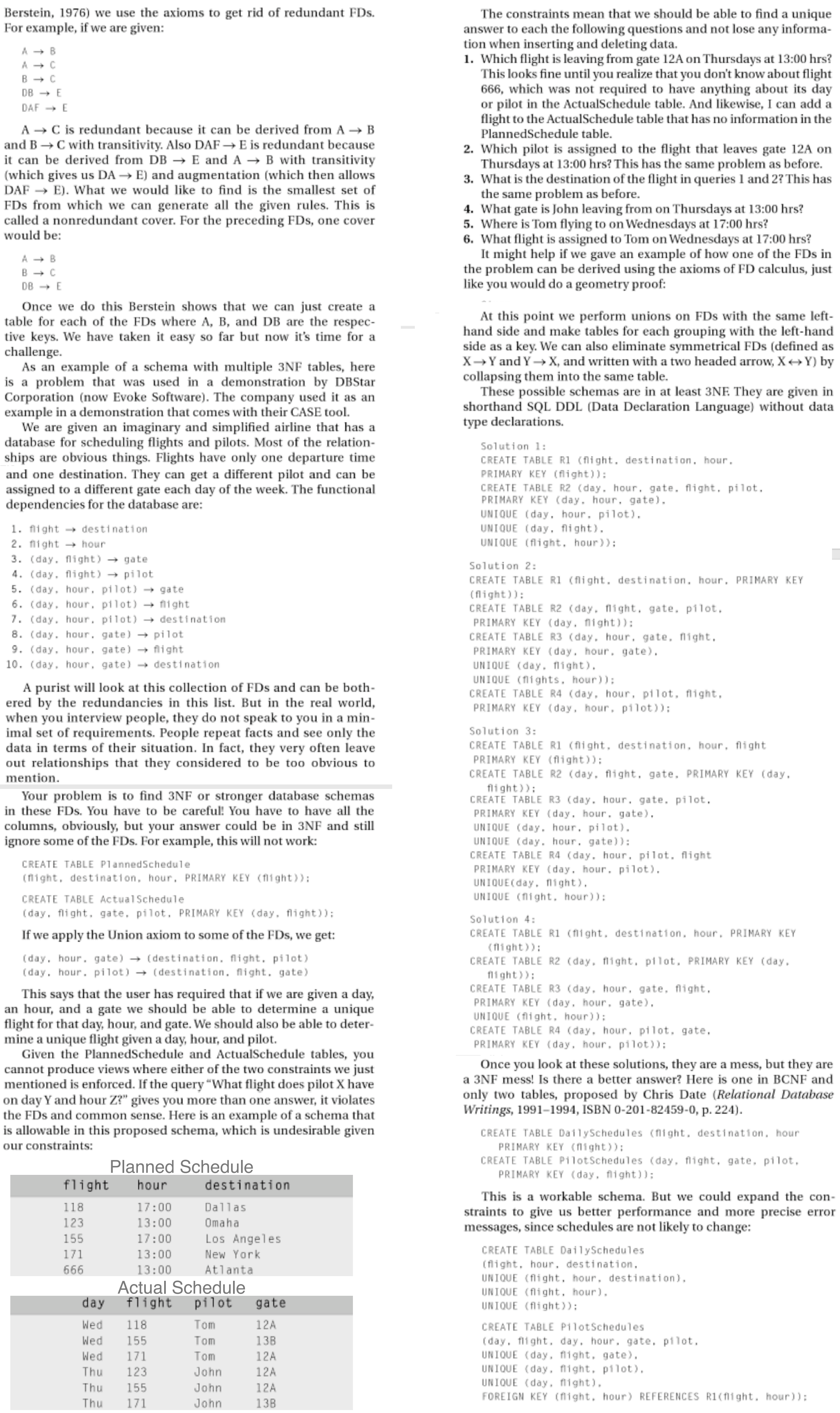
Airline 10

Celko Extract

For this case study, we will consider the Airline-Flight-Pilot database, which is used as an example by DBStar, and made famous by Celko in his chapter on "normalisation" in *SQL for Smarties: Advanced SQL Programming*, for which C J Date has also supplied an 'answer'.



