Database Systems

Course Project Instruction

Database group

刘思彤　张晓航

The Task is

To Implement a DBMS Prototype.

Policy

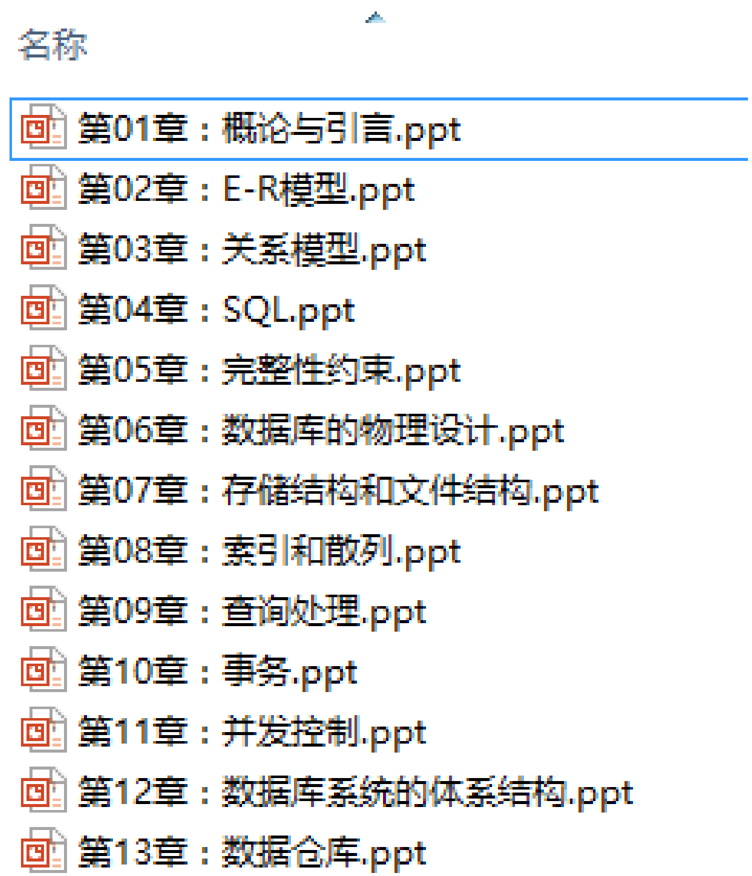
- 2-3 persons form a team.
- 60% of your final score.

DBMS: Problems

- DBMS涉及到的问题
  - 数据库管理系统是为了管理大量、复杂的数据。对数据的管理既涉及到
    - 存：数据存储结构的定义：存得好！
    - 取：数据操作机制的提供：取得快！
  - 如果数据被多用户共享，那么DBMS还必须设法避免可能产生的异常结果，即并发控制；
  - 如果系统发生故障，那么DBMS必须保证将数据恢复到故障发生前的状态，即故障恢复；
  - DBMS还必须保证所存储数据的安全性，不被非法访问和操作，即访问控制……

What We Care

- Correctness
- Response Time
  – Storage
  – Index
  – Caching Strategy
  – Optimizing
  – Query Processing

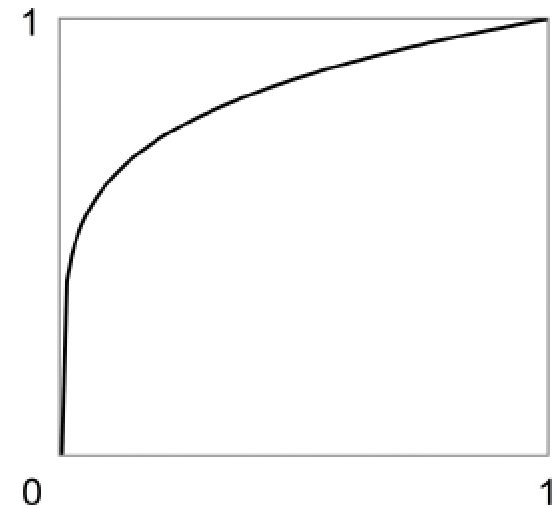


Grading Criteria

Accomplishment	At least one correct run	10
Overall Evaluation	Correctness & Design & Code Quality & Contrib.	10
Performance	S_j = sum(($T_{i,best}$ / $T_{i,j}$)$_{0.2}$) Full * (S_j / S_{best})	30
Documentation	Content & Feature	10
Presentation	For some teams only	≤ 5

