Dr. Stefan Brass
School of Information Sciences
University of Pittsburgh

July 24, 2001

INFSCI 2711 "Database Analysis and Design" — Example II for Final Exam —

Instructions

- This was the midterm exam from the Thursday session of INFSCI 2711 in Fall 1999.
- There was 1 hour and 50 minutes time to work on the exercises.
- In the multiple-choice questions, was possible that more than one answer is correct. This was not very fair, since points were given only if all correct boxes were check. I will not repeat this error on our exam.
- In our final exam, there will be no exercise like Exercise 1 in this exam. Nevertheless, knowledge of SQL is required for data dictionary queries.

Exercise 1 (SQL Errors) 6 Points

In this exam, we consider a database for a small video rental service. For simplicity, we do not consider multiple copies of the same video cassette which might be available (they are treated like distinct cassettes).

- CUSTOMER(<u>CUST_NO</u>, NAME, ADDRESS, PHONE)
- RENTED(<u>CUST_NO</u>→CUSTOMER, <u>VID_NO</u>→VIDEO, <u>START_DATE</u>, RETURNED_DATE)
  When the video cassette is rented, only START_DATE is set (to the current date), and RETURNED_DATE is null. When the cassette is returned, RETURNED_DATE is set to the then current date. A customer can in principle rent the same cassette more than once, therefore START_DATE is also part of the key.
- VIDEO(<u>VID_NO</u>, TITLE, RENT_COUNT)
  The column RENT_COUNT contains the number of times this cassette was rented.

Suppose we want to find customers who have rented at least two times a "Star Wars" cassette. Let us assume that there are different such cassettes, but all contain the substring "Star Wars" in their title.

The queries in a), b) and c) try to solve this request. Which of these solutions are correct, and which are incorrect? If a solution is incorrect, please give a very short explanation.

a)
```
SELECT C.NAME
FROM   CUSTOMER C, RENTED X, RENTED Y, VIDEO V
WHERE  C.CUST_NO = X.CUST_NO AND C.CUST_NO = Y.CUST_NO
AND    X.VID_NO = V.VID_NO AND Y.VID_NO = V.VID_NO
AND    V.TITLE LIKE '%Star Wars%'
```

☐ Correct
☐ Wrong, Reason: ______________________________

b)
```
SELECT C.NAME
FROM   CUSTOMER C
WHERE  2 <= (SELECT COUNT(*) FROM VIDEO V
              WHERE V.TITLE LIKE '%Star Wars%')
```

☐ Correct
☐ Wrong, Reason: ______________________________

c)
```
SELECT   C.NAME
FROM     CUSTOMER C, RENTED R, VIDEO V
WHERE    C.CUST_NO = R.CUST_NO AND R.VID_NO = V.VID_NO
AND      V.TITLE LIKE '%Star Wars%'
GROUP BY C.CUST_NO, C.NAME
HAVING   COUNT(*) >= 2
```

☐ Correct
☐ Wrong, Reason: ____________________

Suppose that now we want to find the cassette which was rented most often. The answer does not have to be unique: If two cassettes were rented 150 times, and no cassette was rented more than 150 times, both of these cassettes should be printed.

Which of these solutions are correct, and which are incorrect?

d)
```
SELECT V.VID_NO, V.TITLE
FROM   VIDEO V
WHERE  V.RENT_COUNT = MAX
```

☐ Correct
☐ Wrong, Reason: ____________________

e)
```
SELECT V.VID_NO, V.TITLE
FROM   VIDEO V
WHERE  RENT_COUNT = (SELECT MAX(X.RENT_COUNT) FROM VIDEO X)
```

☐ Correct
☐ Wrong, Reason: ____________________

f)
```
SELECT V.VID_NO, V.TITLE
FROM   VIDEO V
WHERE  NOT EXISTS(SELECT * FROM VIDEO X
                  WHERE X.RENT_COUNT > V.RENT_COUNT)
```

☐ Correct
☐ Wrong, Reason: ____________________

Exercise 2 (Disks and Buffering) 6 Points

a) The time needed to access a given sector on a disk consists of three components. Which of these components improve when the disk platters spin faster (e.g. 10000 rpm instead of 5800 rpm)? As always, more than one answer may be correct.

