Number on the (numbered) answer sheet:

M2DK, exam: datawarehouses.

07/11/2017

**** Answer exercices 1, 2, 3, 4 directly on these exam sheet ****

Bareme is indicative only and remains subject to change.

Exercise 1 (Short questions) grade can go negative **(1.5+1+1+1.5+1+1 = 7 pts)**

1. Answer what type of index is (best) used in each of the following cases (no justification, but +.5 /-.75 per correct/wrong answer):
   (a) to speedup range queries over a numeric attribute (arbitrary; e.g., sales.amount): .....
   (b) to maintain a primary key: .....
   (c) to speedup queries on a low-cardinality (and scarcely updated) attribute: .....

2. Answer one — and only one; the one you like better — of the following two questions:
   (a) Discuss briefly the chief difference between a datawarehouse and mediator-wrapper architecture. Explain which architecture you would adopt for a mashup application combining a few services, like an application combining rss newsfeeds with Google Map and wikipedia (trying to add localization/description information to a few newsfeed)?
   (b) What kind of metadata are typically recorded in datawarehouses regarding the ETL processes?

   .....
   .....
   .....
   .....

3. Assume that table `t(date_arrival : date, date_departure : date, category: int)` in an Oracle database is partitioned on `date_arrival`. Could the partitioning be a reason why a statement like `UPDATE t SET date_arrival= date_arrival+1` would fail? Explain in detail why (or why not).

   .....
   .....
   .....
   .....

4. +.25 per correct answer (no justification), but total is 0 if 2 wrong answers, $-.75$ if ≥ 3 wrong answers

	True	False
Reads are more frequent than writes in OLAP	☐	☐
Dimension tables are generally normalized in star schemas	☐	☐
DRAM is faster than SRAM because it uses fewer transistors	☐	☐
Operation `'TRUNCATE PARTITION t21'` is generally slower than `'DELETE FROM t21'`	☐	☐
Bloom filters are generally used to record dictionaries (key-value pairs seachable by key)	☐	☐
To speedup insertions, *bulk loading* tools generally bypass integrity constraints	☐	☐

5. +.25 per correct answer (no justification), but total is -.25 if ≥ 2 wrong answer...

*Assume table t is **hash**-partitioned into 8 partitions, on some integer attribute **a***:	True	False
The risk of skew is higher when a is unique than when it has few possible values	☐	☐
The optimizer may perform partition pruning for queries like `'SELECT ... WHERE a < 100'`	☐	☐
(idem) for queries like `'SELECT ... WHERE a = 100'`	☐	☐
(idem) for queries like `'SELECT ... WHERE a < 100 AND a > 50'`	☐	☐

6. Give the assets and drawbacks of column storage with respect to row storage

   .....
   .....
   .....
   .....

estates

id_estate	id_program	surface	price
101	1	30,04	180,000
201	1	30,04	185,000
391	2	40,04	145,000
...	...	...	...

Primary Key: id_estate
Foreign Key: estates.id_program→ program.id

program

id	nature	launch_year	nb_housing	manager
1	family housing	2016	14	Arnold
2	student housing	2010	24	Hamilton
3	family housing	2015	16	Arnold
4	family housing	2016	15	Arnold
5	family housing	2014	20	Gates
6	student housing	2012	21	Hamilton
...	...	...	...	...

Primary Key: id

Figure 1: Database on housing projects

Exercise 2 (Understanding and fixing SQL queries) **(1+1+2=4pts)**

For each query below

- If the query is correct and well written, provide its result on the instance in Figure 1 (assuming in that case that the tables contain no tuple but those displayed)
- If the query is correct but poorly written, (can be simplified), simplify it. (the alternative query must of course return the same result)
- if the query is incorrect, point out all the mistakes that prevent its proper execution

The queries are on the tables from Figure 1: you must take into account keys and foreign keys (if relevant).

0.
```sql
-- (Illustrative example)
SELECT a, b, SUM(c) FROM t t1, t t2 GROUP BY a;
-- Answer:  incorrect because b is not a 'GROUP BY' expression, and column 'a'
-- in SELECT and GROUP BY is ambiguous: it may come from t1 or t2

SELECT  COUNT(*), SUM(surface) FROM estates ;
-- Answer: 3 | 100,12
```

1.
```sql
SELECT id_estate, id_program,
  SUM(SUM(price)) OVER(PARTITION BY id_estate, id_program)
FROM estates
GROUP BY id_estate, id_program
ORDER BY id_estate, id_program;
```

2.
```sql
SELECT nature, launch_year, SUM(SUM(nb_housing))
  OVER(PARTITION BY nature ORDER BY launch_year DESC)
FROM program
GROUP BY nature, launch_year
ORDER BY nature, launch_year;
```

3.
```sql
SELECT REGEXP_REPLACE(nature,
'^([[:alpha:]]*[[:space:]])(.)(.*)',
 '\1.\2'), launch_year, COUNT(*)
FROM program
GROUP BY nature, ROLLUP(launch_year)
ORDER BY nature, launch_year;
```

(this query is correct, don't try to simplify)

Number on the (numbered) answer sheet:

*** Answer exercices 1, 2, 3, 4 directly on these exam sheets ***

Exercise 3 (SQL queries (again), modelization, indexing) (1.5+2+1.5=5pts)

