P. Ø.

DNMI DET NORSKE METEOROLOGISKE INSTITUTT

klima

DATABASE-PROSJEKTET I KLIMAAVD. ETABLERING AV VALGT DATASTRUKTUR PÅ TYPHOON. DELPROSJEKT 3.

Å.M. VIDAL, S.L. LYSTAD, P. ØGLAND, M. MOE

RAPPORT NR. 40/92

DET NORSKE METEOROLOGISKE INSTITUTT
POSTBOKS 43 BLINDERN 0313 OSLO 3

TELEFON: (02) 96 30 00

ISBN

RAPPORT NR.
40/92 KLIMA

DATO
07.10.1992

TITTEL

DATABASE-PROSJEKTET I KLIMAAVD.
ETABLERING AV VALGT DATASTRUKTUR PÅ TYPHOON.
DELPROSJEKT 3.

UTARBEIDET AV

Å.M. Vidal, S.L. Lystad, P. Øgland, M. Moe

OPPDRAGSGIVER

DNMI

SAMMENDRAG
Rapporten beskriver hvordan valgt datastruktur kan etableres på Typhoon. Det er utarbeidet rutiner både for etablering av datastruktur og for overføring av data. Datastrukturer er implementert og noe data overført. Dette utgjør test-databasen som videre prosjekter vil jobbe mot.

Rutinene for kombinert etablering av datastrukturer og lasting av data er også ment brukt når systemet kommer i drift. Overordnet dataflyt er spesifisert og vil danne grunnlaget for videre arbeid.

Det er gjort eksperimenter med innlasting fra magnetbånd til hovedlager og arbeidslager, kontrollrutiner og indeksering. Informasjonslager er implementert v.h.a. Oracles CASE-verktøy.

UNDERSKRIFT

Åse Moen Vidal
PROSJEKTLEDER

Bjørn Aune
FAGSJEF

I N N H O L D

1. Innledning

1.1 Møtevirksomhet

Delprosjekt 3 har følgende bemanning: Åse Moen Vidal (delprosjektleder), Margareth Moe, Petter Øgland, Sofus Linge Lystad.

Prosjektplan ble levert 22.04.92, men videre arbeid ble først startet opp 03.09.92 med faste møter tirsdag og torsdag fra kl. 12 - 14 i lille møterom. Prosjektet avsluttet 06.09.92.

Referat er skrevet for hvert møte. Tidsplan er holdt selv om flere prosjektdeltakere hadde andre prosjekter og gjøremål å ta seg av i samme periode.

1.2 Gjennomføring av prosjektet

Prosjektgruppen var sammensatt av personer som hver har bidratt med nødvendig og tilstrekkelig kompetanse for å utføre prosjektet. Kompentansen har bestått av kjennskap og erfaring med ORACLE RDBMS og dens verktøy, operativsystemene SINTRAN og UNIX, programmering i C, FORTRAN og UNIX-script, meteorologiske data og stasjonsdata, bruk av data i dag og de forbedringer en ønsker å oppnå i det nye systemet med hensyn til datatilgjengelighet, datakvalitet og brukervennlighet og til slutt og ikke minst - praktisk sans.

Selve overføringen av data er gjort både via magnetbånd og diskett. Data fra ND-788 ble lastet på magnetbånd. Overføringen fra magnetbånd innebar dumping av magnetbånd på TAPESERV som er arbeidsstasjonen magnetbåndstasjonen er tilknyttet. Videre overføring til TYPHOON er gjort via FTP. Filoverføring fra diskett er gjort fra PC via FTP.

Programmering og uttesting av programmer, script, indekser etc. er utført på TYPHOON via SG arbeidsstasjoner.

1.3 Rapportbeskrivelse

Rapporten beskriver implementasjon av samtlige datastrukturer utredet av delprosjekt 2E.

Prosjektgruppen har basert seg på en dynamisk opprettelse av tabeller i hovedlager, dvs. stasjonsspesifikk parameterrekkefølge i hver enkelt tabell, og tabellene må følgelig opprettes simultant med at data leses inn.

Tabeller for arbeidslager og informasjonslager er

statiske, og følgelig opprettet en gang for alle.

Kapittel 2 beskriver generelle studier av dataflyt mot systemet og internt i systemet. En basismodell og en alternativ modell er presentert. Dataflyten er imperativ for automatisk opprettelse av tabeller i hovedlager.

Kapittel 3 beskriver etableringsspesifikke studier av teoretisk og praktisk karakter. Dette inkluderer studier av integritetsregler, fysisk avsetting av plass for lagring av tabeller, konstruksjon av informasjonslager og innlasting av nødvendig og tilstrekkelig informasjons-lagerdata, indeksering og konstruksjon av innlastingsprogram.

Kapittel 4 beskriver faktiske innlastinger utført gjennom dataflyten i kapittel 2 (modell 1), og med programmer og prinsipper beskrevet i kapittel 3.

Kapittel 5 beskriver tilrettelegging for endelig innlasting, dvs. anbefalinger angående hvordan det testmiljø som har vært benyttet under delprosjekt 3 gradvis bør utvikles til et endelig dataflytmiljø med de krav og modifikasjoner dette innebærer.

1.4 FORKORTELSER

Nedenforstående forkortelser er brukt i rapporten:

AL	: Arbeidslager - midlertidig tabell for å bearbeide, kontrollere data før de legges i permanent lager. Det finnes 1 arbeidslager for hver stasjonstype.
CASE	: Computer Aided Systems Engineering
EL	: Ekstremlager - absolutt høyeste og laveste verdi en parameter har hatt i observasjonsperioden samt tidangivelse for målingen. Dette er redundante data fra HL hvis hele observasjonsperioden ligger i HL.
FELTDATA	: Data for prognosesystemet LAMxx
FL	: Frekvenslager - relativ hyppighet av feil, bruk av parametre o.a.
FTP	: File Transfer Protokoll
HL	: Hovedlager - permanente stasjonsvise tabeller (ca. 2000) tilgjengelig for sluttbruker
IL	: Informasjonslager - tabeller med administrativ og teknisk informasjon om stasjon og parametre.
ML	: Midlertidig lager - til midlertidig bruk for raskt å kunne skaffe statistikk til veie.
NULL	: Tomt felt i ORACLE som tar 1 byte.
SL	: Statistikklager - tabeller med diverse statistikk som ekstremstatistikk, hyppighetsstatistikk o.a.
SG	: SILICON GRAPHICS - maskinleverandør
TYPHOON	: Databasemaskin
FORMS	: Et verktøy hvor en i et interaktivt skjermbilde kan laste, kontrollere, oppdatere og slette data i en database.
SQLLOADER	: Datainnlastingverktøy for ORACLE
UTPLUKKSRUTINE	: Et program som henter ut data fra databasen. Utplukksprogrammet skal sikre datakvalitet og fornuftig bruk av maskinressurser.

2. Dataflytmodell

I forbindelse med helautomatisk etablering av datastrukturer på TYPHOON er det nødvendig å spesifisere dataflyt. Historisk data skal lastes inn på samme måte som sanntidsdata, og tabeller i hovedlageret med stasjonsspesifikk parameterrekkefølge opprettes automatisk etter behov.

Hovedflyt

Det er presentert to dataflytmodeller.

Modell 1 (kap 2.1) er utviklet med tanke på overføring av såvel historiske som sanntidsdata på en effektiv, kontrollert og enkel måte. Modellen er blitt benyttet som utgangspunkt for de fysiske tester utført av prosjektgruppen.

Modell 2 (kap 2.5) er utviklet for å ta hensyn til reorganisering av tabeller i hovedlager (blokksplitting), hensyn til færrest mulig tabelloppslag, og effektiv bruk av SQL*FORMS.

Da det ikke er avklart hvilken modell som er å foretrekke i praktisk bruk av databasen fremlegger prosjektgruppen to modeller. Modell 1 er basis for de praktiske studier gjennomført på nåværende nivå. Dersom det over tid skulle være behov for finjustering av innlastingens yteevne, eller det på annen måte skulle avsløres svakheter ved modell 1, anbefaler vi at modell 2 implementeres og uttestes.

Dataflyt i infrastruktur

I kapitlene 2.1-2.3 følger en mer detaljert beskrivelse av den interne data- og informasjonsflyt mellom de forskjellige lagre i modell 1.

Først presenteres dataflyten inn i systemet for automatisk og manuell innlasting av rådata eller korrigert data. (kap 2.2)

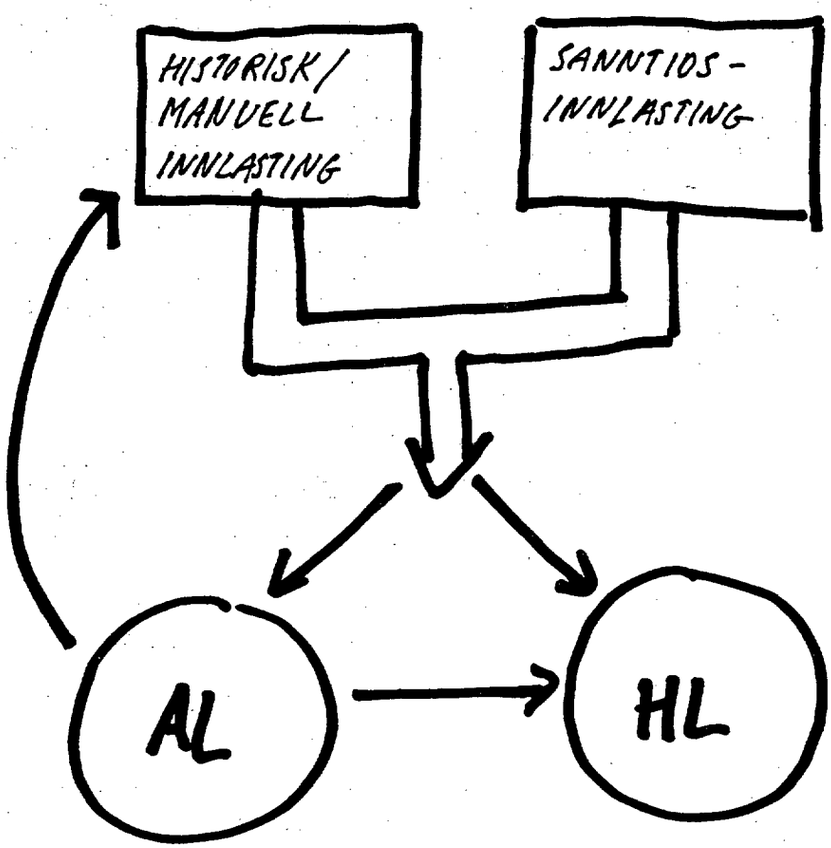
Deretter dataflyten fra arbeidslager til hovedlager i forbindelse med kontroll. (kap 2.3)

Tilslutt presenteres informasjonsutvekslingen mellom arbeidslager og informasjonlager under innlasting av data. (kap 2.4)

2.1 Basismodell

1. Hovedstruktur

Hovedstrukturen for basismodellen kan illustreres gjennom følgende diagram (figur 2.1.1)



Figur 2.1.1 Hovedstruktur for basismodell

All ekstern data (historisk-, sanntids-, punche-data, korrigert data, ...) flyter inn i systemet via en felles kanal.

