INFO20003 Semester 1, 2024

Assignment 2: SQL
Due: 6:00pm Friday, 26th April 2024
Weighting: 10% of your total assessment

Conference Track-r

Description

Conference Track-r is a platform you're creating to help keep track of academic conferences and paper submissions.

An academic conference is an event in which researchers present their findings and hear about the latest work within their research area. Researchers can submit their research papers to conferences, where high-quality papers are accepted for presentation.

Conference event

For each conference event, the system records its details, that are – name (e.g., International Conference on Data Engineering), occurrence number (e.g., 22, which means it's the 22nd occurrence of the International Conference on Data Engineering), location (e.g., Melbourne, Australia) of the conference, the start date, and the end date of the conference. The location and dates of a conference may/will differ for each occurrence.

Each conference has at least one 'track' (e.g, the 'Conference on data engineering' might have a 'relational databases' track type). Each track is associated with the following information: the name of the track, and paper submission guideline (as a text description of maximum 500 characters). Each track is associated with exactly one conference. The tracks / guidelines of a track can be different for different occurrences of the conference.

Each track has at least one 'session'. Each session has a session name (e.g., 'Indexing'), start date and time, end date and time, and is associated with the number of accepted papers to be presented in that session.

Researchers

For each researcher, the system records the researcher's name and a unique email address. The system also stores the research supervision relations of the researchers. Each researcher can have any number of supervisors, where each supervisor is also a researcher. A researcher can supervise any number of other researchers. For each supervision relation, the system stores the start and end date of the supervision (e.g. "Farhana Choudhury and Renata Borovica-Gajic supervised Timothy Hermanto between 01/01/2020 and 31/12/2022"). If the supervision is ongoing, the end date is unpopulated.

We also record which conferences (if any) a researcher has attended.

Papers

The system maintains the information of the papers that are submitted to the conferences: unique ID, title, abstract, and the authors. There can be one or more authors of a paper, where each author is a researcher. A paper is submitted (only once) to a particular track of a conference. Each track of a conference can have any number of papers submitted to it.

Papers must be reviewed to be presented at a conference. After the reviewing process, a paper can be either rejected or accepted to that track. For each rejected paper, the reasons for rejection (as a text description of maximum 100 characters) is stored. Each accepted paper is assigned a start page number and an end page number (a note as the explanation of the page numbers - conferences publish a booklet called conference proceedings, where the start and end page numbers of each accepted paper stored in the system correspond to the page numbers in that conference proceeding). The accepted papers are assigned to sessions (at least one session), where the paper can be presented and discussed. Each session has at least one paper assigned to it. Each paper assigned to a particular session will be presented by one of the authors. A researcher can present any number of papers on the same session or different sessions. For the given paper presentation, the system captures the presenter, the paper and the session.

