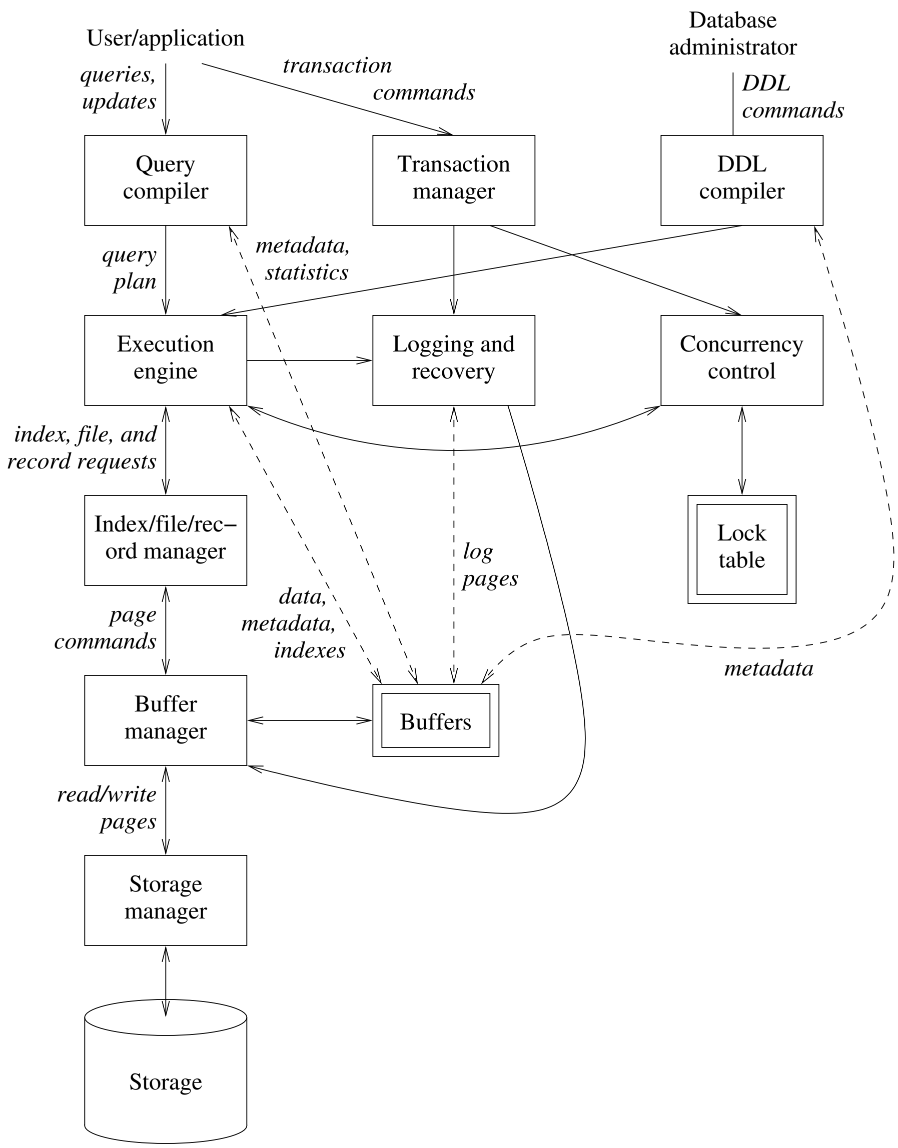
CSCE 520 Final Exam Answer Key
Fall 2018

Do all problems, putting your answers on separate paper. All answers should be reasonably short. The exam is open book, open notes, but no electronic devices. You have two and a half hours.

When you submit your exam pages, staple them together with a single staple in the upper left-hand corner. A stapler will be provided. Make sure your name is on at least the first page. During the test, be sure not to write anything in the area that will be underneath the staple.

There are 150 points total. 130 points constitutes full credit, and undergraduates get a free 13-point boost. Any score in excess of full credit counts as extra credit.

1. (10 points) This question refers to the diagram below, redrawn from Figure 1.1 of the text, describing the major components of a DBMS and their interactions:



Briefly answer ONE of the following questions, specifying which question you are answering:

(a) What is the function of the logging and recovery module, and when is it needed?

(b) What kind of communication occurs between the transaction manager and concurrency control?

Answer:

(a) The logging and recovery module records all changes made during a transation. It is needed when a transaction needs to abort (rollback) either because of a system crash or user command [Optional: or to resolve a deadlock condition.]