Kanalen splittes i to.

All data flyter til arbeidslager (AL), mens et utvalg flyter til hovedlager (HL). Innlastingsrutinen til HL inneholder med andre ord et filter som forkaster fullstendig uakseptable verdier.

Data flyter videre fra AL til HL gjennom et kontrollfilter. Filteret er vesentlig mer finmasket enn forkastningsfilter mot HL. Data kan slippe gjennom filteret på to måter, enten som kontrollert, eller som

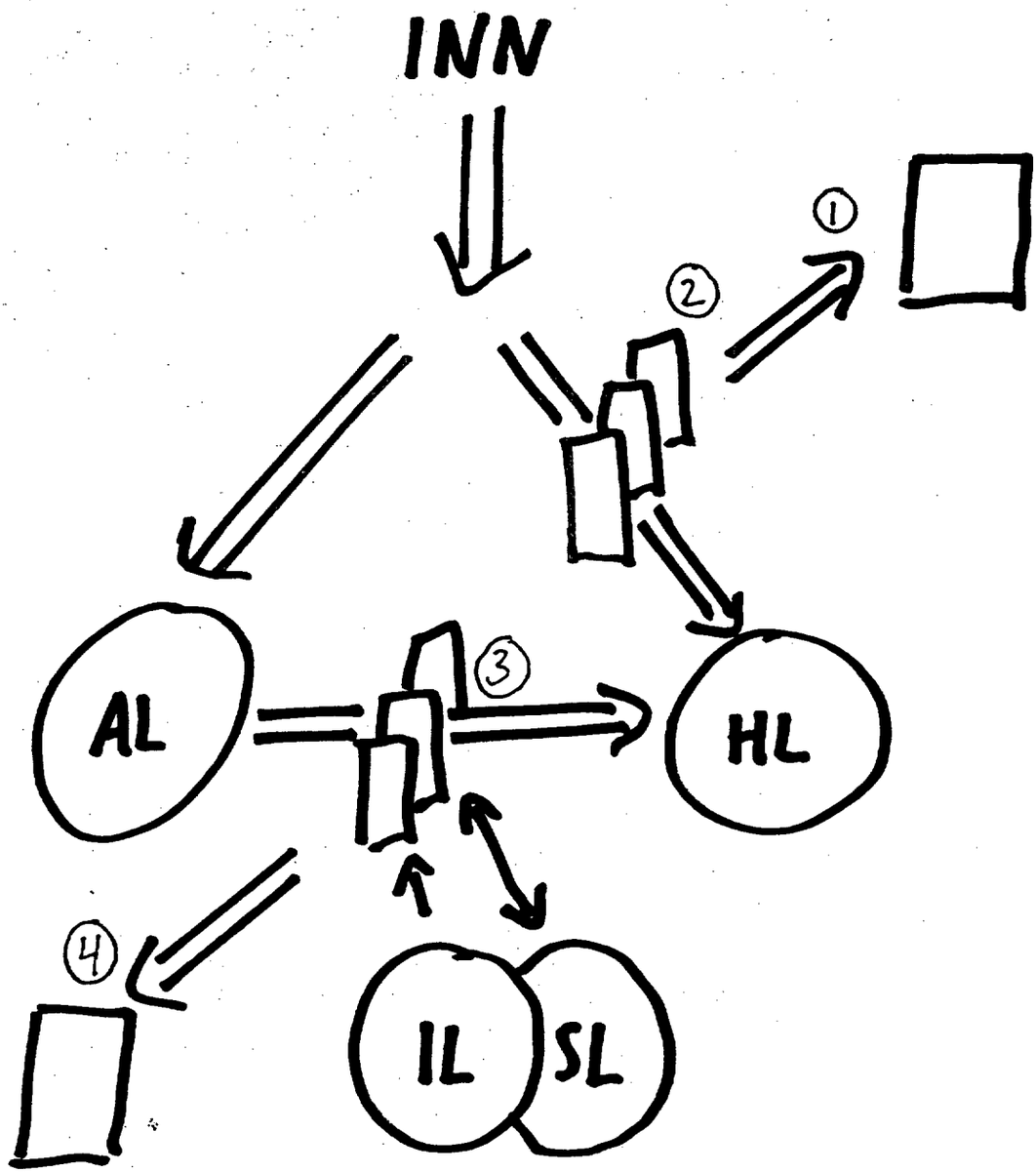
midlertidig interpolert inntil tilstrekkelig kontroller er utført.

Data som ikke tilfredstiller kontrollen rapporteres og sendes ut av systemet for ny innlasting.

All lesing mot databasen (utpluksrutine o.l.) skjer kun mot HL og IL. Arbeidslageret kan ikke manipuleres av sluttbruker.

2. Detaljstruktur

Følgende diagram skal gi en mer detaljert beskrivelse av dataflyten fra figur 2.1.1, nedre halvdel (figur 2.1.2):



Figur 2.1.2 Detaljstruktur for overordnet dataflyt

All data kommer inn i samme kanal til AL og HL. Alle data flagges UKONT (ukontrollert).

I (1) legges rapporter om data, dvs. mangler, status, eksterne flagg etc.

Filteret (2) består av ulike delfiltre for ulike datatyper (SYNOP, METAR, PUNCH,...). All data mot HL gjennom disse filterene flagges bare UKONT.

Filteret (3) består også av ulike delfiltere for ulike datatyper. Ingen data overføres via dette filteret, kun korreksjon av flaggene i HL til KONT eller INTERP. Statistikklager (SL = EL + FL) oppdateres.

I logfilen (4) legges statusrapporter fra kontrollfiltreringen.

Logfilene (1) og (4) gir opphav til manuell reaksjon.

All lesning mot database (utpluksrutine o.l.) skjer mot HL og IL. Arbeidslageret kan ikke manipuleres av sluttbruker.

2.2 Dataflyt inn i systemet

Figur 2.1.1 viser to entiteter for henholdsvis innlasting av historisk- og sanntids-data med felles inngang mot arbeidslager og hovedlager. I kapittel 2.2 presenteres en mer detaljert beskrivelse av dataflyten inn i systemet.

1. Automatisk innlasting

Automatisk innlasting i modell 1 baserer seg på at data ankommer innlastingssystemet i form av en eller flere ASCII-filer. Disse leses automatisk av et innlastingsprogram (script), prepareres og lastes parallelt inn i AL og HL.

I værprognoser genereres såkalt feltdata, hvor diverse kontroller og interpolasjoner er foretatt. Flagging brukt i feltdata bør registreres på en eller annen form for å generere optimale kontrollrutiner, dvs. være behjelpelige for å avsløre ukorrekt data og å gjøre gode interpolasjoner.

Innlastingsformat for feltdata som ivaretar flagginformasjonen og eventuelt lagrer denne i IL (SL) må vurderes av senere prosjekter (7.1 "Data inn").

2. Manuell innlasting

All manuell dataoverføring, enten i form av korreksjoner eller førstegangspunching lagres på temporære ASCII-filer før de lastes parallelt inn i AL og HL via sqlloader.

Når det gjelder punche-verktøy for modell 1 har man satt opp tre alternativer:

a) SQL*FORMS knyttet til midlertidige ORACLE-tabeller som lastes og tømmes inn i hovedstrømmen.

b) SQL*FORMS knyttet til arbeidslageret, men hjulpet av en programrutine som sender de samme data inn gjennom hovedlagerets kontrollfilter

c) Egenutviklet programvare som operer på ASCII-nivå og sender data til systemet via standard inngang.

3. Forkastingsfilter mot hovedlager

Forkastningskriteriet i filteret fra rådata mot hovedlager vil kun bestå i intervalltesting. Intervallgrensene vil leses automatisk fra informasjonslageret, og forkastede data havner på en log-fil beskrevet i seksjon 2.1.

I enkelte situasjoner kan det være ønskelig å overstyre filteret, dvs. laste inn data som normalt ville blitt forkastet ("Naturkatastrofer"). Dette tenkes løst ved

hjelp av et såkalt superkontrollert-flagg, som er en beskjed fra felles inngang mot kontrollfilteret om at vedkommende data ikke skal kontrolleres.

4. Stasjonstypevise innlastingsformater

Innlastingsrutinen skal selv kunne finne ut hva slags innlastingsformat som skal benyttes på basis av informasjonslageret (IL).

Prosjektgruppen har funnet det hensiktsmessig å benytte de standardformater som allerede eksisterer for operative stasjoner i så stor grad som mulig.

For stasjonstypevise innlastingsformater

format K Vær-, Klima-, Linke- og Fordampningsstasjoner
format N Nedbørstasjoner
format P Plumatic stasjoner
format M Maritime + bøyer
format A Automatstasjoner (Edas, Scanmatic, Campbell, DNMI spec.)
format E E-data
format L Aanderaa
format R Radiosondestasjoner
format F Flyværstastasjoner (Metar)

basert på stasjonstypeinndelingen i [4], må det svare en sqlloader-kontrollfil (sml. neste side) for hvert format. I tillegg til stasjonstypevise formater må man ta hensyn til variasjoner av punche-formater innefor hver stasjon, vi tenker for eksempel på periodene 1875-1930, 1931-1936, 1937-1945, 1946-1956, 1956-1981, 1982-..., men også variasjoner innenfor disse gruppene. En vurdering av formater overlates til prosjekt 7.1 "Data inn".

