In the Lecture Series Introduction to Database Systems

Integrity Constraints

Presented by Stéphane Bressan

Database Design

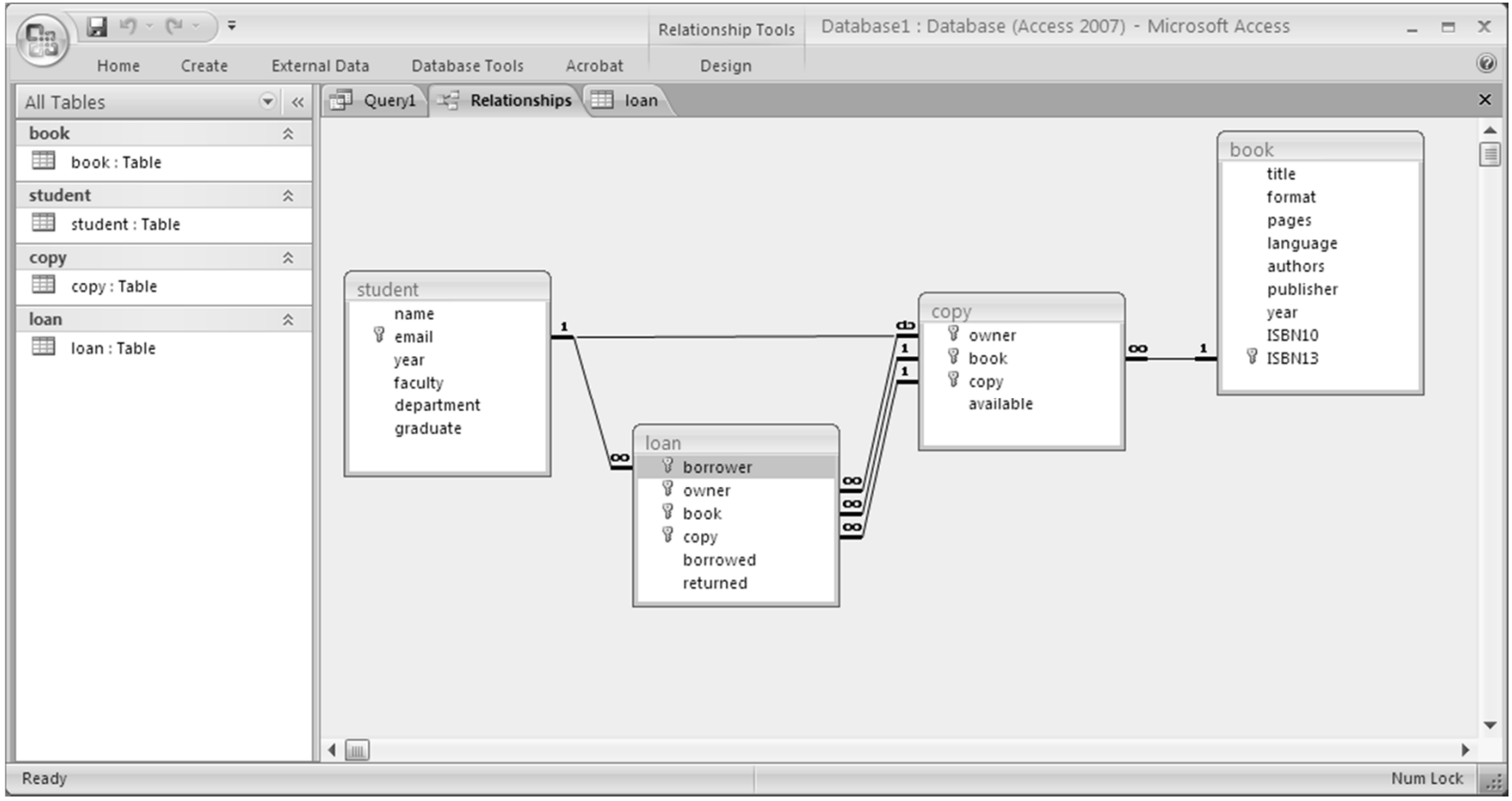
- The database records the name, faculty, department and other information about students. Each student is identified in the system by its email.