1. Modify the query below so that the result displayed satisfies the following order: all the lines corresponding to a total over the same kind of group are displayed together[1]. You will observe that this is a partial order (you may choose to put the "GROUP BY b" results before or after "GROUP BY a" results, as you prefer, and there is no order among the "GROUP BY a" results)

```
SELECT a,b,c, SUM(d) S
FROM t
GROUP BY CUBE(a,b,c)
```

.....
.....
.....
.....

```
What we want to avoid:
A        B     S
-------- ----- -----
Q1 2010  mar10   400
Q1 2010  jan10   800
Q1 2010         1200
Q4 2010  dec10   253
Q4 2010          253
...
```

```
What we want to see:
A        B     S
-------- ----- -----
Q1 2010  mar10   400
Q1 2010  jan10   800
Q4 2010  dec10   253
Q1 2010         1200
Q4 2010          253
...
```

2. Suppose the manager of a program (on Figure 1) may change (possibly several times) over time. We want to track these variations.

   Assume that

   - on 20/01/2017 the manager of program 1 is fired from program 1 and replaced with Cornwallis (Arnold remains the manager of programs 3&4).
   - on 22/01/2017 (we don't care about the exact date, but after 20/01), a new estate is added to the program 1. The `id_estate` of this new estate is 403, its `surface` is 45,02, its `price` is 150,000.

   Show how you would modify the database tables to represent these informations: your answer will be the database instance on 30/01/2017 (write directly over Fig 1 what should be modified) .....

3. How would a bitmap index on `program.manager` look like? How would a bitmap join index of `program.manager` over `estates` look like? (just give the bitvectors, I don't care whether horizontally or vertically)

.....
.....
.....

Exercise 4 (Bloom filters) (1+1+1=3pts)

We wish to build a Bloom filter on estate prices. This filter will use $m = 10$ bits for the array A, and the $k = 3$ hash functions h_1, h_2 and h_3 below:

h_1:		h_2:		h_3:	
	180,000 → 0		180,000 → 1		180,000 → 1
	185,000 → 5		185,000 → 5		185,000 → 1
	145,000 → 2		145,000 → 3		145,000 → 8
	120,000 → 1		120,000 → 4		120,000 → 5
	160,000 → 8		160,000 → 2		160,000 → 3

1. show how array A would look like if we register in the filter the set S of cities from the table, i.e, $S = \{180{,}000; 185{,}000; 145{,}000\}$?

.....
.....
.....
.....

2. What will the filter answer (explain briefly why, i.e., what makes you say the answer is that) if we use the filter to test whether 120,000 belongs to the set S? Same for 180,000? and 160,000?

.....
.....
.....
.....

[1] We will consider there may be NULL in the data, and will *not* accept an answer relying on CASE ... WHEN ... IS NULL

3. Designing good hash functions is sometimes tricky. Here are a 3 propositions (% is the modulo operator = remainder of euclidean division). Discuss which you consider best *and explain precisely why*:

1: $m = 10000, k = 3$, $h_1 : x \mapsto x\%1000$, $h_2 : x \mapsto (2 * x)\%1000$, $h_3 : x \mapsto (3 * x)\%1000$
2: $m = 1000, k = 3$, $h_1 : x \mapsto x\%1000$, $h_2 : x \mapsto (2 * x)\%1000$, $h_3 : x \mapsto (3 * x)\%1000$
3: $m = 100, k = 3$, $h_1 : x \mapsto x\%97$, $h_2 : x \mapsto (h_1(x) + x\%89)\%100$, $h_3 : x \mapsto (h_1(x) + 2 * (x\%89))\%100$

.....

.....

.....

.....

Exercise 5 (Column store) **(3pts)**

We consider a table `Sales (employee, amount)`[2] in column-major storage according to the Sanssouci(≃SAP Hana ≃ Delta-with-main) architecture discussed in the lectures. Suppose the differential buffer is merged into the main while the database is in the state illustrated below. How will the database look like after the merge operation if operations `INSERT INTO Sales VALUES ('Drake',3.3 )`, `INSERT INTO Sales VALUES ('de Monteil', 4.3 )` are performed while the merge operation is running? BONUS if you also consider the operation `DELETE FROM Sales WHERE amount=12.5` in addition to (say, after) the 2 inserts above.

Main Store for table `Sales`

Dictionary

IDValue	employee
0	de Grasse
1	Hood
2	Rochambeau

Attribute vector (employee)

IDLine	IDValue
0	0
1	0
2	2
3	1
4	0
5	1

Attr. vect. (sales)

IDLine	Value
0	10.5
1	4.5
2	11.5
3	8
4	12.5
5	1.5

Validity vector

IDLine	valid
0	0
1	1
2	0
3	1
4	1
5	1

Differential Buffer

Dictionary

IDValue	employee
0	de Grasse
1	Graves
2	Barras

Attribute vector (employee)

IDLine	IDValue
0	0
1	1
2	2
3	0

Attr. vect. (sales)

IDLine	Value
0	8.75
1	3.75
2	12.75
3	11.75

Validity vector

IDLine	valid
0	1
1	1
2	1
3	1

*** Return this exam sheet together with the other answer sheet and your draft ***
(*I* will throw away the draft without evaluating it)

[2]The `amount` column is not dictionary encoded but stored 'as is'.