På de neste sidene vises de stasjonstyper med tilhørende formater uttestet av prosjektgruppen. Formatene valgt med hensyn på å utføre de nødvendige og tilstrekkelige tester for delprosjekt 3:

K.ctl K-format
M.ctl M-format
N.ctl N-format

```
tions (errors=50)
AD DATA
FILE HL.dat
place
clen 135
TO TABLE k18700
STNR POSITION (01:05) integer external,
AAR  POSITION (06:09) integer external,
                  MND  POSITION (10:11) integer external,
                  DAG  POSITION (12:13) integer external,
                  TIM  POSITION (14:15) integer external,
                  TT   POSITION (34:38) decimal external NULLIF (TT = '999.9
                  TN   POSITION (39:43) decimal external NULLIF (TN = '999.9
                  TX   POSITION (44:48) decimal external NULLIF (TX = '999.9
                  TG   POSITION (49:53) decimal external NULLIF (TG = '999.9
                  TW   POSITION (54:58) decimal external NULLIF (TW = '999.9
                  UU   POSITION (59:61) integer external NULLIF (UU = ' -1')
                  RR   POSITION (70:74) decimal external NULLIF (RR = '  -.1
                  DD   POSITION (64:65) integer external NULLIF (DD = '-1'),
                  FFX  POSITION (108:109) integer external NULLIF (FFX = '-1
                  F    POSITION (68:69) integer external NULLIF (F = '-1'),
                  SD   POSITION (77:78) integer external NULLIF (SD = '-1'),
                  SS   POSITION (79:81) integer external NULLIF (SS = ' -1')
                  VV   POSITION (86:87) integer external NULLIF (VV = '-1'),
                  V1   POSITION (88:89) char NULLIF (V1 = '  '),
                  V2   POSITION (90:91) char NULLIF (V2 = '  '),
                  WW   POSITION (94:95) integer external NULLIF (WW = '-1'),
                  W1   POSITION (104:105) integer external NULLIF (W1 = '-1'
                  W2   POSITION (106:107) integer external NULLIF (W2 = '-1'
                  P0   POSITION (16:21) decimal external NULLIF (P0 = '   -.
                  P    POSITION (22:27) decimal external NULLIF (P  = '   -.
                  A    POSITION (28:29) integer external NULLIF (A  = '-1'),
                  PP   POSITION (30:33) decimal external NULLIF (PP = ' -.1'
                  FF   POSITION (66:67) integer external NULLIF (FF = '-1'),
                  E    POSITION (75:76) integer external NULLIF (E  = '-1'),
                  EM   POSITION (128:129) integer external NULLIF (EM = '-1'
                  N    POSITION (82:83) integer external NULLIF (N  = '-1'),
                  H    POSITION (84:85) integer external NULLIF (H  = '-1'),
                  NH   POSITION (112:113) integer external NULLIF (NH = '-1'
                  CL   POSITION (114:115) integer external NULLIF (CL = '-1'
                  CM   POSITION (116:117) integer external NULLIF (CM = '-1'
                  CH   POSITION (118:119) integer external NULLIF (CH = '-1'
                  NS1  POSITION (120:121) integer external NULLIF (NS1 = '-1
                  C1   POSITION (122:123) integer external NULLIF (C1 = '-1'
                  HS1  POSITION (124:125) integer external NULLIF (HS1 = '-1
                  S    POSITION (62:63) integer external NULLIF (S = '-1'),
                  FG   POSITION (110:111) integer external NULLIF (FG = '-1'
                  FX   POSITION (126:127) integer externaL NULLIF (FX = '-1'
```

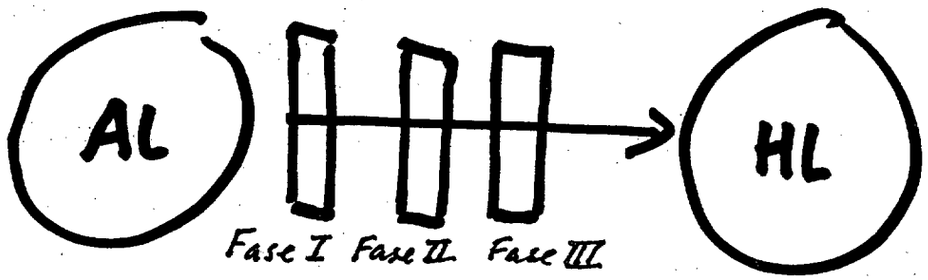
```
)AD DATA
FILE HL.dat
eplace
eclen 124
TO TABLE m0169     ( STNR position (69:71) integer external,
                   DATO position (02:09) DATE "YYYYMMDDHH24",
                   AAR  position (02:03) integer external,
                   MND  position (04:05) integer external,
                   DAG  position (06:07) integer external,
                   TIM  position (08:09) integer external,
                   iw   position (10:10) decimal external,
                   q    position (11:11) integer external,
                   la   position (12:14) integer external,
                   lo   position (15:17) integer external,
                   n    position (22:22) decimal external,
                   Td   position (31:34) decimal external,
                   W1   position (41:41) integer external,
                   ww   position (39:40) integer external,
                   W2   position (42:42) integer external,
                   P    position (35:38) decimal external,
                   A    position (90:90) integer external,
                   PP   position (91:93) decimal external,
                   H    Position (19:19) integer external,
                   NH   Position (43:43) integer external,
                   CL   Position (44:44) integer external,
                   CM   Position (45:45) integer external,
                   CH   Position (46:46) integer external,
                   S    Position (62:63) integer external,
                   pwper Position (53:54) integer externaL,
                   pw   position (59:60) integer external,
                   hwh  position (55:56) integer external,
                   hw   position (61:62) integer external,
                   dw   position (57:58) integer external,
                   wds  position (94:94) integer external,
                   wvs  position (95:95) integer external,
                   dw1  position (96:97) integer external,
                   pw2  position (98:99) integer external,
                   hw2  position (100:101) integer external,
                   ci   position (102:102) integer external,
                   si   position (103:103) integer external,
                   bi   position (104:104) integer external,
                   di   position (105:105) integer external,
                   zi   position (106:106) integer external,
                   wis  position (63:63) integer external,
                   wes  position (64:64) integer external,
                   rs  position (65:65) integer external)
```

```
ad data
FILE HL.dat
place
clen 52
TO TABLE N05293 ( STNR position (01:05) integer external,
                  DATO position (06:15) DATE "YYYYMMDDHH24",
                  AAR  position (06:09) integer external,
                  MND  position (10:11) integer external,
                  DAG  position (12:13) integer external,
                  TIM  position (14:15) integer external,
                  RR   position (16:20) decimal external NULLIF (RR = '-.1')
                  V4   position (23:28) char,
                  V5   position (29:33) char,
                  V6   position (35:40) char,
                  SS   position (41:43) integer external NULLIF (SS = '-1'),
                  SD   position (46:47) integer external NULLIF (SD = '-1'))
```

2.3 Informasjonsflyt fra arbeidslager til hovedlager

1. Innledning

I seksjon 2.1 så man hvordan man konseptuelt kan tenke seg den totale flyten i systemet. I denne seksjonen presenteres en mer nøyaktig beskrivelse av kontrollfilteret fra arbeidslager til hovedlager. Flyten består av kontroll og estimering.



Figur 2.3.1 Informasjonsflyt fra AL til HL

2. Kontroll

Filteret i dataflyten (eg. flaggflyt) fra AL til HL må inneholde kontrollrutiner. Data som passerer kontrollen flagges kontrollert og flagget plasseres i HL.

Forskjellig type data må kontrolleres på forskjellig måte, men all data som slipper gjennom kontroll-filteret i uforandret tilstand betraktes som kontrollert, og kan ikke overskrives i HL (med unntak av databaseadministrator som har visse spesialrettigheter).

Kontrollrutinene bør settes sammen i et fasedelt system basert på hvilke tester som kan utføres ved gitt tid. Dessuten bør generelle tester gjøres først og spesielle tester (parameterspesifikke tester) sist. Suspekte data bør rapporteres så raskt som mulig, og det bør gis rapport om hva slags svar maskinen kunne tenke seg, dvs. konsekvensen av maskinell glatting/estimering.

2.1 Algoritmeaspekter

Prinsipielt kan vi forespeile oss to teststrukturer, auto-kontroll (kontroller basert på tidsrekker av enkeltvise parametre) og inter-kontroller (tidsrekker med varierende innslag av parametre).

2.1.1 Konsistens-kontroll

Før de mer avanserte kontroller kan settes i gang må det sjekkes at data er konsistent. Herunder hører max-min-

kontroller (en observert verdi må alltid befinne seg mellom observert maksimum og minimum), kontroller av typen dersom det er observert regn må det være observert skyer, ... etc.

Kontroll om det har foregått en forskyvning i synopdata er også en aktuell test her.

Egne konsistenskontroller må introdusere auto- og inter-kontrollene skissert nedenfor.

2.1.2 Auto-kontroll

En enkelt parameter betraktes som en stokastisk prosess X(t), og prosessen estimeres ved prosessestimator Y(t)=E(X(t)).

Avstanden

$$|| X(t) - Y(t) || < e \qquad (2.3.1)$$

beregnes i henhold til en passende integralnorm, og observasjonen forkastes dersom verdien overstiger en toleransegrense e.

Normen $|| . ||$ i (2.3.1) kan være en L2-norm passende til den underliggende sannsynlighetsfordelingen i det enkleste tilfellet, men en Sobolev-seminorm for måling av tidsgradienter.

Enkle realiseringer av normen kan f.eks. være

i) Grensesjekk (intervallsjekk)
ii) Frekvenssjekk (FL-test beskrevet i kap 2.4)
iii) Sprangsjekk
iv) Feltdatasjekk

Testene i) og ii) gjøres mot nå-data (Integrasjon med Dirac-mål, t=t0).
Testen iii) på basis av to tidspunkter (nå-sist-Sobolev-Dirac-integral)

2.1.3 Inter-kontroll

En matematisk formulering av interkontrollen følger samme linjer som auto-kontrollen, kun med den modifikasjon at prosessene i (2.3.1) er vektorielle.

Intervalltesting svarer til romkontroll.

Integrasjon over kontinuerlige sannsynlighetsfordelinger svarer til regresjon (flate-ekstrapolasjon).

Feltsjekk kan gjøres både med hensyn til punktverdier og gradienter (flate-ekstrapolasjon).

2.2 Tidsaspekter

2.2.1 Innlastingstempo

Dataflyten inn i AL følger forskjellige diskrete tidsskalaer. Noe data kommer i sanntid (timesvis), noe i ukesintervaller, og noe i månedsintervaller.

For å gjennomføre en komplett sjekk av all data kan man ikke klarere stasjonsverdier før de er sammenliknet med alle influerende nabostasjoner. For at ikke all data skal bli forsinket ved å vente på de siste observasjonene foreslår prosjektgruppen en fasedelt kontroll, hvor kontroller utføres så snart det er mulig avhengig av status i AL, og sender sine midlertidige resultater til HL.

Fase I

Første fase består av umiddelbar kontroll av såvel historisk (ettersendt) som sanntids-data.

I første fase vil det bli gjort logiske kontroller (konsistens-kontroller), og forøvrig auto- og inter-kontroller i den grad det lar seg gjøre.

Fase II

Annen fase består av ukentlig kontroll, dvs. tester basert på siste syv (fjorten) dager. Mangelverdier for enkelte stasjoner er blitt fylt ut, og de aktuelle deler av AL kan testes mot disse.

Fase III

Hver måned (annenhver måned) vil de siste hullene i AL bli fylt ut. Dette gir da opphav til å rekontrollere alle observasjoner som er avhengig av disse.

