Introduction to Database Design, E2015
IT University of Copenhagen
Carsten Schürmann

Hand-in 3

This is an individual assignment, and not a group assignment.

Problem 1: Data Modeling

A database containing information about flight departures is under development. The goal is to support a number of queries that are traditionally not supported by booking systems. Instead of a specification of the database schema, we give examples of the queries the system should support. *Note, that for this problem, we do not expect you to implement the queries in SQL.*

Connection queries:

- Find all direct flights from A to B departing between time t_1 and t_2.
- Find all direct flights from A to B arriving between time t_1 and t_2.
- Find all connections from A to B, involving exactly 1 stopover of maximum 3 hours.
- Find the fastest connection between A and B (total travel time).

Specified connection queries:

- Find all flights from A to a place where the average temperature in the given month is at least 25 degrees.
- Find all flights from A to Africa (or any continent of your choice) arriving between t_1 and t_2.
- Find one direct connection from A to B with an airline from USA (or any country of your choice). A flight can be associated with more than one airline ("code share").

Ticket queries:

- Find available ticket types and their prices for a flight.
- Find the cheapest available ticket type for a given connection query.
- Find the cheapest available business class ticket for a given connection query.

a) Create an E-R model for a database containing enough information to answer all of the above queries. If you make any assumptions about the data that has not been made explicit in the problem description, you are expected to describe them explicitly. Your model should express as many properties about the data as possible, such as participation constraints and a normalized database design. Data not needed for the above (e.g., detailed information about bookings) should not appear in the model.
Important: Do not *autogenerate any parts of an answer. When drawing E-R diagrams, use the notation which was used in class.*

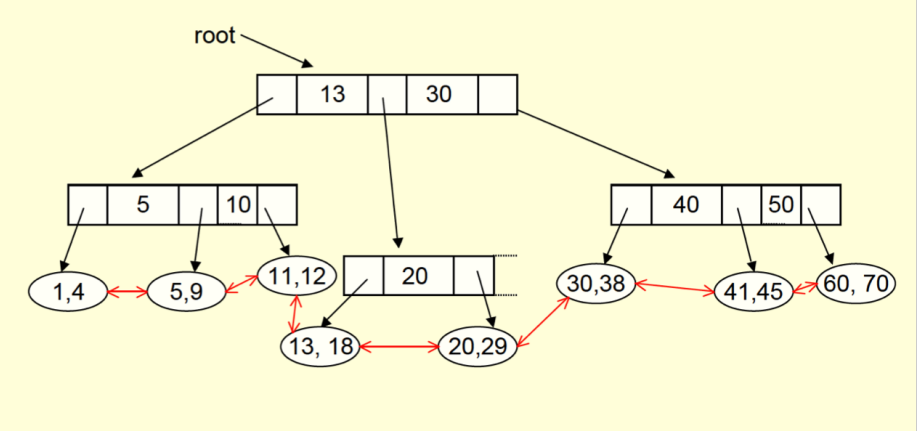
b) Create a relational data model that represents the E-R model. For each relation you need to state attributes, primary keys (underline) and foreign keys (dashed underline).

Problem 2: Indexing

Consider the database in the first problem, together with the query "Find all direct flights from A to B arriving between time t_1 and t_2, sorted by price".

a) Explain how you would use indexing to speed up this query. Define the appropriate index, and justify why you think this index will speed up your query.

Some indexes use special purpose data structures, just like the following B+-tree. In the following we ask you to add a particular element to and from the index.



b) Add 8 to the B+-tree above, and draw the resulting tree.

c) Add 29 to the B+-tree above, and draw the resulting B+-tree. Use the original tree, and *not* the one that resulted from (b).

Problem 3: SQL queries

Consider the following relations containing information on the inspection of airplanes:

```
Inspector(iid, planeId, date)
Employee(eid, name, title, employedSince)
InspectedBy(employeeId, inspectionId)
Plane(pid, airline, modelId, buildDate)
Model(mid, name, manufacturer, engineType)
```

Attributes in primary keys are underlined. In the following we assume that no foreign keys have been assigned in the database.

a) Write an SQL query that returns `pid` on all SAS airplanes built after `2000-01-01` (`buildDate > '2000-01-01'`).

b) Write an SQL query that calculates the number of employees who have inspected an SAS airplane (`airline='SAS'`).

c) Write one or more SQL commands that will delete all information about inspections before 2000-01-01.

d) Write an SQL query that returns the names of the employees who have only worked with airplanes from one manufacturer (`manufacturer`).

Problem 4: Normalization

Consider the two relations from the previous problem,

```
Plane(pid, airline, modelId, buildDate)
Model(mid, name, manufacturer, engineType),
```

where `pid` and `mid` are primary keys for each relation. Furthermore, `modelId` is a foreign key that references Model(mid). Now assume that each manufacturer (`manufacturer`) only builds one plane per date (`buildDate`).

a) Argue that {`modelId, buildDate`} is a key in `Plane`.

Someone is considering to merge the relations into one (by a join operation):

```
Plane2(pid, airline, buildDate, modelId, modelName,
        manufacturer, engineType)
```

b) Argue that this relation is not in BCNF. Your argumentation should use the concept of functional dependencies (FDs).

To be handed in

The second hand-in should be handed in by each group no later than

Friday 4th of November, before class.

Please upload your reply to learn-it. This project will be graded and contribute with 15% to your final grade.