School of Computer Science, McGill University

COMP-421 Database Systems, Winter 2019

Written Assignment 2: SQL and Indexing

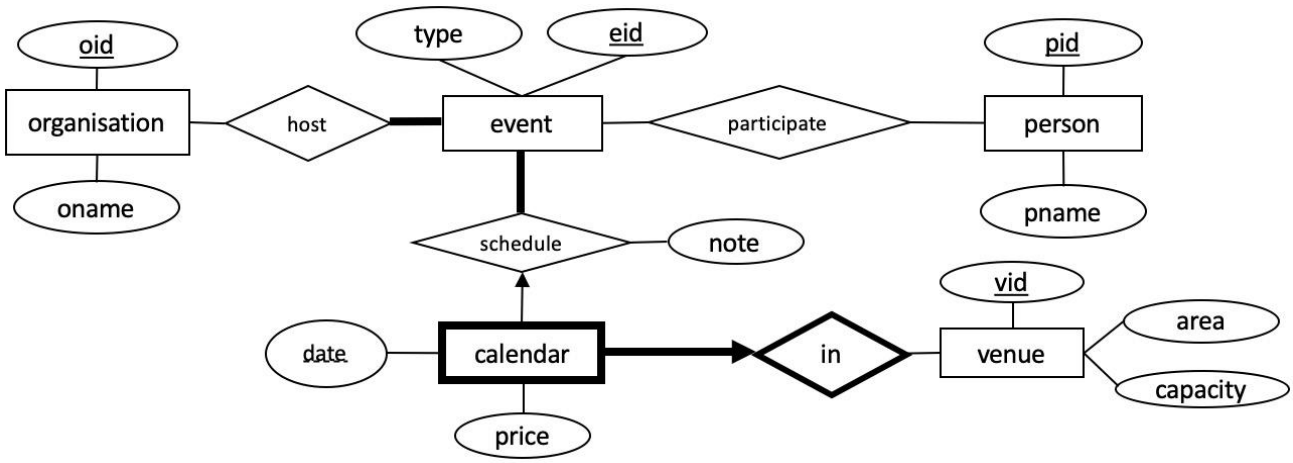
Due Date March 09, 11:59pm

This is an individual assignment. You are required to work on your own to create the solution.
There is no late submission or extension allowed for this assignment as we need to post the solutions so that you can be prepared for the midterm.

Ex1 will be graded by an auotmated system. It is very important to read the instructions and follow them exactly. **Ex2** will be graded by the TAs.

Turn in two attachments. One for **Ex1** the tar file for automated system and another attachment for **Ex2** for the TAs to grade. For the later you may turn in a doc/pdf/txt format.

Ex. 1 — SQL (70 Points)
The following ER model and corresponding relational model are designed for an event management system. There are various event venues in different areas (A, B and C). Each venue has a unique id number, a capacity of how many people the venue can hold, and a calendar to show when the events are scheduled in this venue and the corresponding price to rent the venue. A venue can only be scheduled for one event on any given day, but an event can span multiple days. If needed, a schedule can be associated with a note. The system also records the organizations that host events. An event can be hosted by one organization or multiple organizations. Events can be one of many types (such as *music*, *fundraising*, etc.). The system also keep track of people involved with the events.
A sample schema and few records have been provided as **setup.sh** Please add more records into this as you may need to test various scenarios. The automated system will use slightly different data set. So ensure your logic works even for different data sets. The table definitions used by the automated system will be the same as the ones in the sample scripts given to you. Use your individual database accounts to work on the assignment. **DO NOT** use your project groups database account as it is shared between all of your team members.



person(pid, pname)
organization(oid, oname)
venue (vid, area, capacity)
calendar(vid, date, price)
vid is the foreign key to venue
event(eid, type)
schedule(vid, date, eid, note)
eid is the foreign key to event
(vid, date) is the foreign key to calendar
participate(pid, eid)
pid is the foreign key to person
eid is the foreign key to event
host(oid, eid)
oid is the foreign key to organization
eid is the foreign key to event

Important !!

All the sql solutions will be evaluated by an automated system, which compares the output data produced by executing your query on our dataset with the expected output result for the correct query. So it is important that you include the correct column names, in the correct order, perform any ordering on output tuples as asked etc.

Double check your SQL for **typos**, for example if you spelt '*Joane*' instead of '*Joanne*', a query might not return the correct records and you will not get any points.
While the columnn and table names are not case sensitive, **the data itself can be case sensitive**. So do not write '*JOANNE*', where it was required to write '*Joanne*' this can produce no results or wrong results.
For more details read the attached sql formatting guide. If you have questions about this post it in the discussion forum for assignment 2. **Remember you will either get 0 or all points for a given SQL question !!**

For this assignment you MUST NOT create views or tables in your solution. You can however use derived tables as we saw in class in your SQL. You may also use the SQL `WITH` clause if you wish to (not covered in lectures and not essential to writing solutions). Do not use any explicit SQL `CAST` operations to perform data conversions of your own. All your answers should be comprised of only a select query. **Output ONLY the attributes in the question, following the exact order mentioned in the question**. Adding attributes not mentioned can result in a 0 score !

