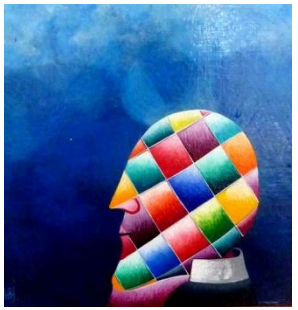
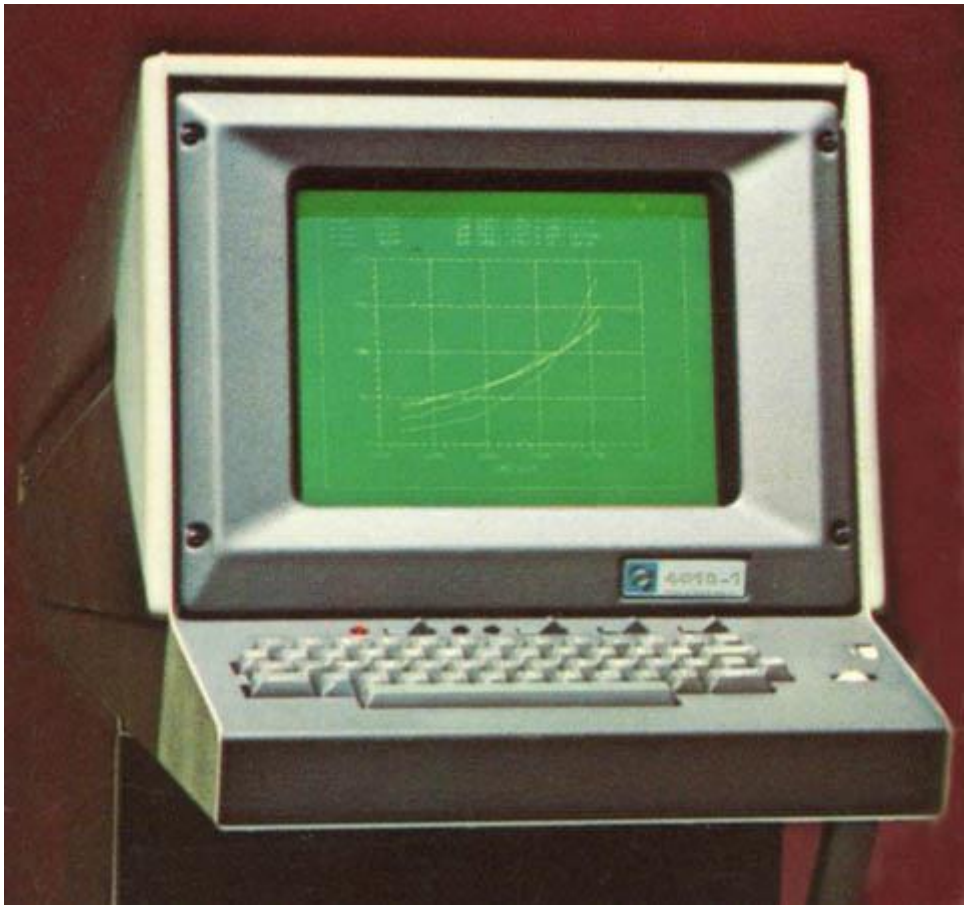
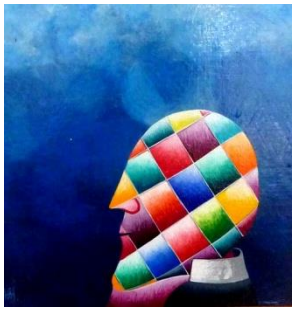
”Spatial is not special”

Bo Victor Thomsen
Civilingeniør

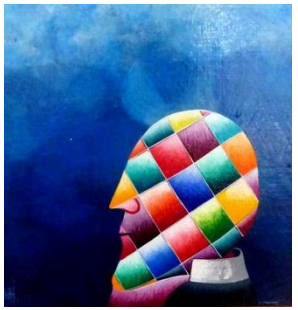


Lidt historie….



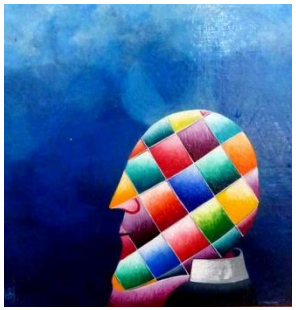


Udvikling - mængder af data => Hvorfor databaser



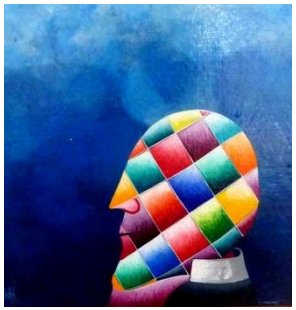
*”**S**patial **I**s **N**ot **S**pecial”*
Spatial data processing simply is a branch in computing between many others: and spatial data are too strategic and pervasive to be confined to niche applications

