Chapter 10

Database Design and Programming

© Pearson Education Limited 2005

Topics

- Quick tutorial in relational databases from a software engineering viewpoint
  - tables and referential integrity
  - conceptual versus logical database models
  - business rules and application logic
- Mapping transient objects to persistent records
  - object databases, SQL:1999, and impedance mismatch
  - object-relational mapping
- Database schema for EM

Database characteristics

- large
- persistent
- multi-user sharable
- recoverable
- consistent
- secure
- extensible

Relational model

- Database model – an abstraction that presents the server data to the client programs in more understandable terms than the bits and bytes
  - The dominant database model for business information systems is the relational model
  - Structured Query Language (SQL) is the relational language that the application programs must use to gain access to the database
- The relational model
  - presents data as records (rows) in tables (relations)
  - records in different tables (or in the same table) cannot be linked by user-visible navigational links
  - referential integrity

Table

- Table consists of a
  - fixed number of columns
    - must be primitive data types, such as numbers or strings of characters
  - varying number of records (rows)

Referential integrity

- uses the notion of a foreign key to link records in one table to records in another (or even the same) table
  - foreign key is a set of columns (frequently just one column), which values correspond to the values of the primary key in another (or the same) table

Select statement to get table content:

```
SELECT * FROM movie;
```

```
MOVIE_CODE   MOVIE_TITLE              DIRECTOR
----------------- ------------------------------ ----------------------
10 Interview with the Vampire     Neil Jordan
11 The Birdcage                   Mike Nichols
```

© Pearson Education Limited 2005

© Pearson Education Limited 2005

© Pearson Education Limited 2005

© Pearson Education Limited 2005
Conceptual ER database model

- Model at a higher-level of abstraction, which does not make a commitment to the database technology
- The Entity-Relationship (ER) is the best known technique for conceptual database modeling

Conceptual UML database model

Implementing business rules

- Declarative referential integrity can be used in relational database to implement many simple and repetitive business rules
- More complex and exclusive business rules require the procedural referential integrity enforcement = triggers

Declarative referential integrity

- SQL: CREATE TABLE listed_as (actor_code NUMBER(5), movie_code NUMBER(5), position NUMBER(2), PRIMARY KEY (actor_code, movie_code));
- SQL: CREATE TABLE actor (actor_code NUMBER(5), actor_name VARCHAR2(25), PRIMARY KEY (actor_code));
- SQL: ALTER TABLE listed_as ADD CONSTRAINT fk_const3 FOREIGN KEY (actor_code) REFERENCES actor ON DELETE CASCADE;

Procedural referential integrity

- SQL: CREATE TRIGGER tda_movie AFTER DELETE ON movie FOR EACH ROW BEGIN
  -- Delete all children in "listed_as"
  DELETE listed_as WHERE movie_code = :old.movie_code;
END;

Programming with embedded SQL

- SQL: EXECUTE SELECT movie_title, actor_name FROM movie m, listed_as l, actor a WHERE m.movie_code = l.movie_code AND l.actor_code = a.actor_code AND l.position = 1;

Declarative referential integrity with on delete cascade in MovieActor

Procedural referential integrity with delete trigger in MovieActor
Programming with stored procedures

Stored procedure in MovieActor database

SQL> -- String search for a movie using stored procedure
2 CREATE OR REPLACE PROCEDURE string_search (string IN VARCHAR2) AS
3 BEGIN
4 dbms_output.put_line ('Found movie title: '||LPAD(c1rec.found,30));
5 END LOOP;
6 END string_search;

Stored procedure in MovieActor database

1 SQL> EXECUTE string_search('vamp');
2 Procedure created.

3 SQL> -- String search for a movie using stored procedure
4 Stored procedure in MovieActor database
5 11 /
6 10  END string_search;
7 9  END LOOP;
8 8  dbms_output.put_line
9 7  FOR c1rec IN c1 LOOP
10  BEGIN
11 WHERE UPPER(movie_title) LIKE '%'||UPPER(string)||'%';
12  FROM movie
13  SELECT movie_title AS found
14  END LOOP;
15  END PROCEDURE create.

Object Databases and SQL:1999

- **Object database model** is a competitor of the relational model
  - Object DBMS can serve as Object Storage API
    - to provide a layer of software between the application program and the relational (or other) database in order to map application's objects to relational records and vice versa
    - the mapping may involve the use of a persistent object database
    - results in moving much of the responsibilities performed by the domain and foundation classes to the Object Storage API
- **SQL:1999** – the latest standard that attempts to add object-oriented features to the relational model.

Impedance mismatch

- **Impedance mismatch** – the impossibility of using the application language (such as Java) to directly manipulate the data in the database without the need to engage SQL
  - SQL manipulates data as sets of records
  - application languages are procedural in nature and manipulate data as individual records
  - SQL provides the cursor mechanism to address the mismatch
- Impedance mismatch is a reason to develop a mapping strategy from classes to tables and vice versa.

Indexes

- Index is a data structure, separate from data pages that store table records, which consists of a hierarchical tree of index nodes
  - physical data independence – application programs can perform their database tasks with or without indexes.
  - SQL:1999

Indexes

From one-to-many association and aggregation...

...to relational database
Many-to-many association...

XXX
 Attribute_1 : int
 Attribute_2 : String
 Attribute_3 : int

YYY
 Attribute_4 : int
 Attribute_5 : int
 Attribute_6 : int

Many-to-many association...