Stasjoner som ikke er komplette etter månedsrutinen gir opphav til egne behandlingsregimer bestående av aksjon fra databaseadministrator og innhenting av data fra HL for rekontroll.

2.2.2 Periodiske kontroller

I tillegg til restarting av kontrollrutiner, kan faseinndelingen benyttes til å beregne kontrollstatistikk

På basis av ukeverdien kan man studere regresjoner og ukesprofiler.

En månedskontroll (månedmiddel o.l.) kan gjennomføres for å forhindre at akkumuleringsfeil ikke oppdages. Tilsvarende akkumuleringsovervåkere burde kunne implementeres på basis av ukes-normaler (pentader).

Det kan imidlertid understrekes at disse periodiske

kontrollene ikke nødvendigvis må synkroniseres med innlastingsfasene, men kan beregnes for de enkelte stasjoner eller grupper av stasjoner etterhver som datasettene blir komplette.

Data som passerer denne siste testen flagges som kontrollert i arbeidslager, og kan kun forandres av databaseadministrator eller ved hjelp av superkontroller (kap 2.2).

3. Automatisk estimering

Dersom data ikke slipper gjennom kontrollrutinene brukes informasjon ervervet gjennom kontrollprogrammet til å lage et estimat over hva slags verdi som kunne vært rimelig å legge inn i HL. Estimatet lagres så i HL som ukontrollert og interpolert.

Hver gang det gjøres tester i AL vil nye estimater legges i HL slik at det dannes en konvergent følge av verdier. Ved å gjøre sammenlikninger med nytt og gammelt estimat skulle det være mulig å sikre at følgen er Chauchy-konvergent (avstanden mellom to påfølgende estimater er avtagende).

4. Statusrapport fra filteret

Filteret gir fra seg informasjon om hvilke data som ikke tilfredstiller kontrollen.

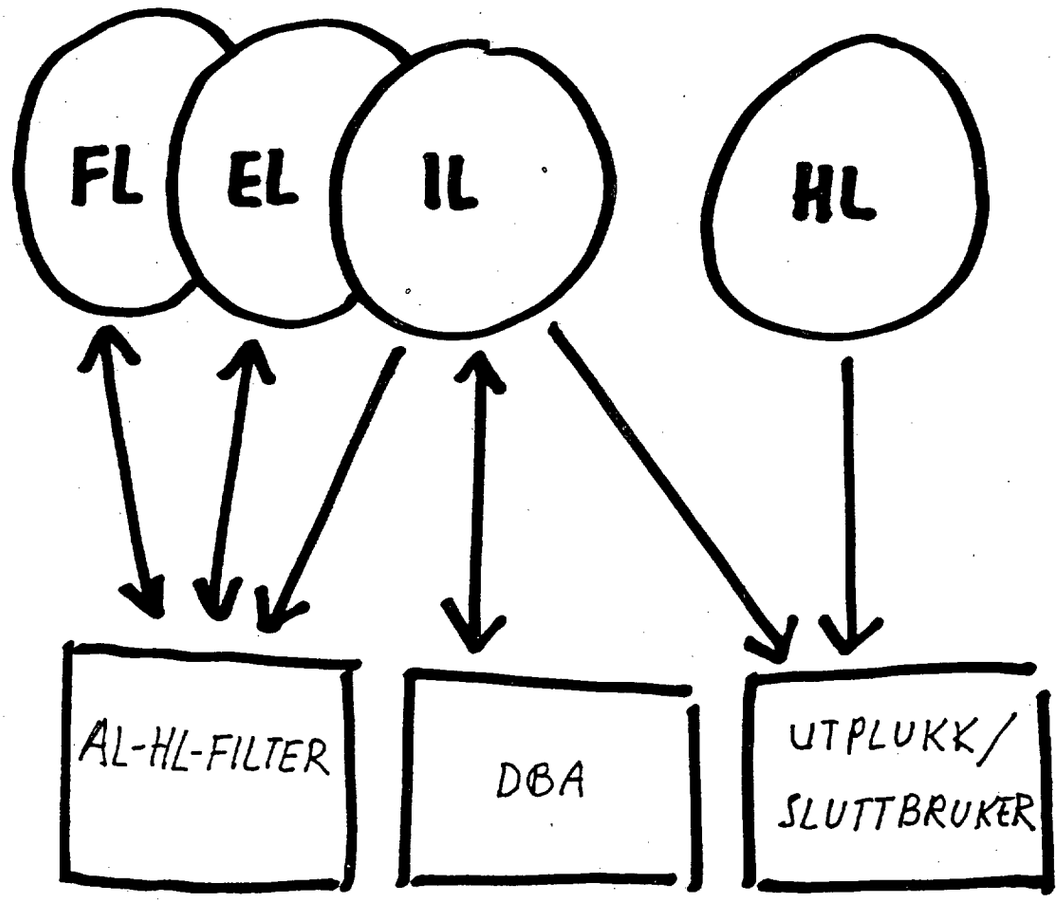
I tvilstilfeller, dvs. i situasjoner der det er nødvendig med meteorologisk skjønn, bør maskinen kjøre en automatisk hypotesetesting med sannsynligheten for forkastning av korrekt data i henhold til forskjellig signifikansintervaller.

Når det gjelder rapportering av feil for manuell korreksjon bør disse rangeres med utgangspunkt i hva maskinen tror er grove feil, midlere feil og små feil.

2.4 Informasjonsflyt mellom arbeidslager og informasjonslager

1. Aksessering av informasjonslager

Informasjonslageret (IL) kan aksesseres fra tre forskjellige kilder; via utpluksrutinen, via databaseadministrator (den som måtte være ansvarlig for manuelle oppdateringer av IL i situasjoner hvor stasjoner initieres, termineres, får nye eiere osv), og via automatisk aksessering fra arbeidslager (AL) eller hovedlager (HL).



Figur 2.4.1 Aksess mot IL

Utpluksrutinen har kun lov til å hente informasjon fra IL-tabellene, den automatiske aksesseringen skal også ha lov til å legge nytt innhold i eksisterende tabellrader i deler av IL (dvs. EL og FL, men ikke i IL generelt), mens kun databaseadministrator har alle rettigheter.

2. IL-informasjon av interesse for AL og HL

Forut for selve innlastingen vil det automatisk bli sjekket mot IL for å finne det rette innlastingsformatet. Under innlasting av historisk data vil det også parametre bli sjekket for parameter-rekkefølge-spesifikk konstruksjon av tabeller.

AL, HL og IL vil være synkrone for hver sammenhengende driftsperiode for den enkelte stasjon (så fremt man benytter statiske tabellstrukturer med fast familie av parametre for hver stasjonstype), og det er følgelig ikke nødvendig å oppdatere IL fra AL eller HL eller omvendt innen en slik periode.

Asynkronitet kan imidlertid oppstå i drifts-diskontinuitetene, dvs. når en stasjon skifter parametre. Asynkronitet i databasen må forhindres ved database-administrators direkte inngripen. Databaseadministrator må varsles om et diskontinuitetspunkt enten på forhånd via kommunikasjon (brev, telefon) eller i ettertid via logfilene til HLs forkastingsfilter. Synkronitets-problemer bør løses manuelt, da en utidig automatisk sjekk kan bli unødvendig ressurskrevende.

3. AL- og HL-informasjon av interesse for IL

Statistikklageret (SL) i IL må oppdateres automatisk avhengig av status i arbeidslager og hovedlager. SL består av to dellagre, frekvenslager og ekstremlager.

3.1 Frekvenslager (FL)

Relative hyppigheter for forekomster av feil i parametre (AL) kan være interessant generell informasjon om en stasjon: Telling og lagring av rettelser.

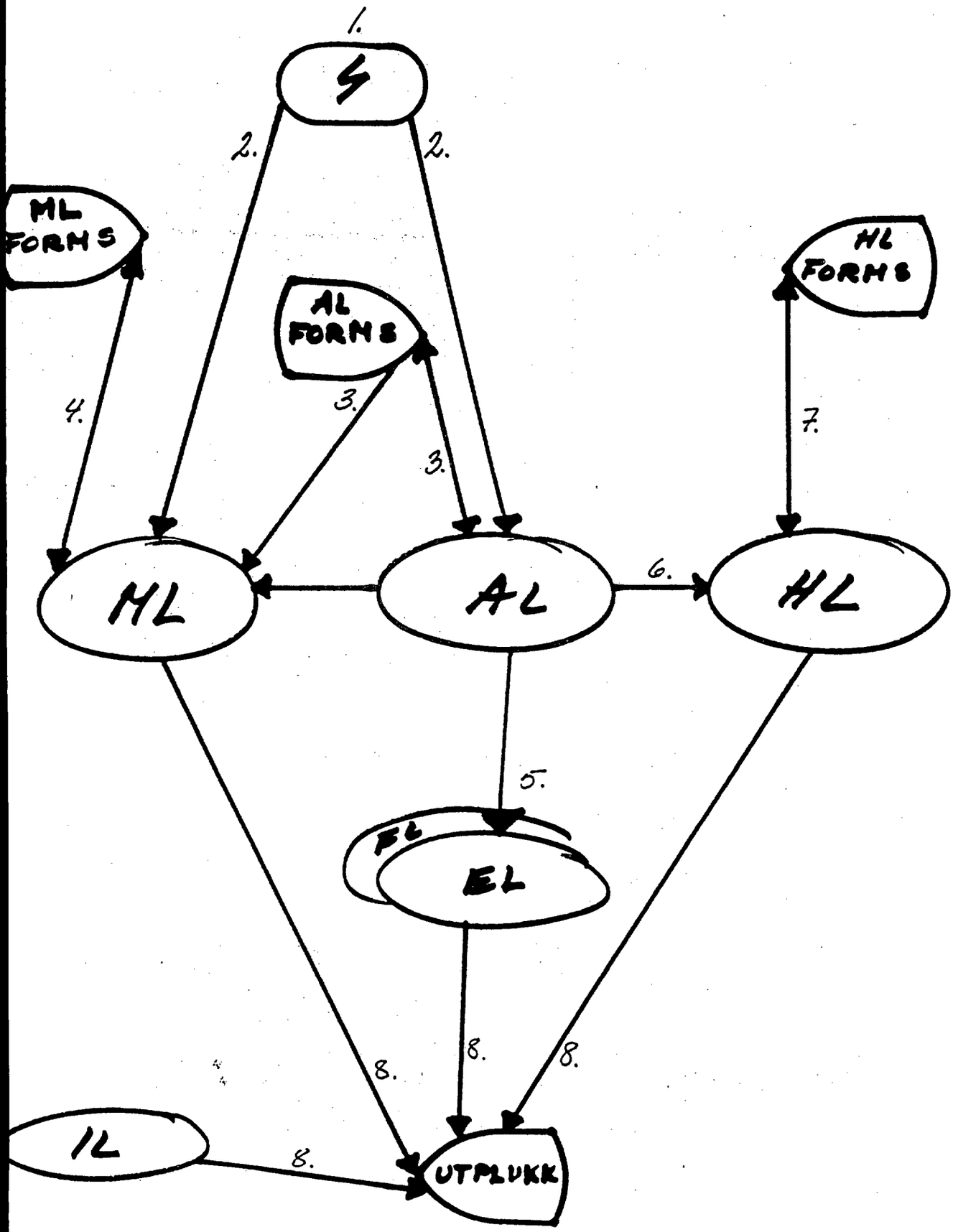
Relativ hyppighet for bruk av parametre, dvs. relativ-hyppighets-tabell for hvor mange ganger en parameter er not-null, kan være interessant.