- ☐ Seek time
- ☐ Latency time
- ☐ Transfer time (from disk surface to disk cache/memory)
  Starts when the disk head is over the beginning of the sector/block and ends when the disk head is at the end of the block.

b) If you read blocks which are consecutively stored on the disk, how much data can you read per second?

- ☐ Less than 50 KByte
- ☐ 50–200 KByte
- ☐ 200 KByte–1 MByte
- ☐ 1–20 MByte
- ☐ Above 20 MByte

c) Suppose your buffer cache has space for 200 database blocks of 2 KByte each. Let us assume that it contains some useful information, e.g. the root blocks of some indexes. Now you do a full table scan on a table which is 1 MByte long (and has no special declarations). Will this overwrite the entire contents of the buffer cache in Oracle?

- ☐ Yes. Afterwards the buffer cache contains only blocks from this table.
- ☐ No. Only some blocks of the buffer cache are used in the full table scan.

d) Suppose you increase the `DB_BLOCK_SIZE` from 2 KByte to 8 KByte. Will an average block access then need 4 times as much time?

- ☐ Yes.
- ☐ No, the average access to one 8 KByte block is faster than 4 independent accesses to 2 KByte blocks.
- ☐ No, reading one 8KByte block actually takes more time than reading 4 blocks of 2 KByte each.

You can assume that 8 KByte are still within the operating system limits, i.e. the OS buffer is large enough to read a block of 8 KByte in a single request to the disk.

e) Suppose you do RAID Level 4 (striping with block-wise parity information). You use 5 disks (4 for the data, 1 for the parity blocks). You have mainly single-block read accesses, which distribute well over the 4 disks, so you can process nearly 200 read requests per second (assuming a single disk can process 50 requests per second). Now one of your 4 data disks fails. How many read requests per second can you now process?

- ☐ None. I must first replace the disk.
- ☐ Around 50.
- ☐ Around 100.
- ☐ Around 150.
- ☐ Unchanged: 200.

f) Suppose the `CUSTOMER` table is quite large, and is accessed very often via the index on its key. Table and index would fit on one disk, but you have another empty disk. What would give you better performance?

- ☐ Put table and index on the same disk.
- ☐ Put table and index on different disks.

Exercise 3 (Access Paths) — 6 Points

a) Suppose you accidentally deleted an index. Will your application programs which previously used this index still run (maybe slower)?

- ☐ They will run, but probably slower.
- ☐ The end user will not note any difference (also not in response time).
- ☐ They will not run.

b) Consider the table `RENTED(CUST_NO, VID_NO, START_DATE, RETURNED_DATE)`. Suppose you have created a B-tree index with the command

```
CREATE INDEX I_RENTED_START_RETURNED ON
             RENTED(START_DATE, RETURNED_DATE)
```

Which of the following conditions can be evaluated using the index? Please check all correct answers.

- ☐ `START_DATE BETWEEN '15-OCT-99' AND '20-OCT-99'`
- ☐ `RETURNED_DATE >= '20-OCT-99'`
- ☐ `RETURNED_DATE - START_DATE > 4`
- ☐ `START_DATE = '15-OCT-99' AND RETURNED_DATE > '16-OCT-99'`
- ☐ `START_DATE = '15-OCT-99' OR RETURNED_DATE > '16-OCT-99'`

c) Suppose you store `RENTED` in an index cluster, clustered by `CUST_NO`. Will this impose any limit on the table?

