Third Normal Form Proofs

Player

Are all attributes atomic? Yes. Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Id 🡪 id, first\_name, last\_name, number, team\_name, position, image\_url

Nothing on the table is uniquely identified by:

first\_name (could have the same first name as another player)

last\_name (could have the same last name as another player)

number (could have the same number as another player)

team\_name (players can and will be on the same team)

position (players will have the same position as other players)

image\_url (not likely, but possible that two players share an image, or more than one

player could have a null image url)

Therefore, the only dependency on the player table is the primary key created by the id

attribute

Fc:

Id 🡪 first\_name, last\_name, number, team\_name, position, image\_url

Therefore 3NF:

Id, first\_name, last\_name, number, team\_name, position, image\_url (Q.E.D)

Users

Are all attributes atomic? Yes (will never need to parse first\_name and last\_name for a login).

Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Id 🡪 id, email, full\_name, password

email 🡪 id, email, full\_name, password

Nothing on the table is uniquely identified by:

full\_name (unlikely, but possible that two users could have the same full name)

password (two users could have the same password)

Therefore, the only dependencies on the user table are created by primary and candidate keys.

Fc:

Id 🡪 email

Email 🡪 full\_name, password

Therefore 3NF:

Id, email // email, full\_name, password (Q.E.D)

HOWEVER:

Id was only added to the table in order to act as a useful primary key (in a full-fledged application, emails could be added/removed/changed too many times to act as a useful primary key). Therefore, id isn’t relevant for the integrity of the table, and it wouldn’t make sense to split up the table into 3NF in this case. Id acts in the same capacity as email. Therefore, we chose to keep the single, unified table:

Id, email full\_name, password

Card

Are all attributes atomic? Yes (just ID’s).

Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Card\_id 🡪 card\_id, player\_id, current\_owner\_id

Nothing on the table is uniquely identified by:

player\_id (multiple copies of cards can exist)

current\_owner\_id (this value can change regularly, so nothing will be identifiable by it)

Therefore, the only dependency on the user table is created by a primary key.

Fc:

Card\_id 🡪 player\_id, current\_owner\_id

Therefore 3NF:

Card\_id, card\_id, player\_id, current\_owner\_id (Q.E.D)

Team

Are all attributes atomic? Yes (team names are indistinguishable from their location, and divisions names are atomic elements).

Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Name 🡪 name, division

Nothing on the table is uniquely identified by:

Division (multiple teams can come from the same division)

Therefore, the only dependency on the team table is created by a primary key.

Fc:

Name 🡪 division

Therefore 3NF:

Name, division (Q.E.D)

Packs

Are all attributes atomic? Yes. Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Pack\_id 🡪 pack\_id, name, points

Name 🡪 name, pack\_id, points

Nothing on the table is uniquely identified by:

Points (multiple packs can have the same point cost)

Therefore, the only dependencies on the packs table are created by primary and candidate keys.

Fc:

Pack\_id 🡪 name

Name -> points

Therefore 3NF:

Pack\_id, name // name, points (Q.E.D)

Pack\_id was only added to the table in order to act as a useful primary key (in a full-fledged application, pack names could be added/removed/changed too many times to act as a useful primary key). Therefore, pack\_id isn’t relevant for the integrity of the table, and it wouldn’t make sense to split up the table into 3NF in this case. Id acts in the same capacity as email. Therefore, we chose to keep the single, unified table:

Pack\_id, name, points

Packs\_players

Are all attributes atomic? Yes. Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Pack\_id 🡪 pack\_id, player\_id

Nothing on the table is uniquely identified by:

Player\_id (one pack will certainly have multiple unique players, and although unlikely, it is possible that re-issues of specific player cards could occur in multiple packs)

Therefore, the only dependency on the packs\_players table is created by a primary key.

Fc:

Pack\_id 🡪 player\_id

Therefore 3NF:

Pack\_id, player\_id (Q.E.D)

Division

Are all attributes atomic? Yes (name is not divisible – each name is its own entity).

` Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Name 🡪 name, conference\_name

Nothing on the table is uniquely identified by:

Conference\_name (multiple divisions are members of one conference)

Therefore, the only dependency on the division table is created by a primary key.

Fc:

Name 🡪 conference\_name

Therefore 3NF:

Name, conference\_name (Q.E.D)

Conference

Are all attributes atomic? Yes. Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Name 🡪 name

Therefore, the only dependency on the conference table is created by a primary key.

Fc:

Name 🡪 name (trivial)

Therefore 3NF:

Name (Q.E.D)

Trade

Are all attributes atomic? Yes (won’t need to parse dates). Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Trade\_id 🡪trade\_id, prop\_id, accepter\_id, proposed\_at, accepted\_at, confirmed\_at

Nothing on the table is uniquely identified by:

Prop\_id (a user could propose trades to multiple users simultaneously)

Accepter\_id (a user could have multiple trades proposed to them simultaneously)

Proposed\_at (could be null, different trades could be proposed simultaneously)

Accepted\_at (could be null, different trades could be accepted simultaneously)

Confirmed\_at (could be null, different trades could be confirmed simultaneously)

Therefore, the only dependency on the packs\_players table is created by a primary key.

Fc:

Trade\_id 🡪 prop\_id, accepter\_id, proposed\_at, accepted\_at, confirmed\_at

Therefore 3NF:

Trade\_id, prop\_id, accepter\_id, proposed\_at, accepted\_at, confirmed\_at (Q.E.D)

Trade\_cards

Are all attributes atomic? Yes. Therefore, in 1NF and can proceed.

Functional Dependencies (F+):

Id ->id, trade\_id, card\_id, from\_id, desired

Nothing on the table is uniquely identified by:

Trade\_id (each trade process can and probably will have more than one associated card

being traded, which will involve more than one user)

card\_id (the same card could be traded back and forth multiple times between the

same users)

from\_id (the same card could be traded back and forth multiple times between the

same users, or the same user could be trading different cards)

desired (any card, in any trade, from any user, could either be desired or offered)

Therefore, the only dependency on the packs\_players table is created by a primary key.

Fc:

Id 🡪 trade\_id, card\_id, from\_id, desired

Therefore 3NF:

Id, trade\_id, card\_id, from\_id, desired (Q.E.D)