COMP3311 22T1 Database Systems

Week 09
Relational Design Theory

Notation: in the relational schemas below, primary key attributes are underlined (e.g. pk), foreign key attributes are shown in italic font (e.g. *fk*) and primary key attributes that are also foreign keys are underlined and in italic font (e.g. *pk+fk*).

Example:

```
Student(id, name, degreeCode)
Degree(code, name, requirements)
Subject(code, name, syllabus)
Marks(studentId, subjectCode, teachingTerm, mark)
```

In their respective relations, the student id, the degree code and the subject code are primary keys. In the Student relation, the degree code is a foreign key. In the Marks relation, the three attributes student id, subject code and teaching term together form the primary key; the first two (student id and subject code) are also foreign keys.

1. Functional dependencies.

   a. What functional dependencies are implied if we know that a set of attributes *X* is a candidate key for a relation *R*?

   Answer:

   X functionally determines all of the other attributes (i.e. $X \rightarrow R\text{-}X$)

   b. What functional dependencies can we infer *do not hold* by inspection of the following relation?

A	B	C
a	1	x
b	2	y
c	1	z
d	2	x
a	1	y
b	2	z

   Answer:

   The tuples (a,1,x) and (a,1,y) ensure that $A \rightarrow C$ and $B \rightarrow C$ and $AB \rightarrow C$ do not hold.
   The tuples (a,1,x) and (d,2,x) ensure that $C \rightarrow B$ and $C \rightarrow A$ do not hold.
   The tuples (a,1,x) and (c,1,z) ensure that $B \rightarrow A$ and $B \rightarrow C$ do not hold.
   Note that $C \rightarrow A$ is not disproved by (a,1,x) and (a,1,y) or (b,2,y) and (b,2,z). For other combinations of *ABC* there is only one instance in the relation and so we cannot infer anything about dependency.

c. Suppose that we have a relation schema *R(A,B,C)* representing a relationship between two entity sets *E* and *F* with keys *A* and *B* respectively, and suppose that *R* has (at least) the functional dependencies $A \rightarrow B$ and $B \rightarrow A$. Explain what this tells us about the relationship between *E* and *F*.

Answer:

The $A \rightarrow B$ tells us that every *A* value in *R* has exactly one corresponding *B* value, and similarly $B \rightarrow A$ tells us that every *B* value has exactly one corresponding *A* value. In other words, the relationship must be 1:1.

2. Consider the relation *R(A,B,C,D,E,F,G)* and the set of functional dependencies $F = \{ A \rightarrow B, BC \rightarrow F, BD \rightarrow EG, AD \rightarrow C, D \rightarrow F, BEG \rightarrow FA \}$ compute the following:

a. A^+

Answer:

Derivation of A^+ ...
given {A} ... using A → B gives {A,B} ... no further attributes can be added
=> A^+ = {A,B}

b. $ACEG^+$

Answer:

Derivation of $ACEG^+$
given {A,C,E,G} ... using A → B gives {A,B,C,E,G} ... using BC → F gives {A,B,C,E,F,G} ... no more
$ACEG^+$ = {A,B,C,E,F,G}

c. BD^+

Answer:

Derivation of BD^+
given {B,D} ... using BD → EG gives {B,D,E,G} ... using BEG → FA gives {A,B,D,E,F,G} ... using AD → C gives {A,B,C,D,E,F,G} ... no more
BD^+ = {A,B,C,D,E,F,G}

3. Consider the relation *R(A,B,C,D,E)* and the set set of functional dependencies $F = \{ A \rightarrow B, BC \rightarrow E, ED \rightarrow A \}$

a. List all of the candidate keys for *R*.

Answer:

A candidate key is any set X, such that $X^+ = R$ and there is no Y subset of X such that $Y^+ = R$.

In this case, the candidate keys are CDE, ACD, BCD.

b. Is R in third normal form (3NF)?

Answer:

Yes, because the right hand sides of all dependencies (i.e. B, E, A) are parts of keys.

c. Is R in Boyce-Codd normal form (BCNF)?

Answer:

No, because none of the left hand sides (i.e. A, BC, ED) contains a key.