- ☐ No. A cluster behaves in the same way as a heap file in this aspect.
- ☐ If the number of `RENTED` rows for a single customer becomes larger than was assumed in the `SIZE` calculation, performance might suffer.
- ☐ If the number of `RENTED` rows for a single video becomes larger than was assumed in the `SIZE` calculation, performance might suffer.
- ☐ The total number of rows is limited. When you create the cluster, you define its expected `SIZE`. Once this space is used up, you cannot insert any further rows.

d) Can it have any use to declare a composed index on all columns of a table? E.g. an index on `VIDEO(VID_NO, TITLE, RENT_COUNT)`? Please check all correct statements.

- ☐ If the key consists of all columns (not in the `VIDEO` example), we still need an index to enforce it (as for any other key).
- ☐ This avoids the TABLE ACCESS (BY INDEX ROWID) whenever the index is usable.
- ☐ For every attribute `A` of the table, a condition A='c' can be evaluated faster via this index than with a full table scan of the table.

e) Consider the table

RENTED(<u>CUST_NO</u>, <u>VID_NO</u>, <u>START_DATE</u>, RETURNED_DATE)

There will be many queries refering to currently borrowed cassettes. We want to keep information about all renting events in the past 12 months in the table `RENTED`. Video cassettes are usually borrowed only for a few days. One option would be to partition the table into

RENTED_ARCHIVE(<u>CUST_NO</u>, <u>VID_NO</u>, <u>START_DATE</u>, RETURNED_DATE)
RENTED_OPEN(<u>CUST_NO</u>, <u>VID_NO</u>, <u>START_DATE</u>)

Here `RENTED_OPEN` will contain information only about the currently borrowed cassettes. When a cassette is returned, the entry in `RENTED_OPEN` is deleted and an entry is written into `RENTED_ARCHIVE`. What do you think of this solution? Please check all true statements.

- ☐ We will save significant storage space by not including the column `RETURNED_DATE` in `RENTED_OPEN`. In this way, we do not have to store the null value explicitly.
- ☐ `RENTED_OPEN` will be much smaller than `RENTED`, so a full table scan of `RENTED_OPEN` might be feasable, whereas a full table scan of `RENTED` might be prohibitively expensive.
- ☐ Indexes on `RENTED_OPEN` will be more effective than indexes on the entire table `RENTED` because they are smaller.
- ☐ If an index on `RENTED_OPEN` returns multiple ROWIDs, and we need to look up the corresponding rows, the chances that two ROWIDs are in the same block and that this block is still in the cache are higher than for indexes on the entire table `RENTED`.
- ☐ `CREATE VIEW RENTED_OPEN AS`
  `SELECT * FROM RENTED WHERE RETUNED_DATE IS NULL`
  has the same advantages for performance (efficient query evaluation).

f) Suppose you access a table by ROWID:

```
SELECT * FROM R WHERE ROWID = 'AAAAkPAA/AAAAADAAL'
```

How many block accesses does this query need? A single row will be not more than 200 Bytes long (shorter than a block). Rows in this table can be updated, and can grow (but still remain less than 200 Bytes). You cannot assume anything about `PCTFREE`.

- ☐ At most one (0 if the block is in the cache)
- ☐ Normally one, at most 2 (0 if in cache)
- ☐ Normally one, but theoretically there is no upper limit (0 if in cache).

Exercise 4 (Heap Files) 1+4+1=6 Points

a) Suppose it is an option to export the data of a table, recreate the table, and import all the data again. Please check all true statements:

- ☐ Afterwards there will be no migrated rows (at least until new updates are done).
- ☐ It is possible that the rows will be stored afterwards in fewer blocks, because no holes will remain from deleted rows.
- ☐ The ROWIDs of the rows might change during this step.