Relative hyppigheter for verdier av parametre kan også være gunstig både for testing av instrumenter og målinger.

3.2 Ekstremlager (EL)

Kontrollfilteret fra AL til HL vil benytte seg av ekstremer under testing.

2.5 Alternativ modell



DATAFLYT I Modell 2 :

1. Data overføres via telegram/oppringning og legges på flatfil.

2. SQLLOADER laster data parallelt til ML og AL.

3. AL-FORMS-programmer laster data inn/ut av AL og inn i ML.
   Dette kan være registrering, oppdatering, uthenting av data for rutinebruk.

   Ved registrering av data vil parametre (felt) som tilhører samme observasjon (rad) og som allerede ligger i AL bli synlig i skjermbildet. Kontroller på manglende data, intervallkontroll, parameterkontroll o.a. skal kunne gjøres mot tabelldata i AL og skjermbilde-data som enda ikke er lagret (committed) i AL.

   Å legge mange kontroller på registreringsnivå vil kunne redusere arbeidsmengden i den generelle rutinen som færre iterasjoner av oppdateringer og kontroller. Ulempen med for mange kontroller på dette nivå er frustrasjon for den som registrerer ved at terminalen stadig gir ulyd fra seg og avbrytelser i registrerings-rytmen. Dette kan imidlertidig løses ved at en kan velge mellom å la maskinen kontrollere umiddelbart eller ved et senere tidspunkt før utlogging.

4. ML-FORMS-programmer laster data inn/ut av ML.
   Kontroller på registreringsnivå gjelder for ML som AL.

5. Når en gitt bolk med data er ferdigkontrollert i AL oppdateres EL.

6. Etterat pkt. 5 er utført overføres data fra AL til HL og slettes så i AL og ML.

7. Tilfeldige oppdateringer i HL.

8. Utplukk av data for sluttbruker gjøres fra utplukksprogrammet.
   Sluttbruker kan hente ut data fra IL, HL, EL og ML.

1. HOVEDLAGER

HL vil bestå av stasjonsvise tabeller i størrelsesorden 2000.
Krav til HL :

- kun ha kontrollerte data
- data som finnes i HL skal ikke ligge redundant
- ha 1 flagg (interpolert verdi eller ikke)
- kun ha 1 inputkanal
- få personer med skriverettigheter

Reorganisering av tabeller (spesielt SYNOP/AUTOMAT stasjoner)

Som tidligere beskrevet må en datablokk ha størrelsesorden 1 rad ledig plass for å kunne foreta en oppdatering (UPDATE) uten at blokksplitting oppstår. Modell 2 vil kun få blokksplitting ved historiske oppdateringer i motsetning til Modell 1 som vil få hyppige blokksplittinger p.g.a at ikke alle data for samme observasjon (rad) lastes samtidig.

Kontrollerte data

Med Modell 2 er det intet behov for sjekk om data er kontrollert eller ikke da alle data skal være kontrollert i HL. Dette kan ha positiv effekt på responstiden. Kun flagg for om data er interpolert er nødvendig. Færre flagg er plassbesparende.

2. ARBEIDSLAGER

AL vil bestå av stasjonstypevise tabeller i størrelsesorden 10. AL skal inneholde følgende data :

- synop/oppringte data
- registrerte data fra dokumenterte kilder
- dokumenterte interpolasjoner/oppdateringer

AL skal kun ha 2 outputkanaler utenom rutinen :

-> HL når data er ferdigkontrollert og OK
-> ML a) etterat data er registrert fra terminal
b) etterat data er oppdatert

Arbeidslager bør ha diverse flagg som beskriver :

- synop (synop lagret)
- registrering (registrering fra FORMS-reg)
- oppdatert (oppdatert fra FORMS-oppdat)
- kontrollert (kontrollert med diverse KONTROLL-program)
- ok (kontrollert med SISTEKONTROLL-program)

Færre tabellaksesser

Hvis en som eksempel overfører 1 observasjon for alle stasjoner som lastes både i AL og HL samtidig, vil det etter Modell 1 bli åpnet rundt 2000 tabeller. Ved overføring av samme data til AL og ML samtidig etter Modell 2, åpnes 2 tabeller.

3. MIDLERTIDIG LAGER

Når det gjelder SYNOP-stasjoner er det svært få med dagens system som får overført et fullstendig datasett over en gitt periode. På AUTOMAT-stasjoner kan problemene hardware, klokkefeil, brudd på linja o.a. Det vil si at disse stasjonstypene kan ha både mangelfulle og feilaktige sanntidsdata.

For å kunne tilfredsstille tidens krav fra omverdenen (media o.a.) om å skaffe tilveie statistikk umiddelbart, kan løsningen være å opprette et midlertidig lager hvor kravet til oppdatering ikke behøver å være så strengt. Disse oppdateringer vil aldri bli overført til AL eller HL, men bli overskrevet så snart dokumenterte oppdateringer og kontroller er gjort i AL. Data slettes i ML når samme data overføres fra AL til HL.

ML er i utgangspunktet det samme som AL. Periode og struktur skal

være den samme, men innholdet varierer. ML kunne være en felles tabell for VA og Klimaavdelingen. Dermed ville Klimaavdelingen slippe å interpolere om igjen data som VA kanskje allerede hadde interpolert.

4. ORACLE SQL*FORMS

Modell 2 bygger på bruk av FORMS. Ved å bruke FORMS reduseres egenprogrammeringen av et tilsvarende verktøy.

FORMS er et kommersielt registreringsvertøy mot databasen. FORMS er basert på at alle data innen 1 entitet ligger i en og samme tabell. Med MI's store datamengder er ikke dette mulig med krav om en rimelig god responstid. Ergo må stasjonsdata fordeles i stasjonsvise tabeller som vil medføre opprettelse av størrelsesorden 2000 tabeller.

Da FORMS ikke kan brukes mot variable tabellnavn, men binder seg fysisk mot eksisterende tabeller, vil det i MI's tilfelle si at det må opprettes like mange FORMS som tabeller en vil bruke FORMS mot.

Med en god del egenprogrammering i FORMS - ifølge Syverinsen fra ORACLE - skulle det være mulig å få FORMS til å knytte seg til et variabelt tabellnavn. Men med tanke på den tid og ressurser dette krever og dertil uvisst hva resultatet blir, vil det være spørsmål om et slikt prosjekt ville være gunstig å sette igang.

5. TILSYNELATENDE ULEMPE MED MODELL 2

Det er ikke fortatt noen uttesting av Modell 2. Modell 2 bygger på arkitekturen som brukes på SYNOP-stasjoner i dag.

Hva som kan betraktes som ulempe med Modell 2 er oppslag i 2 tabeller hvis en ønsker utplukk av data fra en periode som delvis ligger i HL og delvis i ML.

Hvis ML plasseres (på disk) i forhold til HL med tanke på optimal performance, er det uvisst om dette vil øke responstiden. Kanskje bruk av ML kan bedre performance ved samtidig IO på forskjellige disker.

Sanntidsdata eller data for den 'siste tiden' er hyppig brukt, den vil være av en kortere periode og ha krav om god performance (oftest som telefonforespørsler).

3. Teoretisk forarbeid for innlasting

Nedenfor en kort beskrivelse av begrepene indekser, statisk tabellstruktur i forhold til dynamisk, bruk av integritetsregler og CASE.

Begrepsforklaringer :

Indekser

Da valg av riktige indekser i de fleste tilfeller bedrer et programs performance, er det viktig å nøye vurdere hvilke data som skal indekseres. Indekser tar relativt stor plass og dette må tas i betraktning både når diskplass skal estimeres og når plassering av indeksene skal vurderes.

Tabellstruktur

Å finne fram til riktig tabellstruktur er viktig i ORACLE. Selvom ORACLE beskriver V6 som om felt ytterst på en rad ikke opptar plass, tar nevnte felt 1 byte. Tomme felt mellom utfylte felt tar også 1 byte. Derfor må tabellstrukturen vurderes med hensyn til både plass og vedlikehold. Riktig valg av struktur kan også ha stor effekt på performance.

Integritetsregler

Deklarativ referanseintegritet sørger for en engangs forhånds-definering av relasjoner mellom tabeller. Ved å beskrive sammenhengen mellom primærnøkler og fremmednøkler i selve databasen ved opprettelse av tabeller, vil databasen automatisk sørge for å holde dataene konsistente etter referanse-reglene. Andre forretningsregler som DEFAULT-verdier og CHECK-verdier o.a. sørger for at slike verdikontroller ikke behøver å legges inn i hver applikasjon.

CASE - verktøy

CASE-method, CASE-dictionary, CASE-designer og CASE-generator er ORACLE-produkter som kan være til hjelp for å automatisere utviklingen av et databasedesign i et ORACLE RDBMS miljø. Ved å bruke disse verktøy, kan en sikre seg at utviklingen av databasen skjer på en kontrollert måte og at ingenting blir glemt.

3.1 Integritetsforanstaltninger

1.0 INNLEDNING

Integritetsregler skal sikre at databasen er konsistent, at den ikke inneholder "gale data" (ulovlige verdier) og at det ikke forekommer redundans. Dette er meget viktig både ved bruk av (SELECT), og ved operasjoner (UPDATE, INSERT, DELETE) på dataene.

Regler som ivaretar ovenforstående kan alltid implementeres manuelt i applikasjoner (programmeres), men ORACLE gir mulighet for å legge disse reglene inn "sammen med dataene", direkte, i CREATE eller ALTER setningene. Enhver operasjon av data (INSERT, UPDATE, DELETE), enten interaktivt eller fra egenutviklede applikasjoner, vil da bli underkastet det samme "regelverk" uten at man behøver å tenke på dette selv.

ORACLE versjon 6 gir muligheter for å spesifisere en rekke integritetsregler i CREATE / ALTER setningene (ref. kap. 2), men bortsett fra NULL VALUE blir det ikke tatt hensyn til disse spesifiserte reglene i versjon 6 av databasen. I versjon 7 vil imidlertid disse reglene bli fulgt fullt ut.