(b) The transaction manager tells concurrency control what elements of the DB the transaction accesses, as well as the isolation level of the transaction. Concurrency control decides whether the resources are available for the transaction.

2. (30 points total) Let $R(A, B, C, D, E)$ be a relation whose schema satisfies the following functional dependencies:

- $AC \to D$
- $AD \to B$
- $AE \to C$
- $B \to D$
- $CDE \to A$

(a) (5 points) Compute the closures $\{A, C\}^+$, $\{A, D\}^+$, and $\{A, E\}^+$.

(b) (5 points) List all keys of R. [Hint: there are exactly three keys.]

(c) (5 points) List the FDs that hold in $\pi_{A,D,E}(R)$. You can omit FDs that follow from those you list.

(d) (15 points) Decompose R completely into BCNF relations as efficiently as possible without losing any information (lossless join). Use the method described in the textbook or in class. (You may lose some FDs in the decomposition; that is OK.) There may be more than one correct answer.

Answer:

(a) (Attribute order does not matter.)

$$AC^+ = ABCD \qquad AD^+ = ABD \qquad AE^+ = ABCDE$$

(b) The three keys are AE, BCE, and CDE.

(c) There is only one FD that holds in $\pi_{A,D,E}(R)$. It is $AE \to D$.

(d) There are two possibilities for a *final* decomposition, although each can be obtained via different possible sequences of splits:

$(A, B),\ (A, C, D),\ (A, C, E),\ (B, D),$
$(A, B, C),\ (A, C, E),\ (B, D)$

3. (5 points) Let R and S be the following two tables, respectively:

A	B
1	3
4	2
3	3
3	1

C	D
2	4
3	2
2	2
1	2

Write a table for $\pi_{A,C}(R \bowtie (\gamma_{C,\,\mathtt{avg}(D)\to B}(S)))$, assuming bag operations. Tuple order does not matter.

Answer:

A	C
1	2
4	3
4	1
3	2

4. (45 points total) Assume our usual relational database schema for students taking classes:

```
Student(sid, name, status)
Class(crn, semester, instructor)
Course(crn, prefix, courseNo, title)
Takes(sid, crn, semester, grade)
```

Where:

- The primary key for `Student` is (`sid`).
- The primary key for `Class` is (`crn`, `semester`).
- The primary key for `Course` is (`crn`).
- The primary key for `Takes` is (`sid`, `crn`, `semester`).
- `grade` is of numerical type, between 0.0 and 4.0.

For parts (b,c,d) you may use subqueries as you see fit, provided they are reasonable.

(a) (10 points) Give a relational algebra expression that returns, for each instructor and possible student status, the average grade among all students with the given status taking all courses with that instructor prior to Fall 2018 (a single number for each instructor/status combination). You may use $<$, $\leq$, etc. to compare semesters chronologically.

(b) (10 points) Give an SQL query for the above that is not gratuitously complex.

(c) (10 points) Give a data modification statement in SQL that removes all undergraduate students (status 'UG') from all CSCE courses numbered 700 and above.

(d) (15 points) Recall that the CREATE ASSERTION command in SQL has syntax

```
CREATE ASSERTION <assertion-name> CHECK (<condition>);
```

and creates the global constraint that `<condition>` must hold at all times.

Give a CREATE ASSERTION command that enforces the requirement that no student in any class can have the same name as that class's instructor.

Answer:

(a) (The renaming of the average grade is optional.)

$$\gamma_{\texttt{instructor, status, AVG(grade)}\to\texttt{avgGrade}}(\sigma_{\texttt{semester}<\texttt{Fall2018}}(\texttt{Student} \bowtie \texttt{Class} \bowtie \texttt{Takes}))$$

(b)
```
SELECT instructor, status, AVG(grade) avgGrade
FROM Student, Class, Takes
WHERE Student.sid = Takes.sid AND
      Class.crn = Takes.crn AND
      Class.semester = Takes.semester AND
      Takes.semester < 'Fall2018'
GROUP BY instructor, status;
```

The renaming is optional. There are some other correct answers.

(c)
```
DELETE FROM Takes
WHERE sid IN (SELECT sid FROM Student WHERE status = 'UG') AND
      crn IN (SELECT crn FROM Course WHERE prefix = 'CSCE' AND courseNo >= 700);
```

(d)
```
CREATE ASSERTION CHECK (
    NOT EXISTS (
        SELECT * FROM Student NATURAL JOIN Class NATURAL JOIN Takes
        WHERE name = instructor)
);
```

Other answers are possible.