Of course, you should assume relatively normal cases, e.g. not `PCTFREE=99` or rows which are longer than block.

b) What would be a good `INITIAL` size and `PCTFREE` for the table `RENTED`? It is declared as follows:

```
CREATE TABLE RENTED(CUST_NO NUMBER(6) REFRENCES CUSTOMER,
                    VID_NO NUMBER(4) REFERENCES VIDEO,
                    START_DATE DATE,
                    RETURNED_DATE DATE NULL,
                    PRIMARY KEY(CUST_NO, VID_NO, START_DATE))
```

The `CUST_NO` values really contain 6 digits (they start with `100001`), and the `VID_NO` values contain 4 digits (starting with `1001`). `DATE` values need 7 data bytes. Note that when rows are inserted, `RETURNED_DATE` will be null. In Oracle, if the last column is null, no length byte is needed for this column. The table should be designed for 100 000 rows. It will initially be empty and grow through insertions (whenever a customer takes a video out of the shop) and updates (`RETURNED_DATE` is set when he/she brings the video back). `DB_BLOCK_SIZE` is 2048 Byte. Please show the main calculations.

Row length when `RETURNED_DATE` is null (without row directory entry): ________

Row length when `RETURNED_DATE` is later set (without row directory entry): ________

`PCTFREE` ______________

`INITIAL` (don’t forget the row directory here) ______________

c) Suppose there are only insertions into a table (e.g. `RENTED`). No rows are ever deleted. In addition, all rows have the same length (which is not very big, less than 10% of the block size). Does the value for `PCTUSED` matter?

- ☐ Yes, it must be chosen high (e.g. 80%).
- ☐ Yes, it must be chosen low (e.g. 20%).
- ☐ Under these circumstances, it does not matter. `PCTUSED` is only important for insertions when rows have different lengths and for deletions.

Exercise 5 (Data Dictionary) 6 Points

Please write three SQL queries involving the Oracle data dictionary. You can answer them using the following tables (but you can also use other tables from the Oracle data dictionary):

- `TABS` with the following columns:

```
TABLE_NAME, TABLESPACE_NAME, CLUSTER_NAME, IOT_NAME,
PCT_FREE, PCT_USED, INI_TRANS, MAX_TRANS, INITIAL_EXTENT,
NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE,
FREELISTS, FREELIST_GROUPS, LOGGING, BACKED_UP, NUM_ROWS,
BLOCKS, EMPTY_BLOCKS, AVG_SPACE, CHAIN_CNT, AVG_ROW_LEN,
AVG_SPACE_FREELIST_BLOCKS, NUM_FREELIST_BLOCKS, DEGREE,
INSTANCES, CACHE, TABLE_LOCK, SAMPLE_SIZE, LAST_ANALYZED,
PARTIONED, IOT_TYPE, TEMPORARY, NESTED, BUFFER_POOL.
```

- `IND` with the following columns:

```
INDEX_NAME, INDEX_TYPE, TABLE_OWNER, TABLE_NAME, TABLE_TYPE,
UNIQUENESS, TABLESPACE_NAME, INI_TRANS, MAX_TRANS,
INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS,
PCT_INCREASE, PCT_THRESHOLD, INCLUDE_COLUMN, FREELISTS,
FREELIST_GROUPS, PCT_FREE, LOGGING, BLEVEL, LEAF_BLOCKS,
DISTINCT_KEYS, AVG_LEAF_BLOCKS_PER_KEY,
AVG_DATA_BLOCKS_PER_KEY, CLUSTERING_FACTOR, STATUS, NUM_ROWS,
SAMPLE_SIZE, LAST_ANALYZED, DEGREE, INSTANCES, PARTITIONED,
TEMPORARY, GENERATED, BUFFER_POOL.
```

- `USER_IND_COLUMNS` with the following columns:

  `INDEX_NAME, TABLE_NAME, COLUMN_NAME, COLUMN_POSITION,`
  `COLUMN_LENGTH.`

  This table contains one row for every column in an index.