- The database records the title, authors, the ISBN-10 and ISBN-13 and other information about books. The International Standard Book Number, ISBN-10 or -13, is an industry standard for the unique identification of books.

- The database records information about copies of books owned by students.

- The database records information about the book loans by students.

Database Design



Creating a Table for Students

```
CREATE TABLE student (
name VARCHAR(32),
email VARCHAR(256),
year DATE,
faculty VARCHAR(62),
department VARCHAR(32),
graduate DATE);
```

Creating a Table for Books

```
CREATE TABLE book (
title VARCHAR(256),
format CHAR(9),
pages INT,
language VARCHAR(32),
authors VARCHAR(256),
publisher VARCHAR(64),
year DATE,
ISBN10 CHAR(10),
ISBN13 CHAR(14));
```

Creating a Table for Copies

```
CREATE TABLE copy (
owner VARCHAR(256),
book CHAR(14)
available BOOLEAN BIT CHAR(6));
```

Creating a Table for Loans

```
CREATE TABLE loan (
borrower VARCHAR(256),
owner VARCHAR(256),
book CHAR(14),
copy INT,
borrowed DATE,
returned DATE);
```

SQL Integrity Constraints

We can add constraints to our design in order to improve the management of database integrity by the database management system:

- PRIMARY KEY
- NOT NULL
- UNIQUE
- FOREIGN KEY
- CHECK

Structural Constraints

- The choice of the number of columns and their domains imposes structural constraints

```
CREATE TABLE registration(
Student VARCHAR(10);
Module VARCHAR(6));
```

- No student without a module and no module without a student, unless we use NULL values

Integrity Constraint: What Do They Do?

- Integrity constraints are **checked** by the DBMS before a **transaction** (BEGIN...END) modifying the data is committed;
- If an integrity constraint is **violated**, the transaction is **aborted** and **rolled back**, the changes are not reflected;
- Otherwise the transaction is **committed** and the changes are effective.

Note: Integrity constraints can be immediate or deferred

Integrity Constraints – Primary Key

- A **Primary Key** is a set of attributes that identifies uniquely a t-uple:
  - People
    - national identification number
    - email address
    - first name and last name
  - Flights
    - Airline name and flight number
  - Books
    - ISBN

Integrity Constraints – Primary Key

- You cannot have two t-uples with the same Primary Key in the same table.
- Let us make ISBN13 the primary key of the book table.

```
CREATE TABLE book (
      title VARCHAR(256),
      authors VARCHAR(256),
      publisher VARCHAR(64),
      ISBN13 CHAR(14),
      ISBN10 CHAR(10));
```

Column (Value) Constraint – PRIMARY KEY

```
CREATE TABLE book (
title VARCHAR(256),
authors VARCHAR(256),
publisher VARCHAR(64),
ISBN13 CHAR(14) PRIMARY KEY,
ISBN10 CHAR(10));
```

book(title VARCHAR(128), authors VARCHAR(128), publisher VARCHAR(32), ISBN10 CHAR(10), ISBN13 CHAR(14))

book(title , authors, publisher, ISBN10, ISBN13 CHAR(14))

Column (Value) Constraint – PRIMARY KEY

We cannot insert two books with the same ISBN13.

```
INSERT INTO book VALUES('The Digital
  Photography Book', 'Scott Kelby','Peachpit
  Press','032147404X','978-0321474049');

INSERT INTO book VALUES('Digital
  Photography', 'S.Kelby','Peachpit Press
  Ltd', '032147404X','978-0321474049');

Error: UNIQUE constraint failed: book.ISBN13
```

Table Constraint – PRIMARY KEY

In some cases, we need composite primary keys: combinations of attributes. The primary key of the copy table is the combination of owner, book and copy

```
CREATE TABLE copy (
owner VARCHAR(256),
book CHAR(14),
copy INT,
PRIMARY KEY (owner, book, copy));
```

Column Constraint – NOT NULL

Every domain (type) has an additional value: the NULL value (read Ramakrishnan). We can forbid its usage.

```
CREATE TABLE book (
title VARCHAR(256),
authors VARCHAR(256),
publisher VARCHAR(64),
ISBN13 CHAR(14) PRIMARY KEY
ISBN10 CHAR(10) NOT NULL);
```

Column Constraint - UNIQUE

The two constraints together, NOT NULL and UNIQUE, have the same logical effect as a PRIMARY KEY

```
CREATE TABLE book (
title VARCHAR(256),
authors VARCHAR(256),
publisher VARCHAR(64),
ISBN13 CHAR(14) PRIMARY KEY
ISBN10 CHAR(10) NOT NULL UNIQUE);
```

Why this redundancy?

