University of Toronto
CSC343, Winter 2025

Assignment 1

Due: Wednesday 29th January by 4 pm

Learning Goals

This assignment aims to help you learn to:

- Read a new relational schema and determine whether or not a particular instance is valid with respect to that schema.
- Apply the individual techniques for writing relational algebra queries and integrity constraints that we learned in class.
- Combine techniques to solve complex problems
- Identify the limits of what can be expressed in relational algebra

These skills will leave you well prepared to be a strong SQL programmer.

Schema

For this assignment, you will write queries and integrity constraints on a database for airport information. This database will store information about the airports, airlines, and the flights they operate.

Note: Below we use the 24-hour clock format e.g., 23:00 refers to 11:00 pm and 11:00 refers to 11:00 am. All date time values are in Coordinated Universal Time (UTC) - Think about why that is important for this specific domain.

Relations and Integrity Constraints (ICs)

- City(<u>id</u>, <u>name, country</u>)

  A tuple in this relation represents a city with the name *name* in the country *country*.

- Airline(<u>code</u>, name)

  A tuple in this relation represents an airline. *name* is the name of the airline e.g., "Air Canada".

- Passenger(<u>id</u>, firstName, surName)

  A tuple in this relation represents a potential passenger. *firstName* and *surName* is their first name and last name respectively.

- Plane(<u>tailNumber</u>, model, airline)

  A tuple in this relation represents a plane with the unique identifier *tailNumber*. *model* is the plane's model e.g. "Boeing 777", and *airline* is the code of the airline that owns the aircraft.

  ICs:

  - Plane[airline] $\subseteq$ Airline[code]

- Seat(<u>id</u>, <u>plane, row, letter</u>, class)

  A tuple in this relation represents a seat in row *row* on the plane identified by *plane*. *letter* specifies the specific seat in the row, and *class* is the corresponding seating class.

  ICs:

  - Seat[plane] $\subseteq$ Plane[tailNumber]

- Plane[tailNumber] $\subseteq$ Seat[plane]
- Seat[class] $\subseteq$ { "First", "Business", "Economy" }

- Airport(<u>code</u>, <u>name, city</u>)

  A tuple in this relation represents an airport. *code* is the three-character IATA airport code e.g., "YYZ". *name* is the airport's name e.g., "Toronto Pearson International Airport" and *city* is the city where the airport is located.

  ICs:

  - Airport[city] $\subseteq$ City[id]

- Route(<u>flightNumber</u>, airline, source, destination)

  A tuple in this relation represents a route operated by the airline identified by *airline* from the source airport *source* to the destination airport *destination*. *flightNumber* is the alphanumeric flight designator e.g. "WS579".

  ICs:

  - Route[airline] $\subseteq$ Airline[code]
  - Airline[code] $\subseteq$ Route[airline]
  - Route[source] $\subseteq$ Airport[code]
  - Route[destination] $\subseteq$ Airport[code]
  - $\sigma_{\text{source=destination}}\text{Route} = \phi$

- Flight(<u>id</u>, route, <u>plane, schedDeparture</u>, schedArrival)

  A tuple in this relation represents a scheduled flight, which would follow the route identified by *route*, and would use the plane *plane*. The flight is scheduled to leave at *schedDeparture* and arrive at the destination at *schedArrival*. Both *schedDeparture* and *schedArrival* are date-time attributes.

  ICs:

  - Flight[route] $\subseteq$ Route[flightNumber]
  - Flight[plane] $\subseteq$ Plane[tailNumber]
  - $\sigma_{\text{schedDeparture}\geq\text{schedArrival}}\ \text{Flight} = \phi$

- Departure(<u>flight</u>, dateTime)

  A tuple in this relation represents the actual departure time of the flight identified by *flight* from the source airport.

  ICs:

  - Departure[flight] $\subseteq$ Flight[id]

- Arrival(<u>flight</u>, dateTime)

  A tuple in this relation represents the actual arrival time of the flight identified by *flight* at its destination airport.

  ICs:

  - Arrival[flight] $\subseteq$ Departure[flight]

- FlightPrice(<u>flight, class</u>, price)

  A tuple in this relation represents the fact that the seating class *class* on flight *flight* has the price *price*.

