嵌入式数据库

李龙海

西安电子科技大学

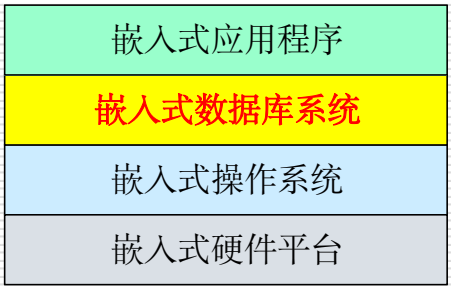
本章要点

- 嵌入式数据库基本知识：
  - 嵌入式数据库的产生
  - 嵌入式数据库的特点
  - 嵌入式数据库的分类
  - 常用的嵌入式数据库
- SQLite嵌入式数据库：（实验）
  - SQLite应用程序设计方法
  - 常用的SQLite API
  - SQLite安装与移植

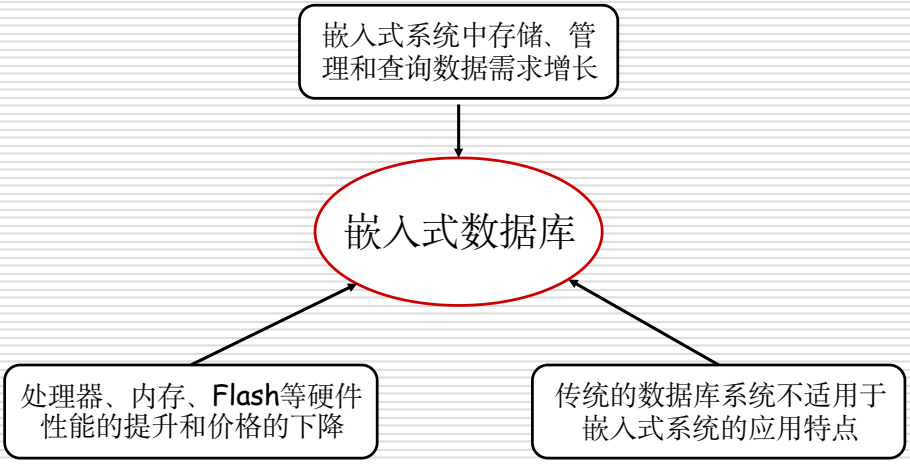
什么是嵌入式数据库？

数据库系统（DBMS）：把数据按照特定模型在存储介质中进行存储组织和管理的软件系统；为应用程序提供简便的数据定义、查询和修改接口；为管理员提供数据管理工具。

嵌入式数据库系统：应用于嵌入式系统的数据库系统称为嵌入式数据库系统，简称为嵌入式数据库。



嵌入式数据库的产生



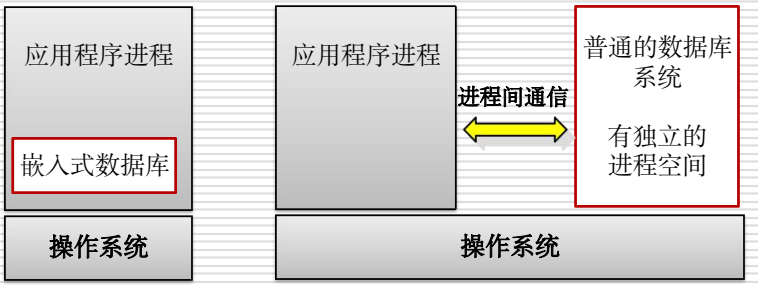
嵌入式数据库解决什么问题?

假定你需要在自己设计的嵌入式应用程序中管理大量的数据（比如联系人信息、通话记录、日志等），如果没有数据库系统支持，你该如何实现?

- 数据怎么存?
  - 存储介质：内存、Flash存储器（有文件系统支持？）
  - 数据组织方式：数组、链表
- 数据怎么取?
  - 数据查询功能的实现：折半查找、索引、BST、B树
- 数据怎么修改?
  - 数据一致性、原子性、隔离性、持久性、安全性
  - 如何实现锁？事务？

嵌入式数据库的特点

- 嵌入性
  - 一般与应用程序处于同一个进程上下文和地址空间
  - 常常表现为静态库或动态库形式，或者以源代码形式与应用程序源代码共同编译成可执行文件



嵌入式数据库的特点（cont）

- 精简
  - 编译之后常常只有几百KB
- 功能做了裁剪（与普通的DBMS比）
- 可移植性好
- 实时性

嵌入式数据库的分类

- 按数据存储介质分：
  - 内存式数据库
  - 文件式数据库
  - 网络式数据库
- 按数据模型分：
  - 关系型数据库（对象型数据库）
  - 非关系型数据库