4. Consider a relation $R(A,B,C,D)$. For each of the following sets of functional dependencies, assuming that those are the only dependencies that hold for R, do the following:

   a. List all of the candidate keys for R.

   b. Show whether R is in Boyce-Codd normal form (BCNF)?

   c. Show whether R is in third normal form (3NF)?

   i. $C \rightarrow D$, $C \rightarrow A$, $B \rightarrow C$

   Answer:

   a. Candidate keys: B
   b. Not BCNF ... e.g. in $C \rightarrow A$, C does not contain a key
   c. Not 3NF ... e.g. in $C \rightarrow A$, C does not contain a key, A is not part of a key

   ii. $B \rightarrow C$, $D \rightarrow A$

   Answer:

   a. Candidate keys: BD
   b. Not 3NF ... neither right hand side is part of a key
   c. Not BCNF ... neither left hand side contains a key

   iii. $ABC \rightarrow D$, $D \rightarrow A$

   Answer:

   a. Candidate keys: ABC BCD

b. 3NF ... *ABC* → *D* is ok, and even *D* → *A* is ok, because *A* is a single attribute from the key
c. Not BCNF ... e.g. in *D* → *A*, *D* does not contain a key

iv. *A* → *B*, *BC* → *D*, *A* → *C*

Answer:

a. Candidate keys: *A*
b. Not 3NF ... e.g. in *BC* → *D*, *BC* does not contain a key and D is not part of a key
c. Not BCNF ... e.g. in *BC* → *D*, *BC* does not contain a key

v. *AB* → *C*, *AB* → *D*, *C* → *A*, *D* → *B*

Answer:

a. Candidate keys: *AB* *BC* *CD* *AD*
b. 3NF ... for *AB* case, first two fd's are ok, and the others are also ok because the RHS is a single attribute from the key
c. Not BCNF ... e.g. in *C* → *A*, *C* does not contain a key

vi. *A* → *BCD*

Answer:

a. Candidate keys: *A*
b. 3NF ... all left hand sides are superkeys
c. BCNF ... all left hand sides are superkeys

5. Specify the non-trivial functional dependencies for each of the relations in the following Teams-Players-Fans schema and then show whether the overall schema is in BCNF.

```
Team(name, captain)
Player(name, teamPlayedFor)
Fan(name, address)
TeamColours(teamName, colour)
FavouriteColours(fanName, colour)
FavouritePlayers(fanName, playerName)
FavouriteTeams(fanName, teamName)
```

Answer:

Functional dependencies:

- `Team ... name → captain`
- `Player ... name → teamPlayedFor`
- `Fan ... name → address`
- `TeamColours` ... no non-trivial *fd*s
- `FavouriteColours` ... no non-trivial *fd*s

- `FavouritePlayers` ... no non-trivial *fd*s
- `FavouriteTeams` ... no non-trivial *fd*s

For each relation, every non-trivial *fd* has a left hand side which is a super key ⇒ each relation is in BCNF and the whole schema is in BCNF.

6. Specify the non-trivial functional dependencies for each of the relations in the following Trucks-Shipments-Stores schema and then show whether the overall schema is in BCNF.

```
Warehouse(warehouse#, address)
Source(trip, warehouse)
Trip(trip#, date, truck)
Truck(truck#, maxvol, maxwt)
Shipment(shipment#, volume, weight, trip, store)
Store(store#, storename, address)
```

Answer:

Functional dependencies:

- `Warehouse ... warehouse# → address`
- `Source` ... no non-trivial *fd*s
- `Trip ... trip# → date,truck`
- `Truck ... truck# → maxvol,maxwt`
- `Shipment ... truck# → volume,weight,trip,store`
- `Store ... store# → storename,address`

For each relation, every non-trivial *fd* has a left hand side which is a super key ⇒ each relation is in BCNF and the whole schema is in BCNF.

This just goes to show that ER design generally leads to well-structured relational designs.

