Referentiel integritet:

Hvis man har en fremmed-nøgle (FK) der refererer til en primær-nøgle (PK) skal alle FK-værdier findes som PK-værdier (i.e. FK'ere skal referere til eksisterende PK'ere).

Hvis man sletter rækker med PK'ere kan der derfor opstå problemer. Disse problemer kan håndteres på en af følgende måder:

1. RESTRICT: Checker om det vil give et problem at slette, og hvis det gør, så bliver det ikke gjort (i.e. man får en "exception"). Man kan så løse de problemer der måtte være, og så dernæst slette.

2. CASCADE: Sletter rækker med tilhørende FK (cascade-effekten gentages om nødvendigt også for disse, osv.). En automatisk proces, men til gengæld har vi ikke rigtig styr på hvad der sker.

3. SET TO NULL: Man sætter FK til null, for at indikere manglende reference.

Normalformer:

Man normaliserer fordi:

- Det fjerne redundans (i.e. at det samme står flere steder)

- Anomalier:

- Update: ændrer et sted men ikke alle steder

- Insert: indsætter allerede eksisterende data, men "staver" forkert

- Det minimere brug af null-værdier

- tvetydig semantik (i.e. uklar betydning)

("hvad betyder semantik, og hvordan staver man til syntaks?")

- Man vil undgå tab af information

- at man sletter en ting, kan bevirke at andre informationer kan gå tabt

(at slette en studerende kan f.eks. slette oplysninger om et fag)

1NF:

- Alle attributter må kun have én værdi ("ingen repeterende felter")

= Man laver \_ikke\_ flere tabeller (det gør man ellers ifm. 2NF og 3NF)

= Man laver flere rækker (én for hver af værdierne i attributterne med flere værdier)

+ Det skaber mere redundans!

2NF:

- 1NF + Alle ikke-PK-attributter skal være \_fuldt\_ funktionelt afhængige (FD) af PK. Dvs.

at ved en sammensat PK er en attribut ikke kun afhængige af dele af PK.

= Man finder først en PK for hver tabel, ved at se på FD imellem attributterne.

= \_Fuld\_ afhængig er kun et problem hvis man har en \_sammensat\_ PK

= Man deler tabeller i to

= i.e. laver en ny tabel med de attributter, der ikke er \_fuldt\_ FD af PK, med

de dele af PK, som de er (fuldt) FD af (som bliver PK i den nye tabel).

= den oprindelige tabel mister disse attributter, dog beholder den PK fra den nye tabel som FK.

+ Fjerner normalt redundans i den gamle tabel (i.e. den nye får færre rækker)

3NF:

- 2NF + Ingen transitive afhængigheder (i.e. ingen FD mellem ikke-PK-attributter)

= Man deler tabellen i to

= i.e. laver en ny tabel med de attributter, der er FD af ikke-PK-attribut,

med den attribut som de er FD af (som bliver PK i den nye tabel).

= den oprindelige tabel mister disse attributter, dog beholder den PK fra den nye tabel som FK.

+ Fjerner normalt redundans i den gamle tabel (i.e. den nye får færre rækker)

= Postnummer-eksempel er et velkendt eksempel på 3NF