Berstein, 1976) we use the axioms to get rid of redundant FDs. For example, if we are given:

```
A → B
A → C
B → C
DB → E
DAF → E
```

A → C is redundant because it can be derived from A → B and B → C with transitivity. Also DAF → E is redundant because it can be derived from DB → E and A → B with transitivity (which gives us DA → E) and augmentation (which then allows DAF → E). What we would like to find is the smallest set of FDs from which we can generate all the given rules. This is called a nonredundant cover. For the preceding FDs, one cover would be:

```
A → B
B → C
DB → E
```

Once we do this Berstein shows that we can just create a table for each of the FDs where A, B, and DB are the respective keys. We have taken it easy so far but now it's time for a challenge.

As an example of a schema with multiple 3NF tables, here is a problem that was used in a demonstration by DBStar Corporation (now Evoke Software). The company used it as an example in a demonstration that comes with their CASE tool.

We are given an imaginary and simplified airline that has a database for scheduling flights and pilots. Most of the relationships are obvious things. Flights have only one departure time and one destination. They can get a different pilot and can be assigned to a different gate each day of the week. The functional dependencies for the database are:

```
1. flight → destination
2. flight → hour
3. (day, flight) → gate
4. (day, flight) → pilot
5. (day, hour, pilot) → gate
6. (day, hour, pilot) → flight
7. (day, hour, pilot) → destination
8. (day, hour, gate) → pilot
9. (day, hour, gate) → flight
10. (day, hour, gate) → destination
```

A purist will look at this collection of FDs and can be bothered by the redundancies in this list. But in the real world, when you interview people, they do not speak to you in a minimal set of requirements. People repeat facts and see only the data in terms of their situation. In fact, they very often leave out relationships that they considered to be too obvious to mention.

Your problem is to find 3NF or stronger database schemas in these FDs. You have to be careful! You have to have all the columns, obviously, but your answer could be in 3NF and still ignore some of the FDs. For example, this will not work:

```
CREATE TABLE PlannedSchedule
(flight, destination, hour, PRIMARY KEY (flight));

CREATE TABLE ActualSchedule
(day, flight, gate, pilot, PRIMARY KEY (day, flight));
```

If we apply the Union axiom to some of the FDs, we get:

```
(day, hour, gate) → (destination, flight, pilot)
(day, hour, pilot) → (destination, flight, gate)
```

This says that the user has required that if we are given a day, an hour, and a gate we should be able to determine a unique flight for that day, hour, and gate. We should also be able to determine a unique flight given a day, hour, and pilot.

Given the PlannedSchedule and ActualSchedule tables, you cannot produce views where either of the two constraints we just mentioned is enforced. If the query "What flight does pilot X have on day Y and hour Z?" gives you more than one answer, it violates the FDs and common sense. Here is an example of a schema that is allowable in this proposed schema, which is undesirable given our constraints:

Planned Schedule

flight	hour	destination
118	17:00	Dallas
123	13:00	Omaha
155	17:00	Los Angeles
171	13:00	New York
666	13:00	Atlanta

Actual Schedule

day	flight	pilot	gate
Wed	118	Tom	12A
Wed	155	Tom	13B
Wed	171	Tom	12A
Thu	123	John	12A
Thu	155	John	12A
Thu	171	John	13B

The constraints mean that we should be able to find a unique answer to each the following questions and not lose any information when inserting and deleting data.