7. For each of the sets of dependencies in question 4:

i. if *R* is not already in 3NF, decompose it into a set of 3NF relations

ii. if *R* is not already in BCNF, decompose it into a set of BCNF relations

a. $C \rightarrow D, \quad C \rightarrow A, \quad B \rightarrow C$

Answer:

i. application of the 3NF algorithm decomposes *R* into three relations based on the dependencies ($R_1(CD)$, $R_2(CA)$, $R_3(BC)$); this decomposition leaves enough "connectivity" between the relations that no extra "candidate key" relation is needed to make them join-preserving (from now on, we denote a relation like $R_1(CD)$ simply by *CD*)

ii. the above 3NF decomposition is also in BCNF

b. $B \rightarrow C, \quad D \rightarrow A$

Answer:

i. application of the 3NF algorithm decomposes *R* into *BC, AD* (using the dependencies), but also requires the addition of a "linking" relation *BD* containing the candidate key

ii. this is the same decomposition that would be reached by following the BCNF algorithm, although it would proceed by first producing *AD, BCD* and then decomposing *BCD* further into *BC, BD*

c. $ABC \rightarrow D, \quad D \rightarrow A$

Answer:

i. *R* is already in 3NF

ii. applying the BCNF algorithm means decomposition to "fix" the violation caused by $D \rightarrow A$ giving *AD, BCD* (note that this does not preserve the dependency $ABC \rightarrow D$, and, in fact, it is not possible to find a BCNF decomposition which does preserve it)

d. $A \rightarrow B, \quad BC \rightarrow D, \quad A \rightarrow C$

Answer:

i. the standard algorithm produces *AB, BCD, AC*, which contains enough connectivity to not require a "linking" relation

ii. $BC \rightarrow D$ violates BCNF because *BC* does not contain a key, so we split *R* into *BCD, ABC*, and this is now BCNF

e. $AB \rightarrow C, \quad AB \rightarrow D, \quad C \rightarrow A, \quad D \rightarrow B$

Answer:

i. *R* is already in 3NF

ii. applying the standard BCNF decomposition algorithm requires us to "fix" the BCNF violations caused by $C \rightarrow A$ and $D \rightarrow B$; this could be achieved by decomposing *R* into *AC* and *BCD*, and then decomposing *BCD* into *BC* and *BD*; this does not preserve all dependencies (e.g. $AB \rightarrow C$ no longer applies).

f. $A \rightarrow BCD$

Answer:

i. *R* is already in 3NF

ii. *R* is already in BCNF

8. Consider (yet another) banking application that contains information about

accounts, branches and customers. Each account is held at a specific branch, but a customer may hold more than one account and an account may have more than one associated customer.

Consider an unnormalised relation containing all of the attributes that are relevant to this application:

- *acct#* - unique account indentifier
- *branch#* - unique branch identifier
- *tfn* - unique customer identifier (**t**ax **f**ile **n**umber)
- *kind* - type of account (savings, cheque, ...)
- *balance* - amount of money in account
- *city* - city where branch is located
- *name* - customer's name

i.e. consider the relation *R(acct#, branch#, tfn, kind, balance, city, name)*

Based on the above description:

a. Devise a suitable set of functional dependencies among these attributes.

Answer:

acct# → kind, balance, branch#
branch# → city
tfn → name

These all come from the semantics of the problem (e.g. each account has exactly one type and balance, and is held at a specific branch).

b. Using these functional dependencies, decompose *R* into a set of 3NF relations.

Answer:

Computing the minimal cover for the above FDs produces the same set of FDs. In other words, they were already a minimal cover.

In this case, the 3NF decomposition from applying the above three *fd*s produces a BCNF decomposition as well. This is the simplest and most natural decomposition of the problem.

```
Account(acct#, kind, balance, branch)
Branch(branch#, city)
Customer(tfn, name)
CustAcc(customer, account)
```

The first three relations come directly from the dependencies; the last relation comes from the final check for a relation with a candidate key for *R* in the 3NF algorithm.

c. State whether the new relations are also in BCNF.

Answer:

The first three tables are in BCNF, since they have single-attribute primary keys and no FDs that don't involve the primary key. The final table has two attributes, but both are part of the primary key, so it is also in BCNF. So, the schema is in BCNF.

