In the Lecture Series Introduction to Database Systems

The Relational Model

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.

Data Model

- A **data model** is a framework for the **definition** of the general form (**schema**) of the **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 meant to be updated*

Designing Database Applications

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

(Logical) 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 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

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 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)

Physical Data Independence

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

Relation Instance

relation name ***column***

book ***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***

Relation Scheme

- A relation or **relation schema** R is a 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)**
- The number of attributes in a relation schema is called the **degree** or **arity** of the relation

Relation Instance

- A **relation instance** 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

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 schema, 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!*

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

SQL

We mostly refer to SQL-92 (or SQL2) and to Oracle's PL/SQL

We use SQLite for demonstration and Oracle for tutorials and project.

SQL

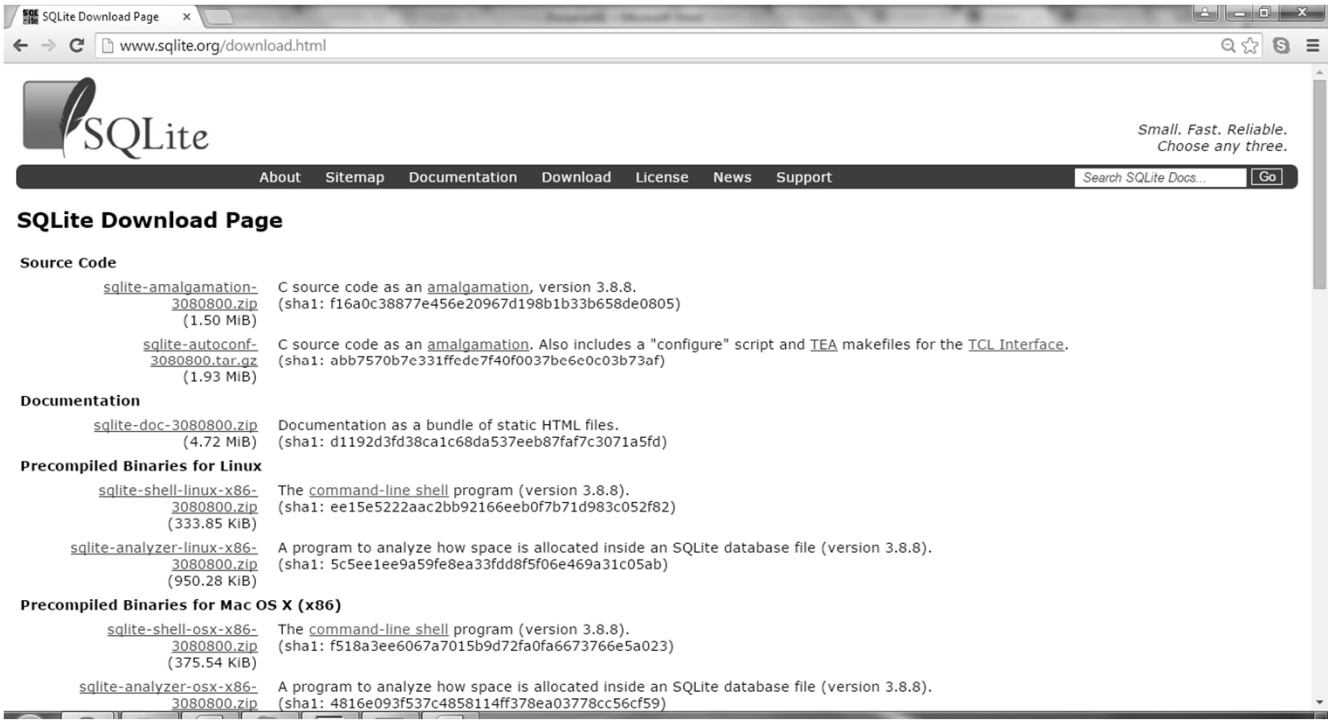
- ***Data Definition Language (DDL)***
  - *Creating, altering and deleting tables and other database objects*
- ***Data Manipulation Language (DML)***
  - *Querying tables*
  - *Inserting, updating and deleting rows*
- Database Control language (DCL)
  - Controlling access to the database