(Alessandro Furieri –
- main developer of the spatialite dataformat)



Fordele ved brug af en database

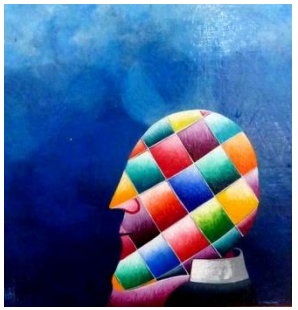
- Spatiale data i samme database som øvrige data.
- Kvalitetssikring
- Data tilgås på en standardiseret metode via ODBC og SQL.
- I SQL kan man specificere og opbygge vilkårligt komplekse forespørgsler
- SQL indeholder geografiske operatorer til spatiale analyser direkte uden brug af GIS
- Forespørgsler kan ”gemmes” i database som et ”view”.



Fordele ved brug af en database (2)

Dataadgang styres af DBMS programmet og ikke af GIS
Mulighed for ”intelligens” i tilgangen til data, f.eks:

- Finmasket styring af brugerettigheder
- Finmasket datavalidering før data oprettes eller opdateres i databasen
- For- eller efterbehandling af data ved oprettelse/opdatering/sletning i databasen (SQL Triggers).



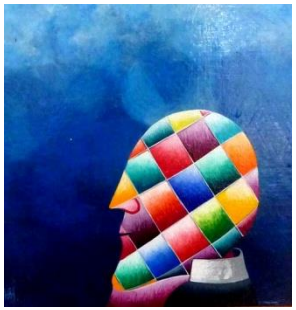
Trigger funktioner - eksempler

- Implementere historik, som fungerer uafhængig hvilket bruger program, data kommer fra.

- Udføre meget komplekse kontrolfunktioner før data indlægges i databasen. F.eks udføre multiple kontrol lookups i andre tabeller kombineret med a'la posteringer i et dobbelt bogholderi.

- Tilføje data til en post, f.eks UUID data eller administrative informationer såsom brugerid, rettelsestidspunkt automatisk.

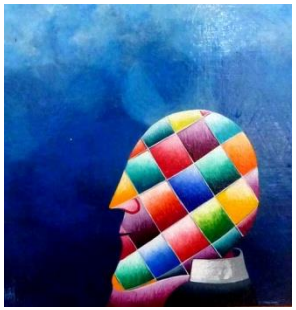
- Bearbejde data efter vilkårlig komplekse regler, før de lægges permanent ind i databasen.



Fordele ved brug af en database (3)

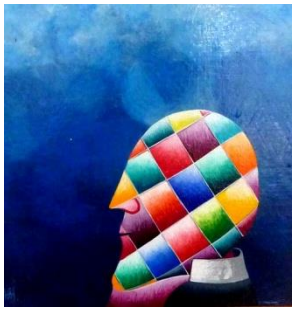
- Multiredigering – Et databasesystem er fra starten beregnet til at mange brugere kan redigere i samme datatabel på samme tidspunkt. Dette kan kun med vanskelighed lade sige gøre ved redigering i filbaserede data.

- Robusthed over for hardware fejl.
  De forskellige databasesystemer er udviklet kontinuert over en horisont på op til 40 år. Der har været en seriøs indsats med at gøre DBMS systemerne robuste over for hardwarefejl og utilsigtede nedlukninger. Dette gør brugen af databaser væsentligt mere robust end ved mange filbaserede systemer som shape- og tab-filer



Ulemper ved databaser

- Omkostninger til anskaffelse
- Større administration, specielt ved brug af mange databaser kørende under samme databasesystem
- Dårligere performance
- Mindre smidighed ved ændring af datastruktur.
- Maksimal udnyttelse af databasen kræver kendskab til SQL
- Slutbruger opfattelse af ”besværlighed”



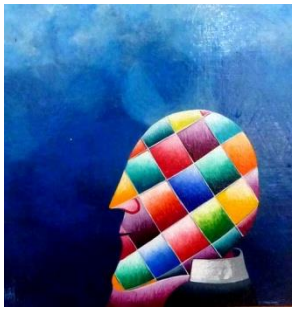
Hvilket databasesystem ?

Følgende databasesystemer understøtter spatiale data:

•MS-SQLServer 2008R2 +, Oracle 8+ (med Spatial eller Locator add-on), Postgres/Postgis, DB2. Der findes andre databasesystemer med understøttelse af spatiale data end disse, men de nævnte systemer dækker vel en 95 % af markedet.