Example

	Workload 0	Workload 1
Team 0	5	100
Team 1	10	1000
Team 2	1	Fail



$S_0 = (1 / 5)_{0.2} + (100 / 100)_{0.2} = 1.725$
$S_1 = (1 / 10)_{0.2} + (100 / 1000)_{0.2} = 1.262$
$S_2 = (1 / 1)_{0.2} + (100 / INF)_{0.2} = 1$

$Score_0 = 30 * 1.725 / 1.725 = 30$
$Score_1 = 30 * 1.262 / 1.725 = 22$
$Score_2 = 30 * 1 / 1.725 = 17$

$(1 / 5)_{0.2} = 0.725$
$(1 / 10)_{0.2} = 0.631$
$(1 / 50)_{0.2} = 0.457$
$(1 / 100)_{0.2} = 0.398$
$(1 / 500)_{0.2} = 0.289$
$(1 / 1000)_{0.2} = 0.251$
$(1 / 5000)_{0.2} = 0.182$

The Environment is

- Ubuntu 08.04 LTS 32bit/ 2CPU*4 cores
- g++ 4.4.3
- Intel(R) Xeon(R) CPU E5420@2.50GHz
- 16GB RAM, 512GB*2 Disk

Nonnegotiable

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

[G:\助教\working\讲义&ppt\course\include\client.h](G:\助教\working\讲义&ppt\course\include\client.h)

You Have to Implement

- create()

  Create a new table.

- train()

  Given some query information, train your system and choose the storage and access methods.

- load()

  Load initial data in csv format. The initial data set might be too large to keep in the main memory entirely.

- preprocess()

  Build the indexes and do other preprocessing.

[*G:\助教\working\讲义&ppt\course\client\client.cpp*](G:\助教\working\讲义&ppt\course\client\client.cpp)

You Have to Implement

- execute()

  Execute a query or insert statement.
- next()

  Get the next row from the result set of the last query.
- close()

  Close the sockets and kill other threads.

See course/include/client.h for more details.

You Have to Implement

- execute()

  Execute a query or insert statement.
- next()

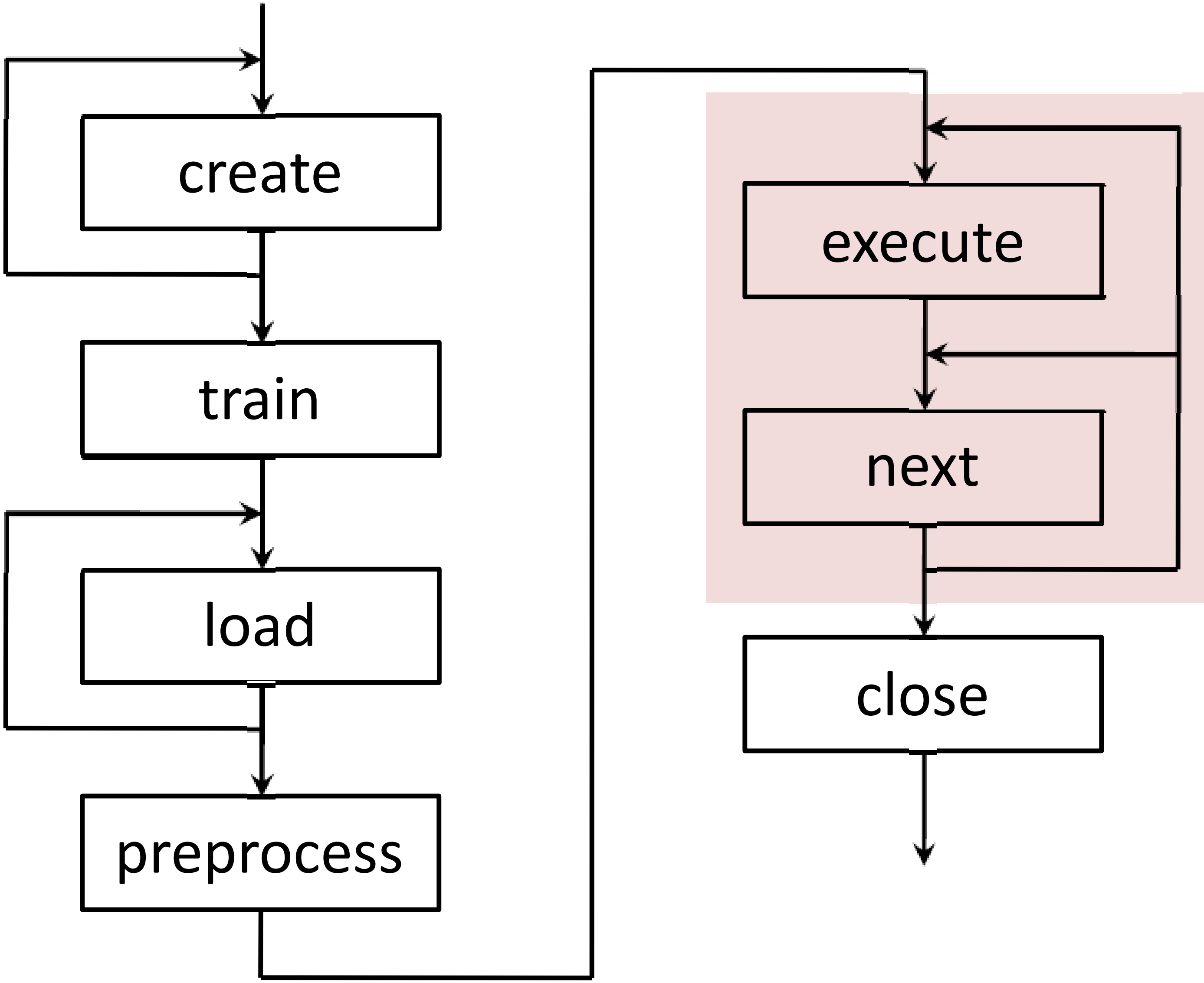
  Get the next row from the result set of the last query.
- close()

