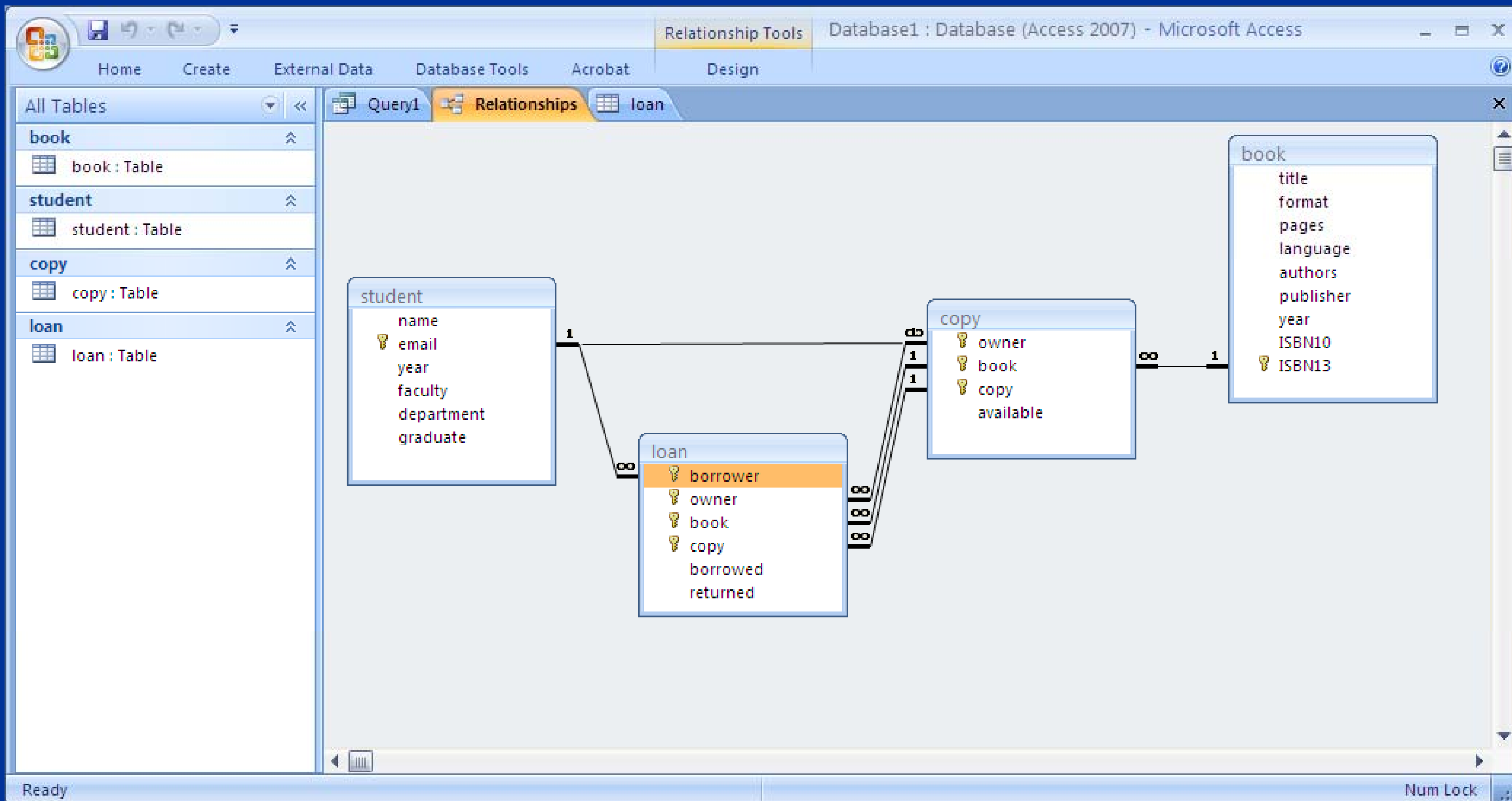
Data Models

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



Data Model

- The framework for **defining** the general form (**schema**) of the objects and **data in the database** (instances)
- *Notice that in the life time of the database the schema is rarely subject to changes while the instances are generally often updated*