  ICs:

  - FlightPrice[flight] $\subseteq$ Flight[id]
  - Each class on a flight has a price recorded in FlightPrice. FlightPrice doesn't include information for classes that are not available on the flight.

- Booking(<u>id</u>, passenger, <u>seat, flight</u>, price, dateTime)

  A tuple in this relation represents a booking, made by the passenger, identified by *passenger* for the seat *seat* on the flight *flight*. *price* is the price they paid, which could be different from the price recorded in FlightPrice, since prices change and/or they may have used a special discount. *dateTime* is the time when they made the booking.

  ICs:

  - Booking[passenger] $\subseteq$ Passenger[id]
  - Booking[seat] $\subseteq$ Seat[id]
  - Booking[flight] $\subseteq$ Flight[id]
  - $[\Pi_{\text{seat, flight}}\text{Booking}] - \left[\rho_{(\text{seat, flight})}\left(\Pi_{\text{Seat.id, Flight.id}}\left(\sigma_{\text{Flight.plane=Seat.plane}}\left(\text{Flight} \times \text{Seat}\right)\right)\right)\right] = \phi$

Warmup: Getting to know the schema

To get familiar with the schema, ask yourself questions like these (but don't hand in your answers):

1. Can a passenger book two seats on the same flight?
2. Identify the constraint that prevents a seat from being double booked on the same flight.
3. What changes should we make if we want each seat on the flight to have a different price?
4. Does the schema allow an airport to have no flights start or end there?
5. Does the schema allow an airline to operate no flights?
6. According to our schema, can a city name exist in two different countries?
7. There are multiple date-time information in the schema. Which ones are enforced to occur in a realistic chronological order? which ones aren't?
8. Does our schema make a distinction between round-trip flights and one-way flights?
9. How would a flight with a layover be represented? A layover refers to a stop or break in a journey, typically during air travel, where a traveler pauses at an intermediate airport before continuing to their final destination.
10. Does our schema allow a plane from one airline to service a route operated by another airline?

Part 1: Create an instance (8%)

Create an instance of the database defined by the schema above that represents the following given scenario. All provided times are in UTC.

Sadia booked a flight (AC890) with "Air Canada" that goes from "Toronto Pearson International Airport" (YYZ) in Mississauga, Canada to "Leonardo da Vinci International Airport" (FCO) in Rome, Italy. Her flight leaves on May 1, 2025 at 23:45 and arrives in Rome on May 2, 2025 at 8:15. Since she made her booking early on Dec 6, 2024 at 21:00, she was able to secure a good price of $600.00. The flight departed 5 mins after the scheduled departure time and arrived 10 mins after the scheduled arrival time.

She booked her return flight while in Italy on May 9, 2025 at 9:00. She found a flight that leaves "Leonardo da Vinci International Airport" May 22, 2025 at 8:00 and arrives at "Frankfurt Airport" (FRA) in Frankfurt, Germany, at 10:00 on the same day. This flight (LH231) is operated by Lufthansa and cost $950.00. She will then leave "Frankfurt Airport" at 11:55 on flight LH470, also operated by Lufthansa, and is expected to arrive in "Toronto Pearson International Airport" on 20:20 of the same day. This second flight cost $1200.00.

The instance you create must match this description of the scenario. Where the data is unspecified, you can choose your own values. Make sure your instance will give a non-empty result in Part 2.

You must make sure all of the constraints defined above in the schema are satisfied. Do not add any additional tuples that are not needed to represent the above.

Part 2: Give the output of a query (2%)

What is the result of the following query on your instance defined in Part 1?

$$\text{Temp(passenger)} := \Pi_{\text{passenger}} \left\{ \sigma_{\substack{\text{B.seat=S.id} \\ \wedge \\ \text{S.class=“First”}}} [(\rho_{\text{B}}\text{Booking}) \times (\rho_{\text{S}}\text{Seat})] \right\}$$

$$\Pi_{\substack{\text{id, passenger,} \\ \text{seat, flight}}} \{\text{Booking} \bowtie [(\Pi_{\text{passenger}}\text{Booking}) - \text{Temp}]\}$$

Part 3: Violating a constraint (8%)

Give a non-empty instance of the database that violates the following constraint:

$$\sigma_{\substack{\text{B1.flight}=\text{B2.flight} \\ \wedge \\ \text{B1.passenger}=\text{B2.passenger} \\ \wedge \\ \text{B1.seat}\neq\text{B2.seat}}} [(\rho_{\text{B1}}\text{Booking}) \times (\rho_{\text{B2}}\text{Booking})] = \emptyset$$

All other constraints defined on the schema for the relations involved must be satisfied.