•Hvis man skal ud og anskaffe et nyt databasesystem:
Er organisationen en ”Microsoft-shop” skal man kigge grundigt på MS-SQLServer (I et sådant miljø vil valg af databasesystem under alle omstændigheder blive målt og vejet mod MS-SQLServer. Der er sandsynligvis mange andre tekniske hensyn at overveje end blot det at skulle opbevare spatiale data).

•Er man mere frit stillet kan jeg personligt klart anbefale PostgreSQL/PostGIS. Set ud fra spatiale forhold er dette database system det hurtigste og nemmeste at håndtere. Og anskaffelsesprisen er til at overskue: 0,- kr. inkl. moms.



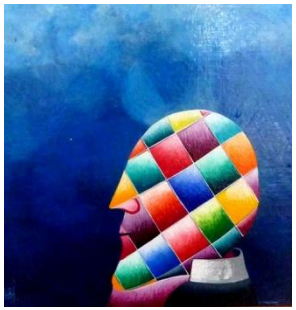
Opsætning af system – generelle anbefalinger:

- Systemer med mange brugere, store datamængder:

  Opsæt to database systemer på hver sin server. Den ene er ”master” og bruges til at indsætte og redigere data. Den anden er ”slave” og bruges kun til analyser (read-only).

  ”Master” kan være relativt lavt ydende maskine

”Slave” bør være en højt ydende maskine



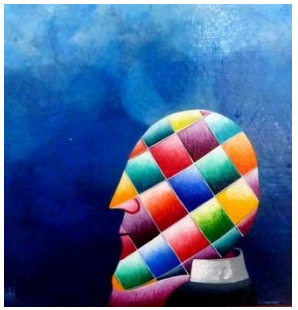
Import af data til databasen

- SAFE Software FME kan flytte stort set alle spatiale data fra hvilket som helst format til og fra ethvert database system (sådan nogenlunde) . Problemet er at det bogstavlig talt koster en ½ bondegård at anskaffe.

- Andre alternativer er div. småprogrammer som typisk følger med databasesystemet. I PostgreSQL er der indbygget et importværktøj til shape filer i selve administrationsprogrammet (PGADMIN III)

- I en nyere udgaver af QGIS er det muligt at indlæse fildata og videre upload dem til forskellige database systemer. Min egen erfaring er dog blandet mht. succes rate. Så derfor:

- Personlige valg er OpenSource systemet GDAL/OGR - det er gratis. Systemet understøtter alle gængse spatiale filformater og de fleste databasesystemer.



Eksempel Markblokke – manuelt i GIS

- Generer en 10 meter bred buffer rundt om vandløb (tabel streams)
  I virkeligheden er vandløbene 4 m. brede, men repræsenteres af polylinier (
- bredde = 0 m.) . Så vi laver bufferen på 12 m for at kompensere.

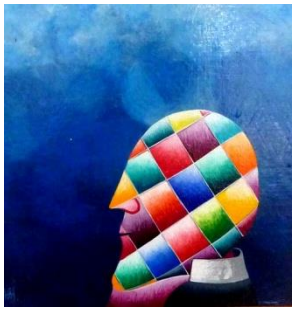
- Find alle søer med et areal > 100 m2

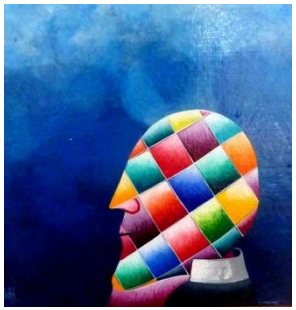
- Genererer en buffer, 10 m. bred rundt om disse søer.

- Kombiner de to buffer lag til et. Fjern overflødigt overlap mellem bufferzoner fra de to lag. Resultatet kalder vi ”vand_buffer”.

- Find overlappet mellem de enkelte markblokke og ”vand_buffer” og vis det på kortet.

- Beregn kompensationsarealet i m2 for hver markblok



Eksempel Markblok – i SQL - automatisk

```
DROP VIEW IF EXISTS compensation_area CASCADE;
CREATE VIEW compensation_area AS

SELECT
  a.gid AS gid,
  MIN(bloknr) AS bloknr,
  MIN(ortoyear) AS ortoyear,
  MIN(dept) AS dept,
  MIN(usage) AS usage,
  st_multi(st_union(st_intersection(a.geom,b.geom))):::geometry(MULTIPOLYGON,25832) AS geom

FROM
  markblok AS a,
  (
    SELECT
      gid,
      st_multi(st_buffer(streams.geom,12.0)):::geometry(MULTIPOLYGON,25832) AS geom

    FROM streams

    UNION

    SELECT
      gid + 100000 AS gid,
      st_multi(st_buffer(lake.geom,10.0)):::geometry(MULTIPOLYGON,25832) AS geom

    FROM lake

    WHERE
      st_area(geom) > 100.0
  )  AS b

WHERE
  st_intersects(a.geom,b.geom)

GROUP BY

  a.gid;
```