刘晓松　樊茗玥　主编

Visual FoxPro 程序设计

Visual FoxPro Programming

江苏大学出版社
JIANGSU UNIVERSITY PRESS

江苏大学英文教材出版基金资助出版

刘晓松　樊茗玥　主编

Visual FoxPro 程序设计

Visual FoxPro Programming

江苏大学出版社
JIANGSU UNIVERSITY PRESS
镇　江

图书在版编目(CIP)数据

Visual FoxPro 程序设计＝Visual FoxPro Programming：英文／刘晓松，樊茗玥主编. —镇江：江苏大学出版社，2017.9
ISBN 978-7-5684-0581-2

Ⅰ.①V… Ⅱ.①刘… ②樊… Ⅲ.①关系数据库系统—程序设计—高等学校—教材—英文 Ⅳ.①TP311.138

中国版本图书馆 CIP 数据核字(2017)第 238572 号

Visual FoxPro 程序设计
Visual FoxPro Programming

主　　编/刘晓松　樊茗玥
责任编辑/孙文婷
出版发行/江苏大学出版社
地　　址/江苏省镇江市梦溪园巷 30 号(邮编：212003)
电　　话/0511-84446464(传真)
网　　址/http://press.ujs.edu.cn
排　　版/镇江文苑制版印刷有限责任公司
印　　刷/镇江文苑制版印刷有限责任公司
开　　本/787 mm×1 092 mm　1/16
印　　张/15
字　　数/610 千字
版　　次/2017 年 9 月第 1 版　2017 年 9 月第 1 次印刷
书　　号/ISBN 978-7-5684-0581-2
定　　价/40.00 元

如有印装质量问题请与本社营销部联系(电话：0511-84440882)

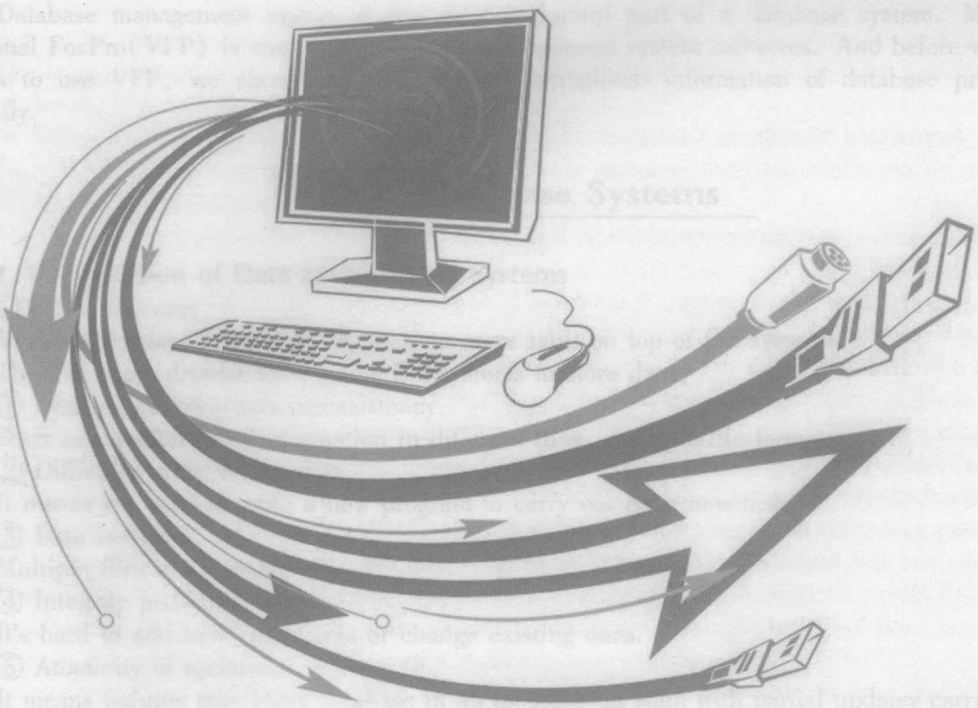
CONTENTS

Section 1 Visual FoxPro Fundamentals

Section 2 Practice

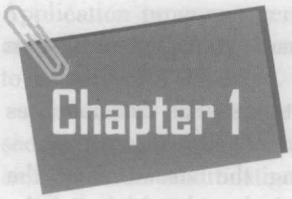
Section 1

Visual FoxPro Fundamentals



Section I

Visual FoxPro Fundamentals



Overview of Database Principles

Nowadays, Database system touches all aspects of our lives. Such as:

- Banking: all transactions.
- Airlines: reservations, schedules.
- Universities: registration, grades.
- Sales: customers, products, purchases.
- Manufacturing: production, inventory, orders, supply chain.
- Human resources: employee records, salaries, tax deductions.

Database management system is the most important part of a database system. Microsoft Visual FoxPro(VFP) is one of the database management system softwares. And before we learn how to use VFP, we should talk about the foundational information of database principles firstly.

1.1 Database Systems

1.1.1 Evolution of Data Management Systems

(1) File Systems

In the early days, database applications were built on top of file systems.

There're some drawbacks of using file systems to store data:

① Data redundancy and inconsistency.

Such as duplication of information in different files, multiple file formats.

② Difficulty in accessing data.

It means we need to write a new program to carry out each new task.

③ Data isolation.

Multiple files and formats.

④ Integrity problems.

It's hard to add new constraints or change existing ones.

⑤ Atomicity of updates.

It means failures may leave database in an inconsistent state with partial updates carried out.

⑥ Concurrent access by multiple users.

Concurrent accessed needed for performance, and uncontrolled concurrent accesses can lead to inconsistencies.

⑦ Security problems.

(2) Database Systems

With the development of technology, the first commercial database management system (DBMS) appeared in the late 1960's.

The early DBMS's, evolving from file systems, encouraged the user to visualize data much as it was stored.

These DBMS's used several different data models for describing the structure of the information in a database, chief among them the "hierarchical" or tree-based model and the graph-based "network" model.

A problem with these early models and systems was that they did not support high-level query languages.

Following a famous paper written by Edgar Frank Codd in 1970, database systems changed significantly. Codd proposed that database systems should present the user with a view of data organized as tables called relations.

Nowadays, almost all database systems use the relational data model.

Advantages of using a DBMS to manage data:

① Data independence.

Provide an abstract view of data to hide the details of data representation and storage.

② Efficient data access.

Store and retrieve data efficiently.

③ Data integrity and security.

It enforces integrity constraints and access controls.

④ Data administration.

Minimize redundancy and make retrieval efficient.

⑤ Concurrent access and crash recovery.

⑥ Reduced application development time.

1.1.2 Important Database Concepts

A number of important database concepts should be learned:

(1) Data

In brief, data is information recorded in the media.

(2) Database(DB)

What is a database? In essence, a database is nothing more than a collection of information that exists over a long period of time, often many years. In common parlance, the term database refers to a collection of data that is managed by a DBMS.

A database is simply a collection of data that is stored on one or more computers. A database can contain any sort of data, such as university student records, a library card catalog, or an individual address book.

(3) Database Management System(DBMS)

Database management system is a software system facilitating the creation and maintenance of a database and the execution of computer programs using the database.

A DBMS allows designers to structure their information, allows users to query and modify that information, and helps manage very large amounts of data and many concurrent operations on the data.

A DBMS is expected to:

① Allow users to create new databases and specify their schema (logical structure of the data), using a specialized language called a data definition language(DDL).

② Give users the ability to query the data and modify the data, using an appropriate language, often called a query language or data manipulation language(DML).

DML is also known as query language. SQL is the most widely used query language.

Application programs generally access databases through one of language extensions to allow embedded SQL or application program interface(e. g. , ODBC/JDBC) which allow SQL queries to be sent to a database.

③ Support the storage of very large amounts of data over a long period of time, keeping it secure from accident or unauthorized use and allowing efficient access to the data for queries and database modifications.

④ Control access to data from many users at once, without allowing the actions of one user to affect other users and without allowing simultaneous accesses to corrupt the data accidentally.

Components of database management systems:

① Data Definition Language(DDL)

Specification notation for defining the database schema.