To relational database

XXX
 Attribute_1 INTEGER not null
 Attribute_2 VARCHAR(30) null
 Attribute_3 INTEGER null

YYY
 Attribute_4 INTEGER not null
 Attribute_5 INTEGER not null
 Attribute_6 INTEGER null

Association

Attribute_1 + Attribute_1
 Attribute_2 + Attribute_4
 Attribute_3 + Attribute_5

New table Association introduced to maintain the many-to-many relationship between XXX and YYY. The primary key of Association is a set of columns consisting of the primary key columns of XXX and YYY. The foreign keys refer to the primary keys of the corresponding primary tables.

One-to-one association...

XXX
 Attribute_1 : int
 Attribute_2 : String
 Attribute_3 : int

YYY
 Attribute_4 : int
 Attribute_5 : int
 Attribute_6 : int

One-to-one association...

To relational database

XXX
 Attribute_1 INTEGER not null
 Attribute_2 VARCHAR(30) null
 Attribute_3 INTEGER null

YYY
 Attribute_4 INTEGER not null
 Attribute_5 INTEGER not null
 Attribute_6 INTEGER null

Association

XXX_Attribute_1 + Attribute_1
 Attribute_2 + Attribute_4
 Attribute_3 + Attribute_5

New foreign key column XXX_Attribute1 introduced to maintain a recursive one-to-many relationship to primary key values. There can be many records in XXX with a foreign key value equal to a primary key value in another record in XXX. The foreign key accepts nulls because of the 0..1 multiplicity from XXX to YYY.

Recursive (self-referential) one-to-many association...

0..1 Role A

XXX
 Attribute_1 : int
 Attribute_2 : String
 Attribute_3 : int

0..* Role B

To relational database

XXX
 Attribute_1 INTEGER not null
 XXX_Attribute_1 INTEGER not null
 Attribute_2 VARCHAR(30) null
 Attribute_3 INTEGER null

XXX
 Attribute_1 INTEGER not null
 XXX_Attribute_1 INTEGER not null
 Attribute_2 VARCHAR(30) null
 Attribute_3 INTEGER null

New foreign key column XXX_Attribute1 introduced to maintain a recursive one-to-many relationship to primary key values. There can be many records in XXX with a foreign key value equal to a primary key value in another record in XXX. The foreign key accepts nulls because of the 0..1 multiplicity of Role A.
From many-to-many recursive association...

Recursive (self-referential) many-to-many association

Role A

Attribute_1 : int
Attribute_2 : String
Attribute_3 : int

Role B

XXX

0..* 1..*

...to relational database

XXX

Attribute_1 : INTEGER
Attribute_2 : VARCHAR2(30)
Attribute_3 : INTEGER

1..* 0..*

Attribute_1 = XXX_Attribute_1

Association_1

Attribute_1 : INTEGER
XXX_Attribute_1 : INTEGER

0..* 1..*

Attribute_1 = Attribute_1

Attribute_2 : INTEGER

Attribute_3 : INTEGER

Association_1

Attribute_1 : INTEGER
XXX_Attribute_1 : INTEGER

0..* 1..*

Any many-to-many association, whether recursive or defined on two classes, results in a "relationship table" with two referential integrity constraints.

From generalization...

Generalization

XXX

Attribute_1 : int
Attribute_2 : String
Attribute_3 : int

YYY

Attribute_4 : int
Attribute_5 : int
Attribute_6 : int

...to relational database

YYY

Attribute_1 : INTEGER
Attribute_2 : VARCHAR2(30)
Attribute_3 : INTEGER

0..* 1..*

Attribute_1 = Attribute_1

Attribute_4 : INTEGER

Attribute_5 : INTEGER

Attribute_6 : INTEGER

XXX

Attribute_1 : INTEGER
XXX_Attribute_1 : INTEGER

0..* 1..*

Attribute_1 = Attribute_1

Attribute_2 : INTEGER

Attribute_3 : INTEGER

XXX

Attribute_1 : INTEGER
XXX_Attribute_1 : INTEGER

0..* 1..*

Attribute_1 = Attribute_1

Attribute_2 : INTEGER

Attribute_3 : INTEGER

YYY "inherits" only the primary key of XXX and uses it as its own primary key as well as the foreign key to XXX.

From conceptual class diagram for EM...

Employee

employee_id : String
first_name : String
family_name : String
login_name : String
employee_email : String

0..* 1..*

creator : String
sender : String

OutMessage

message_id : String
message_text : String
date_created : Date
date_ended : Date

...to logical database schema for EM

Employee

employee_id : CHAR(4)

first_name : VARCHAR2(20)

family_name : VARCHAR2(40)

login_name : VARCHAR2(40)

employee_email : VARCHAR2(60)

OutMessage

message_id : CHAR(12)

message_text : VARCHAR2(255)
date_created : DATE
date_ended : DATE

Contact

...to relational database

Employee

employee_id : CHAR(4)

first_name : VARCHAR2(20)

family_name : VARCHAR2(40)

contact_id : CHAR(5)

employee_email : VARCHAR2(60)

OutMessage

message_id : CHAR(12)

message_text : VARCHAR2(255)
date_created : DATE
date_ended : DATE
**Summary**

- Database is large, persistent, multi-user shareable, recoverable, consistent, secure, extensible
- The dominant database model for business information systems is the relational model
- An important software engineering task is to map between transient objects in an application program and persistent records in a database