1. Which flight is leaving from gate 12A on Thursdays at 13:00 hrs? This looks fine until you realize that you don't know about flight 666, which was not required to have anything about its day or pilot in the ActualSchedule table. And likewise, I can add a flight to the ActualSchedule table that has no information in the PlannedSchedule table.
2. Which pilot is assigned to the flight that leaves gate 12A on Thursdays at 13:00 hrs? This has the same problem as before.
3. What is the destination of the flight in queries 1 and 2? This has the same problem as before.
4. What gate is John leaving from on Thursdays at 13:00 hrs?
5. Where is Tom flying to on Wednesdays at 17:00 hrs?
6. What flight is assigned to Tom on Wednesdays at 17:00 hrs?

It might help if we gave an example of how one of the FDs in the problem can be derived using the axioms of FD calculus, just like you would do a geometry proof:

At this point we perform unions on FDs with the same left-hand side and make tables for each grouping with the left-hand side as a key. We can also eliminate symmetrical FDs (defined as X → Y and Y → X, and written with a two headed arrow, X ↔ Y) by collapsing them into the same table.

These possible schemas are in at least 3NF. They are given in shorthand SQL DDL (Data Declaration Language) without data type declarations.

```
Solution 1:
CREATE TABLE R1 (flight, destination, hour,
PRIMARY KEY (flight));
CREATE TABLE R2 (day, hour, gate, flight, pilot,
PRIMARY KEY (day, hour, gate),
UNIQUE (day, hour, pilot),
UNIQUE (day, flight),
UNIQUE (flight, hour));

Solution 2:
CREATE TABLE R1 (flight, destination, hour, PRIMARY KEY
(flight));
CREATE TABLE R2 (day, flight, gate, pilot,
 PRIMARY KEY (day, flight));
CREATE TABLE R3 (day, hour, gate, flight,
 PRIMARY KEY (day, hour, gate),
 UNIQUE (day, flight),
 UNIQUE (flights, hour));
CREATE TABLE R4 (day, hour, pilot, flight,
 PRIMARY KEY (day, hour, pilot));

Solution 3:
CREATE TABLE R1 (flight, destination, hour, flight
 PRIMARY KEY (flight));
CREATE TABLE R2 (day, flight, gate, PRIMARY KEY (day,
   flight));
CREATE TABLE R3 (day, hour, gate, pilot,
 PRIMARY KEY (day, hour, gate),
 UNIQUE (day, hour, pilot),
 UNIQUE (day, hour, gate));
CREATE TABLE R4 (day, hour, pilot, flight
 PRIMARY KEY (day, hour, pilot),
 UNIQUE(day, flight),
 UNIQUE (flight, hour));

Solution 4:
CREATE TABLE R1 (flight, destination, hour, PRIMARY KEY
   (flight));
CREATE TABLE R2 (day, flight, pilot, PRIMARY KEY (day,
   flight));
CREATE TABLE R3 (day, hour, gate, flight,
 PRIMARY KEY (day, hour, gate),
 UNIQUE (flight, hour));
CREATE TABLE R4 (day, hour, pilot, gate,
 PRIMARY KEY (day, hour, pilot));
```

Once you look at these solutions, they are a mess, but they are a 3NF mess! Is there a better answer? Here is one in BCNF and only two tables, proposed by Chris Date (*Relational Database Writings*, 1991–1994, ISBN 0-201-82459-0, p. 224).

```
CREATE TABLE DailySchedules (flight, destination, hour
   PRIMARY KEY (flight));
CREATE TABLE PilotSchedules (day, flight, gate, pilot,
   PRIMARY KEY (day, flight));
```

This is a workable schema. But we could expand the constraints to give us better performance and more precise error messages, since schedules are not likely to change:

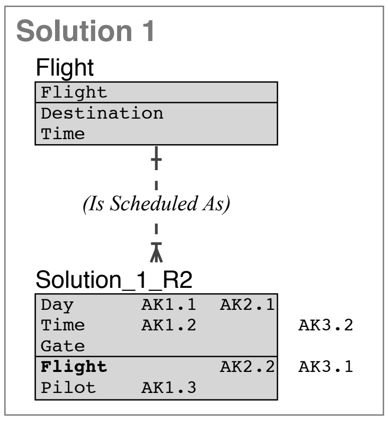
```
CREATE TABLE DailySchedules
(flight, hour, destination,
UNIQUE (flight, hour, destination),
UNIQUE (flight, hour),
UNIQUE (flight));

CREATE TABLE PilotSchedules
(day, flight, day, hour, gate, pilot,
UNIQUE (day, flight, gate),
UNIQUE (day, flight, pilot),
UNIQUE (day, flight),
FOREIGN KEY (flight, hour) REFERENCES R1(flight, hour));
```

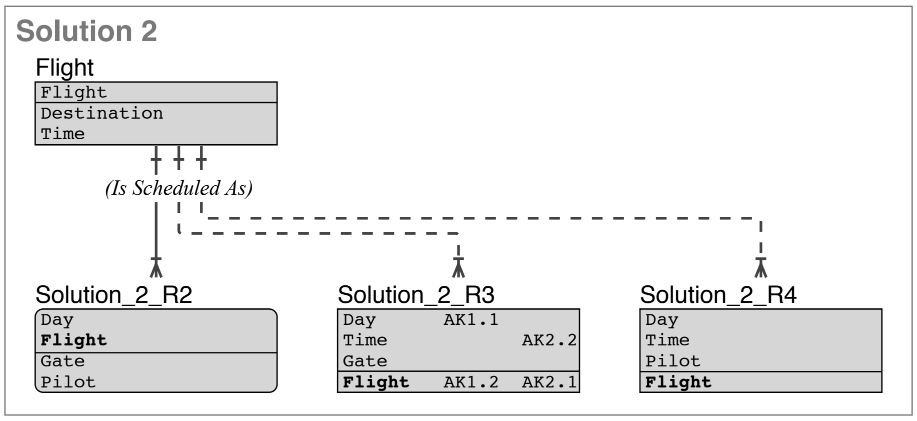
Gotta love the way 3NF has a life and a soul, creative powers, and creates the mess, all on its own. This is their idea of "science".

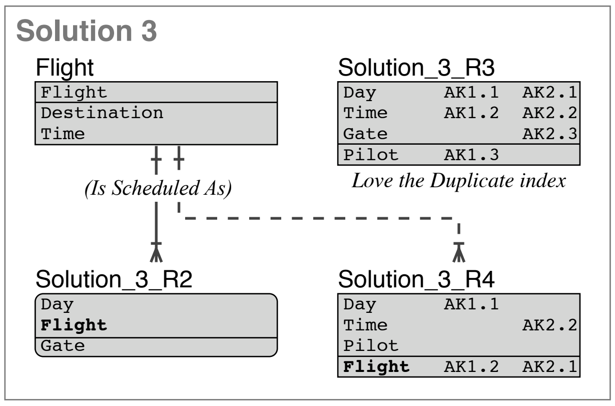
Celko & Date Proposal

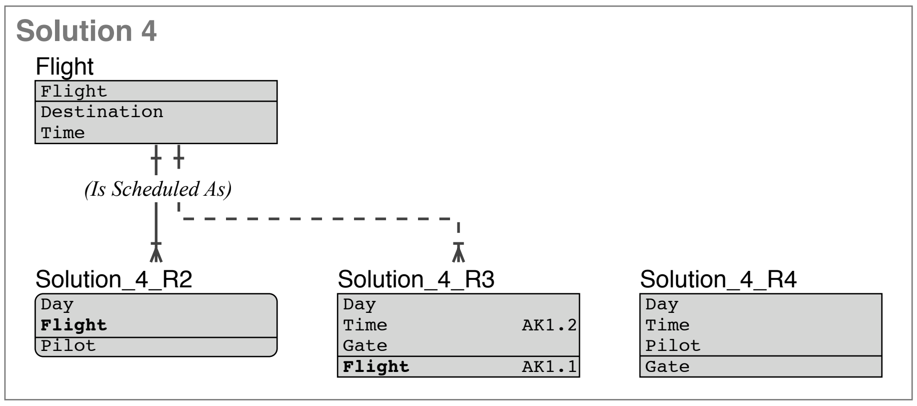
A J Celko: Presented as "3NF", and a "3NF Mess"