② Data Manipulation Language(DML)

Language for accessing and manipulating the data organized by the appropriate data model.

③ Transaction Management

A transaction is a collection of operations that performs a single logical function in a database application.

Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures(e. g. , power failures and operating system crashes) and transaction failures.

Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

④ Storage Management

Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible to the following tasks:

- Interaction with the file manager.
- Efficient storing, retrieving and updating of data.

(4) Database Administrator(DBA)

Coordinate all the activities of the database system.

The database administrator has a good understanding of the enterprise's information resources and needs.

Database administrator's duties include:

① Schema definition.

② Storage structure and access method definition.

③ Schema and physical organization modification.

④ Granting user authority to access the database.

⑤ Specifying integrity constraints.

⑥ Acting as liaison(a channel for communication between groups) with users.

⑦ Monitoring performance and responding to changes in requirements.

(5) Database Users

Users are differentiated by the way they expect to interact with the system.

Application programmers—interact with system through DML calls.

Sophisticated users—form requests in a database query language.

Specialized users—write specialized database applications that do not fit into the traditional data processing framework.

(6) Database System(DBS)

Database system contains DB, DBMS, DBA, DB Users and other parts needed. DBMS is the most important part of a DBS.

1.2 Data Models

Managing large quantities of structured and unstructured data is a primary function of information systems. Data models describe structured data for storage in data management systems such as relational databases.

The term data model has two meanings:

① A data model theory, i. e. , a formal description of how data may be structured and accessed.

② A data model instance, i. e. , applying a data model theory to create a practical data model instance for some particular application.

A data model theory has three main components:

① The structural part: a collection of data structures which are used to create databases representing the entities or objects modeled by the database.

② The integrity part: a collection of rules governing the constraints placed on these data structures to ensure structural integrity.

③ The manipulation part: a collection of operators which can be applied to the data structures, to update and query the data contained in the database.

A data model instance is created by applying a data model theory.

This is typically done to solve some business enterprise requirements. Business requirements are normally captured by a semantic logical data model. This is transformed into a physical data model instance from which is generated a physical database.

For example, a data modeler may use a data modeling tool to create an entity-relationship model of the corporate data repository of some business enterprises. This model is transformed into a relational model, which in turn generates a relational database.

1.2.1 Entity-relationship Model

In software engineering, an entity-relationship model (E-R model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database.

An entity-relationship model is a systematic way of describing and defining a business process.

The process is modeled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one student may elect many courses to study, and one course can be elected by many students.

Entities may have various properties(attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity-relationship diagrams(E-R diagrams).

E-R diagrams have three principal components:

① Entities: members of an entity set.

An entity may be defined as a thing capable of an independent existence that can be uniquely identified.

An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspects of the real world that can be distinguished from other aspects of the real world.

An entity may be a physical object such as a student, a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order.

E. g. , the collection of all the students (courses) in a database is an entity set, and a particular student (course) is an entity.

② Attributes, which are values describing some properties of an entity or a relationship.

Entities and relationships can both have attributes. E. g. , the "student" entity may have a Student Identify Code (SId) attribute; the "elected" relationship may have a final grade attribute.

③ Relationships, which are connections among two or more entity sets.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. E. g. , an "owns" relationship between a company and a computer, a "supervises" relationship between a department and an employee, a "performs" relationship between an artist and a song, a "proved" relationship between a mathematician and a theorem.

In an E-R diagram (see Figure 1. 1. 1) :

① Entities: use rectangles.

② Relationships: use rhombuses.

③ Attributes: use ellipses.

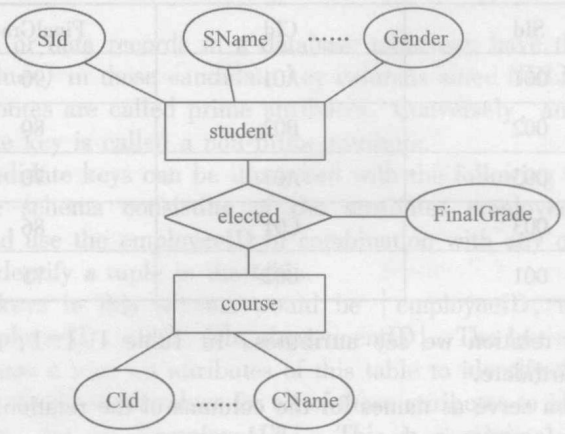


Figure 1. 1. 1 An example of an E-R diagram

E-R model is widely used for database conceptual design. And it is usually converted to design in the relational model (coming up next) which is used for storage and processing.