Using SQLite for the Lecture

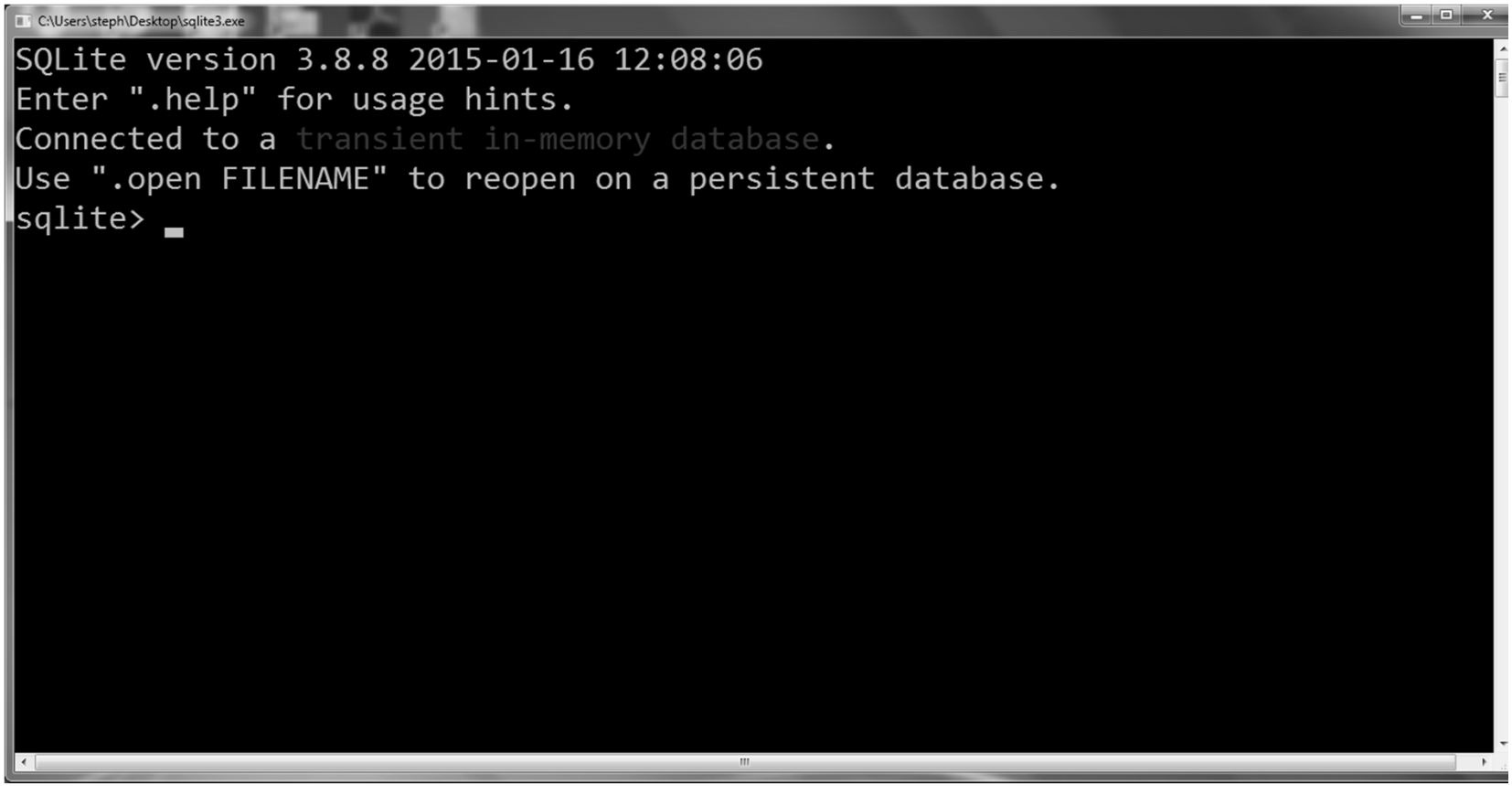
Download SQLite

Link: http://www.sqlite.org/download.html

Download precompiled binaries for your OS and unzip the package



Using SQLite Shell



Using SQLite for the Lecture

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

Creating a Table for Students

We create a table student with six columns of various and appropriate types/domains.

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

SQLite Query Language: C

https://www.sqlite.org/lang_createtable.html

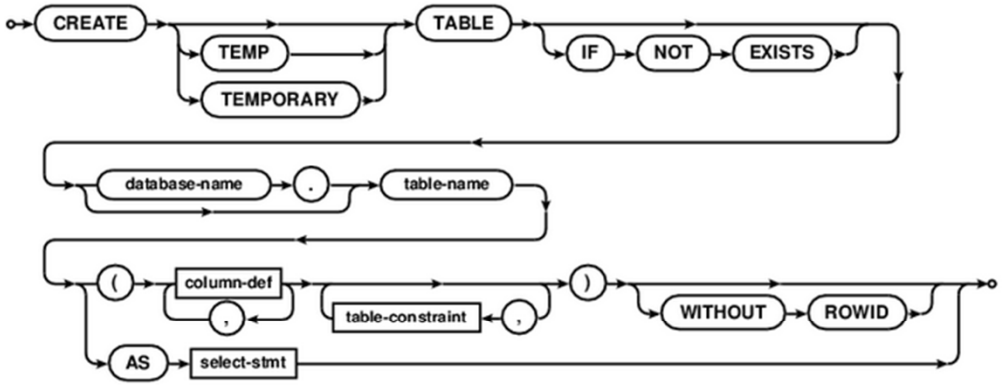
About Sitemap Documentation Download License News Support

SQL As Understood By SQLite

[Top]

CREATE TABLE

create-table-stmt: hide



column-def: show

select-stmt: show

table-constraint: show

The "CREATE TABLE" command is used to create a new table in an SQLite database. A CREATE TABLE command specifies the fol

- The name of the new table.
- The database in which the new table is created. Tables may be created in the main database, the temp database, or in any

Types/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.

Types/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.

Types/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 Types/Domains:

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

Inserting Data

We insert one new student.

```
INSERT INTO student VALUES('JIE JIE’,
'jiejie@hotmail.com',
'2007-01-01',
'School of Computing',
'Computer Science',
'2014-01-01');
```

```
SELECT * FROM student;
```

name	email	year	faculty	department	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	School of Computing	Computer Science	2014-01-01

Inserting Data

```
INSERT INTO student VALUES('DEWI WIJAJA’,
'dw@astaga.co.id',
'2010-01-01',
'School of Computing',
'Computer Science',
'2014-01-01');
```

```
SELECT * FROM student;
```

name	email	year	faculty	department	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	School of Computing	Computer Science	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2010-01-01	School of Computing	Computer Science	2014-01-01

Inserting Data

We insert one new student with incomplete information.

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

```
SELECT * FROM student;
```