  Close the sockets and kill other threads.

WARNING: Run time of execute() and next() will be measured.

See course/include/client.h for more details.

Test Procedure



Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Keep all your source codes in this directory.

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Make sure your Makefile is correct.

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Test procedure. This file will be modified.

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Keep all your data in this directory.

Directory Structure

```
.
├── client
│   ├── client.cpp
│   └── Makefile
├── data
├── include
│   └── client.h
├── instruction.pdf
├── main.cpp
├── Makefile
├── test
│   ├── customer.data
│   ├── item.data
│   ├── _order.data
│   ├── order_line.data
│   ├── query
│   ├── schema
│   └── statistic
└── tool
    ├── hash.cpp
    ├── hash.h
    ├── Makefile
    ├── split_csv.cpp
    ├── split_csv.h
    ├── tokenize.cpp
    ├── tokenize.h
    └── workload.cpp
```

Do NOT use any routines in other folders.

Query Statement

```
SELECT column0, column1, …
FROM table0, table1, …
WHERE condition0 AND … AND conditionN;
```

A condition could be

`column = constant`

`column < constant` (For integers only)

`column > constant` (For integers only)

`column0 = column1` (Join condition)

Query Statement

```
SELECT column0, column1, …
FROM table0, table1, …
WHERE condition0 AND … AND conditionN;
```

No prefix

A condition could be

`column = constant`

`column < constant` (For integers only)

`column > constant` (For integers only)

`column0 = column1` (Join condition)

Query Statement

```
SELECT column0, column1, …
FROM table0, table1, …
WHERE condition0 AND … AND conditionN;
```

The only operator

A condition could be

`column = constant`

`column < constant` (For integers only)

`column > constant` (For integers only)

`column0 =  column1` (Join condition)

Query Statement

```
SELECT column0, column1, …
FROM table0, table1, …
WHERE condition0 AND … AND conditionN;
```

If the FROM-clause contains only one table, there might be no WHERE-clause.

A condition could be

`column = constant`

`column < constant` (For integers only)

`column > constant` (For integers only)

`column0 = column1` (Join condition)

Insert Statement

```
INSERT INTO table
VALUES (value_list0), …, (value_listN);
```

All value lists are in csv format.

```
constant0,constant1,…,constantN
```

Insert Statement

```
INSERT INTO table
VALUES (value_list0), …, (value_listN);
```

No column list

All value lists are in csv format.

```
constant0,constant1,…,constantN
```

Insert Statement

```
INSERT INTO table
VALUES (value_list0), …, (value_listN);
```

Number of rows is important for the train() routine.