Part 4: Queries (50%)

Write the queries below in relational algebra. There are a number of variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in this course. You may use assignment, and the operators we have used in class: $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$. Assume that all relations are sets (not bags), as we have done in class, and do not use any of the extended relational algebra operations from Chapter 5 (for example, do not use the division operator).

Some additional points to keep in mind:

- Do not make any assumptions about the data that are not enforced by the original constraints given above, including the ones written in English. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute. For those of you who know some SQL, in other words, there are no null values.
- Remember that the condition on a select operation may only examine the values of the attributes in one tuple, not whole columns. In other words, to use a value (other than a literal value such as 100 or "YYZ"), you must get that value into the tuples that your select will examine.
- The condition on a select operation can use comparison operators (such as $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). Simple arithmetic is also okay, *e.g.*, attribute1 $\leq$ attribute2 + 5000.
- Some relations in our schema have date-time attribute(s).
  - You may use comparison operators on such values. You may refer to the components of a date-time using a dot notation. For example, you can refer to the year component of a date-time attribute d using the notation d.year or hour using d.hour.
  - You may assume that all date-time attributes follow the UTC time zone.
  - You can take the difference between 2 date-time attributes. You may assume that the resulting value is an interval that includes one or more values in the form "⟨*number*⟩ type", where type is one of "years", "months", "days", "hours", "minutes", "seconds". For example, 2024-09-15 23:00 - 2024-09-10 23:00 is "5 days".
  - You can compare two intervals e.g., "5 days 5 minutes" > "5 days 5 minutes 1 seconds" would evaluate to False.
  - You may use *now* to specify the current date-time. You can therefore find the current day, month and year using *now.day*, *now.month*, and *now.year* respectively.
- You are encouraged to use assignment to define intermediate results.
- It's a good idea to add commentary explaining what you're doing. This way, even if your final answer is not completely correct, you may receive part marks. Note that consistently omitting comments would result in a mark deduction.
- The order of the columns in the result doesn't matter.
- If there are ties, report all of them.

At least one of the queries and/or integrity constraints in this assignment cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed".
Note: The queries are not in order according to difficulty.

1. A flight is considered to "fly through" both its source and destination airports. Find airports that have the largest number of flights flying through them. Report the airport code and the number of flights through it.
2. Find airlines that had three or more delayed flights in 2024. We will define a delayed flight as one with an actual departure date recorded, and its actual departure is at least one hour after its scheduled departure. Only consider flights that have been scheduled to depart in 2024 (their actual departure might have been in 2025 though). Report the airline code, the airline name, and the flight number of the most recently delayed flight (as determined by the flight's scheduled departure). If there are ties, report them all.
3. Find cities that have at least one airport, but there are no routes ending at that city. Report the name and country of the cities.
4. Two routes are equivalent if they have the same source and destination airports. Find pairs of flights that have the same route, but are offered by different airlines. Report the flight id and flight number of the 2 flights. Report the cheaper flight (cheapest seat in that flight) first. If the two flights are tied for price, report the one with the smaller flight id first.
5. Find flight(s) from airport YYZ to airport LIS with the earliest scheduled arrival time *after* June 17, 2025 11:00. Consider direct flights, as well as flights with exactly one layover. Only consider layovers of at least 1 hour and at most 24 hours. Report the scheduled departure time from "YYZ" and the scheduled arrival time at "LIS" of the flight(s).
6. Find passengers that have the same last name and always travel together i.e., the set of flights they booked are the same. Report the first name of the two passengers, their last name and their most recent trip (as determined by the flight's scheduled departure). Only consider passengers who have been on at least one trip. Don't include pseudo-duplicates.
7. Find planes that have been used for less than four different flights. Report their tail number and the airline code of the airline that owns the plane.
8. Find passengers who have booked at least one flight, and for all their booked flights, they have booked exactly two seats. However, the two seats are never in the same row. Report the id(s) of these passengers.
9. Find all the airports that are not reachable from YYZ. An airport A is reachable from another airport B if there is a direct flight between A and B, or if there is a direct flight from A to another airport C and B is reachable from C.
10. Find passengers who have paid strictly less for their seat than the current price recorded in FlightPrice for every flight they have taken. Note that a passenger can book more than one seat on the same flight. Only consider passengers with at least one booking.
11. Airlines offer high discounts to passengers on unpopular routes. We define unpopular route as one where at least 2 different seats were booked with 50% or higher discount (that is booking price is lower than half the listed price). Find all the airlines that have operated a flight on all unpopular routes.

