CS54100 Spring 2012 Midterm *draft solutions*, 7 March, 2012
Prof. Chris Clifton

Turn Off Your Cell Phone. Use of any electronic device during the test is prohibited.

Time will be tight. If you spend more than the recommended time on any question, **go on to the next one**. If you can't answer it in the recommended time, you are either giving too much detail or the question is material you don't know well. You can skip one or two parts and still demonstrate what I believe to be an A-level understanding of the material.

Scoring notes are preliminary and subject to change.

1 Relational Modeling (20 minutes, 17 points)

Given the following relational table **F**unding:

***A**ward#*	***P**I*	***C**oPI*	***T**itle*	***D**ivision*	***V**alue*	***S**tatus*
1012208	Clifton	Raskin	TextAnon	IIS	$42800	Funded
1012208	Clifton	Hill	TextAnon	IIS	$532800	Funded
1012208	Clifton	Jiang	TextAnon	IIS	$38200	Pending
0673957	Clifton	Kisselburgh	Discrimination	III	$28300	Review
0673957	Clifton	Kisselburgh	Discrimination	SBE	$28300	Review
1117766	Aref	Ouzzani	Spatiotemporal	III	$52300	Funded
0492734	Bertino	None	Access	TC	$28300	Review
1016722	Bertino	None	Matching	TC	$24000	Funded
1111512	Bertino	Ouzzani	Provenance	TC	$163000	Funded
1111512	Bertino	Ghinita	Provenance	TC	$163000	Funded

(Note that you can abbreviate attributes using only the first letter.)

1.1 Keys (2 minutes, 2 points)

Give two keys for table **F**unding.

A C D ; A P C D

Scoring: 1 each. Note that there are multiple keys, as a superset of a key is also a key.

1.2 Functional Dependencies (6 minutes, 5 points)

List all functional dependencies that hold on the given data. (It is okay if you give only a minimal cover.) *I suggest you write the ones that are obvious to you, then go on, and come back at the end of the exam if you have time to see if you have them all. The number of points doesn't reflect the number of dependencies you should find.*

A → P

Scoring: 1 for each valid FD, up to 3, 1 for key → everything (either a correct key, or one listed in 1.1), 1 for a valid FD that holds in the data, but isn't particularly sensible, 1 for a dependency involving multiple LHS attributes, -1 for an invalid FD, max 4 points with an invalid FD.

1.3 Queries (2 minutes, 2 points)

Give a queries to return the total ***V**alue* of funding for each ***P**I* (note that this will be a single number for each *PI*.) You may express this in either SQL or in Relational Algebra.

Select PI, sum(value) from Funding group by PI;

$\gamma_{PI,sum(value)} Funding$

Scoring: 1 for basic idea of group-by, 1 for syntactically/semantically correct.

1.4 Join (5 minutes, 4 points)

Assume you are given the following decomposition of the table in Section 1:

Left	**R**ight
A*ward#* ***PI*** ***C****oPI* ***V****alue* ***S****tatus*	***PI*** ***C****oPI* ***T****itle* ***D****ivision*

1.4.1 Queries (3 minutes, 2 points)

Give a query, in both SQL and Relational Algebra, to join the two tables.

Select * from Left, Right where Left.PI = Right.PI and Left.CoPI = Right.CoPI;

$L \bowtie_{PI=PI \cap CoPI=CoPI} R$

Scoring: 1 for correct SQL, 1 for correct Relational Algebra.

1.4.2 Lossless Join (2 minutes, 2 points)

Is this a lossless join decomposition of the table? Explain why or why not.

No, this is not a lossless join. As neither side is a key, we can (and do) end up with tuples that did not exist in the original relation, in particular Prof. Bertino having two entries with different amounts for Awards 0492734 and 10106722.

Scoring: 1 for no, 1 for reason why and/or example.

1.5 Normalization (5 minutes, 4 points)

Assume you are told that the following dependencies form a canonical cover for the dependencies that hold on the data:

A*ward#* $\longrightarrow$ ***PI*** ***A****ward#* $\twoheadrightarrow$ ***D****ivision*

Give a good decomposition of the database. Explain (briefly) which normal form you are using and why.

$\underline{\text{A}}$ P **$\underline{\text{A D}}$** **A C T V S**

This is a fourth-normal four decomposition, which enforces the multivalued dependency as well as the functional dependency. It is also BCNF, as for every functional dependency, the left hand side is a key for any relation containing all the attributes of the functional dependency.

Scoring: 1 for each dependency, 1 for lossless join decomposition, 1 for proper explanation of form used.

2 Buffer Management (10 minutes, 8 points)

Assume that you are deciding on a page layout for the data pages associated with a primary index supporting range queries. You need to decide between Fixed-Length Records and Variable-Length Records.

A. Give an advantage of Fixed-Length Records.

Fixed-length records can be easier to search in memory, particularly if stored in a sorted order, as we can calculate the location of a record without having a separate directory of records, and quickly scan the records to determine everything in the range of the query.