An E-R model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationships.

1.2.2 Relational Data Model

(1) Two-dimensional Table

Relationship Model is a two-dimensional table which can show us what entity and entities' relationship are.

The relational model gives us a single way to represent data: as a two-dimensional table called a relation. Table 1. 1. 1 is an example of a relation. The name of the relation is students information. Table 1. 1. 2 and Table 1. 1. 3 are two other relations named course information and final grades.

Table 1.1.1 Student information table

SId	SName	Gender	Birthdate
001	Jim	Male	1980.10
002	Mike	Male	1981.1
003	Mary	Female	1979.2

Table 1.1.2 Course information table

CId	CName
A01	VFP
B02	MIS
C01	ERP

Table 1.1.3 Final Grade table

SId	CId	FinalGrade
001	A01	90
002	B02	80
003	A01	70
003	C01	86
001	B02	75

Across the top of a relation we see attributes. In Table 1.1.1, the attributes are SId, SName, Gender and Birthdate.

Attributes of a relation serve as names for the columns of the relation. Usually, the attribute describes the meaning of entries in the column below.

Attribute is the term used in the theory for what is commonly referred to as a column (also called field in VFP). Similarly, table is commonly used in place of the theoretical term relation.

The name of a relation and the set of attributes for a relation is called the schema for that relation. We show the schema for the relation with the relation name followed by a parenthesized list of its attributes. Thus, the schema for relation students information of Table 1.1.1 is:

Student information(SId, SName, Gender, Birthdate)

The rows of a relation, other than the header row containing the attributes, are called tuples. A tuple has one component for each attribute of the relation. A tuple is basically the same thing as a row(also called record in VFP).

Tuples are not ordered; instead, each attribute value is identified solely by the attribute name and never by its ordinal position within the tuple.

Relations, however, are sets of tuples, and it is not possible for a tuple to appear more than once in a given relation.

The relational model requires that each component of each tuple is atomic. It is not permitted for a value to be a record structure, set, list, array, or any other type that can reasonably have its values broken into smaller components.

(2) The Definition of Key

① Super Key

Informally, a super key is a set of attributes within a table whose values can be used to uniquely identify a tuple.

A super key is defined in the relational model of database organization as a set of attributes of a relation variable for which it holds that in all relations assigned to that variable, there are no two distinct tuples(rows) that have the same values for the attributes in this set. Equivalently a super key can also be defined as a set of attributes of a relation schema upon which all attributes of the schema are functionally dependent.

② Candidate Key

A candidate key is a minimal set of attributes necessary to identify a tuple; this is also called a minimal super key. The candidate key may consist of a single attribute or multiple attributes in combination.

Since a relation contains no duplicate tuples, the set of all its attributes is a super key if NULL values are not used. It follows that every relation will have at least one candidate key.

In a relational database, a candidate key uniquely identifies each row of data values in a database table.

No two distinct rows or data records in a database table can have the same data value(or combination of data values) in those candidate key columns since NULL values are not used.

The constituent attributes are called prime attributes. Conversely, an attribute that does not occur in ANY candidate key is called a non-prime attribute.

The definition of candidate keys can be illustrated with the following example.

Given an employee schema consisting of the attributes employeeID, name, job, and departmentID, we could use the employeeID in combination with any or all other attributes of this table to uniquely identify a tuple in the table.

Examples of super keys in this schema would be {employeeID, name}, {employeeID, name, job}, and {employeeID, name, job, departmentID}. The last example is known as the trivial super key, because it uses all attributes of this table to identify the tuple.

