**Hovedemner til Pages**

**Intro**

* Forståelse om og anvendelse af data i brug og ved opbevaring i en database
* Implementering og vedligeholdelse af relationelle databaser
* Indsætning af data på en sikker måde

**Database Designfaser & Målsætninger**

Databasedesignfaserne er processen hvor man skaber rammerne for en database samt hvordan den er struktureret. Vi har 6 forskellige faser, som hver handler omkring sit eget område indenfor databaser. Det er nogle faser, som man følger, som hjælper en med at sikre et godt design til en database.

Designprocessen starter ud med definition af mål med database. Derefter kommer man videre til opbevaring og brug af data i den givne database. Når man er færdig med det går man videre til analysering og organisering, som omhandler undersøgelse af krav for den givne database. Har man de ting på plads kan man bevæge sig videre til implementering af databasen, som selvfølgelig handler om, at den givne database bliver implementeret og igangsat. Processen sluttes af med, at man validerer/kontrollerer og tester sin database.

**Designfaser:**

* Feasibility Study
  + Formål klart defineret
* Requirement collection and analysis
  + Hvordan opbevares og bruges data i databasen?
* Prototyping & Design
  + Analyse og organisering
* Implementation
  + Implementering af database
* Validation and testing
  + Validering og test
* Operation
  + Administration/Vedligeholdelse

**Målsætninger**

* Effektivitet
  + Ydeevne og resourceforbrug
* Integritet
  + Nøjagtig data
* Privathed
  + Kontrolleret adgang
* Sikkerhed
  + Sikre mod indtrængen og hardware/software fejl
* Implementering
  + Let overgang fra konceptuel til fysisk design
* Fleksibilitet
  + Skal kunne håndtere ændringer

**Målsætning**

Målsætninger er vigtig for en database samt databasens formål. Generelt handler effektiviteten af en database om, at man gerne vil sikre at databasen er i stand til at behandle store mængder af data hurtigt. Integriteten handler om at det er en akkurat data vi har i databasen. Privathed kommer i spil, da man gerne vil beskytte følsomme data imod negativ indtrængen. Sikkerheden generelt er en vigtig målsætning eftersom det også sikrer imod diverse hackerangreb. Implementering handler om at igangsætte og aktivere sin database så man sikrer at den fungerer som den skal. Fleksibilitet handler om at den kan ændres og tilpasses alt efter behov. Til sammen udgør disse målsætninger en god og funktionel database, hvis de er brugt korrekt.

**Normalisering og Redundans**

**Normalisering**

Normalisering er en designprocess, hvor man strukturerer sin data på en måde, som minimerer redundans og dermed hjælper med at sikre integritet i data. Hvis en database ikke er normaliseret kan det skabe problemer ved opdatering og vedligeholdelse af databasen. Ved normalisering forstås (atomic values). Der er forskellige niveauer af normalisering som gøres brug af i en database, afhængig af kritikaliteten af data. Formålet med at normalisere er også, at man gerne vil gøre databasen effektiv og sørge for korrekt organisering af data. Tit handler det omkring opdeling af tabeller til mindre tabeller. Første normalform(1NF) går ud på, at alle værdier i ens tabel skal være “atomic values”, hvilket vil sige, at de skal være enkeltstående og umulige at dele op. Anden normalform(2NF) handler om, at hvis en tabel har en sammensat nøgleværdi, så skal de resterende værdier være afhængige af den. 3. Normalform(3NF) går ud på at der ikke må være værdier uden for den primære nøgle, men som samtidig er afhængige indbyrdes.

**Redundans**

* Redundans er når samme information er lagret flere forskellige steder
* Redundant information kan fjernes uden selve informationen forsvinder

Når man arbejder med databaser, så vil størrelsen på databaser være forskellige. Nogle databaser vil have så meget data, at det faktisk har en betydning på serveren. Det er hermed vigtigt, at lave sin database på en måde som ikke gør den ineffektiv og langsom for serveren at håndtere.

Det er her, hvor termet redundans kommer ind. Redundans er når den samme data omgår flere gange i en database. Det vil hermed være vigtigt at sammensætte rækkerne i tabeller med kun den information som er nødvendigt. Efter de nævnte normalformer fra før bruges, så vil vi eliminere redundans og samtidig gør databasen nemmere at aflæse.

Desuden så vil eliminering af redundans også sikre at databasen kører optimalt og konsistent

**Brugerkonti og Privilegier**

Brugerkonti og privilegier er vigtige emner inden for databasedesign, da de hjælper med at beskytte databasen og dens data mod uautoriseret adgang og misbrug. Det er ikke en god ide at give en normal bruger for høje rettighed. Man kigger på hvilke handlinger der er påkrævede for brugeren at kunne udføre og tilretter adgangsniveauet. For eksempel bør en normal bruger ikke kunne udføre arbejde på selve databasen men begrænses til kun at kunne arbejde i de tabeller der er relevante.

**Sikkerhedskopiering og Restore**

Sikkerhedskopiering og restore er afgørende elementer i enhver databaseadministratorers opgaver, da de hjælper med at beskytte databasen mod datatab og genoprette den i tilfælde af en nødsituation. Datatab er en af det største udfordringer man kan støde på ved brugen af en database. Ved designet af databasen bør man sætte sig ned og lave en risikovurdering

* Sikkerhedskopiering
  + Bør automatiseres
  + Vægtes hvordan, hvornår og hvortil sikkerhedkopiering skal foretages

* Restore
  + Skal testes

**Views, stored procedures, stored functions og triggers**

Views, stored procedures, stored functions og triggers er forskellige typer af databaseobjekter, der kan forbedre ydeevnen og funktionaliteten i en database og hjælpe med at automatisere nogle af de almindelige opgaver

* Views
  + Views bruges til at prædefinere gængse søgninger eller for at give en bruger adgang til data i en tabel som de ikke har direkte adgang til