Table Constraint - UNIQUE

UNIQUE constraints can also apply to composite attributes.

```
CREATE TABLE people (
first_name VARCHAR(32),
last_name VARCHAR(32),
UNIQUE (first_name, last_name));
```

Table Constraint - UNIQUE

```
INSERT INTO people VALUES ('John', 'Smith');
INSERT INTO people VALUES ('John', 'Henry');
INSERT INTO people VALUES ('paul', 'Smith');

INSERT INTO people VALUES ('John', 'Smith');

Error: UNIQUE constraint failed:
  people.first_name, people.last_name
```

The **combination** of the two attributes must be unique

Column Constraint – FOREIGN KEY (referential integrity)

Since the relational model is value oriented, references are made by values (references to the primary keys). Foreign key (referential) constraints, RFERENCES, make sure the references are valid (avoid dangling pointers).

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
  student(email),
book CHAR(14) REFERENCES book(ISBN13),
copy INT,
PRIMARY KEY (owner, book, copy));
```

email is an attribute of the relation student

email must be the **primary key** the relation student

Column Constraint - FOREIGN KEY (referential integrity)

There is a new copy (Thomas Lee, tom27@gmail.com)

copy

copy	book	email
1	978-0596101992	jj@hotmail.com
1	978-0596520830	tom27@gmail.com
2	978-0596520830	tom27@gmail.com
2	978-0596101992	ds@yahoo.com

student

email	name	year
jj@hotmail.com	Jong-jin Lee	2009
tom27@gmail.com	Thomas Lee	2008
helendg@gmail.com	Helen Dewi Gema	2009

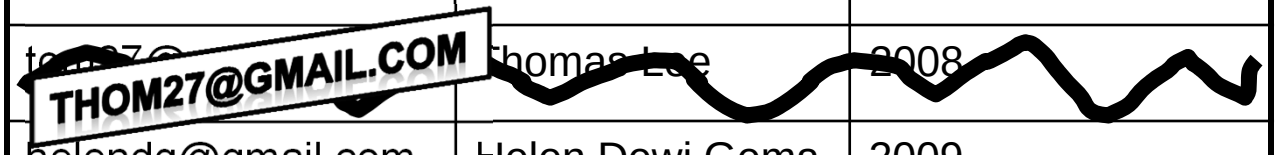


Table Constraint – FOREIGN KEY (referential integrity)

```
CREATE TABLE loan (
borrower VARCHAR(256) REFERENCES
  student(email),
owner VARCHAR(256),
book CHAR(14),
copy INT,
borrowed DATE NOT NULL,
return DATE,
FOREIGN KEY (owner, book, copy) REFERENCES
  copy(owner,book, copy),
PRIMARY KEY (borrower, owner, book, copy)
```

owner, book and copy are attributes of the relation student

owner, book and copy are the **primary key** the relation student

Example of CHECK Table Constraint

All constraints can be expressed by means of a CHECK constraint.

```
CREATE TABLE test (
before NUMBER CHECK (before > 0),
after NUMBER CHECK (after> 0));

INSERT INTO test VALUES (1,0);
Error: CHECK constraint failed: test

INSERT INTO test VALUES (0,2);
Error: CHECK constraint failed: test
```

Column Constraint - CHECK

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
  student(email),
book CHAR(14) REFERENCES  book(ISBN13),
copy INT CHECK(copy > 0),
PRIMARY KEY (owner, book, copy));
```

See also CREATE DOMAIN and CREATE TYPE

Column Constraint - CHECK

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
  student(email),
book CHAR(14) REFERENCES  book(ISBN13),
copy INT,
CONSTRAINT non_zero CHECK(copy > 0),
PRIMARY KEY (owner, book, copy));
```

Example of CHECK Table Constraint

