# LærerVikaren 01

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf4:** anvende centrale metoder og teknikker til at realisere modeller i et databasesystem […] |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Multistrukturel] Du kan beskrive flere kendetegn ved **database-design** |
| **Din forberedelse** | Programmeringssporet:   * Genopfrisk LærerVikar-casen samt systemudviklingsopgaverne fra uge 8:   + Case - LærerVikaren AS.docx   + 08Ma - LærerVikaren I.docx   + 08Ti - LærerVikaren II.docx   Find frem til de artefakter, du tidligere oprettede, og gennemlæs dem; du skal inddrage dem i denne opgave, specielt domænemodellen.   * Genopfrisk efter behov følgende termer fra opgaver i uge 6 og 8 samt udleveret materiale (*Kom i gang med databaser.FORELØBIG.v3.docx*, *WhyUseADatabase.pdf*, *TheRelationalModel.pdf* og *Normalization.pdf*):   + Relation, relationsskema, relationel databaseskema   + Primær nøgle, fremmednøgle, kandidatnøgle, surrogatnøgle   + Funktionel afhængighed samt determinant og velformet relation   + Anomalier, normaliseringsprocessen og de forskellige normalformer * Gense videoen: [Normalization: Operations and Anomalies](https://www.youtube.com/watch?v=mSGoFrOPBJY&list=PLyp-1WQY32mN6U6PUq9c977X9vyHxtohD&index=1) |

Opgaven er den første i en serie af opgaver, hvor du skal arbejde videre med database­temaet primært med udgangspunkt i LærerVikar-casen, som du allerede har arbejdet med i systemudvikling. Denne og de følgende 2 opgaver giver dig mulighed for endnu en gang at fordybe og øve dig yderligere i **de 3 faser i databaseudvikling** (den spirale tilgang).

Denne opgave har fokus på første fase: **database-design**:

* **Fase A: Database-design (fra domænemodel til UML-databasemodel via normaliserede relationsskemaer)**
* Fase B: Database-implementering (fra UML-databasemodel til konkret database i et DBMS via SQL DDL)
* Fase C: Database-tilgang via C# og .NET

Alle opgaver er koblet sammen, så du løser alle øvelser i denne opgave med en iterativ tilgang, før du går videre med den næste.

# Øvelse 1: Terminologi

Del teamet op i to mindre grupper, og brug Ordet rundt til endnu engang at reflektere over begreberne ”funktionel afhængighed (en: functional dependency)”, ”determinant”, ”velformet (en: well formed)”, ”normalisering”, ”normalformer”, ”UML-databasemodel” og ”referentiel integritetsrestrik­tion (en: referential integrity constraint)”.  
Sørg for, at alle får mulighed for at tale.

*Tidsramme: 15 minutter*

# Øvelse 2: Domænemodel og use cases for LærerVikaren

I denne øvelse har du fokus på det første trin i fase A af database-udviklingsprocessen[[1]](#footnote-1), dvs. domæne­modellen og primært udvalgte use cases.

*Du har allerede i dit team udarbejdet domænemodel og use cases for LærerVikaren i systemudviklings­opgaver, så det er dem, du skal tage udgangspunkt i.*

Når du arbejder iterativt, så undgå at designe hele den relationelle database for alle domæneklasser samt alle use cases på én gang (vandfaldstilgang). I stedet start i ”et hjørne”, tag udgangspunkt i relevante domæneklasser for én use case, og gennemfør alle tre faser (database-design, database-implementering og database-tilgang via C#) for dette udvalg i denne iteration. Nye use cases kan altid tilføjes senere efter behov. Sørg for, at de use cases, der vælges, har en vis sammenhæng og central værdi for systemet. Det vil typisk være use cases udvalgt sammen med Product Owner, som driver disse iterationer.

Udfør følgende:

* Udvælg de domæneklasser i domænemodellen samt tilhørende use case, du gerne vil se på i den første iteration. Et godt råd er først at tage udgangspunkt i domæneklasser i domæne­modellen, som har få afhængigheder, og i senere iterationer domæneklasser, som er afhængige af og bygger på disse**. Vælg meget gerne to domæneklasser, som har en 1:M-association mellem dem.**

I følgende øvelser antages, at dataudviklingen gennemføres iterativt.

*Tidsramme: ca. 15 minutter (brug ikke for lang tid)*

# Øvelse 3: Gennemførelse af fase A – Database-design

Den første fase af databaseudvikling omfatter et antal designtrin, som i de følgende øvelser er gjort meget tyde­lige for at tydeliggøre både terminologien og de enkelte trin. Når man er lidt mere øvet, slår man oftest flere trin sammen.

**Benyt parprogrammering**, hvor intet andet er angivet.

## Øvelse 3.1: Relationer

Del teamet op i to mindre grupper, og brug **Ordet rundt** til at afklare følgende spørgsmål:

* Hvilke fælles karakteristika skal en tabel have for, at det kan kaldes en relation?
* Hvilken betegnelse anvender man for en relation, der som minimum opfylder disse karakteristika?

Del jeres svar i hele teamet.

*Tidsramme: ca. 15 minutter*

## Øvelse 3.2: Relationsskema (en: relation schema)

Med udgangspunkt i de i øvelse 2 valgte domæneklasser for LærerVikar-casen, udfør nu (trin 2 i fase A):

* For hver klasse i domænemodellen (iterativt udvalgt):
  + Beskriv domæneklassen med et relationsskema, dvs. relationens navn (senere tabellens navn) samt dens kolonner (attributter). Benyt den tekstuelle notation, du har lært for et relationsskema.