9. Consider a schema representing projects within a company, containing the following information:

   - *pNum* - project's unique identifying number
   - *pName* - name of project
   - *eNum* - employee's unique identifying number
   - *eName* - name of employee
   - *jobClass* - type of job that employee has on this project
   - *payRate* - hourly rate, dependent on the kind of job being done
   - *hours* - total hours worked in this job by this employee

   This schema started out life as a large spreadsheet and now the company wants to put it into a database system.

   As a spreadsheet, its schema is: *R(pNum, pName, eNum, eName, jobClass, payRate, hours)*

   Based on the above description:

   a. Devise a suitable set of functional dependencies among these attributes.
   Answer:

   pNum → pName
   eNum → eName
   jobClass → payRate
   pNum,eNum → jobClass,payRate,hours

   The above implies that *pNum,eNum* is a key for this relation (these attributes determine all of the others).

   Based on semantics given in the descriptions *and* on some further assumptions, such as:

   - one employee can work on several projects
   - they may be doing a different job in each project

   b. Using these functional dependencies, decompose *R* into a set of BCNF relations.
   Answer:

Following the BCNF decomposition algorithm ...

- *pNum → pName* is a dependency on part of the key
  to fix: decompose to
  R1(pNum,eNum,eName,jobClass,payRate,hours) and
  R2(pNum,pName)
- *eNum → eName* is a dependency on part of the key
  to fix: decompose to *R1(pNum,eNum,jobClass,payRate,hours)*
  and *R2(pNum,pName)* and *R3(eNum,eName)*
- *jobClass → payRate* is a dependency on a non-key attribute
  to fix: decompose to *R1(pNum,eNum,jobClass,hours)* and
  R2(pNum,pName) and *R3(eNum,eName)* and
  R4(jobClass,payRate)

The relevant functional dependencies are now:

pNum → pName
eNum → eName
jobClass → payRate
pNum,eNum → jobClass,hours

With these dependencies there are no violations of BCNF, so the schema is now in BCNF, and we could rename the relations as:

```
Project(pNum, pName)
Employee(eNum, eName)
AwardRates(jobClass, payRate)
Assignment(pNum, eNum, jobClass, hours)
```

c. State whether the new relations are also in 3NF.
Answer:

The new schema is *not* in 3NF because we have lost the dependency: *pNum,eNum → payRate*

10. Real estate agents conduct visits to rental properties

- need to record which property, who went, when, results
- each property is assigned a unique code (P#, e.g. PG4)
- each staff member has a staff number (S#, e.g. SG43)
- staff members use company cars to conduct visits
- a visit occurs at a specific time on a given day
- notes are made on the state of the property after each visit

The company stores all of the associated data in a spreadsheet. Describe any functional dependencies that exist in this data. The table of sample data below below may give some ideas:

```
P#  | When        | Address    | Notes         | S#   | N
----+-------------+------------+---------------+------+-
PG4 | 03/06 15:15 | 55 High St | Bathroom leak | SG44 | P
```

```
PG1 | 04/06 11:10 | 47 High St | All ok        | SG44 | F
PG4 | 03/07 12:30 | 55 High St | All ok        | SG43 | D
PG1 | 05/07 15:00 | 47 High St | Broken window | SG44 | F
PG2 | 13/07 12:00 | 12 High St | All ok        | SG42 | F
PG1 | 10/08 09:00 | 47 High St | Window fixed  | SG42 | F
PG3 | 11/08 14:00 | 99 High St | All ok        | SG41 | J
PG4 | 13/08 10:00 | 55 High St | All ok        | SG44 | F
PG3 | 05/09 11:15 | 99 High St | Bathroom leak | SG42 | F
```

State assumptions used in determining the functional dependencies.

Answer:

- *P* → *A* ... the property code identifies a particular address
- *PW* → *N* ... notes are based on a particular visit to a property
- *PW* → *S* ... one staff member carries out each property visit
- *S* → *M* ... the staff number determines the staff member's name
- *S* → *C* ... each staff member uses a particular car (from table)

Other dependencies are possible (e.g. *M* → *S*), but they appear less plausible, given the semantics of the application.