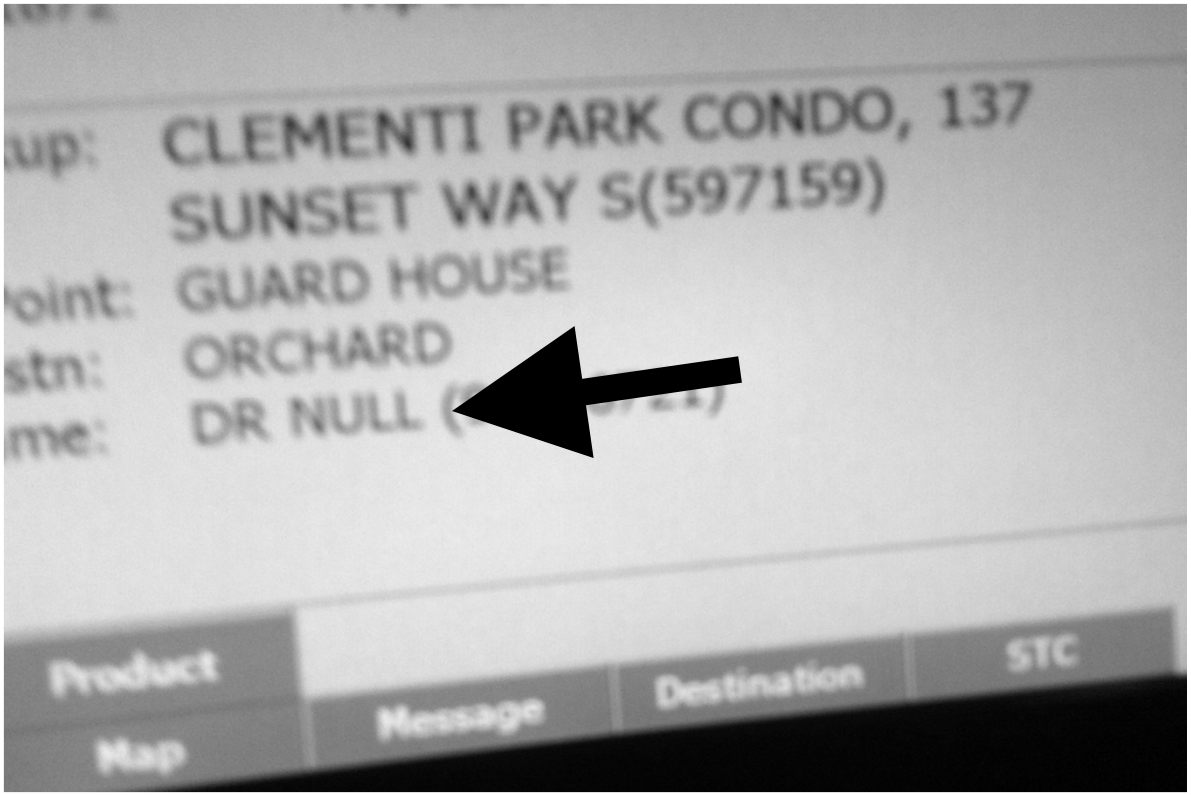
name	email	year	faculty	department	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	School of Computing	Computer Science	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2010-01-01	School of Computing	Computer Science	2014-01-01
XIE XIN	xiexin2011@gmail.c	★	Faculty of Science	Chemistry	★

It creates NULL values.

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

I AM DR NULL


up: CLEMENTI PARK CONDO, 137
SUNSET WAY S(597159)
oint: GUARD HOUSE
stn: ORCHARD
me: DR NULL
Product
Map
Message
Destination
STC

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

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

Modifying the tables

We can modify the schema of existing tables.

```
ALTER TABLE student ADD COLUMN facebook
  VARCHAR(32);
```

```
ALTER TABLE student RENAME TO members;
```

The following do not work with SQLite.

```
ALTER TABLE student MODIFY COLUMN test
  CHAR(6);
```

```
ALTER TABLE members DROP COLUMN year;
```

Copying Some Data

We create a table to copy computer science students.

```
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';
```

Copying Some Data

All computer science students are in the new table.

```
SELECT * FROM cs_student;
```

```
name        email               year        graduate
----------  ------------------  ----------  ----------
JIE JIE     jiejie@hotmail.com  2007-01-01  2014-01-01
DEWI WIJAJ  dw@astaga.co.id     2010-01-01  2014-01-01
```

Creating Views

WE create a view for the computer science students.