SQL*FORMS v.3 kan i visse tilfeller (ved default skjermbilde-kreering) benytte integritetsreglene fra v.6 av databasen.

2.0 INTEGRITETSREGLENE

Integritetsreglene er:

*	**NOT NULL**	Krever at kolonnen er fylt ut med noe (verdi). Denne regelen blir utnyttet i v.6.
*	**PRIMARY KEY**	Spesifisering av primærnøkkel (kolonne(r) som entydig definerer raden).
*	**FOREIGN KEY**	Henvisning til en annen tabell. Fremmednøkkelen viser til primærnøkkelen i en annen tabell og krever at kolonnen(e) finnes her (har verdi), ellers må den ha NULL VALUE i fremmednøkkeltabellen.
*	**UNIQUE**	Krever at kolonnen(e) er unik innen tabellen. Dette kan i v.6 oppnås ved at det opprettes UNIQUE INDEX på kolonnen(e), men er ikke helt "renslig" da man jo kanskje må opprette en unødvendig index.
*	**CHECK**	Spesifiserer en liste av verdier / tekster, verdiområde (intervall), eksakt verdi / tekst eller produkt fra en oracle-funksjon.
*	**DEFAULT**	Kolonnen gis en default verdi.

PRIMARY KEY og UNIQUE kolonnen(e) må være NOT NULL. Antagelig impliserer PRIMARY KEY og UNIQUE direkte NOT NULL, men siden versjon 6 av databasen bare tar hensyn til NOT NULL regelen, må også denne spesifiseres eksplisitt sammen med de andre.

FOREIGN KEY må referere en PRIMARY KEY eller UNIQUE i "primærnøkkeltabellen". Hvis ikke sammenhengen (constraint) er angitt, refererer FOREIGN KEY primærnøkkelen. Ved bruk må man ha privilegium for å referere aktuell kolonne / tabell i "primærnøkkeltabellen". Datatypen må overensstemme. Hvis NULL VALUE er lovlig for kolonnen(e) i "fremmednøkkeltabellen", behøver ikke FOREIGN KEY å finnes som primærnøkkel.

Samtlige integritetsregler unntatt NULL VALUE kan enten gis som "tabell regel" eller som "kolonne regel". NULL VALUE kan kun gis som "kolonne regel". Virkningen av reglene er den samme om de gis som tabell eller kolonne regel. Den eneste forskjellen er hvor i CREATE setningen reglene angis. Kolonne reglene angis etter angivelsen av den enkelte kolonne (attributt) og kan kun referere dette, mens tabell reglene angis etter at alle kolonnene (attributtene) er spesifisert og kan referere hvilke(n) som helst kolonne(r) i tabellen.

Enhver regel får et eget navn. Dette kan spesifiseres når regelen lages (i CREATE / ALTER) setningen. Hvis ikke noe navn spesifiseres, vil regelen få et navn som ORACLE bygger opp.

3.0 MULIGHETER FOR SENERE IMPLEMENTERING / ENDRING AV INTEGRITETSREGLENE.

I utgangspunktet er integritetsreglene ment å følge datamodellen. Det vil derfor være naturlig å angi integritetsreglene samtidig med at man genererer datastrukturen (i CREATE setningene). Det er også mulig å angi kolonne-regler når man legger til nye kolonner (ADD) i ALTER tabell setningen. Nye integritetsregler, eller endring av allerede spesifiserte regler, kan også legges til i ALTER tabell setningen (med ADD) uten at nye kolonner spesifiseres. Disse integritetsreglene må legges inn som tabell-regler.

Integritetsreglene kan slettes i ALTER setningen med DROP CONSTRAINT. Dette vil normalt være den eneste gangen man må angi regelens navn, og derved ha glede av logisk, egendefinerte navn etter et kjent mønster. Navnet vil imidlertid alltid kunne finnes i ORACLE DICTIONARY.

4.0 KONSEKVENSER VED INNLEGGING / BRUK AV INTEGRITETSREGLER

Konsekvenser ved å **nå** legge inn integritetsregler vil være:

- at de enkelte tabellene må gås gjennom slik at integritetsregler kan bli spesifisert. Spesielt for CHECK regelen kan dette antagelig bety en del arbeid.
- at CREATE setningene vil bli noe lenger.

Konsekvenser ved **bruk** av integritetsregler vil være:

- Ingen foreløpig, da v.6 ikke tar hensyn til integritetsreglene selv om de er generert. UNNTATT for NULL VALUE da denne blir tatt hensyn til i v.6. Alle kolonner har i utgangspunktet (default) NULL VALUE; dvs. at de kan inneholde NULL VALUE. Ved å spesifisere NOT NULL VALUE forlanger man at kolonnen får en verdi for hver enkelt rad. Dette vil være spesielt aktuelt for primærnøkkelkolonnen(e). Ved i tillegg å spesifisere UNIQUE INDEX på denne / disse kolonnen(e) vil man oppnå at hver rad i tabellen er entydig identifisert. Dette innebærer det samme i v.6 som PRIMARY KEY senere (i v.7) vil medføre.

- Det er helt klart at bruk av integritetsregler (når dette blir mulig (v.7)), vil sikre at dataene ikke antar "spinn ville" verdier, at man unngår redundant lagring av data og at "databasen henger sammen" (f.eks. at data fra en stasjon ikke kan legges inn uten at stasjonen er registrert i informasjonsarkivet). Disse forhold vil også kunne sikres gjennom applikasjoner, men muligheten for å legge inn data uten å gå gjennom disse vil alltid være til stedet.

- SQL*FORMS kan utnytte integritetsregler som er lagt inn i databasen ved at det automatisk genereres triggere (prosedyrer) eller attributter i FORMS-applikasjonen som sørger for at integritetsreglene overholdes. Data som kommer inn på ulike måter (f.eks. både gjennom SQL*FORMS og SQL*LOADER eller C program), vil da ikke nødvendigvis underkastes de samme regler.

- Det er usikkert hvordan bruk av integritetsreglene (v.7) vil påvirke responstider ved INSERT, UPDATE og DELETE, men at den vil øke, er det vel ikke tvil om!

5.0 VURDERING AV FORDELER / ULEMPER NÅ (V.6) OG SIDEN (V.7) SAMT ANBEFALING.

Etter å ha gått gjennom integritetsreglene er vi ikke i tvil om at bruk av samtlige av disse vil være svært nyttig (nødvendig ?) i en stor database som vår, med høye krav til "datariktighet". Kanskje bør dette overordnes eventuelle problemer med lenger responstider som heller forsøkes løst på andre måter, og tas i bruk "uansett" ?

Når det gjelder implementasjon av integritetsregler nå vil fordelene være at vi nå genererer databasen og derved er "godt inne i datamodellen og forholdene her", og "like vel" utfører CREATE setningene hvor jo integritetsreglene angis.

Integritetsreglene vil dessuten kunne bli utnyttet i SQL*FORMS applikasjoner som utvikles.

Ulemper ved å implementere integritetsregler nå vil være at integritetsregler må spesifiseres og det vil kanskje gå noe mere tid før vi har en database klar. Vi kan dessuten ikke teste ut integritetsreglene i praksis, noe som kan bety at vi får uforutsette problemer / feil ved en overgang til v.7 av databasen ("alt burde jo

funket uten videre !").

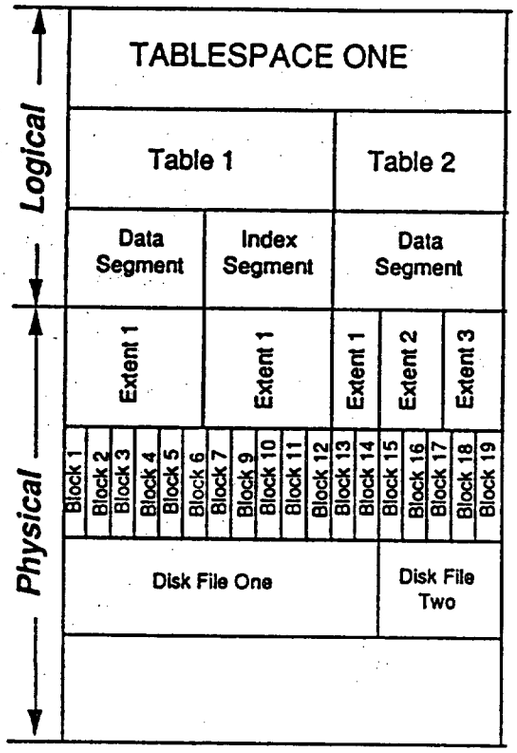
Det synes forholdsvis enkelt å implementere integritetsregler senere (ved ALTER setningen), da de også kan testes ut. Ved å utsette implementeringen til senere vil vi nå kunne konsentrere oss om andre ting (f.eks. om å få generert databasen, lagt over data og applikasjonene,.....), og det kan senere nedsettes et eget prosjekt som bare ser på implementering av integritetsreglene og uttesting av dette.

Vi mener integritetsregler absolutt bør implementeres i database, men at dette godt kan utsettes til vi får v.7 av ORACLE som benytter integritetsreglene. Dette fordi det synes forholdsvis enkelt å legge inn integritetsregler senere (ALTER tabell) og fordi vi "nå egentlig har mere enn nok annet å tenke på" og dessuten ikke kan få testet ut integritetsreglene enda.

Det må imidlertid vurderes ved bruk av SQL*FORMS, om integritetsregler likevel bør implementeres i v.6 av databasen, spesielt for IL og AL. Dette bør prosjekt 7.1 "Data inn" ta i betraktning for SQL*FORMS applikasjoner og evt. legge inn integritetsregler i databasen v.6 for IL og AL.

Når det gjelder NULL VALUE mener vi det for alle primærnøkkelkolonner bør spesifiseres NOT NULL og genereres UNIQUE INDEX slik at vi kan finne dataene våre igjen og at vi ikke får redundante data. Dette anser vi for fundamentalt viktig og da dette vil få umiddelbar virkning (i v.6) bør det implementeres for alle tabeller nå i det databasen genereres (i CREATE tabell).

3.2 ORACLE-struktur



Definisjoner:

TABLESPACE	Logisk lagring av flere objekter som tabeller, indekser, rollback segments, temporary segments - spre IO på flere disker - skille datasegment fra indekssegment - kan tas 'offline' - forskjellige tablespases til forskjellig bruk - backup unit
SEGMENT	logisk lagring av ett objekt
EXTENT	contigues space for et segment (INITIAL, NEXT)
PCTINCREASE	størrelse i % på NEXT extent i forhold til INITIAL
PCTFREE	avsatt plass til utvidelse av allerede lagrede rader i en blokk
PCTUSED	hvor mye i % av en blokk som må slettes før nye rader kan lastes inn i gitt blokk

1. ORACLE tabellstruktur

ORACLE's tabellstruktur består av rader (observasjoner) og felt (parametre). Rekkefølgen av feltene defineres når tabellen opprettes.