5. (40 points total) Consider the following relational database schema:

```
Book(ISBN, title, edition, year, publisherName)
Writes(authorName, ISBN)
Publisher(name, address)
```

Assume the following constraints on the data:

Primary Keys: A book is uniquely identified by its ISBN. A publisher is uniquely identified by its name.

Uniqueness: No two books (different ISBNs) may have the same title and edition.

Value Constraints: The title of a book cannot be null.

Referential Integrity: An ISBN appearing in the Writes table must also appear in the Book table. A publisherName appearing in the Book table must appear as a name in the Publisher table.

(a) (15 points) Give `CREATE TABLE` commands in SQL for the three relations above, giving attribute types that are reasonably appropriate and consistent. Also incorporate the given constraints. (Note that `edition` should be an integer, 1 for first, 2 for second, etc.)

(b) (10 points) Express the constraint (in the form $R = \emptyset$, where R is some expression in relational algebra) that no two editions of any title can appear in the same year.

(c) (15 points) Suppose the tables described in the last problem above are made up of the following tuples (rows sorted by key and blank entries NULL):

Book:

ISBN	*title*	*edition*	*year*	*publisherName*
140	Winning Ways	1	1982	Academic Press
143	Automatic Sequences		2003	Cambridge
207	Structural Complexity	1	1988	Springer
354	Quantum Mechanics	1	1979	Springer
399	Structural Complexity	2	1990	Springer
446	Algebraic Coding Theory		1968	McGraw-Hill
591	Aperiodic Order	1	2013	Cambridge
651	The Domino Problem		1966	AMS
829	Proofs from THE BOOK	4	2010	Springer
904	On Numbers and Games	2	1976	Academic Press

Writes:

authorName	*ISBN*
Aigner, M.	829
Allouche, J.-P.	143
Baake, M.	591
Balcázar, J.L.	207
Balcázar, J.L.	399
Berger, R.	651
Berlekamp, E.R.	140
Berlekamp, E.R.	446
Böhm, A.	354
Conway, J.H.	140
Conway, J.H.	904
Díaz, J.	207
Díaz, J.	399
Gabarró, J.	207
Gabarró, J.	399
Grimm, U.	591
Guy, R.	140
Shallit, J.	143
Ziegler, M.	829

Publisher:

name	*address*
Academic Press	New York
ACM	New York
AMS	Providence
Cambridge	Cambridge
Elsevier	Amsterdam
IEEE	New York
McGraw-Hill	New York
Springer	Heidelberg
Trauner	Linz

What is returned by the following SQL queries?

i.
```
SELECT name FROM Publisher
WHERE address NOT IN (
    SELECT address FROM Book, Publisher
    WHERE publisherName = name AND year < 1980);
```

ii.
```
SELECT title, year
FROM Book NATURAL JOIN (
    SELECT publisherName, min(ISBN) minISBN FROM Book
    GROUP BY publisherName) P
WHERE ISBN = minISBN;
```

iii.
```
(SELECT title
 FROM Book, Writes
 WHERE Book.ISBN = Writes.ISBN AND authorName < 'B')
UNION
(SELECT title
 FROM Book
 WHERE edition > 1)
UNION
(SELECT title
 FROM Book, Publisher
 WHERE publisherName = name AND address = 'New York');
```

Answer:

(a) There is some leeway in the data types here.

```
CREATE TABLE Book (
    ISBN           INTEGER(3)    PRIMARY KEY,
    title          VARCHAR(50)   NOT NULL,
    edition        INTEGER,
    year           INTEGER(4),
    publisherName  VARCHAR(30)   REFERENCES Publisher(name),
    UNIQUE(title, edition)
);

CREATE TABLE Writes (
    authorName  VARCHAR(25),
    ISBN        INTEGER(3)    REFERENCES Book
);

CREATE TABLE Publisher (
    name      VARCHAR(30)   PRIMARY KEY,
    address   VARCHAR(20)
);
```