All value lists are in csv format.

```
constant0,constant1,…,constantN
```

Format

```
SELECT_a_,_b_
FROM_A_,_B_
WHERE_a_=_5_AND_b_<_10_;

INSERT_INTO_A_VALUES_
(_0,’Stalin’,1879_)_,_
(_1,’Roosevelt’,1882_)_;
```

Data Types

- INTEGER

  32-bit unsigned integer, ‘int’ is OK.

- VARCHAR(d)

  Consist of _, a-z, A-Z, or 0-9. Enclosed by single quotes.
  At most d characters (excluding the quotes).

NOTE:

All identifiers (table names and column names) are string constants not starting with 0-9.

Columns in different tables have distinct names.

String constants don’t contain space, quote, or comma.

Primary Keys

- Primary keys will be assigned to all relations.
- The primary keys will be unique. There is no need to check this constraint.
- The primary keys will be given in ascending order.
- You can just ignore them.

Join Operations

Let nodes represent tables and edges represent join conditions, then each query can be transformed into a graph. This graph should be a tree which

- is connected;
- contains no self-cycles;
- contains no duplicate edges;
- contains no cycles (at least 3 nodes).

Workloads

- Projection
- Selection
- Join

- TPC-C
- TPC-H

About Third-party Library

- You are free to use any third-party library or code about storage, index, multi-thread, network, etc.

  e.g. Boost, Berkeley DB, open-source disk-based B-tree / hash table implementation, etc

- You are forbidden to use any system that is capable to process a SQL query.

  e.g. MySQL, PostgreSQL, etc

- Ask for confirmation if you are not sure.

Document / Presentation

- System Architecture
- Storage Model and the Selection Strategy
- Index Structure and the Selection Strategy
- Caching Strategy
- Query Processing Strategy
  - Heuristic Rules
  - Cost Model
- Other features of your system
- References
- Personal Contribution Rate (For document)

Submission

- Prepare your submission with *make tar*.
- Submit the **.tar.gz* to …
- One submission every 2 hours.
- Only the last submission counts.

Notice

- Some tests may depend on the result of another one.
- Time limitation applies to the whole program, although only execute() and next() affect your final scores.
- Not all the queries are provided in train(), although only those included are measured.

Warnings

- Never do irrelevant operations
- Never replicate other team’s work

Hints

- Read some books and research papers
- Discuss with others
- Start ASAP

create()

Keep the schema safe.

train()

- Find affinitive tables.
- Find affinitive attributes.
- Choose access methods.
- Read-intensive or update-intensive?

load()

Keep the data safe.

preprocess()

- Make some useful statistics.
- Build some indexes.
- Start some threads.

Statistics

- For uniform distribution
  - Size(R), Cnt(R), Card(A), Min(A), Max(A)
  - SF(A = value) = 1 / Card(A)
  - SF(A > value) = (Max(A) - value) / Range(A)
  - SF(A < value) = (value – Min(A)) / Range(A)
  - SF(A0 $\wedge$ A1) = SF(A0) * SF(A1)

- For skewed distribution (e.g., Zipf)
  - Histogram

execute() and next()

- Do all the jobs in execute().
- Do all the jobs in next().
- Do all the jobs in independent threads.

Query Processing

- Google ‘query processing’
- Search on ACM Digital Library (*dl.acm.org*)