11. Consider a company supplying temporary employees to hotels:

    - the company has contracts with different hotels
    - it may have several contracts with a given hotel
    - contracts are identified by a code (e.g. C12345)
    - staff work at different hotels as needed
    - staff have tax file #'s (TFN, e.g. T123)
    - hotels have Aus business #'s (ABN, e.g. H234)

    Describe any functional dependencies that exist in this data. The table of sample data below below may give some ideas:

```
Contract | TFN  | Name        | Hrs | ABN  | Hotel
---------+------+-------------+-----+------+-------------
C12345   | T311 | John Smith  |  12 | H765 | Four Seasons
C18765   | T255 | Brad Green  |  12 | H234 | Crown Plaza
C12345   | T311 | John Smith  |  12 | H765 | Four Seasons
C12345   | T255 | Brad Green  |  10 | H765 | Four Seasons
C14422   | T311 | John Smith  |   6 | H222 | Sheraton
C14422   | T888 | Will Smith  |   9 | H222 | Sheraton
C18477   | T123 | Clair Bell  |  15 | H222 | Sheraton
```

    State assumptions used in determining the functional dependencies.

    Answer:

    Describe any functional dependencies that exist in this data.

    - *C* → *AH* ... a contract is with a particular hotel
    - *A* → *H* ... each hotel has a unique ABN
    - *T* → *N* ... each employee has a unique TFN

- *CT → R* ... each employee works specified hours on a contract

Other potential dependencies that **do not** apply:

- *C → T* ... because many employees can work on a contract
- *H → A* ... two different hotels may have the same name (e.g. Hilton)

12. What functional dependencies exist in the following table:

A	B	C	D
1	a	6	x
2	b	7	y
3	c	7	z
4	d	6	x
5	a	6	y
6	b	7	z
7	c	7	x
8	d	6	y

How is this case different to the previous two?

Answer:

Since *A* has a different value in every tuple, it determines all other attributes:

A → B, A → C, A → D or A → BCD

Also, *A* in combination with any other subset of attributes determines all the others.

Similar reasoning could be applied to the combination *BCD*, since it's unique over all tuples.

It also appears that *B → C*, but not *C → B*.

This case is different to the previous two because there are no application semantics to draw on to determine reasonableness of choice of FDs. All you can do is look for counter-examples to eliminate certain potential dependencies. Example: since *(C=6,B=a)* and *(C=6,B=d)* we can eliminate *C → B*.

13. Compute a minimal cover for:

F = { B → A, D → A, AB → D }

Answer:

Steps in converting to a minimal cover:

1. put FDs into canonical form:
   B → A, D → A, AB → D is already in canonical form.

2. eliminate redundant attributes:
   The only possible redundant attributes are *A* or *B* in $AB \rightarrow D$.

   We can prove that *A* is redundant as follows:

   $B \rightarrow A \;\Rightarrow\; BB \rightarrow AB \;\Rightarrow\; B \rightarrow AB$

   $AB \rightarrow D + B \rightarrow AB \;\Rightarrow\; B \rightarrow D$

   Since we have $AB \rightarrow D$ and $B \rightarrow D$, *A* is redundant.

3. eliminate redundant dependencies:
   The above elimination leaves: $B \rightarrow A,\ D \rightarrow A,\ B \rightarrow D$

   But $D \rightarrow A,\ B \rightarrow D \;\Rightarrow\; B \rightarrow A$

So, the minimal cover is: $B \rightarrow D,\ D \rightarrow A$

14. Using the functional dependencies you produced in Q10, convert the real-estate inspection spreadsheet (single table), into a BCNF relational schema.

