Relational Model

Data Models

- Hierarchical Model 1965 (IMS)
- Network Model 1965 (DBTG)

- **<u>Relational Model</u>** (1NF) 1970s

(*E.F. Codd "A Relational Model for Large Shared Data Banks" Communication of the ACM, Vol 13, #6*)

- 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)*

Designing Database Applications

- Real World

- Logical Model (Relational Model)

  (Logical Data Independence: ``Future users of large data banks must be protected from having to know how the data is organized in the machine'' E.F Codd)

  (We need a model for design and implementation)

- Physical Model

  (need to be understood for tuning – cs3223, cs4221)

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
- 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 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 refer to SQL-92*** *(or SQL2))*

SQL DDL Statement

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

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

- Relations have a **name** : book, student
- Relations have a **schema** which is a list of attributes: title, authors, name, etc.
- Attributes have a **domain**: CHAR(14), DATE, VARCHAR(32), etc. (atomic values).
- The **database** schema is the schema of all the relations.

Relation Instance

relation name ***book***

column

attribute name: domain (or type)

number of columns: degree or arity

table

relation schema

row *t-uple*

number of rows: cardinality

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 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.

SQLite

```
> .help
> .open cs2102.db
> .mode column
> .header on
```

Creating Tables

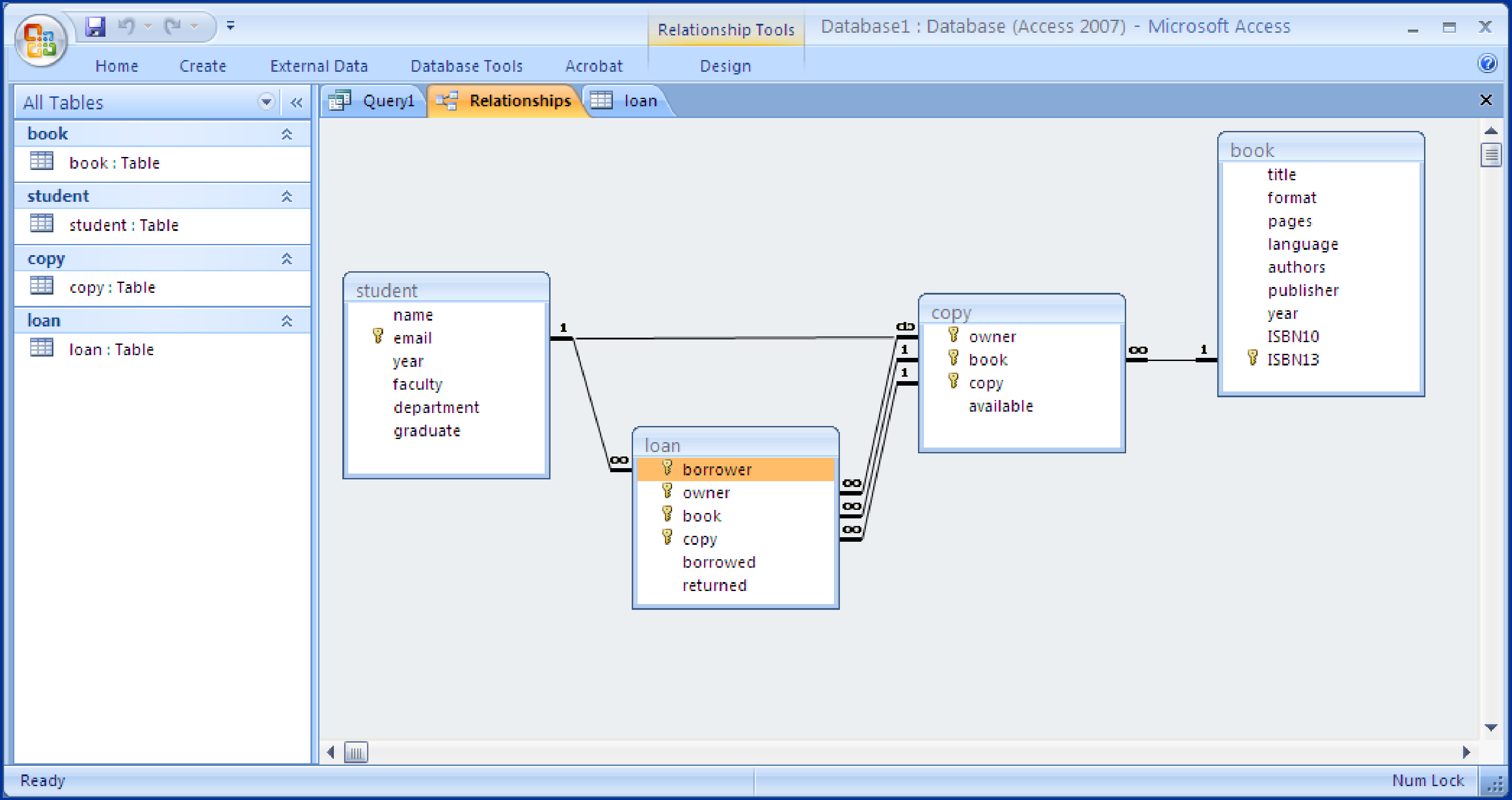
```
CREATE TABLE student (
name VARCHAR(32),
email VARCHAR(256),
year DATE,
faculty VARCHAR(62),
department VARCHAR(32),
graduate DATE);

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