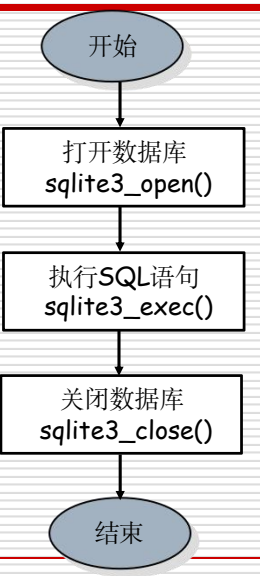
常用的嵌入式数据库

- Berkeley DB:
  - 开源文件型数据库
  - 嵌入应用程序进程内（300KB左右）
  - 简单的API接口（旧版本不支持SQL）
  - 可以管理2TB数据
  - 快速高效
- SQLite:
  - 开源关系型数据库（内存、文件都支持）
  - 支持SQL、支持ACID、支持事务、触发器
  - 嵌入应用程序进程内（250KB左右）

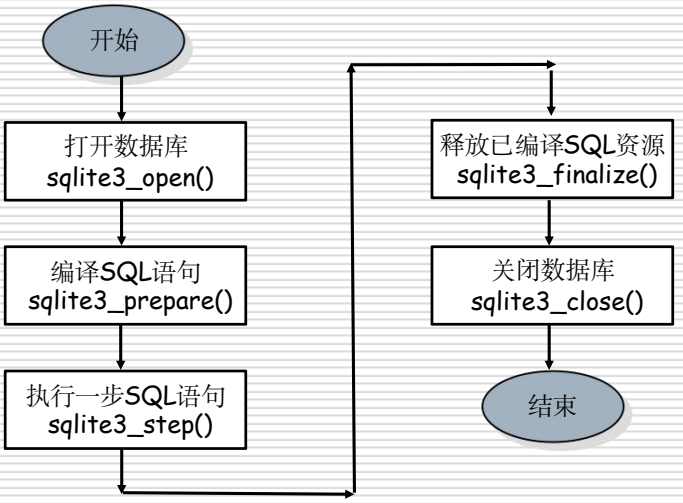
常用的嵌入式数据库（cont）

- eXtremeDB:
  - 内存嵌入式实时数据库
  - 高性能、低开销、稳定可靠
- Empress
- mSQL

SQLite应用程序设计方法1



SQLite应用程序设计方法2



SQLite常用API

- int sqlite3_open(
  const char *filename, /* Database filename (UTF-8) */
  sqlite3 **ppDb /* OUT: SQLite db handle */
  );
- int sqlite3_close(
  sqlite3* /* SQLite db handle */
  );

西安电子科技大学 XIDIAN UNIVERSITY

SQLite常用API

- int sqlite3_exec(
  sqlite3*, /* An open database */
  const char *sql, /* SQL to be evaluated */
  int (*callback)(void*,int,char**,char**),
  /* Callback function */
  void *, /* 1st argument to callback */
  char **errmsg /* Error msg written here */
  );

SQLite常用API

- int sqlite3_prepare(
  sqlite3 *db, /* Database handle */
  const char *zSql, /* SQL statement */
  int nByte, /* length of zSql in bytes. */
  sqlite3_stmt **ppStmt, /* OUT: Statement handle */
  const char **pzTail /* OUT: Pointer to unused portion of zSql */
  );

西安电子科技大学 XIDIAN UNIVERSITY

SQLite常用API

- int sqlite3_step(sqlite3_stmt* pStmt);
  - Return SQLITE_ROW or SQLITE_DONE or SQLITE_ERROR
- int sqlite3_finalize(sqlite3_stmt *pStmt);

SQLite常用API

- const void *sqlite3_column_blob(sqlite3_stmt*, int iCol);
- int sqlite3_column_bytes(sqlite3_stmt*, int iCol);
- int sqlite3_column_bytes16(sqlite3_stmt*, int iCol);
- double sqlite3_column_double(sqlite3_stmt*, int iCol);
- int sqlite3_column_int(sqlite3_stmt*, int iCol);
- sqlite3_int64 sqlite3_column_int64(sqlite3_stmt*, int iCol);
- const unsigned char *sqlite3_column_text(sqlite3_stmt*, int iCol);
- const void *sqlite3_column_text16(sqlite3_stmt*, int iCol);
- int sqlite3_column_type(sqlite3_stmt*, int iCol);
- sqlite3_value *sqlite3_column_value(sqlite3_stmt*, int iCol);