    Answer:

    FDs: $P \rightarrow A$, $PW \rightarrow NS$, $S \rightarrow MC$ (reduced from the 5 FDs given above)

    - We start from a schema: *PAWNSMC*, which has key *PW* (work it out from FDs).
    - The FD $S \rightarrow MC$ violates BCNF (FD with non-key on LHS).
    - To fix, we need to decompose into tables *PAWNS* and *SMC*.
    - Key for *PAWNS* is *PW*, and key for *SMC* is *S*.
    - Since all attributes in *SMC* are f-dependent on whole key, *SMC* is in BCNF.
    - The FD $P \rightarrow A$ violates BCNF in *PAWNS* (FD with partial key on LHS).
    - To fix, we need to decompose into tables *PWNS* and *PA*.
    - Key for *PAWNS* is *PW*, and key for *SMC* is *S*.
    - In both tables, all attributes are f-depdendent on whole key ⇒ BCNF.
    - Final schema (with keys boldened): ***PW***NS, ***P***A, ***S***MC

15. Consider the schema *R* and set of fds *F*

    $R = ABCDEFGH$
    $F = \{ ABH \rightarrow C,\ A \rightarrow DE,\ BGH \rightarrow F,\ F \rightarrow ADH,\ BH \rightarrow GE \}$

    Produce a BCNF decomposition of *R*.

    Answer:

    This is just one of many possible decompositions. Variations occur because of choice of candidate key (although not in this example) and

choice of non-BCNF FD to resolve at each step.

- We start from a schema: *ABCDEFGH*, with key *BH* (work it out from FDs).
- The FD $A \rightarrow DE$ violates BCNF (FD with non key on LHS).
- To fix, we need to decompose into tables: *ADE* and *ABCFGH*.
- FDs for *ADE* are $\{ A \rightarrow DE \}$, therefore key is *A*, therefore BCNF.
- FDs for *ABCFGH* are $\{ ABH \rightarrow C,\ BGH \rightarrow F,\ F \rightarrow AH,\ BH \rightarrow G \}$
- Key for *ABCFGH* is *BH*, and FD $F \rightarrow AH$ violates BCNF (FD with non key on LHS).
- To fix, we need to dcompose into tables: *AFH* and *BCFG*.
- FDs for *ADE* are $\{ F \rightarrow AH \}$, therefore key is *F*, therefore BCNF.
- FDs for *BCFG* are $\{ \}$, so key is *BCFG* and table is BCNF.
- Final schema (with keys boldened): ***A***DE, ***F***AH, ***BCFG***

16. Using the functional dependencies you produced in Q10, convert the real-estate inspection spreadsheet (single table), into a 3NF relational schema.

    Answer:

    FDs: $P \rightarrow A,\ PW \rightarrow N,\ PW \rightarrow S,\ S \rightarrow M,\ S \rightarrow C$

    This looks like a minimal cover; no redunant attributes; no redundant FDs.

    3NF is constructed directly from the minimal cover, after combinining dependencies with a common right hand side.

    $F_c = P \rightarrow A,\ PW \rightarrow NS,\ S \rightarrow MC$

    Gives the following tables (with table keys in **bold**):

    PA ***PW***NS ***S***MC

    Since *PW* is a key for the whole schema, and since a table contains this key, the 3NF decompostion is complete:

    3NF = ***P***A ***PW***NS ***S***MC

17. Consider the schema *R* and set of fds *F*

    $R = ABCDEFGH$
    $F = \{ ABH \rightarrow C,\ A \rightarrow D,\ C \rightarrow E,\ BGH \rightarrow F,\ F \rightarrow AD,\ E \rightarrow F,\ BH \rightarrow E \}$
    $F_c = \{ BH \rightarrow C,\ A \rightarrow D,\ C \rightarrow E,\ F \rightarrow A,\ E \rightarrow F,\ BH \rightarrow E \}$

    Produce a 3NF decomposition of *R*.

    Answer:

    3NF is constructed directly from the minimal cover, after combinining dependencies with a common right hand side where possible.

    $F_c = BH \rightarrow CE,\ A \rightarrow D,\ C \rightarrow E,\ F \rightarrow A,\ E \rightarrow F$

Gives the following tables (with table keys in **bold**):

BH*CE* ***A****D* ***C****E* ***F****A* ***E****F*

A key for *R* is *BHG*; *G* must be included because it appears in no functional dependency.

Since no table contains the whole key for *R*, we must add a table containing just the key, giving:

3NF = ***BH****CE* ***A****D* ***C****E* ***F****A* ***E****F* ***BGH***