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

All computer science students are in the view.

```
SELECT * FROM better_cs_student;
```

```
name        email               year        graduate
----------  ------------------  ----------  ----------
JIE JIE     jiejie@hotmail.com  2007-01-01  2014-01-01
DEWI WIJAJ  dw@astaga.co.id     2010-01-01  2014-01-01
```

Views vs Copies

What is the difference?

```
INSERT INTO student VALUES('VIJAY KUMAR’,
'kumar@gmail.com’,
'2010-01-01',
'School of Computing',
'Computer Science',
'2013-01-01');
```

Views vs Copies

A new computer science student is not in the copy table but is in the view. The view is always fresh!

```
SELECT * FROM cs_student;
```

name	email	year	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2010-01-01	2014-01-01

```
SELECT * FROM better_cs_student;
```

name	email	year	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2010-01-01	2014-01-01
VIJAY KUMA	kumar@gmail.com	2010-01-01	2013-01-01

Logical Data Independence

Views achieve Logical data Independence: User can see what they need in the form they need.

Deleting Data vs Dropping Tables

There is a difference between deleting the content of a table and dropping the table.

```
DELETE FROM cs_student;
```

```
SELECT * FROM cs_student;
```

The result is empty

```
DROP TABLE cs_student;
```

```
SELECT * FROM cs_student;
```

```
Error: no such table: cs_student
```

Selectively Deleting Data

```
SELECT * FROM student;
```

```
name        email               year        faculty              department        graduate
----------  ------------------  ----------  -------------------  ----------------  ----------
JIE JIE     jiejie@hotmail.com  2007-01-01  School of Computing  Computer Science  2014-01-01
DEWI WIJAJ  dw@astaga.co.id     2010-01-01  School of Computing  Computer Science  2014-01-01
XIE XIN     xiexin2011@gmail.c              Faculty of Science   Chemistry
VIJAY KUMA  kumar@gmail.com     2010-01-01  School of Computing  Computer Science  2013-01-01
```

We can selectively delete students.

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

```
SELECT * FROM student;
```

```
name        email               year        faculty              department        graduate
----------  ------------------  ----------  -------------------  ----------------  ----------
JIE JIE     jiejie@hotmail.com  2007-01-01  School of Computing  Computer Science  2014-01-01
DEWI WIJAJ  dw@astaga.co.id     2010-01-01  School of Computing  Computer Science  2014-01-01
VIJAY KUMA  kumar@gmail.com     2010-01-01  School of Computing  Computer Science  2013-01-01
```

Updating Views

In some systems an under some conditions (when it make sense), we can delete/update from a view (view update). It is not the case with SQLite.

```
SELECT * FROM cs_student;
```

```
DELETE FROM better_cs_student WHERE
  graduate<'2014-01-01';
```

```
Error: cannot modify better_cs_student because it is a view
```

When it works, the change is propagating to the base table.

```
SELECT * FROM cs_student;
SELECT * FROM student;
```

Selectively Updating Data

We can selectively update students. Deletions and updates are set-at-a-time: all the matching students are deleted/updated.

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

```
SELECT * FROM student;
```

name	email	year	faculty	department	graduate
JIE JIE	jiejie@hotmail.com	2007-01-01	School of Computing	CS	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2010-01-01	School of Computing	CS	2014-01-01
VIJAY KUMA	kumar@gmail.com	2010-01-01	School of Computing	CS	2013-01-01

Selectively Updating Data

The update can make use of the values before the update to define the values after the update.

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

```
SELECT * FROM student;
```

name	email	year	faculty	department	graduate
JIE JIE	jiejie@hotmail.com	2008	School of Computing	CS	2014-01-01
DEWI WIJAJ	dw@astaga.co.id	2011	School of Computing	CS	2014-01-01
VIJAY KUMA	kumar@gmail.com	2011	School of Computing	CS	2013-01-01

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