The Data Model

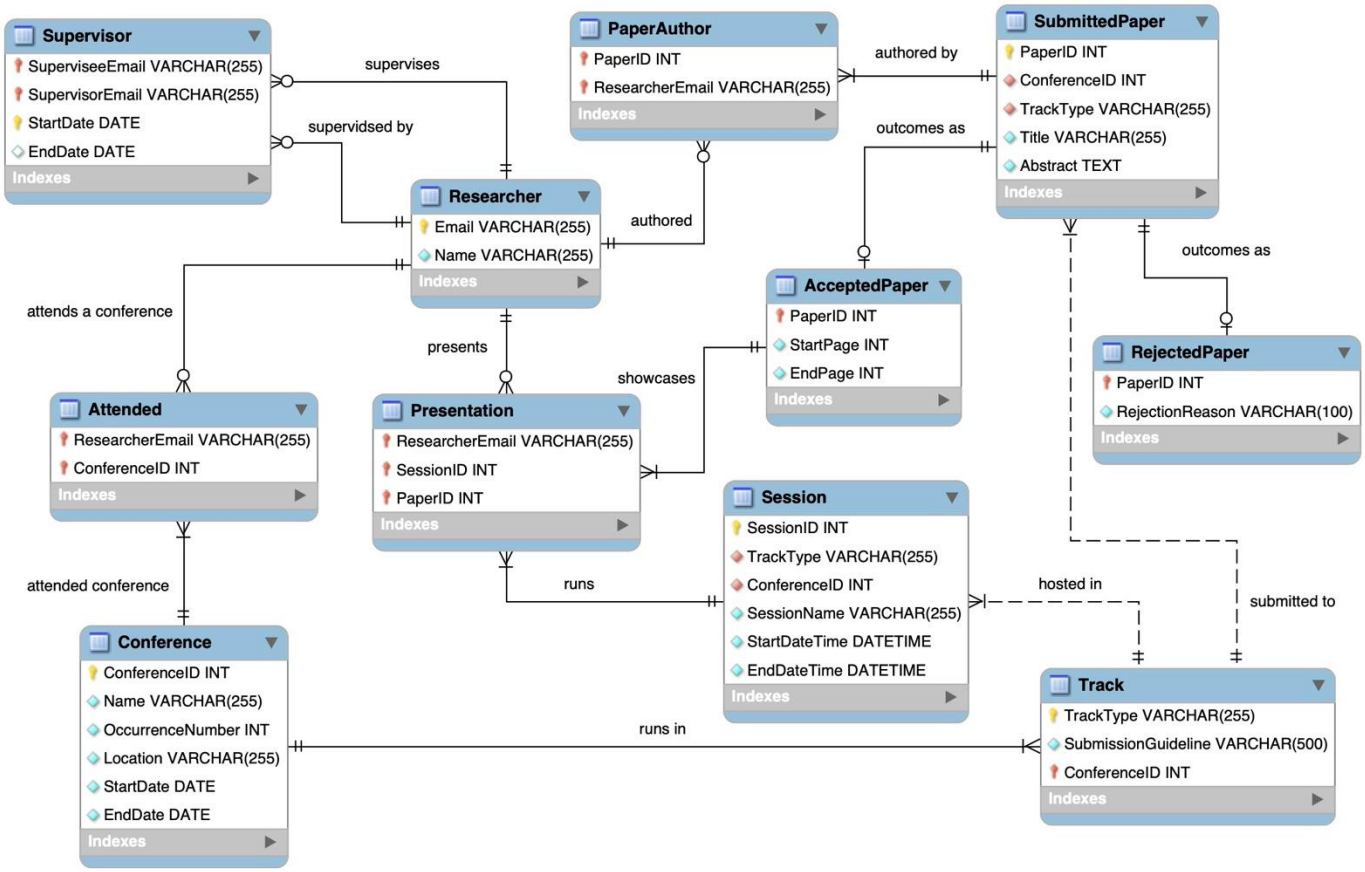


Fig 1: The physical ER model of Conference Track-r database.

Assignment 2 Setup

A dataset is provided which you can use when developing your solutions. To set up the dataset, download the file **conference_2024.sql** from the Assignment link on Canvas and run it in Workbench. This script creates the database tables and populates them with data. Note that this dataset is provided for you to ***experiment*** with, but it is ***not*** the same dataset as what your queries will be tested against (the schema will stay the same, but the data itself may be different). *This means when designing your queries you must consider edge cases even if they are not represented in this particular data set, and should not hardcode information like IDs into your queries.*

The script is designed to run against your account on the Engineering IT server (info20003db.eng.unimelb.edu.au). If you want to install the schema on your own MySQL Server installation, *uncomment the lines at the beginning of the script.*

⚠️ ***Do NOT disable only_full_group_by mode when completing this assignment****. This mode is the default, and is turned on in all default installs of MySQL workbench. You can check whether it is turned on using the command "SELECT @@sql_mode;". The command should return a string containing "ONLY_FULL_GROUP_BY" or "ANSI". When testing, our test server WILL have this mode turned on, and if your query fails due to this, you will lose marks.*

The SQL tasks

In this section are listed 10 questions for you to answer. Write one (single) SQL statement per question. Subqueries and nesting are allowed within a *single* SQL statement

⚠ In general, we care more about correctness than constructing the 'most efficient' query (computationally, or in terms of number of characters/lines). However, you may be penalized for writing *overly* complicated SQL statements (e.g the query is 2-3x longer than required, using superfluous joins, etc), using very poor formatting, using very poor alias naming, or other decisions that make it hard for us to read/understand what you're trying to do when marking!

⚠ **DO NOT USE VIEWS (or 'WITH' statements/common table expressions) to answer questions.**

1. Find the total number of attended researchers each conference had. Your query should return results of the form (ConferenceID, ResearcherAttendedCount). **(1 mark)**

2. Find the rejected paper with the shortest RejectionReason. Assume there are no ties (i.e., only one paper has the shortest rejection reason). Your query should return results of the form (PaperID, RejectionReason, LengthRejectionReason). *HINT: use the "LENGTH()" function.* **(1 mark)**

3. List all conferences that have 5 or more papers submitted across all tracks. Your query should return results of the form (ConferenceID, PapersCount). **(1 mark)**

