MODULE 5 ASSIGNMENT 1 - NEW FLIGHTS DATABABASE

BY RANDY LEON

Step one:
Going back to the original ***flights*** database, you can generate CREATE TABLE scripts for each table.

For airlines:

Query Editor | Query History

```
-- Table: public.airlines(1sttable)
-- DROP TABLE public.airlines;
CREATE TABLE public.airlines
(
    carrier character(2) COLLATE pg_catalog."default" NOT NULL,
    name character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT airlines_pkey PRIMARY KEY (carrier)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
-- Table: public.airports(2ndtable)
-- DROP TABLE public.airports;

CREATE TABLE public.airports
(
    faa character(3) COLLATE pg_catalog."default",
    name character varying COLLATE pg_catalog."default",
    lat double precision,
    lon double precision,
    alt integer,
    tz integer,
    dst character(1) COLLATE pg_catalog."default"
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 55 msec.
```

Step one:

Going back to the original ***flights*** database, you can generate CREATE TABLE scripts for each table.

For airports:

```
16  CREATE TABLE public.airports
17  (
18      faa character(3) COLLATE pg_catalog."default",
19      name character varying COLLATE pg_catalog."default",
20      lat double precision,
21      lon double precision,
22      alt integer,
23      tz integer,
24      dst character(1) COLLATE pg_catalog."default"
25  )
26  WITH (
27      OIDS = FALSE
28  )
29  TABLESPACE pg_default;
30  -- Table: public.flights(3rdtable)
31  -- DROP TABLE public.flights;
```

Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 50 msec.
```

Step one:

Going back to the original ***flights*** database, you can generate CREATE TABLE scripts for each table.

For flights:

```
-- Table: public.flights(3rdtable)
-- DROP TABLE public.flights;
CREATE TABLE public.flights
(
    year integer,
    month integer,
    day integer,
    dep_time integer,
    dep_delay integer,
    arr_time integer,
    arr_delay integer,
    carrier character(2) COLLATE pg_catalog."default",
    tailnum character(6) COLLATE pg_catalog."default",
    flight integer,
    origin character(3) COLLATE pg_catalog."default",
    dest character(3) COLLATE pg_catalog."default",
    air_time integer,
    distance integer,
    hour integer,
    minute integer
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 42 msec.
```

Step one:

Going back to the original ***flights*** database, you can generate CREATE TABLE scripts for each table.

For planes:

```
-- Table: public.planes(4thtable)
-- DROP TABLE public.planes
CREATE TABLE public.planes
(
    tailnum character(6) COLLATE pg_catalog."default",
    year integer,
    type character varying COLLATE pg_catalog."default",
    manufacturer character varying COLLATE pg_catalog."default",
    model character varying COLLATE pg_catalog."default",
    engines integer,
    seats integer,
    speed integer,
    engine character varying COLLATE pg_catalog."default"
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

Explain | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 44 msec.
```

Step one:

Going back to the original ***flights*** database, you can generate CREATE TABLE scripts for each table.

For weather:

```
-- Table: public.weather(5thtable)
-- DROP TABLE public.weather;
CREATE TABLE public.weather
(
    origin character(3) COLLATE pg_catalog."default",
    year integer,
    month integer,
    day integer,
    hour integer,
    temp double precision,
    dewp double precision,
    humid double precision,
    wind_dir integer,
    wind_speed double precision,
    wind_gust double precision,
    precip double precision,
    pressure double precision,
    visib double precision
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