2E har konkludert med at stasjoner grupperes i stasjonstyper og at alle stasjoner tilhørende gitt stasjonstype skal ha definert et fast antall felt - statisk struktur. Men innenfor denne statiske strukturen, kan rekkefølgen være variabel.

Blant stasjoner som er definert under samme stasjonstype kan antall observerte parametre variere fra 10 til 50. Dette vil si at stasjoner under en stasjonstype som totalt observerer mange parametre, kan ha mange forkomster av tomme felt som vil oppta plass.

For å frigjøre denne dødplassen, er det mulig å omgå problemet ved ta en ALTER-tabell kommando. For å få det til må tabellen først opprettes etter stasjonstypestrukturen, dernest må 'stasjonsvis struktur' (felt med verdier sorteres først i raden og tomme felt til slutt) defineres og til slutt må tabellen reorganiseres (ALTER-table).

Det kan synes merkelig at en reorganisering må til for å fjerne dødplassen som består av NULL-felt sist på en rad, men dette er bare sånn.

Ved en reorganisering som nevnt ovenfor oppnås en bedre utnyttelse diskplass som igjen kan bidra til bedre performance.

2. Tablespace

Med tanke på performance bør en spre diskaksessene mot flere disker for samtidig IO. Dette kan oppnås ved å fordele tablespacene som inneholder IL, HL og tilhørende indekser på forskjellige disker.

Et tablespace kan spenne over 1 til flere disker. En ulempe er at hvis en disk i et tablespace som spenner over flere disker skulle kræsje, blir hele tablespacet satt ut av drift inntil den ødelagte disken er erstattet.

Tablespace er enhet for backup. Hvis et tablespace spenner over flere disker, kreves det like mange backupdisker tilgjengelig samtidig og jobben tar forholdsvis lengre tid.

Alt dette tatt i betraktning er gruppens konklusjon er som følger :

- De meteorologiske data bør sorteres stasjonstypevis i egne tablespace. På hver disk bør det opprettes 1 datatablespace og 1 indekstablespace. Indekser og data for samme stasjon skal spres på forskjellige disker. Dette vil redusere samtidig IO. IL-tablespacet bør ikke ligge på samme disk som HL eller dens indekser.

  Samme regler bør også gjelde for AL og dens indekser sålangt det er mulig med hensyn til antall disker tilgjengelig. Men HL må prioriteres med hensyn til performance.

En ulempe med dette valget er at dersom en disk kræsjer, vil dette indirekte føre til performance-reduksjon på andre tablespace grunnet krysslagring av indekser. Dette er ikke noe stort problem da indekser kan regenereres utfra datagrunnlaget.

3. Storageparametre

INITIAL og NEXT EXTENT kan initialiseres etter behov. Å opprette store extent medfører at det vil avsettes mye ubrukt plass. Fordelen med å avsette god plass er å hindre fragmentering av tabeller. Fragmentering vil øke IO og dermed også responstiden. Bare en reorganisering kan rydde opp i fragmenterte tabeller. Reorganisering innen 1 disk er enkelt, men reorganisering av hele databasen er en omfattende oppgave.

Gruppens konklusjon er som følger :

INITIAL EXTENT - bør være stort nok til å romme historiske data pluss data for 3 år framover.
NEXT EXTENT - bør være stort nok til å romme datamengde for 1 år.
Nye EXTENT - (nyopprettede stasjoner) som INITIAL EXTENT.

De samme forhold gjelder indeksene.

PCTINCREASE - 0%

PCTFREE - En hvilken som helst oppdatering innen 1 rad i en blokk krever at det finnes ledig plass som tilsvarer minst 1 rads størrelse i blokken for å unngå blokksplitting. Vi vet det vil bli tilfeldige oppdateringer i mange tabeller. Men istedenfor å avsette størrelsesorden 1 rad for oppdateringer som vil bli mange KB dødplass totalt, kan en reorganisering ved behov være en bedre løsning.

PCTFREE - 0% ved innlasting av historiske data.

PCTFREE for nye data - bør settes i forhold til hvordan data lastes inn. Hvis alle data i en rad lastes inn samtidig, er det nok å sette PCTFREE til størrelsesorden 1 rad - hvis tilfeldig innlasting bør PCTFREE settes til 40-50% av blokkstørrelse.

PCTUSED - 0% for den historiske delen.

Etterat den historiske delen er lastet inn forandres PCTUSED til standardstørrelse som er 40%. Dette kan skape en del dødplass, men effektiviserer innlasting av data. PCTUSED bør revurderes etter at det er tatt standpunkt til hvilken dataflyt som blir valgt.

ORACLE BLOKKSTØRRELSE - 8KB.

Blokkstørrelsen er pr. i dag satt til 4095 bytes som er en standardstørrelse . Eksempelvis tilsvarer dette plass for 1 måned med SYNOP-data. Hvis det som oftest hentes ut flere måneder med data om gangen, kan det lønne seg å sette blokkstørrelse relativt stor. Med det spares overhead i den logiske blokken, og IO speedes opp. Blokkstørrelsen defineres når databasen opprettes og kan ikke forandres.

4. Andre storageparametre

Delprosjekt 3 har tatt hensyn til storageparametre som har direkte tilknytning til lagring av data og indekser.

Spesifisering av tablespacene SYSTEM, ROLL, TOOL og TEMP, ROLLBACK SEGMENT og CONTROL FILER og andre parametre vil videreført av delprosjekt 7.3.

3.3 Konstruksjon av informasjonslager ved hjelp av CASE

Hensikten med dette kapittelet er å redegjøre for de nødvendige og tilstrekkelige deler av informasjonslageret til bruk for innlasting av data.

Kapittelets første del tar for seg spesifikasjoner og utvalg basert på spesifikasjoner i [3], mens annen del dokumenterer konstruksjonen blant annet ved hjelp av skjermbilder og automatisk genererte delrapporter fra henholdsvis CASE*Designer og CASE*Dictionary.

1. Spesifikasjoner

1.1. Valg av entiteter

Prosjektgruppen har kommet frem til at entitetene STASJON, STASJONSTYPE og PARAMETER skal implementeres, mens de vurderte entitetene GEOGRAFISK_INFO, ENHET, UTSTYR, INSTRUMENTER, METODE, KODE skal ligge inntil videre. De kommuniserende (mellomliggende) entitetene STASJON-AV-TYPE og PARAMETER-PÅ-STASJON skal også implementeres. I tillegg til entitetene i [3] har vi funnet det nødvendig å ta med entiteter PARAMETER_FOR_STASJONSTYPE, INNLESNINGSFORMAT og INNLESNINGSFORMAT_FOR_STASJONSTYPE.

1.2. Spesifikasjon av entiteter (attributter)

Avsnittet har til hensikt å spesifisere entitetene med hensyn på attributter (tabell-kolonner).

STASJON skal inneholde attributtene

- stnr (stasjonsnummer - primærnøkkel)
- navn
- synopnr
- historisknr
- addresse
- godsaddresse
- tlf
- matrikkelnr (gårds/bruksnr)
- gardsnavn
- kommentar

STASJONSTYPE skal ha følgende attributtliste:

- Hgr (hovedgruppe - basert på inndelingen K, N, P, M, A, E, L, R, F fra [4])
- Ugr (Undergruppe - Vær, Klima, Linke, Fordampning, Nedbør, Plumatic, Maritim, Bøye, Edas, Scanmatic, Campbell, DNMI_spec, E-data, Aanderaa, Radiosonde, Flyvær, Metar - også fra [4])
- Beskrivelse (tekststreng)

Kommentar: Hovedgruppen skiller mellom forskjellige tabeller modulo parametervalg, mens undergruppen skiller med hensyn på stasjonsfunksjon.

STASJON-AV-TYPE skal ha følgende attributtliste:

- stnr (primærnøkkel for stasjon)
- ugr (primærnøkkel for stasjonstype)
- hgr
- init_dato (initiering av driftsperiode - f.o.m. dato)
- term_dato (terminering av driftsperiode - t.o.m. dato)

Attributtene stnr, ugr pg init_dato utgjør en nøkkel for STASJON-AV-TYPE.

PARAMETER skal ha følgende attributtliste:

- para (forkortet standardisert navn - primærnøkkel)
- fullst_navn
- beskrivelse

PARAMETER-PÅ-STASJON skal inneholde attributtlisten

- stnr (primærnøkkel for stasjonsentiteten)
- para (nøkkel for parameterintiteten)
- init_dato (initiering av observasjonsperiode - f.o.m. dato)
- term_dato (terminering av observasjonsperiode - t.o.m. dato)
- H (måling er tatt i gitt avstand fra bakkenivå)

- kommentar

De tre første attributtene danner en nøkkel for PARAMETER-PÅ-STASJON

PARAMETER_FOR_STASJONSTYPE skal inneholde attributtlisten

- para (primærnøkkel parameter)
- hgr (primærnøkkel stasjonstype)
- lager (AL/HL)
- type (parameterformat for oracle-tabeller, eks number(3,1))
- bgint (nedre grense for intervallkontroll mot HL)
- endint (øvre grense for intervallkontroll mot HL)

INNLESNINGSFORMAT skal inneholde attributtene

- filnavn
- hgr
- lager (AL/HL)

INNLESNINGSFORMAT_FOR_STASJONSTYPE skal inneholde attributtene

- filnavn
- hgr
- lager (AL/HL)
- fom (fra og med, dvs. brukes for å spesifisere innlesningsformat som eksempelvis skal brukes fra 1957 til 1981)
- tom

1.3. Spesifikasjon av relasjoner

Ved bruk av CASE-verktøy kreves det at relasjonene spesifiseres etter mønsteret

```
    xxx
A >-------------- - - - - - - - - B
                          yyy
```

"Hver A må være xxx en og bare en B, hver B kan være yyy en eller flere A", hvor xxx og yyy typisk er preposisjoner.

Vi har funnet det uhensiktsmessig å gå i detaljer her. En effektiv spesifikasjon er å substituere xxx og yyy med "relatert til" da denne typen informasjon synes overfødig til delprosjektets formål.

1.4. Eksempler

Kun et utvalg av de faktiske kolonner er vist:

STASJON-AV-TYPE:

STNR	STASJONSNAVN	F.O.M DATO	T.O.M DATO	TYPE
18700	Oslo-Blindern	1951.01.01	1965.05.31	Nedbør
18700	Oslo-Blindern	1965.06.01		Klima
39170	Bergen-Fredriksberg	1957.09.30	1975.09.30	Klima

Kommentar: F.O.M. DATO brukes som header for attributtet init_dato, TYPE for ugr.

