1, 字段2, …) values (字段1的值, 字段2的值, …) ;

- 示例

p insert into t_student (name, age) values (‘mj’, 10) ;

- 注意

p 数据库中的字符串内容应该用单引号 ’ 括住

更新数据（update）

- 格式

p update 表名 set 字段1 = 字段1的值, 字段2 = 字段2的值, … ;

- 示例

p update t_student set name = ‘jack’, age = 20 ;

- 注意

p 上面的示例会将t_student表中所有记录的name都改为jack，age都改为20

删除数据（delete）

- 格式

p delete from 表名 ;

- 示例

p delete from t_student ;

- 注意

p 上面的示例会将t_student表中所有记录都删掉

条件语句

- 如果只想更新或者删除某些固定的记录，那就必须在DML语句后加上一些条件

- 条件语句的常见格式

p where 字段 = 某个值 ;

p where 字段 is 某个值 ; // is 相当于 =

p where 字段 != 某个值 ;

p where 字段 is not 某个值 ; // is not 相当于 !=

p where 字段 > 某个值 ;

p where 字段1 = 某个值 and 字段2 > 某个值 ;

p where 字段1 = 某个值 or 字段2 = 某个值 ;

条件语句练习

- 示例

p 将t_student表中年龄大于10 并且 姓名不等于jack的记录，年龄都改为 5

✓ update t_student set age = 5 where age > 10 and name != ‘jack’ ;

p 删除t_student表中年龄小于等于10 或者 年龄大于30的记录

✓ delete from t_student where age <= 10 or age > 30 ;

- 猜猜下面语句的作用

p update t_student set score = age where name = ‘jack’ ;

✓ 将t_student表中名字等于jack的记录，score字段的值 都改为 age字段的值

DQL语句

- 格式

p select 字段1, 字段2, … from 表名 ;

p select * from 表名; // 查询所有的字段

- 示例

p select name, age from t_student ;

p select * from t_student ;

p select * from t_student where age > 10 ; // 条件查询

起别名

- 格式(字段和表都可以起别名)

p select 字段1 别名 , 字段2 别名 , … from 表名 别名 ;

p select 字段1 别名, 字段2 as 别名, … from 表名 as 别名 ;

p select 别名.字段1, 别名.字段2, … from 表名 别名 ;

- 示例

p select name myname, age myage from t_student ;

✓ 给name起个叫做myname的别名，给age起个叫做myage的别名

p select s.name, s.age from t_student s ;

✓ 给t_student表起个别名叫做s，利用s来引用表中的字段

计算记录的数量

- 格式

p select count (字段) from 表名 ;

p select count (*) from 表名 ;

- 示例

p select count (age) from t_student ;

p select count (*) from t_student where score >= 60;