Unless specified, your output query should not produce duplicate results in your output resultset. Use the technique taught in class to eliminate duplicate records from the output.
Where an output ordering is asked for, remember to order the output records. The technique for this was also shown in class.

1. (2 Pts) List all the venue ids and capacities of the venues in area `B` and `C` with a capacity greater than `100`. Order the output by the descending order of capacity and ascending order of vid.
2. (2 Pts) List all the venue ids and their capacities for the venues in area `C` whose price is less than `100.00` on `2020-01-16` - using joins. Order the output by venue id.
3. (2 Pts) List all the venue ids and their capacities for the venues in area `C` whose price is less than `100.00` on `2020-01-16` - using a subquery. Order the output by venue id.
4. (2 Pts) List all the venue ids and their capacities for the venues in area `C` whose price is less than `100.00` on `2020-01-16` - using a correlated subquery. Order the output by venue id.
5. (3 Pts) List the pids and names of all the people who participate in the event with eid `5`. Order the output by pid.
6. (3 Pts) List the organization ids and names for all the organizations that host events on `2020-01-16`. Order the output by oid.
7. (2 Pts) Find the vid of the venues in area `A` that are available (not scheduled for an event) on `2020-01-17`. List the vids in ascending order.
8. (3 Pts) Find the eid and date for `fundraising` type of events that BOTH the people with pids `12345678` and

`12345679` participated in. List the event ids and event dates. Output multiple records for the events that span multiple days. Order the output by the descending order of eid and ascending order of date.

9. (3 Pts) Find the eid and date for `music` type of events that the person with pid `12345678` participated in but the person with pid `12345679` did not. List the event ids and event dates. Output multiple records for the events that span multiple days. Order the output by the descending order of eid and ascending order of date.
10. (4 Pts) List the venue ids, areas, capacities of all the venues that the organization with oid `6` has ever used to host events. - using a correlated subquery. Order the output by vid and capacity.
11. (2 Pts) List the pids and names of all the other people that participated in the same event as that of the person with pid `12345678`. Order the output by pid.
12. (3 Pts) Find the total number of organizations. Give the output column the name `numorganizations`.
13. (3 Pts) Find the number of distinct individuals who have ever participated in the events hosted by the organization with oid `1`. Name the output column `numpeople`.
14. (4 Pts) List the event ids and the total amount of money that each of them spent on renting the venues. Name the latter `totalamount`. Order the output by the descending order of the eid.
15. (5 Pts) List the organization ids and names of all the organizations that have hosted an event that spans more than 2 days. You can assume that all the events are scheduled on consecutive days. Order the output by oid.
16. (6 Pts) Among the fundraising events, list the event id and the total attendees of the event with the largest participation. Order the output by the event id. Name the second column `numpeople`.
17. (5 Pts) List all the venue ids and the total number of events ever scheduled in the venue. Name the second column `numevents`. If a venue has not been scheduled to host any event, it should show `0` for `numevents`. Write this query without using any outer joins. Order the output by vid.
18. (6 Pts) Redo the above question with an outer join. **Hint:-** use COALESCE.
19. (4 Pts) List the organization ids of the organizations that have used all the venues in area `C` for holding events. Order the output by oid.
20. (6 Pts) Find the average venue cost per event for each type of event. List the types and the average amounts. Order the output by type of the event. Name the second column `averageamount`.

Ex. 2 — Indexing (30 Points)

Consider the `schedule` table from the previous question.

```
schedule(vid INTEGER, date DATE, eid INTEGER, note VARCHAR(50))
```

Here is some additional information.

- An INTEGER has 64 bits, DATE has 4 bytes and the average size of note is 32 bytes.
- There are 3,000 venues and 1,350,000 events.
- The system keeps records for 1,000 days.
- The average vacancy rate for venues is 10%.
- Each rid has 10 bytes, each pointer (of internal index pages) has 6 bytes.
- Leaf pages are filled on average of 60% and page size is 4000 bytes.
- Intermediate pages can have a fill factor between 50 - 100%. The root might have any fill factor to accomodate as many child nodes as it can.

Now assume there exists an indirect, clustered type II B+-tree index on `eid` and an unclustered type II B+-tree indirect index on `(vid, date)`. A single data entry may NOT spread over more than one leaf page.

1. For both indices calculate:
   (a) (10 Points) *The avg. number of rids per data entry, the size of the data entry and the total number of data entries and the number of leaves.*
   (b) (5 Points) *The maximum and minimum possible number of intermediate nodes in the index (for the given possible fill factor range of 50-100%) and the height of the tree in each case.*