Designing Database Applications

- Conceptual Model
  (for analysis and design)
  - Entity-Relationship
- Logical Model
  (for design and implementation)
  - Relational Model
- Physical Model
  (usually not visible,need to be understood for tuning)
  - CS3223

Physical Data Independence

- Interactions with the database can **ignore the actual representation** of data on the disk
- The physical representation can change

Data Models

- Hierarchical Model
- Network Model 1965 (DBTG, IMS)
- **Relational Model** (1NF) 1970s
- Nested Relational Model 1970s
- Complex Object 1980s
- Object Model 1980 (OQL)
- Object Relational Model 1990s (SQL)
- *XML (DTD), XML Schema 1990s (Xpath, Xquery)*
- *NoSQL Databases (MongoDB)*

DBTG (from Silberschatz, Korth, Sudarsan)

Print the total number of accounts in the Perryridge branch with a balance greater than $10,000.

count := 0;

branch.branch-name := “Perryridge”;

find any *branch using branch-name;*

find first *account within account-branch;*

while DB-status = 0 do

begin

get *account*

if *account.balance > 10000 then count := count + 1;*

find next *account within account-branch;*

end

print *(count);*

See: http://www.db-book.com/

The Same in the Relational Model with SQL

Print the total number of accounts in the Perryridge branch with a balance greater than $10,000.

```
SELECT COUNT(*)
FROM account
WHERE account.branch = ‘Perryridge’
AND account.balance > 10000;
```

The same with MongoDB

Print the total number of accounts in the Perryridge branch with a balance greater than $10,000.

```
db.account.find({branch:“Perryridge“, balance:{$gt:33}}).count()
```

See: http://www.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart

Logical and Knowledge Independence

- Logical Data Independence
  - **Views**: User Interactions with the database can **ignore the entirety of the schema**, they see what they need in the form they need

- Knowledge Independence
  - **Complex queries** are available as views
  - **Procedures** are stored in the database and the details of their implementation hidden
  - **Business rules** are captured with integrity constraints and triggers and automatically maintained

Relational Model

E.F. Codd “A Relational Model for Large Shared Data Banks” Communication of the ACM, Vol 13, #6

SQL DDL Statement

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

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

ALTER, DROP

SQL DML Statements

```
INSERT INTO book
VALUES (
        'Database Management Systems',
        'Raghu Ramakrishnan, Johannes Gehrke'),
        'McGraw-Hill',
        '978-0072465631')
```

DELETE, UPDATE, SELECT-FROM-WHERE

Idea

- Use mathematics to describe and represent records and collections of records: the relation
  - can be understood formally
  - leads to formal query languages
  - properties can be explained and proven

Idea

- Use a simple data structure: the Table
  - simple to understand
  - useful data structure (capture many situations)
  - leads to useful yet not too complex query languages

Relation Scheme

- A relation or **relation scheme** (or schema) R is a list or set of (distinct) **attribute names**
- R also called the **name** of the relation
  - **R(A, B, C, D, E)**
  - **R = {A, B, C, D, E}**
- Each attribute has a **domain**
  - **R(A:STRING, B:NUMERIC, C:DATE)**

Relations in First Normal Form (1NF)

- A **domain** is a set of atomic values (integer, char[24], boolean, date, blobs)
  - Complex domains (sets, lists, etc) are forbidden in First Normal Form (1NF)
  - There exists complex object and nested relational models (called Non First Normal Form or NF^2)

Degree, Arity

- The number of attributes in a relation scheme is called the **degree** or **arity** of the relation

Relation Instance

- A **relation instance** [R] is a sub**set** of the Cartesian product of the domains of the attributes in the schema
- An element of the relation instance, a record, is called a **t-uple**
- The number of t-uples is called the **cardinality** of the relation instance