Explain Messages Notifications

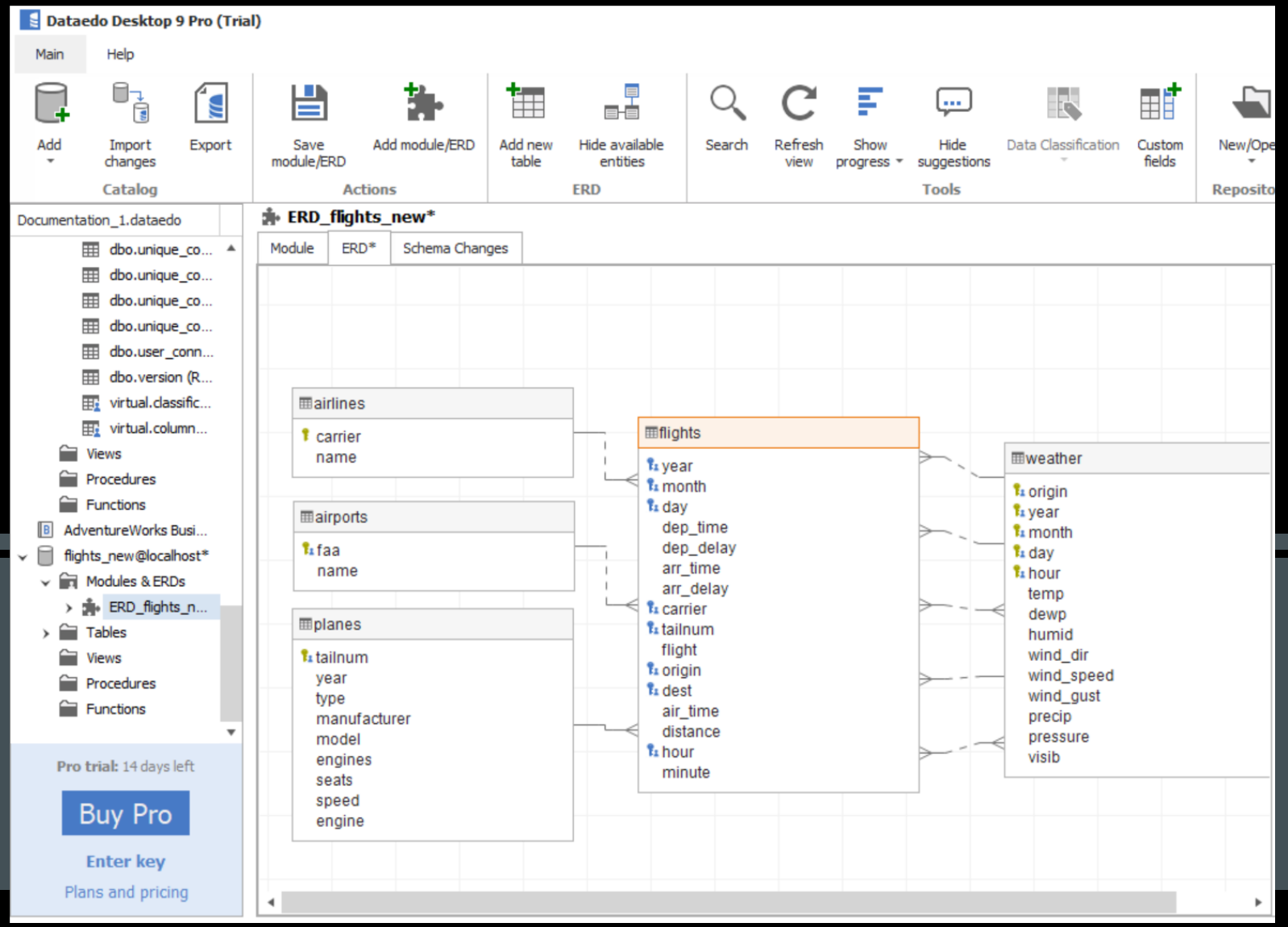
```
CREATE TABLE

Query returned successfully in 44 msec.
```

Step two:

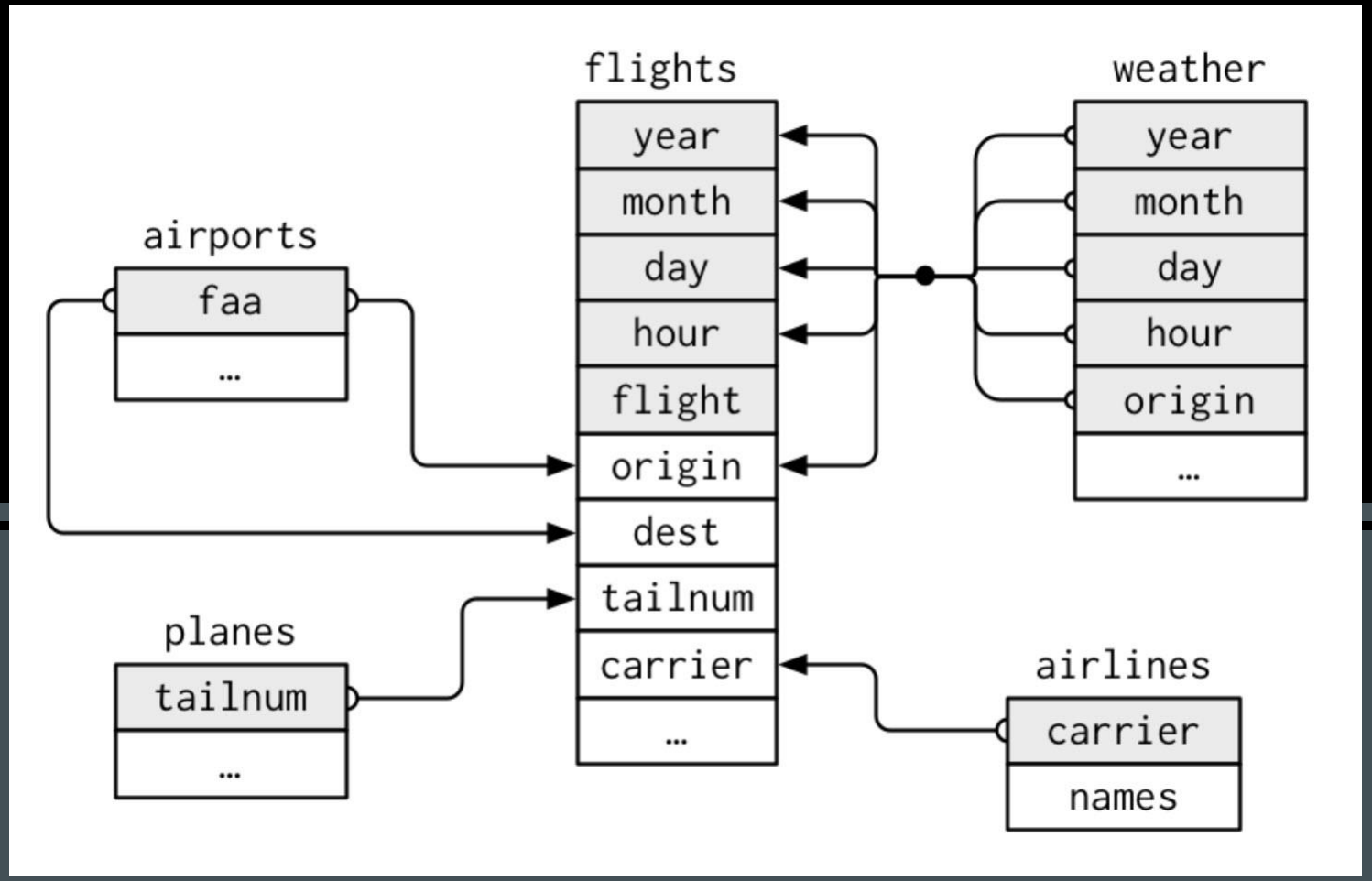
Now create the FOREIGN Keys in the flights_new database based on the models.

```
98  --alter table statements to add foreign keys
99  ALTER TABLE public.airlines
100     ADD FOREIGN KEY (carrier)
101     REFERENCES public.flights ("carrier");
102
103 ALTER TABLE public.airports
104     ADD FOREIGN KEY ("faa")
105     REFERENCES public.flights ("origin", "dest");
106
107 ALTER TABLE public.planes
108     ADD FOREIGN KEY ("tailnum")
109     REFERENCES public.flights ("tailnum");
110
111 ALTER TABLE public.weather
112     ADD FOREIGN KEY (year, month, day, hour, "origin")
113     REFERENCES public.flights (year, month, day, hour, origin);
```

Step three: Once complete, reverse engineer the flights_new database into Dataedo and compare to your original model.



Step three: Once complete, reverse engineer the flights_new database into Dataedo and compare to your original model.



Flights table is the unifying table where all other tables in this database have foreign keys to reference to.

In this case, it is probably an authority in the airline industry keep track of every flight in and out, and the other tables serve to supplement the flights table with more information.