Part 5: Additional Integrity Constraints (30%)

Express the following integrity constraints with the notation $R = \emptyset$, where R is an expression of relational algebra. You are welcome to define intermediate results with assignment. The last step for each question must be a single assertion of the form expression $= \emptyset$.

Remember that at least one of the queries and/or integrity constraints in this assignment cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed".

1. For every plane the rows of seats in the plane should all be consecutively numbered starting with 1. For example, a plane with 25 rows of seats should have them from 1 to 25 without missing a number.

2. All of the following must hold:
    - The booking date-time must occur before the flight's scheduled departure.
    - The flight's actual departure can't be earlier than its scheduled departure.
    - The flight's actual arrival must be later than its actual departure.
3. A plane must depart from the same airport where it last landed (unless this is its very first trip), and there must be at least two hours between a plane's scheduled arrival and its next scheduled departure.
4. We define an airline's home country as the country where the airport(s) that the maximum number of that airline's flights fly through is located. While we can have multiple such airports for any given airline, all of those airports must be in the same country.
5. Each class available on a flight has a price recorded in FlightPrice. FlightPrice doesn't include information for classes not available on a flight.
6. A domestic flight is one whose source and destination are both within the same country. Planes exclusively used for domestic flights can travel a maximum of three times per month (refers to a calendar month instead of the past month), as determined by the flights' scheduled departure. This applies to planes with at least one recorded flight only.

Formatting Instructions

Your assignment must be typed to produce a PDF document named **a1.pdf** (hand-written submissions are not acceptable). You may use any word-processing software you like. Many academics use LaTeX, and some use a tool called Overleaf to work on LaTeX documents in the cloud, which makes collaborating with a partner easier. (Note that if you choose to use some kind of online storage of your assignment, make sure it is private to only you and your partner.)

Please use a font size of at least 10 (or larger). The default LaTeX font size is acceptable.

Regardless of which tool you use, make sure you leave yourself enough time to type up your answers. **Formatting may take longer than you expect!**

Submission Instructions

You may work on the assignment in groups of 1 or 2, and submit a single assignment for the entire group on MarkUs. Partners do not have to be in the same section, but must be in CSC343 at St. George this semester. You must establish your group well before the due date. Instructions on how to form a group can be found here. No changes may be made to groups after the original deadline (Wednesday 29th January) has passed.

To declare a group in MarkUs, one partner creates the group and the other must accept the invitation.

When you submit, check that you have submitted the correct version of your file by downloading it from MarkUs; new files will not be accepted after the due date. If you are working with a partner, only one of you needs to upload the file for your group. You can upload new files as many times as you like before the deadline.

How your assignment will be marked

Most of the marks will be for the correctness of your answers. However, there will be additional marks allocated to each of these:

- Comments:
  Does every assignment statement have a comment above it specifying clearly exactly what rows get to be in this relation? Comments should describe the data (*e.g.*, "The student number of every student who has passed at least 4 courses.") not how to find it (*e.g.*, "Find the student numbers by self-joining ..."). Comments should appear *above* each assignment, and start with two dashes.

- Attribute names given on the LHS:
  Does every assignment statement name the attributes on the LHS? This should be done even if the names are not being changed. Together with the comments, it allows you to understand what a "subquery" is supposed to do without reading it. Think of this as analogous to good parameter names and good comments on a function.

- Relation and attribute names:
  Does every relation and every attribute have a name that will assist the reader in understanding the query quickly? Apply the same high standards you would when writing code.

- Formatting:
  Is the algebra formatted with appropriate line breaks and indentation to reveal the structure of the algebra for ease of understanding?

A clear and easy to read submission means we will have an easier time finding ways to give you part marks if you have correctness issues, and you will have an easier time checking over and fixing your work if you keep it clear as you go.