Relation Instance

relation name ***column***

book ***attribute name: domain (or type)***

table

relation schema

row ***t-uple***

title:VARCHAR(128)	authors:VARCHAR(128)	publisher:VARCHAR(32)	ISBN13:CHAR(14)
The Future of Learning Institutions in a Digital Age	Cathy N. Davidson, David Theo Goldberg	The MIT Press	978-0262513593
Introduction to Algorithms	Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein	The MIT Press	978-0262033848
The Shallows: What the Internet Is Doing to Our Brains	Nicholas Carr	W. W. Norton & Company	978-0393072228
The Digital Photography Book	Scott Kelby	Peachpit Press	978-0321474049
Computer Organization and Design	David A. Patterson, John L. Hennessy	Morgan Kaufmann	978-0123744937
Introduction to Algorithms	Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein	The MIT Press	978-0262033848

Database Schema

- A **database schema** is the set of schemes of the relations in the database
- A **database instance** is the set of instances of the relations in the database
- *Very often we will confuse*
  - *the relation, its scheme, and its instance*
  - *the instance and the table*
  - *the attribute and the column*
  - *the t-uple and the row*
- *Ask for precision if there is ambiguity!*

Integrity Constraints in SQL

SQL Integrity Constraints

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

Remark: Structural Constraints

- The choice of a database schema imposes some constraints (structural)
- e.g. a database scheme:
  {flight(pilot, plane)}
  does not allow a plane without a pilot
  (unless using null values, read Ramakrishnan))

Integrity Constraint: What Do They Do?

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

Column Constraint – PRIMARY KEY

```
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), <u>ISBN13 CHAR(14) </u>)

book(title , authors, publisher, ISBN10, <u>ISBN13 CHAR(14) </u>)

Table Constraint – PRIMARY KEY

```
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)

```
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

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

Table Constraint - UNIQUE

```
CREATE TABLE student (
first_name VARCHAR(32)
last_name VARCHAR(32),
UNIQUE (first_name, last_name))
```

- The <u>**combination**</u> of the two attributes must be unique

Column Constraint – FOREIGN KEY (referential integrity)

```
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 [illegible] (ds@yahoo.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~~ THOM27@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

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))

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

```
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 again until it is returned''

Assertions

CREATE ASSERTION name
CHECK(*some condition*)

Enforcing Integrity Constraints

```
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

```
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

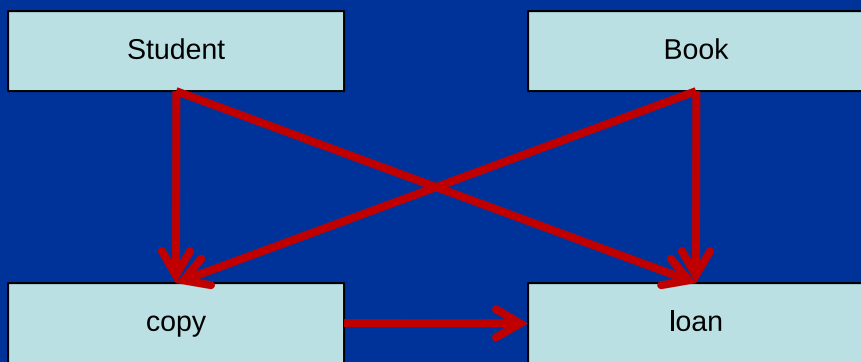
Multiple Cascade Paths

```
CREATE TABLE book (
title VARCHAR(256) NOT NULL,
format CHAR(9) CHECK(format = 'paperback' OR format='hardcover'),
pages INT,
language VARCHAR(32),
authors VARCHAR(256),
publisher VARCHAR(64),
year DATE,
ISBN10 CHAR(10) NOT NULL UNIQUE,
ISBN13 CHAR(14) PRIMARY KEY)

CREATE TABLE student (
name VARCHAR(32) NOT NULL,
email VARCHAR(256) PRIMARY KEY,
year DATE NOT NULL,
faculty VARCHAR(62) NOT NULL,
department VARCHAR(32) NOT NULL,
graduate DATE,
CHECK(graduate >= year)
```

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))
```

Multiple Cascade Paths



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

Animated characters are animated using VocaliseTTS under license from Digital Curiosty

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