FIT2094-FIT3171 Databases

Session 5 Tutorial Suggested Solution

NORMALISATION

FIT Database Teaching Team

FIT2094-FIT3171 2021 Summer B

FIT2094-FIT3171 Databases

Author: FIT Database Teaching Team

Important

Remember, before starting any tutorial activity which involves working with files, first use SQL Developer to pull from the FIT GitLab server so as to ensure your local and server files are in sync.

5.1 Steps on Normalisation -- Tutor Explanation

UNF
APPOINTMENT(dentist_no, dentist_name, patient_no, patient_name, app_datetime, surgeryroom_no)

1NF
APPOINTMENT(<u>dentist_no</u>, dentist_name, patient_no, patient_name, <u>app_datetime</u>, surgeryroom_no)
**note that there are 3 candidate keys:*

- *(dentist_no, app_datetime),*
- *(patient_no, app_datetime)*
- *(surgeryroom_no, app_datetime)*

and (dentist_no, app_datetime) is picked as PK

Partial dependencies:
dentist_no → dentist_name
patient_no → patient_name

**note that we use general definition, partial dependency is based on PK and all candidate keys*

2NF
APPOINTMENT(<u>dentist_no</u>, patient_no, <u>app_datetime</u>, surgeryroom_no)

DENTIST(<u>dentist_no</u>, dentist_name)

PATIENT(<u>patient_no</u>, patient_name)

Transitive dependencies:
No transitive dependency

3NF
There is no transitive dependency, the 3NF is the same as the 2NF. Note that you are **required** to show all forms, even if they are the same as a previous form.

APPOINTMENT(<u>dentist_no</u>, patient_no, <u>app_datetime</u>, surgeryroom_no)

DENTIST(<u>dentist_no</u>, dentist_name)

PATIENT(<u>patient_no</u>, patient_name)

Full Dependencies:
dentist_no, app_datetime → pat_no, surgeryroom_no
dentist_no → dentist_name
patient_no → patient_name

5.2 Multiple Forms Normalisation -- Part 1

APPROVED UNITS REPORT

UNF
UNIT (unit_no, unit_name, unit_desc, unit_value)

1NF
UNIT (unit_no, unit_name, unit_desc, unit_value)

Partial Dependencies:
No Partial Dependency

2NF
UNIT (unit_no, unit_name, unit_desc, unit_value)

Transitive Dependencies:
No Transitive Dependency

3NF
UNIT (unit_no, unit_name, unit_desc, unit_value)

Full Dependencies:
unit_no → unit_name, unit_desc, unit_value

LECTURER REPORT

UNF
LECTURER (lect_no, lect_name, lect_office, lect_phone, (unit_no, unit_name))

1NF
LECTURER (lect_no, lect_name, lect_office, lect_phone)
**Note: lect_phone is one of the candidate keys*

ADVICE (lect_no, unit_no, unit_name)

Partial Dependencies:
unit_no -> unit_name

2NF
LECTURER (lect_no, lect_name, lect_office, lect_phone)

ADVICE (lect_no, unit_no)

UNIT (unit_no, unit_name)

Transitive Dependencies:
No Transitive Dependency

**Note: There is no transitive dependency here related to lect_phone as lect_phone is a candidate key - transitive dependency is about the removal of non-key dependencies ie. dependencies between non-key attributes (lect_phone is not a non-key attribute)*

3NF
LECTURER (lect_no, lect_name, lect_office, lect_phone)

ADVICE (lect_no, unit_no)

UNIT (unit_no, unit_name)

Full Dependencies:

lect_no → lect_name, lect_office, lect_phone
unit_no → unit_name

STUDENT REPORT

UNF
STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no, lect_name, (unit_no, unit_name, year, semester, grade))

Note: replacement of mentor details with lecturer details - a mentor is a lecturer - this prevents the introduction of synonyms (attributes with different names but representing the same thing)

1NF
STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no, lect_name)

AC_REC (stu_no, unit_no, year, semester, unit_name, grade)

Partial Dependencies:
unit_no -> unit_name

2NF
STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no, lect_name)

AC_REC (stu_no, unit_no, year, semester, grade)

UNIT (unit_no, unit_name)

Transitive Dependencies:
lect_no → lect_name

3NF
STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no)

LECTURER (lect_no, lect_name)

AC_REC (stu_no, unit_no, year, semester, grade)

UNIT (unit_no, unit_name)

Full Dependencies:
stu_no → stu_name, stu_address, stu_crse, stu_mode, lect_no

lect_no → lect_name
stu_no, unit_no, year, semester → grade
unit_no → unit_name

COLLECTED 3NF RELATIONS:

1. UNIT (unit_no, unit_name, unit_desc, unit_value)

2. LECTURER (lect_no, lect_name, lect_office, lect_phone)

3. ADVICE (lect_no, unit_no)

4. UNIT (unit_no, unit_name)

5. STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no)

6. LECTURER (lect_no, lect_name)

7. AC_REC (stu_no, unit_no, year, semester, grade)

8. UNIT (unit_no, unit_name)

ATTRIBUTE SYNTHESIS

Join together relations, which have an **identical** PK – ie. represent the same entity:

1. 4. & 8.
UNIT (unit_no, unit_name, unit_desc, unit_value)

2. & 6.
LECTURER (lect_no, lect_name, lect_office, lect_phone)

3.
ADVICE (lect_no, unit_no)

5.
STUDENT (stu_no, stu_name, stu_address, stu_crse, stu_mode, lect_no)

7.
AC_REC (stu_no, unit_no, year, semester, grade)

Prior to building the logical model, so as to maintain relation name prefixes to attributes AC_REC attributes year, semester and grade will be renamed to:

AC_REC (stu_no, unit_no, ar_year, ar_semester, ar_grade)

Please note that the above steps show the standard of the normalisation process and the format that we expect all students to produce in their assignment submissions.

5.3 Normalise Multiple Forms -- Part 2

PROPERTY MAINTENANCE REPORT

**Note: in normalisation you have to decompose attribute when it is necessary (i.e. stated either in case study or in the form/report)*

UNF
PROPERTY(prop_no, prop_address, owner_no, owner_title, owner_givname, owner_famname, owner_address, (maint_datetime, maint_desc, maint_cost))

1NF
PROPERTY(<u>prop_no</u>, prop_address, owner_no, owner_title, owner_givname, owner_famname, owner_address)

MAINTENANCE(<u>prop_no</u>, <u>maint_datetime</u>, maint_desc, maint_cost)

Partial Dependencies:
No Partial Dependency

2NF
PROPERTY(<u>prop_no</u>, prop_address, owner_no, owner_title, owner_givname, owner_famname, owner_address)

MAINTENANCE(<u>prop_no</u>, <u>maint_datetime</u>, maint_desc, maint_cost)

Transitive dependencies:
owner_no → owner_title, owner_givname, owner_famname, owner_address

3NF
OWNER(<u>owner_no</u>, owner_title, owner_givname, owner_famname, owner_address)
PROPERTY(<u>prop_no</u>, prop_address, owner_no)

MAINTENANCE(<u>prop_no</u>, <u>maint_datetime</u>, maint_desc, maint_cost)

Full Dependencies:
owner_no → owner_title, owner_givname, owner_famname, owner_address
prop_no → prop_address, owner_no
prop_no, maint_datetime → maint_desc, maint_cost

PROPERTY TENANT LEDGER REPORT

UNF
PROPERTY_TENANT(prop_no, prop_address, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no, tenant_title, tenant_givname, tenant_famname, (pay_no, pay_date, pay_type, pay_amount, pay_paidby))

1NF
PROPERTY_TENANT(prop_no, prop_address, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no, tenant_title, tenant_givname, tenant_famname)

**note: prop_no and rent_lease_startdate is the only candidate key, hence the PK. The combination of tenant_no and prop_no is not unique since a tenant can rent the same property more than once. The combination of tenant_no and rent_lease_startdate is also not unique since a tenant may rent more than two properties at the same time.*

PAYMENT(prop_no, rent_lease_startdate, pay_no, pay_date, pay_type, pay_amount, pay_paidby)

**note: pay_no is unique for each payment, thus this new relation brings along prop_no and rent_lease_startdate (PROPERTY_TENANT PK) as part of repeating group removal, but these attributes are not part of PAYMENT PK*

Partial dependencies:
prop_no → prop_address

2NF
PROPERTY(prop_no, prop_address)

PROPERTY_TENANT(prop_no, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no, tenant_title, tenant_givname, tenant_famname)

PAYMENT(prop_no, rent_lease_startdate, pay_no, pay_date, pay_type, pay_amount, pay_paidby)

Transitive dependencies:
tenant_no → tenant_title, tenant_givname, tenant_famname

3NF
PROPERTY(prop_no, prop_address)

TENANT(tenant_no, tenant_title, tenant_givname, tenant_famname)

PROPERTY_TENANT(prop_no, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no)

PAYMENT(prop_no, rent_lease_startdate, pay_no, pay_date, pay_type, pay_amount, pay_paidby)

Full dependencies:
prop_no → prop_address
tenant_no → tenant_title, tenant_givname, tenant_famname
prop_no, rent_lease_startdate → rent_weekly_rate, rent_bond, tenant_no
pay_no → prop_no, rent_lease_startdate, pay_date, pay_type, pay_amount, pay_paidby

COLLECTED 3NF RELATIONS:

1. OWNER(owner_no, owner_title, owner_givname, owner_famname, owner_address)
2. PROPERTY(prop_no, prop_address, owner_no)
3. MAINTENANCE(prop_no, maint_datetime, maint_desc, maint_cost)
4. PROPERTY(prop_no, prop_address)
5. TENANT(tenant_no, tenant_title, tenant_givname, tenant_famname)
6. PROPERTY_TENANT(prop_no, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no)
7. PAYMENT(prop_no, rent_lease_startdate, pay_no, pay_date, pay_type, pay_amount, pay_paidby)

ATTRIBUTE SYNTHESIS

Join together relations, which have an **identical** PK – ie. represent the same entity:

1.
OWNER(owner_no, owner_title, owner_givname, owner_famname, owner_address)

2. & 4.
PROPERTY(prop_no, prop_address, owner_no)

3.
MAINTENANCE(prop_no, maint_datetime, maint_desc, maint_cost)

5.
TENANT(tenant_no, tenant_title, tenant_givname, tenant_famname)

6.
PROPERTY_TENANT(prop_no, rent_lease_startdate, rent_weekly_rate, rent_bond, tenant_no)

7.
PAYMENT(prop_no, rent_lease_startdate, pay_no, pay_date, pay_type, pay_amount, pay_paidby)

REMINDER: Again, the above steps show the standard of the normalisation process and the format that we expect all students to produce in their assignment submissions.

5.4 Additional Normalisation Exercise

UNF

BOOKING (booking_no, client_no, client_name, (flight_no, dep_date,dep_time,dep_air_code, dep_air_name, arr_date, arr_time, arr_air_code, arr_air_name, flight_duration))

1NF
BOOKING (booking_no, client_no, client_name)

BOOKING_LEG (booking_no, flight_no, dep_date, dep_time,dep_air_code, dep_air_name, arr_date, arr_time, arr_air_code, arr_air_name, flight_duration)

CKs:
booking_no, flight_no, dep_date
booking_no, flight_no, arr_date

Partial Dependencies:
flight_no → dep_time, dep_air_code, dep_air_name, arr_time, arr_air_code, arr_air_name, flight_duration

flight_no, dep_date → arr_date*
flight_no, arr_date → dep_date*
**Note: these two partial dependency removals create two relations which have the same structure which is (flight_no, dep_date, arr_date) in 2NF, the difference is only the PK choice, so we need to pick one of them.*

2NF
BOOKING (booking_no, client_no, client_name)

BOOKING_LEG (booking_no, flight_no, dep_date)

FLIGHT_INSTANCE (flight_no, dep_date, arr_date)*

FLIGHT (flight_no, dep_time, dep_air_code, dep_air_name, arr_time, arr_air_code, arr_air_name, flight_duration)

Transitive Dependencies:
client_no → client_name
dep_air_code → dep_air_name
arr_air_code → arr_air_name

3NF
CLIENT (client_no, client_name)

BOOKING (booking_no, client_no)

BOOKING_LEG (booking_no, flight_no, dep_date)

FLIGHT_INSTANCE (flight_no, dep_date, arr_date)

FLIGHT (flight_no, dep_time, dep_air_code, arr_time, arr_air_code, flight_duration)

DEP_AIRPORT (dep_air_code, dep_air_name)

ARR_AIRPORT (arr_air_code, arr_air_name)

Combined DEP_AIRPORT and ARR_AIRPORT into AIRPORT(airport_code, airport_name) - attribute synthesis:

FINAL 3NF

CLIENT (client_no, client_name)
BOOKING (booking_no, client_no)
BOOKING_LEG (booking_no, flight_no, dep_date)
FLIGHT_INSTANCE (flight_no, dep_date, arr_date)
FLIGHT (flight_no, dep_time, dep_air_code, arr_time, arr_air_code, flight_duration)
AIRPORT (airport_code, airport_name)

Full dependencies:
client_no → client_name
booking_no → client_no
flight_no, dep_date → arr_date
flight_no → dep_time, dep_air_code, arr_time, arr_air_code
airport_code → airport_name

Prior to building the logical model, so as to maintain relation name prefixes for the attributes the 3NF above will be renamed for the attributes in flight and flight_instance as follows:

CLIENT (client_no, client_name)
BOOKING (booking_no, client_no)
BOOKING_LEG (booking_no, flight_no, fi_dep_date)
FLIGHT_INSTANCE (flight_no, fi_dep_date, fi_arr_date)
FLIGHT (flight_no, flight_dep_time, flight_dep_air_code, flight_arr_time, flight_arr_air_code, flight_duration)
AIRPORT (airport_code, airport_name)

Important

After you have completed your current lab activities, at the end of each session remember to add, commit and push any changes you have made to the FIT GitLab server.

You need to get into the habit of establishing this as a standard FIT2094-FIT3171 workflow - Pull at the start of your working session, work on the activities you wish to/are able to complete during this session, add files (stage)/commit changes and then Push the changes back to the FIT GitLab server.
Remember you should also regularly use the Web UI (login to the web interface of the server) to check that your files are correctly being pushed.