Scoring: 1 for showing understanding of fixed length records, 1 for good answer.

B. Give an advantage of Variable-Length Records.

Variable-length records may take less space, allowing us to put more records on a page and reduce the number of IOs to retrieve the full range.

Scoring: 1 for showing understanding of variable length records, 1 for good answer.

C. Which do you expect would result in more I/Os? Explain.

I would expect additional I/Os for fixed-length records, as for a given range, it may require more data pages to hold all records in that range.

Scoring: 1 for showing understanding of evaluation of cost based on I/O, 1 for good answer explaining tradeoff.

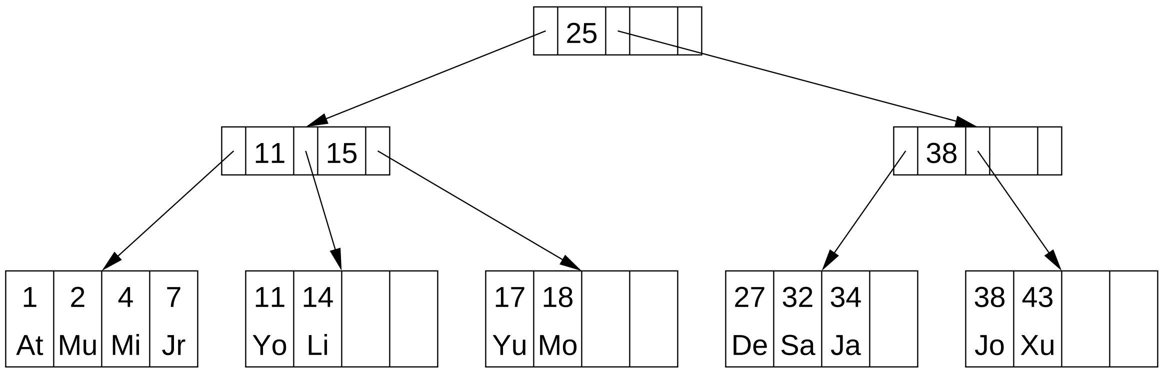
D. Would any of your answers change if this were a secondary index?

In a secondary index, the records could not be sorted by the index key (since they would already be sorted by the primary index's key.) Thus the advantage given for fixed-length records would disappear – we would still have to scan the entire page for records in the range of the query.

Scoring: 1 for showing understanding of term secondary index, 1 for good answer.

3 Indexing (10 minutes, 8 points)

Given the following B+ tree for relation R attribute a:



A. Explain how to use the index for the query:

```
SELECT * FROM R WHERE 20 < a AND a <= 35;
```

By going left at the first node and picking the right pointer from the 11/15 node, we get the first page that could contain values in the range > 20. By going right at the first node (≥ 35) and left at the second node ($35 < 38$), we get a pointer to the last page needed. Assuming the pages were chained, we could then start from the first and scan to the last. As shown we would need to get them from the intermediate nodes.

Scoring: 1 for basic search idea, 1 for showing each end of range, 1 for picking up pointers from intermediate notes or saying chained.

B. Show the tree after insert(36, Cl). *You may write your answer on the tree.*

Place 36 Cl in the empty spot in the fourth bucket.

Scoring: 1 for showing idea, 1 for getting it right.

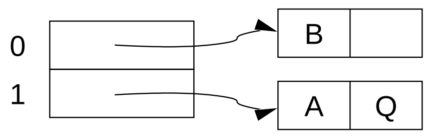
C. Show the tree after insert(5, Ta) *You may write your answer on the tree, as long as you distinguish it from your answer to B.*

This requires splitting buckets or reorganizing at lower nodes. In addition, the intermediate node would then need to be either split or have values changed. I haven't shown the solution to avoid confusion between the question and this answer.

Scoring: 1 for insert, 1 for split/move at lower nodes, 1 for split/change internal nodes.

4 Hashing (10 minutes, 5 points)

Given the following extensible hash table and hash function:



x	*h(x)*	*x*	*h(x)*
A	00100001	N	00101110
B	00100010	O	00101111
C	00100011	P	00110000
D	00100100	Q	00110001
E	00100101	R	00110010
F	00100110	S	00110011
G	00100111	T	00110100
H	00101000	U	00110101
I	00101001	V	00110110
J	00101010	W	00110111
K	00101011	X	00111000
L	00101100	Y	00111001
M	00101101	Z	00111010

A. Give an insert that would not cause a split.

insert(F)

Scoring: 1 for insert into 0.

B. Give an insert that would more than double the size of the directory.

insert(I)

Scoring: 1 for insert causing directory doubling, 1 for causing doubling twice.

C. What is the greatest number of additional *data* pages (i.e., not counting the directory) that could be required for inserting a single value? Explain.

One. Even though the directory may grow, it will point to existing datapages except for the one page being split - and a split page will always have room for one more...

Scoring: 1 for "1", 1 for explanation.