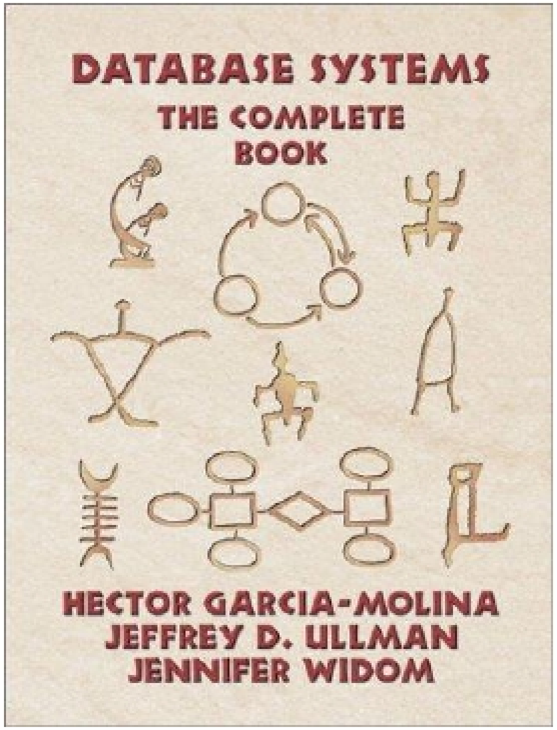
- Cost model vs. Rules

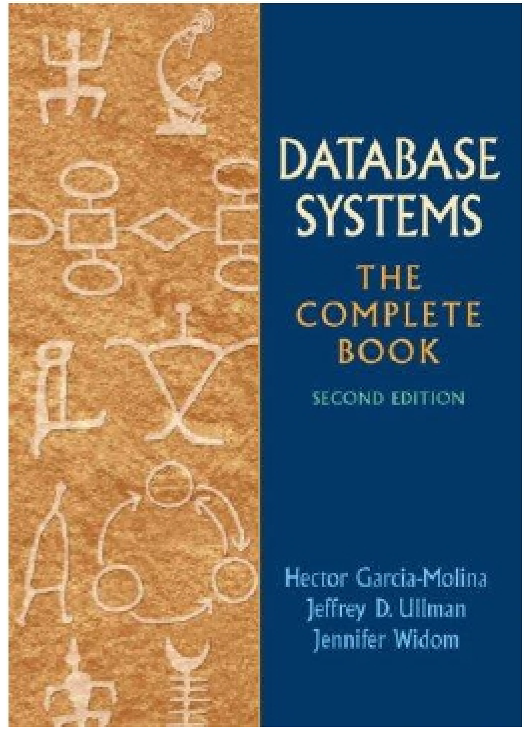
Join Operation

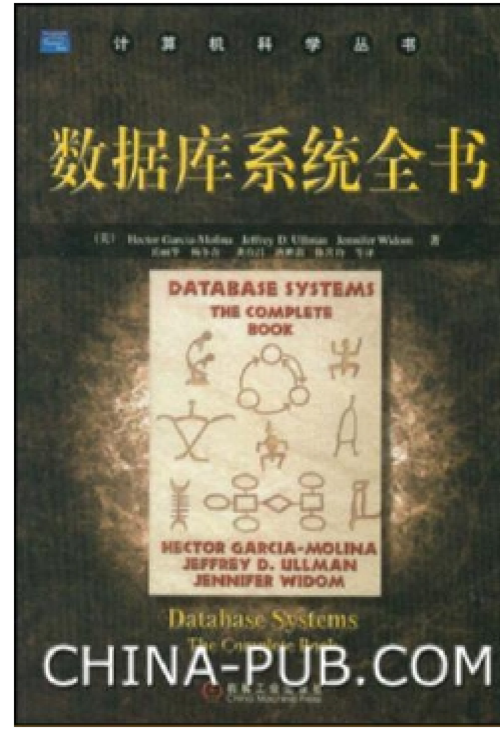
- Nested Loop Join
- Index-based Nested Loop Join
- Sort-Merge Join
- Hash Join (Pruning)

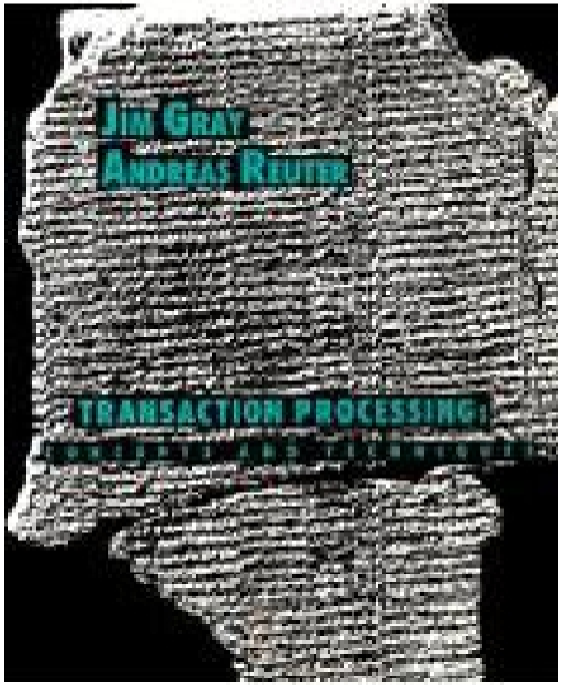
close()

Close your program safely.


