In the Lecture Series Introduction to Database Systems

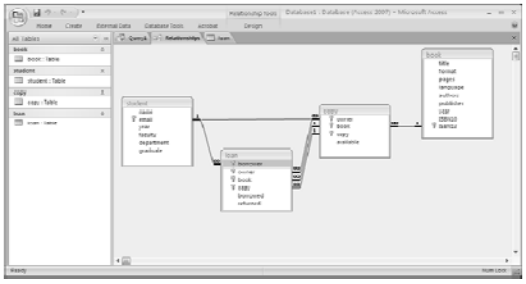
Data Models

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

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 (SQL)

SQL

- SQL was invented by D. Chamberlain and R. Boyce in 1974 at IBM for the first relational database management system System R
- SQL is an ANSI standard since1986
- SQL is an ISO standard since 1987
- We mostly refer to SQL-92 (or SQL2) and to Oracle's PL/SQL

SQL

- Data Definition Language (DDL)
  - Creating, altering and deleting tables and other database objects
- Data Manipulation Language (DML)
  - Querying tables
  - Inserting, updating and deleting rows
- Database Control language (DCL)
  - Controlling access to the database

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);
```

variable char vs char (cannot type less)

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
  - 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 NF²)

Domains: CHAR

- CHAR(*size*)
- Fixed-length character data of length *size bytes.* Maximum *size is 2000 bytes or characters. Default and* minimum *size is 1 byte.* Size indicates the length.

Domains: VARCHAR

- VARCHAR[(*size*)]
- Variable-length character string having maximum length *size bytes or characters. Maximum size is 4000* bytes or characters, and minimum is 1 byte or 1 character. Size indicates the length.

Domains: NUMBER

- NUMBER(*precision, scale)*
- *Number having precision p and scale s. The precision* can range from 1 to 38. The scale *can range from -84 to* 127.

Domains: DATE

- DATE
- Valid date range from January 1, 4712 BC to December 31, 9999 AD.
- Oracle's default format for DATE is "DD-MON-YY".

```
alter session set
  NLS_DATE_FORMAT='YYYY-MM-DD';
```


More Domains

- NVARCHAR(size)
- NVARCHAR2(size)
- NCHAR(size)
- LONG
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP
- UriType
- XMLType
- RAW(size)
- LONG RAW
- BINARY
- BIT
- CLOB
- NCLOB
- BLOB
- BFILE

a relation is like a set. relation, like a set, has no order. does not matter. but duplicates is a prob

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 subset 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

SELECT DISTINCT: results wth duplicates
SELECT: results with duplicates

Relation Instance

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

table

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

relation schema

row *t-uple*

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);
```


Removing Tables

```
DROP TABLE loan;
```

DROP delete table
DELETE delete data

Modifying Tables

```
ALTER TABLE loan ADD test NUMBER;

ALTER TABLE loan MODIFY test CHAR(6);

ALTER TABLE loan DROP COLUMN test;
```


Inserting New Rows (DML Statement)

```
INSERT INTO student VALUES('XIE XIN',
 'xiexin2011@gmail.com',
'2007-01-01',
'Faculty of Science',
'Chemistry',
'2011-01-01');

INSERT INTO student (email, name, faculty, department)
VALUES('XIE XIN',
'xiexin2011@gmail.com',
'2007-01-01',
'Faculty of Science',
'Chemistry');
```

no date added. NULL

Inserting New Rows (This is a bad idea)

```
CREATE TABLE cs_student (
name VARCHAR(32),
email VARCHAR(256),
year DATE,
graduate DATE);

INSERT INTO cs_student
SELECT name, email, year, graduate
FROM student
WHERE faculty='School of Computing'
AND department='CS';
```

nvr do that

new cs student, need to put in both tables. risk of inconsistency

Deleting Rows

```
DELETE FROM student;

DELETE FROM student WHERE department='Chemistry';
```

Modifying Rows

```
UPDATE student
SET department='Computer Science'
WHERE department='CS';

UPDATE student
SET year=year+1;
```


NULL Values

- Every domain has an additional value noted NULL
- The semantics of NULL is ambiguous:
  - Unknown
  - Does not exists
  - Unknown or does not exists

NULL Values Logic

P	Q	P AND Q	P OR Q	NOT P
True	True	True	True	False
False	True	False	True	True
Unknown	True	Unknown	True	Unknown
True	False	False	True	False
False	False	False	False	True
Unknown	False	False	Unknown	Unknown
True	Unknown	Unknown	True	False
False	Unknown	False	Unknown	True
Unknown	Unknown	Unknown	Unknown	Unknown

NULL Values Arithmetic

- Something = NULL is unknown
- Something < NULL is unknown
- Something > NULL is unknown
- 10 + NULL is unknown
- 10 * NULL is unknown
- COUNT(*) count NULL values
- COUNT, AVG, MAX, MIN eliminate NULL values

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

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

bad design
better design:
create another table for studs, another table for module. 3 tables in total

probs: must update multiple tables

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

transaction: units of work

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

can be more than one attribute

dun use system gen keys as pri keys unless theres a good reason to

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))
```

composite pri keys

actually can be any of the pri keys (ISBN)

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))

underline pri keys!!

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)

whats the diff?? for now, no diff

differences
diff pri keys
efficiency

Table Constraint - UNIQUE

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

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

combi appear only once

note, can have nulls.
null null are 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

always reference pri key of the other table

Column Constraint - FOREIGN KEY (referential integrity)

There is a new copy of book 978-0596520830 (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

foreign key

student

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

THOM27@GMAIL.COM

pri key

change email transaction

change all simulteneously

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))
```

comparison of attributes

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''

oracle will not let u write this constraint

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))

cmd can only be in the table with foreign keys, but updates can only be done on pri key

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))

or ON DELETE CASCADE

update the reference in the other table also when updating the pri key

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

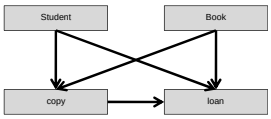
Multiple Cascade Paths

```
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))
```

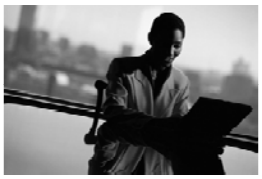

Multiple Cascade Paths



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

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



Copyright © 2013 by Stéphane Bressan