- `USER_SEGMENTS` with the following columns:

  `SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE, TABLESPACE_NAME,`
  `BYTES, BLOCKS, EXTENTS, INITIAL_EXTENT, NEXT_EXTENT,`
  `MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE, FREELISTS,`
  `FREELIST_GROUPS, BUFFER_POOL.`

  `SEGMENT_TYPE` can e.g. be `'CLUSTER'`, `'INDEX'`, `'TABLE'`.

a) Write an SQL query which lists tables with more than 2% of chained or migrated rows. (The data dictionary counts migrated rows as chained rows, it makes no distinction between the two.)

b) Write an SQL query which lists for each table all columns on which a single-column index exists. The output should have the columns `TABLE_NAME`, `COLUMN_NAME`, and `INDEX_NAME`, and be sorted by `TABLE_NAME`. Only single-column indexes should be contained in the output.

c) Write an SQL query which lists tables such that the indexes on the table need together more disk space than the table itself.

Exercise 6 (Query Optimization) 1+1+1+3=6 Points

a) Consider the following query:

```
SELECT *
FROM   CUSTOMER
WHERE  CUST_NO > 200000
AND    NAME = 'Smith'
AND    ADDRESS LIKE '%Pittsburgh%'
```

Which of the following access paths would the rule-based optimizer choose. Here, only one answer is correct (the one ranked highest among the possible ones):

- ☐ Full Table Scan
- ☐ Unique Index on `CUSTOMER(CUST_NO)`
- ☐ Index on `CUSTOMER(NAME)`
- ☐ Index on `CUSTOMER(ADDRESS)`

Of course, you can assume that these three indexes really exist.

b) Consider this query:

```
SELECT C.NAME, V.TITLE
FROM   CUSTOMER C, RENTED R, VIDEO V
WHERE  C.CUST_NO = R.CUST_NO
AND    R.VID_NO = V.VID_NO
AND    R.START_DATE = '20-OCT-99'
```

Suppose that the following access paths exist:

- Unique index on `CUSTOMER(CUST_NO)`
- Unique index on `RENTED(CUST_NO, VID_NO, START_DATE)`
- Unique index on `VIDEO(VID_NO)`
- Index on `RENTED(START_DATE)`
- `RENTED` and `VIDEO` are stored together in an index cluster, clustered by `VID_NO`.

Consider the QEP constructed by the rule-based optimizer which starts by accessing `RENTED`. Which table would the rule-based optimizer join first with `RENTED`?

- ☐ `CUSTOMER`
- ☐ `VIDEO`

c) Consider this query:

```
SELECT R.VID_NO
FROM   RENTED R
WHERE  R.CUST_NO = 123456
AND    R.START_DATE > '20-OCT-99'
```

Suppose that only the index enforcing the key constraint exists, i.e. a unique index on `CUST_NO`, `VID_NO`, `START_DATE`. How would a good QEP look like?

- ☐ Only a Full Table Scan is possible.
- ☐ Index scan evaluating `R.CUST_NO = 123456` and then a table access by ROWID.
- ☐ This query can be answered entirely out of the index. However, only `R.CUST_NO = 123456` would be used for accessing index entries. The table `RENTED` itself will not be accessed.

d) Consider the following query:

```
SELECT R.VID_NO, START_DATE
FROM   CUSTOMER C, RENTED R
WHERE  C.PHONE = '624-9404'
AND    R.CUST_NO = C.CUST_NO
AND    RETURNED_DATE IS NULL
```

The following indexes exist:

- Unique index `I_CUSTOMER_KEY` on `CUSTOMER(CUST_NO)`
- Index `I_CUSTOMER_PHONE` on `CUSTOMER(PHONE)`
- Unique index `I_RENTED_KEY` on `RENTED(CUST_NO, VID_NO, START_DATE)`

Please draw the Oracle QEP which the rule-based optimizer constructs starting with access to `CUSTOMER`. You can use the tree notation with interconnected boxes or use one line for every operation with indentation to clarify the structure. You do not have to assign numbers to every node.