In his chapter on "normalisation", Celko labours over the stated problem, and presents and discusses Functional Dependencies and 'normal forms', intimating that, as recommended by Date, the ***method*** of Normalisation is through the NFs. He presents these four 'solutions' as being "in Third Normal Form"; as being "a mess, but they are a 3NF mess". The allegations are completely false, because (a) none of them are in Third Normal Form, and (b) the Functional Dependencies, that Celko himself identified as demanding resolution, have not been resolved. By his own evidence, Celko is confused re FDs, the NFs, and Normalisation. Also notable, formal normalisation-by-NFs is not attempted. No solution is provided. Given the presentation and discussion, we are not expecting genuine Normalisation, only normalisation-by-NFs; however, not even that is provided.



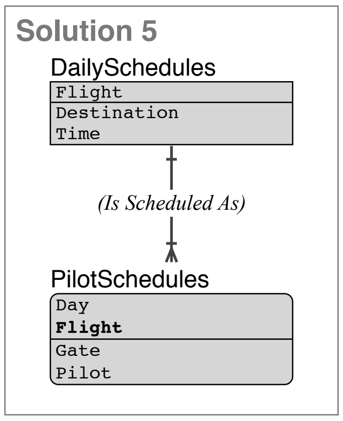


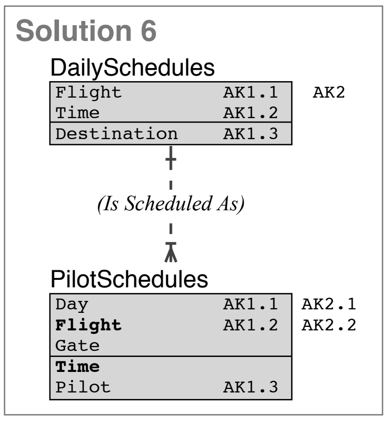




B C J Date: Presented as "BCNF", Celko "Improves" It

Being completely unable to produce a solution, Celko consults with his partner in crime, C J Date. Evidently Date (Solution 5) starts at a somewhat different square one, but he does not understand FDs, the NFs, or Normalisation, either. No solution from the great author of database "science". Celko of course, 'improves' Date's contribution (Solution 6), demonstrating (a) his abject understanding of keys and indices, and (b) his love for complexity that delivers nothing.





- *PK is not given, but it is likely (Flight, Time), since he uses it as FK in the child*
- *One can only laugh or cry, at Alternate Key 1*

- *PK is not given, one of two possibilities is used*
- *No index on (Flight, Time) is proposed*
- *A Duplicated column has been excluded*

The relevant points are, in Celko's chapter titled "normalisation":
- neither Celko nor Date can provide a solution to the stated problem
- neither Normalisation nor his "normalisation" is taught. 'Normalisation' via the 'normal forms' is implied, yet again, but not demonstrated
- the declaration that these are in BCNF is false.

C Choice of Problem

This problem has been chosen for the exercise, because:
- it is a good classroom problem, and one that neither Celko nor Date can solve
- the solution that eludes them is easily derived, using genuine Normalisation (as distinct from the abnormal 'normal forms').

Before commencing the exercise, the following should be noted:
- their struggle with Normalisation
- the difficulty or impossibility of attempting it sans Determination of Entities and Keys
- their confusion with Functional Dependencies
- the common error of uneducated developers: adding indices in a vain attempt to solve the problem. This adds complexity (to both DDL and DML) and harms performance, without delivering anything.
- In case it needs to be said, there is nothing whatsoever to be gained from Celko and Date's various gyrations about either the problem or about Normalisation: completely erase it from your mind, in order to decontaminate it, before commencing the exercise.