4. We'll say that going to a 'unique' conference means attending a conference with a name which is different to every other conference you've attended before. E.g. if last year, you went to the 60th occurrence of a conference named "SIGCSE", and then this year you go to the 61st occurrence of a conference named "SIGCSE", then you have been to only one 'unique' conference. Find the researcher that has the most number of unique conferences attended. If there are ties, you must return all researchers with the equal highest count. Your query should return results of the form (ResearcherEmail, DistinctConferencesCount), with one row per researcher in case of a tie. **(2 marks)**

5. Find all researchers who attended one or more conferences which ran only on days before 01/01/2024 but have not attended any other conferences which ran on days that were on or after that date (essentially, we are looking for researchers who stopped attending conferences from 01/01/2024). Your query should return results of the form (ResearcherEmail). **(2 marks)**

6. Find researchers who have authored (at least) as many rejected papers as they have accepted papers. Your query should return results of the form (ResearcherEmail, RejectedPapersCount, AcceptedPapersCount). **(2 marks)**

7. Find how much time (in minutes) is allocated per presentation for each session. You may assume all sessions have at least one presentation, that all time in a session is used only for presentations (there are no other activities in a session), and that all presentations from a given session receive an equal amount of time. Round the minutes DOWN to the nearest whole minute. Your query should return results of the form (SessionID, MinutesPerPresentation). *HINT: use the "TIMESTAMPDIFF()" function.* **(2 marks)**

   E.g. If a session started at 1:00pm, finished at 2:00pm, and there were 3 presentations in that session, then there are 20 minutes allocated per presentation for that session.

8. Find any researchers who have presented ALL of the accepted papers that they've ever authored, at least once. Your query should return results of the form (ResearcherEmail, presentedPaperCount), where presentedPaperCount is the number of distinct papers presented. *Hint: Consider that a paper might have been presented multiple times…* **(3 marks)**

9. We'll say that researcher A has "Influenced" researcher B if A has supervised B, OR A has supervised researcher C, where C has "Influenced" B (note this is a recursive definition). The "Depth" of an influences relationship is the number of intermediate researchers between the influence and the influencer +1 (see example). You can assume that there are no "Influenced" relationships with a "Depth" greater than 3. Find all influenced-influencer pairs, and their depths. If there are multiple possible depths, you should return the smallest (see example). Your query should return results of the form (InfluencerResearcherEmail, InfluencedResearcherEmail, Depth). **(3 marks)**

   E.g.: Consider the Supervisors table with data as shown:

Supervisor	**Supervisee**
Andrew	Bob
Andrew	Cathy
Bob	Cathy
Bob	David
David	Eve
Fred	Eve

   Then the following is a list of who has been influenced by whom (depth in square brackets):

   - Andrew has influenced (Bob[1], Cathy[1], David[2], Eve[3])
     - Note the depth for Cathy is '1': even though there is an influences relationship Andrew -> Bob -> Cathy, but this has a depth of 2; the shortest path / smallest depth is used.
   - Bob has influenced (Cathy[1], David[1], Eve[2])
   - Cathy has influenced ()
   - David has influenced (Eve[1])
   - Eve has influenced ()
   - Fred has influenced (Eve[1])

10. Find all conferences where the number of researchers who attended dropped from one Occurrence to the next. Your query should return results of the form (ConferenceName, PrevOccurenceNumber, PrevAttendanceCount, NextOccurenceNumber, NextAttendanceCount) **(3 marks)**

    E.g.: Consider the Conference table (+ derived number of attendees from the Attended table) data as shown:

ConferenceID	**Name**	**Occurrence number**	**Number of attendees (derived)**
1	SIGCSE	1	100
2	SIGCSE	2	100
3	DASFAA	1	52
4	DASFAA	2	52
5	SIGCSE	3	98

    Then the query would return:

ConferenceName	**PrevOccurrence**	**PrevAttendanceCount**	**NextOccurrence**	**NextAttendanceCount**
SIGCSE	2	100	3	98

SQL Response Formatting Requirements

To help us mark your assignment queries as quickly/accurately as possible, please ensure that:

1. Your query returns the projected attributes in the same order as given in the question, and does not include additional columns.

E.g., if the question asks 'return as (userId, name)', then:

- DO: **"SELECT userId, name …"**
- ⚠️ DON'T: **"SELECT name, userId…"**

You can, however, rename/name the columns to whatever you'd like using `**AS**`, only the **order** matters.

2. Do NOT use "databaseName.tableName" format.

E.g.:

- DO: **"SELECT userId FROM users…"**
- ⚠️ DON'T: **"SELECT userId FROM coltonc.users …".**

Note that you can use tableName.columnName format, like researchers.email.