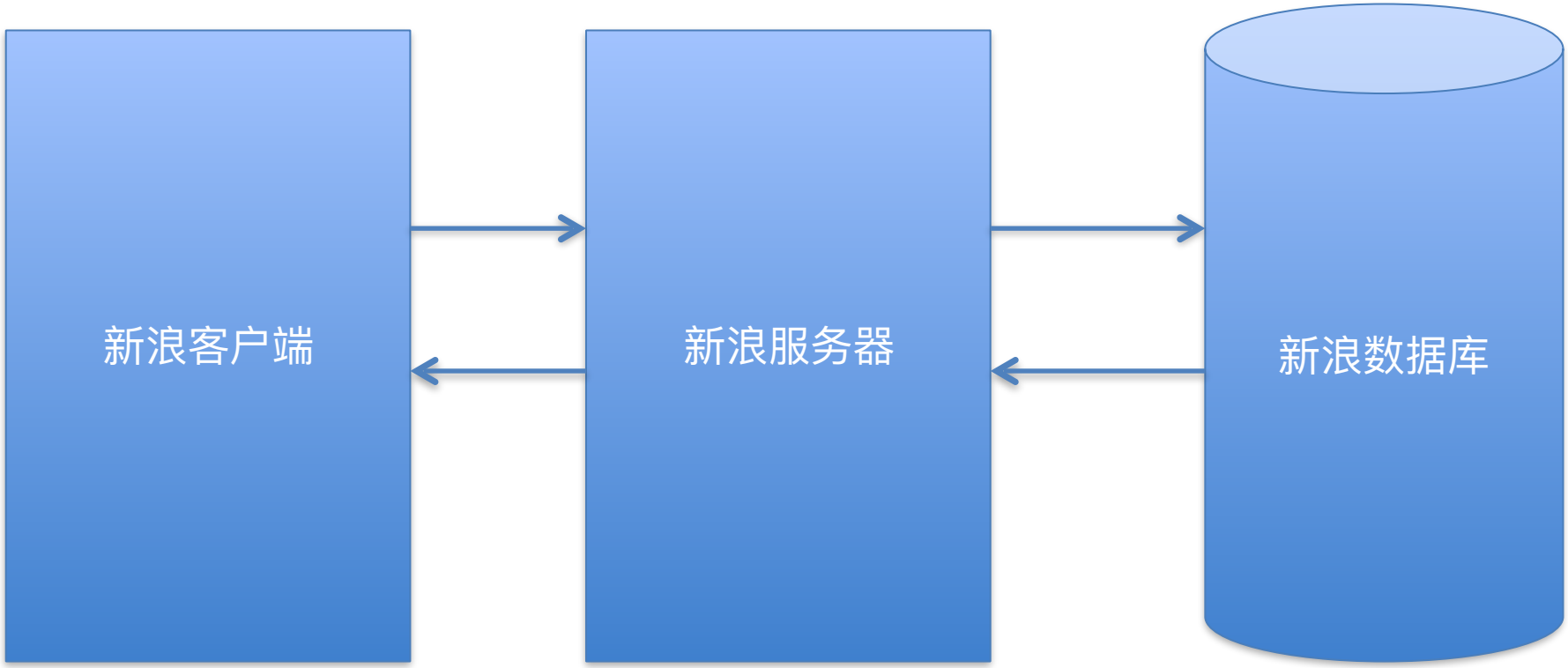
排序

- 按照某个字段的值，进行排序搜索
  - select * from t_student order by 字段 ;
  - ✓ select * from t_student order by age ;

- 默认是按照升序排序（由小到大），也可以变为降序（由大到小）
  - select * from t_student order by age desc ; //降序
  - select * from t_student order by age asc ; // 升序（默认）

- 也可以用多个字段进行排序
  - select * from t_student order by age asc, height desc ;
  - ✓ 先按照年龄排序（升序），年龄相等就按照身高排序（降序）

Select * from t_status where id > sinceID order by id desc limit 20



limit

- 使用limit可以精确地控制查询结果的数量，比如每次只查询10条数据

- 格式

p select * from 表名 limit 数值1, 数值2 ;

- 示例

p select * from t_student limit 4, 8 ;

✓ 可以理解为：跳过最前面4条语句，然后取8条记录

limit

- limit常用来做分页查询，比如每页固定显示5条数据，那么应该这样取数据
  - 第1页：limit 0, 5
  - 第2页：limit 5, 5
  - 第3页：limit 10, 5
  - …
  - 第n页：limit 5*(n-1), 5

- 猜猜下面语句的作用
  - select * from t_student limit 7 ;
    - ✓ 相当于select * from t_student limit 0, 7 ;
    - ✓ 表示取最前面的7条记录

简单约束

- 建表时可以给特定的字段设置一些约束条件，常见的约束有
- not null： 规定字段的值不能为null
- unique： 规定字段的值必须唯一
- default： 指定字段的默认值

（建议：尽量给字段设定严格的约束，以保证数据的规范性）

- 示例
- create table t_student (id integer, name text not null unique, age integer not null default 1) ;
- ✓ name字段不能为null，并且唯一
- ✓ age字段不能为null，并且默认为1

主键约束

- 如果t_student表中就name和age两个字段，而且有些记录的name和age字段的值都一样时，那么就没法区分这些数据，造成数据库的记录不唯一，这样就不方便管理数据

- 良好的数据库编程规范应该要保证每条记录的唯一性，为此，增加了主键约束
- 也就是说，每张表都必须有一个主键，用来标识记录的唯一性

- 什么是主键
- 主键（Primary Key，简称PK）用来唯一地标识某一条记录
- 例如t_student可以增加一个id字段作为主键，相当于人的身份证
- 主键可以是一个字段或多个字段

主键的设计原则

- 主键应当是对用户没有意义的
- 永远也不要更新主键
- 主键不应包含动态变化的数据
- 主键应当由计算机自动生成

主键的声明

- 在创表的时候用primary key声明一个主键
- create table t_student (id integer primary key, name text, age integer) ;
- integer类型的id作为t_student表的主键

- 主键字段
- 只要声明为primary key，就说明是一个主键字段
- 主键字段默认就包含了not null 和 unique 两个约束

- 如果想要让主键自动增长（必须是integer类型），应该增加autoincrement
- create table t_student (id integer primary key autoincrement, name text, age integer) ;

外键约束

- 利用外键约束可以用来建立表与表之间的联系
- 外键的一般情况是：一张表的某个字段，引用着另一张表的主键字段

- 新建一个外键
- create table t_student (id integer primary key autoincrement, name text, age integer, class_id integer, constraint fk_t_student_class_id_t_class_id foreign key (class_id) references t_class (id);
- ✓ t_student表中有一个叫做fk_t_student_class_id_t_class_id的外键
- ✓ 这个外键的作用是用t_student表中的class_id字段引用t_class表的id字段

表连接查询

- 什么是表连接查
- 需要联合多张表才能查到想要的数据

- 表连接的类型
- 内连接：inner join 或者 join （显示的是左右表都有完整字段值的记录）
- 左外连接：left outer join （保证左表数据的完整性）

- 示例
- 查询0316iOS班的所有学生
- ✓ select s.name,s.age from t_student s, t_class c where s.class_id = c.id and c.name = '0316iOS';