In a real database we do not need values for all of those attributes to identify a tuple. We only need, per our example, the set {employeeID}. This is a minimal super key—that is, a minimal set of attributes that can be used to identify a single tuple. Therefore, employeeID is a candidate key.

③ Primary Key

Depending on its design, a database table may have many candidate keys but at most one candidate key may be distinguished as the primary key. This is the key that is allowed to migrate to other entities to define the relationships that exist among the entities.

④ Foreign Key

A foreign key is a column(or collection of columns) in one table that uniquely identifies a row of another table.

In other words, a foreign key is a column or a combination of columns that is used to establish and enforce a link between two tables.

A table may have multiple foreign keys, and each foreign key can have a different parent table. The table containing the foreign key is called the referencing or child table, and the table containing the candidate key is called the referenced or parent table.

For example, consider a database with two tables: a CUSTOMER table that includes all customer data and an ORDER table that includes all customer orders. Suppose the business requires that each order must refer to a single customer. To reflect this in the database, a foreign key column is added to the ORDER table(e.g., CUSTOMERID), which references the primary key of CUSTOMER(e.g., ID). Because the primary key of a table must be unique,

and because CUSTOMERID only contains values from that primary key field, we may assume that, when it has a value, CUSTOMERID will identify the particular customer which placed the order. However, this can no longer be assumed if the ORDER table is not kept up to date when rows of the CUSTOMER table are deleted or the ID column altered, and working with these tables may become more difficult. Many real world databases work around this problem by "inactivating" rather than physically deleting master table foreign keys, or by complex update programs that modify all references to a foreign key when a change is needed.

Foreign keys play an essential role in database design. One important part of database design is making sure that relationships between real-world entities are reflected in the database by references, using foreign keys to refer from one table to another. Another important part of database design is database normalization, in which tables are broken apart and foreign keys make it possible for them to be reconstructed.

Multiple rows in the referencing(or child) table may refer to the same row in the referenced (or parent) table. In this case, the relationship between the two tables is called a one to many relationship between the referenced table and the referencing table.

Here is an example of a primary key becoming a foreign key on a related table. The Author_ ID migrates from the Author Table to the Book Table.

Author Table Schema:

Author Table(Author_ID, AuthorName, CountryBorn, YearBorn)

Book Table Schema:

Book Table(ISBN, Author_ID, Title, Publisher, Price)

Here we can see that Author_ID serves as the primary key in the Author Table but also serves as the foreign key in the Book Table. The foreign key serves as the link and therefore the connection between the two "related" tables in this sample database.

(3) Integrity Constraint of Relations

① Entity Integrity Constraint

The attributes that belong to key can not be set as NULL.

② Referential Integrity Constraint

The value of foreign key can only be NULL or same as what is in the parent table.

Referential integrity is a property of data which, when satisfied, requires every value of one attribute(column) of a relation(table) to exist as a value of another attribute in a different(or the same) relation(table).

For referential integrity to hold in a relational database, any field in a table that is declared a foreign key can contain either a null value, or only values from a parent table's primary key or a candidate key.

In other words, when a foreign key value is used, it must reference a valid, existing primary key in the parent table.

"Referential" the adjective describes the action that a foreign key performs, "referring" to a link field in another table. In simple terms, "referential integrity" is a guarantee that the target it "refers" to will be found.

A lack of referential integrity in a database can lead relational databases to return incomplete data, usually with no indication of an error. A common problem occurs with relational database tables linked with an "inner join" which requires non-NULL values in both tables, a requirement that can only be met through careful design and referential integrity.

For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity.

Some relational database management systems(RDBMS) can enforce referential integrity, normally either by deleting the foreign key rows as well to maintain integrity, or by returning an error and not performing the delete. Which method is used may be determined by a referential

integrity constraint defined in a data dictionary.

③ User-defined Integrity Constraint

Users define the constraints themselves.

(4) NULL

Visual FoxPro provides support for null values. This support simplifies the task of representing unknown data and makes it easier to work with Microsoft Access or SQL databases that may contain null values.

Null values are:

- Equal to the absence of any value.
- Different than zero, the empty string(" "), or blank.
- Sorted ahead of other data.
- Propagated in calculations and most functions.