```
CREATE TABLE test (
before NUMBER,
after NUMBER,
CHECK (before <= after));
INSERT INTO test VALUES (1,2);
INSERT INTO test VALUES (3,2);
Error: CHECK constraint failed: test
```

Table Constraint - CHECK

```
CREATE TABLE loan (
borrower VARCHAR(256) REFERENCES
  student(email),
owner VARCHAR(256),
book CHAR(14),
Copy INT,
borrowed DATE NOT NULL ,
return DATE,
FOREIGN KEY (owner, book, copy) REFERENCES
                                  copy(owner,
  book, copy),
PRIMARY KEY (borrower, owner, book, copy),
CHECK(return >= borrowed OR return IS NULL))
```

Table Constraint - CHECK

The language for CHECK constraints , in practice, is however very limited. Too bad!

```
CHECK(NOT EXISTS
  (SELECT *
    FROM loan l1, loan l2
    WHERE l1.owner=l2.owner AND
  l1.book=l2.book AND    l1.copy=l2.copy AND
  l1.borrowed <= l2.borrowed AND
  (l2.borrowed <= l1.return OR l1.return IS
  NULL));
```

``A copy cannot be borrowed until it is returned''

Subqueries in the table constraints are forbidden in SQLite

Assertions

An even more general form of constraints, assertions, exist in the standard. They can span across tables. They are not implemented in any system. Too bad!

CREATE ASSERTION name
CHECK(*some condition*)

Enforcing Integrity Constraints

Integrity constraints, if violated, abort and rollback the transaction.

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
  student(email),
book CHAR(14) REFERENCES book(ISBN13),
copy INT,
PRIMARY KEY (owner, book, copy));
```

Enforcing Integrity Constraints

Updates and deletions that violates foreign key constraints are rejected.

Could they be compensated?

copy

copy	book	email
1	978-0596101992	jj@hotmail.com
1	978-0596520830	tom27@gmail.com
2	978-0596520830	tom27@gmail.com

student

email	name	year
jj@hotmail.com	Jong-jin Lee	2009
tom27@gmail.com	Thomas Lee	2008
helendg@gmail.com	Helen Dewi Gema	2009

Enforcing Integrity Constraints

Compensation can be implemented, in some systems, by cascading. In others, they have to be handled manually by triggers.

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
          student(email) ON UPDATE CASCADE,
book CHAR(14) REFERENCES
          book(ISBN13) ON UPDATE CASCADE,
copy INT,
PRIMARY KEY (owner, book, copy));
```

Enforcing Integrity Constraints

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES
          student(email) ON UPDATE CASCADE
          ON DELETE CASCADE,
book CHAR(14) REFERENCES
          book(ISBN13) ON UPDATE CASCADE
          ON DELETE CASCADE,
copy INT,
PRIMARY KEY (owner, book, copy));
```

Enforcing Integrity Constraints

ON UPDATE/DELETE

- CASCADE
- NO ACTION
- SET DEFAULT
- SET NULL

Multiple Cascade Paths

```
CREATE TABLE copy (
owner VARCHAR(256) REFERENCES student(email) ON UPDATE CASCADE ON DELETE
    CASCADE,
book CHAR(14) REFERENCES book(ISBN13) ON UPDATE CASCADE,
copy INT CHECK(copy > 0),
available BIT NOT NULL DEFAULT 'TRUE',
PRIMARY KEY (owner, book, copy))

CREATE TABLE loan (
borrower VARCHAR(256) REFERENCES student(email) ON UPDATE CASCADE,
owner VARCHAR(256),
book CHAR(14),
copy INT,
borrowed DATE,
returned DATE,
FOREIGN KEY (owner, book, copy) REFERENCES copy(owner, book, copy) ON UPDATE
    CASCADE ON DELETE CASCADE,
PRIMARY KEY (borrowed, borrower, owner, book, copy),
CHECK(returned >= borrowed))
```

Logical and Knowledge Independence

- Constraints, together with complex queries and views, stored procedures and functions, and triggers contribute to Knowledge Independence.
- Application programmers need not worry in their programs about the application business rules. They can focus on the interfacing aspects.

Credits

The content of this lecture is based on chapter 2 of the book "Introduction to database Systems"
By
S. Bressan and B. Catania, McGraw Hill publisher

Clipart and media are licensed from Microsoft Office Online Clipart and Media