**I første omgang behøver du ikke overveje nøgler og sammenhængene mellem klasserne.**

## Øvelse 3.3: Relationelt databaseskema (en: relational database schema)

Et relationelt databaseskema er en samling af relationsskemaer, hvor også sammenhængen mellem relationerne er repræsenteret med angivelse af primær- og fremmednøgler i relationsskemaerne svarende til associationerne mellem domænemodellens klasser.

Udfør følgende (trin 3 i fase A):

* Omdan nu alle udvalgte relationsskemaer i forrige øvelse til et relationelt databaseskema, hvor associationerne (1-M, M-M, 1-1) samt generaliseringer mellem klasser i domænemodellen bliver repræsenterer via primær- og fremmednøgler. Benyt igen samme tekstuelle notation  
  *Gør det enkelt: vælg to domæneklasser forbundet med en 1:M-association.*
* Beskriv referentielle integritetsrestriktioner for alle relationer, hvor det er relevant (nøgler mm.)

## Øvelse 3.4: Normaliseringsprocessen

Du har nu etableret et relationelt databaseskema for (en del af) LærerVikaren med et antal udvalgte sammenhæn­gende relationsskemaer.

* Tjek om alle relationsskemaer som minimum er på første normalform (1NF). Kan du gøre brug af de begrænsede data, som er beskrevet i casen?

Da relationer i 1NF sagtens kan indeholde alvorlige problemer (anomalier), er du ikke færdig endnu med dit databasedesign. Du skal nu gennemføre en normaliseringsproces, så alle relationsskemaerne i dit relationelle databaseskema overholder Boyce-Codd normalform (BCNF), også betegnet 3.5NF.

For hver relation i dit relationelle databaseskema (se også s. 95 i *Normalization.pdf*), udfør da følgende (stadig trin 3 i fase A):

1. Identificér alle kandidatnøgler i relationen
2. Identificér alle funktionelle afhængigheder i relationen
3. Undersøg determinanterne i de funktionelle afhængigheder. Hvis hvilken som helst determinant ikke er en kandidatnøgle, da er relationen ikke velformet.   
   I dette tilfælde:
   1. Placér den funktionelle afhængighed’s kolonner i en ny relation for sig selv.
   2. Gør den funktionelle afhængighed’s determinant til en primærnøgle i den nye relation
   3. Efterlad en kopi af determinanten i den oprindelige relation som en fremmednøgle
   4. Opret en referentiel integritetsrestriktion (en: referential integrity constraint) mellem den oprindelige relation og den nye relation.
4. Gentag trin 3 så mange gange, det er nødvendigt, indtil hver determinant i hver relation er en kandidatnøgle

Undersøg til sidst trin for trin, om hver relation overholder 1NF, 2NF, 3NF og BCNF (3.5NF). Se f.eks. s. 104-105 i *Normalization.pdf*.

# Øvelse 4: Etablering af UML-databasemodel

Med udgangspunkt i det normaliserede relationelle databaseskema fra øvelse 3, skal du nu afslutte design­fasen (dvs. trin 4 i fase A) med at etablere en UML-databasemodel.

Dette involverer følgende:

1. Brug af et DCD med fokus på de anførte datatyper for domæneklassernes attributter i Model-laget.
2. Etablering en UML-databasemodel, der samler alle nødvendige informationer i én model, for efterfølgende at implementere en database (fase B)

## Øvelse 4.1: Brug af DCD

I en databasesammenhæng bidrager et DCD primært med datatyper til domæneklassernes attributter[[2]](#footnote-2), idet kendskab til SQL-datatyper er nødvendig for en tabels kolonner i et R-DBMS.

Med udgangspunkt i domænemodellen samt dine arkitekturovervejelser, udfør følgende:

* Etabler et DCD med en lagdelt MVVM-arkitektur inklusiv et Persistens-lag med Repository-klasser.
* Videndel DCD’et med de andre i teamet, så du sikrer dig, at alt er med.

Du behøver kun medtage klasser, der relaterer til den use case, du har udvalgt i din iterative tilgang.

## Øvelse 4.2: Etablering af UML-databasemodel

Med udgangspunkt i domænemodellen, DCD’et og dit BCNF-databasedesign fra øvelse 3, udarbejd nu følgende:

* Tegn en UML-databasemodel for din database. Sørg for at anføre:
  + Primærnøgler og fremmednøgler
  + Datatyper for hver attribut (husk det er SQL-datatyper)
  + Sammenhænge mellem tabeller
  + Navngivne constraints
* Videndel din UML-databasemodel med de andre i teamet, så du sikrer dig, at alt er med.

# Øvelse 5: Iterationer

Er du blevet færdig med den udvalgte iteration, da udfør ovenstående ’fase A’-øvelser for en ny use case og tilhørende domæneklasser. Gentag dette, så længe du har tid i dag.

1. Faser og trin refererer til ”*Kom godt i gang med databaser”* - dokumentet i seneste version. [↑](#footnote-ref-1)
2. I sjældne tilfælde, afhængig af den konkrete situation, kan der også være behov for at gemme data for en eller flere af DCD’ets softwareklasser (dvs. andre klasser i DCD’et end domæneklasser); det kunne f.eks. i forhold til gemme en brugers aktiviteter i systemet (en log). Dette vil ikke være tilfældet i LærerVikaren. [↑](#footnote-ref-2)