(b) Here are two slightly different answers:

$$\sigma_{\texttt{B1.title}=\texttt{B2.title}\ \&\ \texttt{B1.edition}<\texttt{B2.edition}\ \&\ \texttt{B1.year}=\texttt{B2.year}}(\rho_{\texttt{B1}}(\textsf{Book}) \times \rho_{\texttt{B2}}(\textsf{Book})) = \emptyset$$

$$\sigma_{\texttt{B1.title}=\texttt{B2.title}\ \&\ \texttt{B1.edition}<>\texttt{B2.edition}\ \&\ \texttt{B1.year}=\texttt{B2.year}}(\rho_{\texttt{B1}}(\textsf{Book}) \times \rho_{\texttt{B2}}(\textsf{Book})) = \emptyset$$

(c) Here is the MySQL output of the three queries in succession. Row ordering is not important.

```
+-----------+
| name      |
+-----------+
| Cambridge |
| Elsevier  |
| Trauner   |
+-----------+
3 rows in set (0.00 sec)

+-------------------------+------+
| title                   | year |
+-------------------------+------+
| Winning Ways            | 1982 |
| Automatic Sequences     | 2003 |
| Structural Complexity   | 1988 |
| Algebraic Coding Theory | 1968 |
| The Domino Problem      | 1966 |
+-------------------------+------+
5 rows in set (0.00 sec)

+-------------------------+
| title                   |
+-------------------------+
| Proofs from THE BOOK    |
| Automatic Sequences     |
| On Numbers and Games    |
| Structural Complexity   |
| Winning Ways            |
| Algebraic Coding Theory |
+-------------------------+
6 rows in set (0.00 sec)
```

6. (20 points total) This problem refers to an XML document with bibliographic data, stored locally with file name `bibliography.xml`.

```
<Bibliography>
    <Bibitem id = "rbg" type = "incollection" cites = "ro st">
        <Author>K. Ambos-Spies</Author>
        <Title>Resource-bounded genericity</Title>
        <Year>1996</Year>
    </Bibitem>
    <Bibitem id = "ro" type = "article" cites = "st">
        <Author>C.H. Bennett</Author>
        <Author>J. Gill</Author>
        <Title>Random oracles</Title>
        <Journal>SIAM J. Comput.</Journal>
        <Year>1981</Year>
    </Bibitem>
    <Bibitem id = "aft" type = "techreport" cites = "qcn">
        <Author>D. Coppersmith</Author>
        <Title>An approximate Fourier transform</Title>
        <Year>1994</Year>
    </Bibitem>
    <Bibitem id = "qcn" type = "article" cites = "ro">
        <Author>D. Deutsch</Author>
        <Title>Quantum computational networks</Title>
        <Year>1989</Year>
    </Bibitem>
    <Bibitem id = "pcpc" type = "inproceedings" cites = "ro qcn">
        <Author>S. Goldwasser</Author>
        <Author>M. Sipser</Author>
        <Title>Private coins versus public coins</Title>
        <Year>1986</Year>
    </Bibitem>
    <Bibitem id = "st" type = "book">
        <Author>T. Jech</Author>
        <Title>Set Theory</Title>
        <Year>1978</Year>
    </Bibitem>
</Bibliography>
```

(a) (10 points) What, specifically, is returned by the following XQuery expression when run on the document above?

```
let $bibs := doc("bibliography.xml")/Bibliography
let $titles := (
        for $paper1 in $bibs/Bibitem[@type != "book"]
        for $paper2 in $bibs/Bibitem[Year >= 1990]
        where $paper1/@id = $paper2/@cites
        return $paper1/Title
    )
return <RecentlyCited>{$titles}</RecentlyCited>
```

For readability's sake you may break lines and indent as appropriate.

(b) (10 points) Write an XQuery expression that returns an element with tagname `BGciters` that contains as subelements those authors that cite the 1981 paper of Bennett and Gill. Your query should work in general, not just on the specific data given above, but you can assume that the `id` attribute value `"ro"` identifies the Bennett and Gill paper.

Answer:

(a)
```
<RecentlyCited>
    <Title>Random Oracles</Title>
    <Title>Quantum Computational Networks</Title>
</RecentlyCited>
```

(b)
```
let $bibs := doc("bibliography.xml")/Bibliography
let $authors := (
        for $item in $bibs/Bibitem
        where $item/@cites = "ro"
        return $item/Author
    )
return <BGciters>{$authors}</BGciters>
```

Here is an alternative answer:

```
let $bibs := doc("bibliography.xml")/Bibliography
let $authors := (
        for $item in $bibs/Bibitem[@cites = "ro"]
        return $item/Author
    )
return <BGciters>{$authors}</BGciters>
```