3. Ensure that you are using single quotes(') for strings

Double quotes should only be used for table names (but you shouldn't need to do this since we don't have spaces in our table names)

E.g.:

- DO: **…WHERE name = 'bob'**
- ⚠️ DON'T: **…WHERE name = "bob"…**

4. Do NOT delete the special comment markers in the SQL template file.

These include (where X is the question number):

-- BEGIN QX

-- END QX

-- END OF ASSIGNMENT

These help us mark your assignment!

5. Comments are optional, but will help tutors to understand your code!

Submission Instructions

Your submission will be in the form of an SQL script. There is a template file on the LMS, into which you will paste your solutions and fill in your student details (more information below).

This .sql file should be submitted on Canvas by **6pm** on the due date of **Friday, 26 April 2024**. Name your submission as 987654.sql, where 987654 corresponds to YOUR student id.

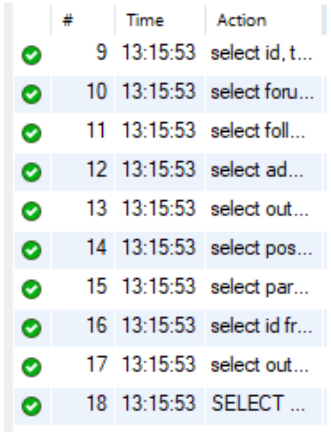
Filling in the template file:

The template file on the LMS has spaces for you to fill in your student details and your answers to the questions. There is also an example prefilled script available on the LMS as well. Below are screenshots from those two documents explaining the steps you need to take to submit your solutions:

Step	Example

1. At the top of the template, you'll need to replace "XXXXXXXX" with your student number and name	*Template*
	`-- Your Name: XXXXXXXX`
	`-- Your Student Number: XXXXXXX`
	Example Filled in
	`-- Your Name: Colton Carner`
	`-- Your Student Number: 693281`

2. For each question 1-10, place your SQL solution in between the "BEGIN QX" and "END QX" markers.
<u>Ensure each query is terminated with a semicolon ";"</u>
	Template
	`-- BEGIN Q1`
	`-- END Q1`
	Example Filled in
	`-- BEGIN Q1`
	`-- **This is an example of a comment which will not be executed`
	`-- **(comments are not NEEDED, but if your query is complex you can leave some)`
	`-- **Below is an example of how you would enter your answer:`
	`SELECT *`
	`FROM delivery`
	`NATURAL JOIN deliveryitem`
	`WHERE supplierid = 101;`
	`-- **It's OK to add more space in between the 'BEGIN QX' and 'END QX' markers`
	`-- **for each question, just don't DELETE the markers!`
	`-- **Make sure you fill out your name + student num at the top of the document`
	`-- **After reading / understanding these comments in the Q1 section`
	`-- **(comments starting with '**'), feel free to delete them`
	`-- **(don't delete lines without **)`
	`-- END Q1`

3. Test that your script is valid SQL by running it from MySQL Workbench. Run the entire script by copy-pasting this entire file into a new workbench tab, placing your cursor at the start of the file (without selecting anything), and pressing the lightning bolt to run the entire file.  All queries should run successfully one after another. If not, check to make sure you added semicolons ';' after each query.	*All 10 queries ran sequentially and were successful.* 

Late submission

Unless you have an approved extension (see below), you will be penalised -10% of the total number of marks in the assignment per day that your submission is late. For instance, if you received a 78% raw score, but submitted 2 days late, you'd receive a 58% score for the assignment.

Requesting a Submission Deadline Extension

If you need an extension due to a valid (medical) reason, you will need to provide evidence to support your request by **5pm on Thursday 25th April**. Extensions received after this time may be rejected. Medical certificates need to be at least two days in length.

To request an extension:

- Email Farhana Choudhury (farhana.choudhury@unimelb.edu.au) from your university email address, supplying your student ID, the extension length that you require and supporting evidence. Please add INFO20003 in the subject title.
- If your submission deadline extension is granted you will receive an email reply granting the new submission date. Do not lose this email!

Reminder: INFO20003 Hurdle Requirements

To pass INFO20003, you must pass two hurdles:

- **Hurdle 1:** Obtain at least 50% (15/30) for the three assignments (each worth 10%)
- **Hurdle 2:** Obtain at least 50% (35/70) for the combination of the quizzes and final exam

Therefore, it is our recommendation that you attempt every assignment and question in the exam.

GOOD LUCK!