DATABASE SYSTEMS
THE COMPLETE BOOK
HECTOR GARCIA-MOLINA
JEFFREY D. ULLMAN
JENNIFER WIDOM


DATABASE SYSTEMS
THE COMPLETE BOOK
SECOND EDITION
Hector Garcia-Molina
Jeffrey D. Ullman
Jennifer Widom


计算机科学丛书
数据库系统全书
DATABASE SYSTEMS
THE COMPLETE BOOK
HECTOR GARCIA-MOLINA
JEFFREY D. ULLMAN
JENNIFER WIDOM
Database Systems
CHINA-PUB.COM


JIM GRAY
ANDREAS REUTER
TRANSACTION PROCESSING:


Transaction Processing
Concepts and Techniques
事务处理
概念与技术
(英文版)
Jim Gray
Andreas Reuter
人民邮电出版社


计算机科学丛书
事务处理
概念与技术
Jim Gray Andreas Reuter 著
JIM GRAY
ANDREAS REUTER
TRANSACTION PROCESSING:
Transaction Processing
Concepts and Techniques

大学计算机教育国外著名教材、教参系列 （影印版）

Principles of Distributed Database Systems

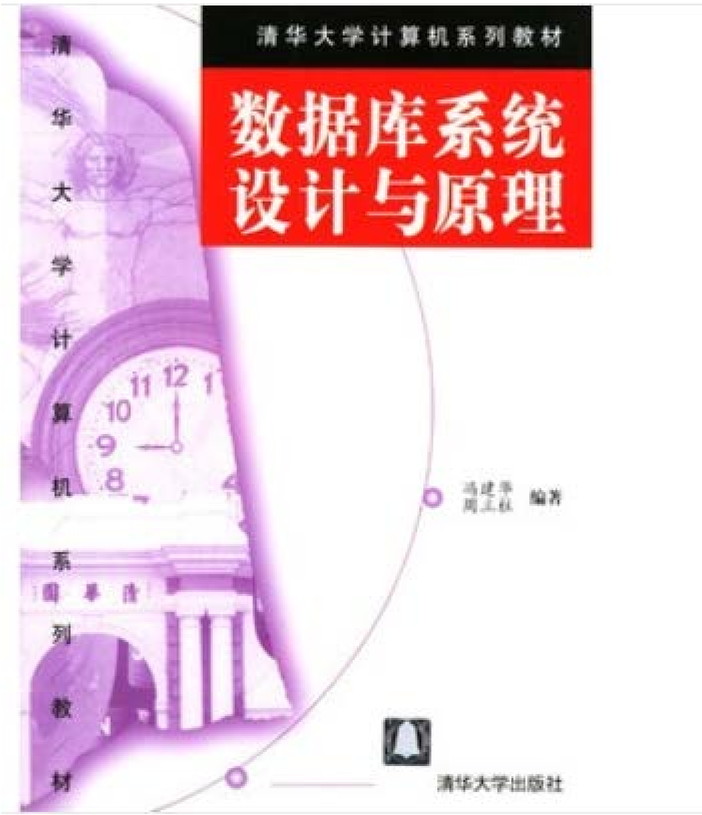
Second Edition

M. Tamer Özsu
Patrick Valduriez

分布式数据库系统原理

（第 2 版）

清华大学出版社


清华大学计算机系列教材
数据库系统
设计与原理
冯建华
周立柱 编著
清华大学出版社


普通高等教育"十一五"国家级规划教材 计算机系列教材
数据库系统
设计与原理（第2版）
冯建华 周立柱 郝晓龙 编著
清华大学出版社

What to do immediately

- Homework： Chapter1-5
- Team partners: 2-3
  - Before 15th Oct.
  - Send me partner name+password
  - stoneliu2010@gmail.com

Contact Information

- 办公室：东主楼 10区204
- 刘思彤
  - 189 1163 5234
  - stoneliu2010@gmail.com
- 张晓航
  - 158 1150 7626
  - zhangxhscut@gmail.com

Good luck and have fun!