```
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));

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

Database Design: Logical Diagram



Removing Tables, Modifying Tables

```
ALTER TABLE loan ADD test NUMBER;

ALTER TABLE loan MODIFY test CHAR(6);

ALTER TABLE loan DROP COLUMN test;

DROP TABLE loan;


SQLite> .dump loan
```

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('abm@hotmail.com',
'Alif Bin Muhammad',
'School of Computing',
'Computer Science');
```

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='Computer Science';
```

Although SQL allows to insert the results of a query into a table, it is generally an unnecessary and bad idea for either permanent or temporary results.

Views (a very good idea)

Instead we shall use **SQL views**.

```
CREATE VIEW cs_student1 AS
   SELECT name, email, year, graduate
   FROM student
   WHERE faculty='School of Computing'
   AND department='Computer Science';

INSERT INTO student (email, name, faculty, department)
VALUES('momo@hotmail.com',
'Maurice Alphon',
'School of Computing',
'Computer Science');
```

Deleting and Updating Rows

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

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

UPDATE student
SET year=year+1;

DELETE FROM student;
```

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

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: In SQL integrity constraints can be immediate or deferred. You should always use deferred constraints.

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
- You cannot have two t-uples with the same Primary Key in the same table

Column (Value) Constraint – PRIMARY KEY

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

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))
```

NULL Values

- Every domain (type) has an additional value: the NULL value (read Ramakrishnan)
- 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

Column Constraint – NOT NULL

```
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 **combination** 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@~~ 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 copy

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

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 (Doesn’t work! ☹)

```
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’’

Assertions (Doesn’t work! ☹)

```
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
                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

```
USE master;
GO
CREATE DATABASE MyDatabase;
GO
USE MyDatabase;
GO
CREATE TABLE Departments
(
 ID VARCHAR(7) PRIMARY KEY CHECK (ID like 'DE%'),
 name VARCHAR(100),
 domain VARCHAR(3) CHECK(domain = 'FIN' OR domain = 'MAR' OR
domain = 'ADM' OR domain = 'HRM' OR domain = 'CRM' OR domain =
'TCD' OR domain = 'TOD'),


)
```

```
CREATE TABLE Employees
(
 ID VARCHAR(50) PRIMARY KEY CHECK (ID like 'EM%'),
 title VARCHAR(3) CHECK (title = 'Mr' OR title = 'Mrs' OR title ='Ms'),
 full_name VARCHAR(40),
position_em VARCHAR(3) CHECK (position_em = 'CHE' OR
position_em = 'CHA' OR position_em = 'MEM'),
 salary_month MONEY,
 department_ID VARCHAR(7) REFERENCES DEPARTMENTS(ID)
ON DELETE CASCADE


)
```

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