PARAMETER-PÅ-STASJON:

STNR	T.O.M.DATO	F.O.M.DATO	PAR
18700	1951.01.01	1961.02.28	PO
18700	1951.01.01	1961.02.28	P
...			
18700	1951.01.01	1961.02.28	RR
18700	1961.03.01	1983.01.31	PO
...			
18700	1961.03.01	1983.01.31	RR
18700	1983.02.28		TT
...			
18700	1983.02.28		RR
39700	1951.01.01	1983.01.31	PO
...			
39700	1951.01.01	1983.01.31	RR

1.5. Effektivisering av oppslag for parameterinformasjon

Radvis lagring av parameterinformasjon kan synes lite effektivt for visse formål. I praktisk bruk av informasjonslageret kan det tenkes at logiske ("true/false") hjelpetabeller med parametre indikert kolonnevis vil være fornuftig.

2. Implementasjon ved hjelp av CASE

Implementasjonen har fulgt CASE*Method dokumentert i ORACLEs CASE-litteratur. Paragrafene nedenfor behandler de tre stadier i CASE*Method som ble benyttet.

Utviklingsmiljø benyttet på typhoon er:

~kabase/src/IL/BASICS

(sml. forøvrig [5])

2.1. Strategi

CASE*Dictionarys entiteter og relasjoner ble initiert gjennom CASE*Designer.

Før diagramgenereringen kunne begynne måtte man spesifisere applikasjon via CASE*Dictionary (figur 3.3.1) Navnekonvensjonen INFARK er benyttet som applikasjonsnavn for informasjonsarkivet.

Entitetene benyttet seg av navn spesifisert ovenfor, med unntak av ord inneholdende bokstaven "Å" som ble byttet ut med "AA", og bindestrek byttet ut med "underscore".

Entitetsrelasjons-diagrammet (ER-diagrammet) følger vedlagt (figur 3.3.2). Vedlegget er et 'screendump', da det ser ut til at de ordinære output-rutinene ikke fungerer slik som de skal.

2.2. Tabell-design

Design av tabell er gjort via default opsjoner, dvs en-en-tydig korrespondanse mellom entiteter-tabeller og attributter-kolonner.

2.3. Implementasjon

For at implementasjonsmenyen skal fungere må man passe på at "create-status" for tabellene er satt under design-fasen. Selve implementasjonen foregår ved at man genererer tabell for tabell gjennom en spørre-meny. CASE dictionary vil da generere passende SQL-kommandofiler og kjøre disse etter valgfritt ønske.

Figur 3.3.1

APPLICATION SYSTEM DEFINITION

Application : INFARK Parent Application :
Version : 1 Version :

Title : Informasjonsarkiv

Description : Modell for lagring av informasjon om stasjonene

Objectives :

Priorities :

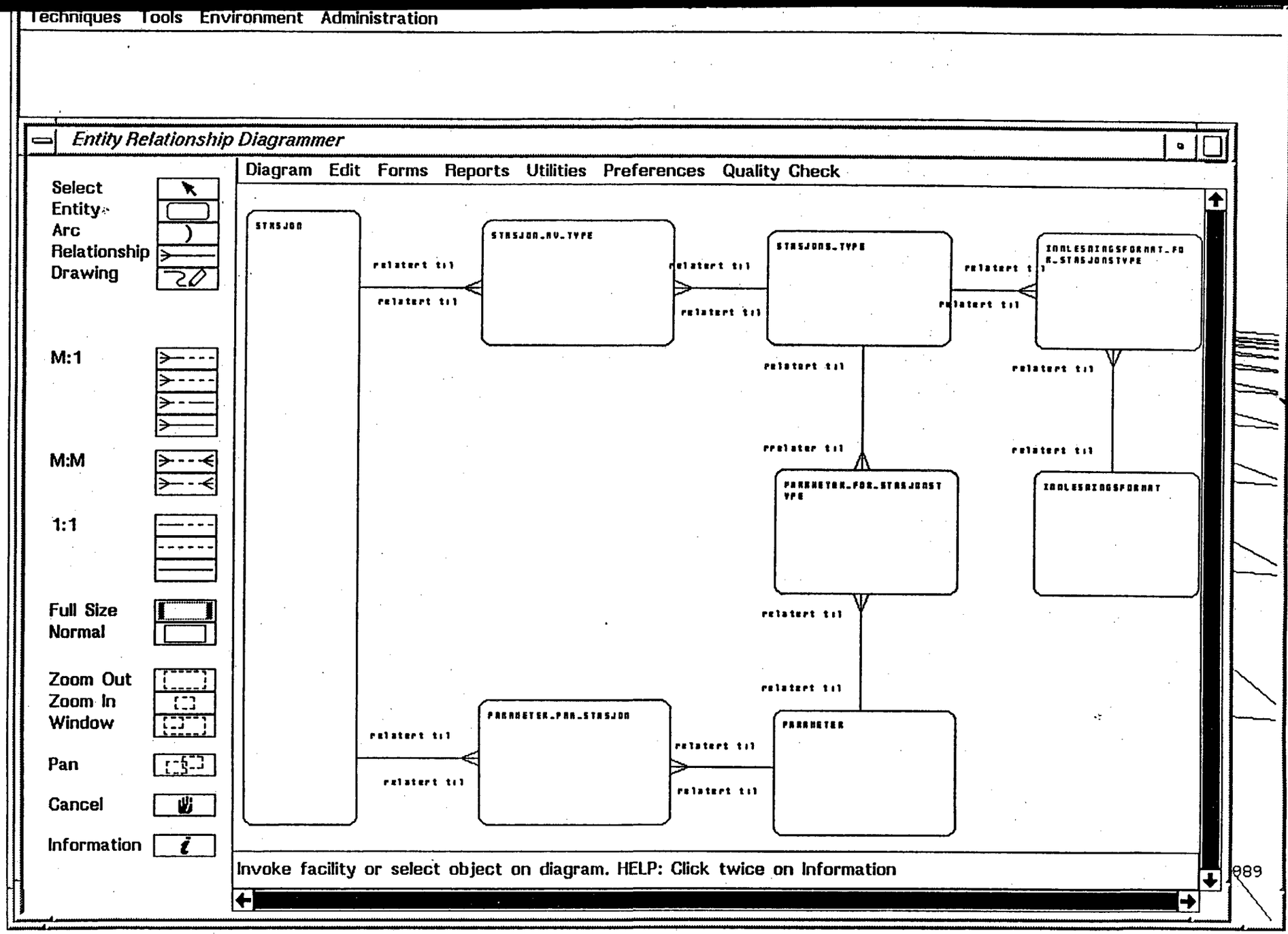
Constraints :

Comment :

Authority :

Owner : MATS

Status :

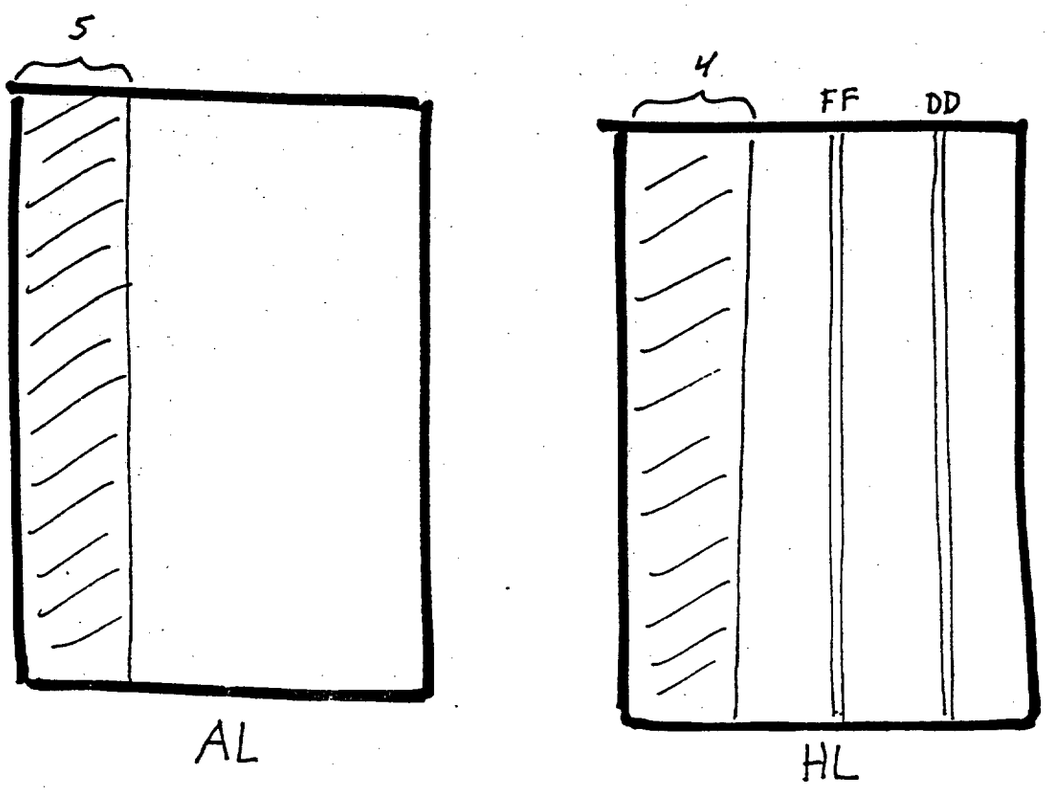

Techniques Tools Environment Administration
Entity Relationship Diagrammer
Diagram Edit Forms Reports Utilities Preferences Quality Check
Select
Entity
Arc
Relationship
Drawing
M:1
M:M
1:1
Full Size
Normal
Zoom Out
Zoom In
Window
Pan
Cancel
Information
STASJON
STASJON_AV_TYPE
STASJONS_TYPE
PARAMETER_FOR_STASJONST YPE
PARAMETER_PAR_STASJON
PARAMETER
relatert til
Invoke facility or select object on diagram. HELP: Click twice on Information

Figur 3.3.2

3.4 Indeksering

1. Innledning

Arbeidslager og hovedlager skal benyttes for forskjellige formål, og må også indekseres på forskjellig måte (figur 3.4.1).



Figur 3.4.1 Indeksering i AL og HL

Det endelige valg av indekser i begge lagre er avhengig av responstid for applikasjoner definert under delprosjekt 7.2, 7.3 og 7.4, delprosjekt 3 har imidlertid funnet det naturlig å gjøre noen vurderinger av bruk av indekser på basis av visse eksperimenter.

I den applikasjonsavhengige hovedlagertabellen ønsker man å indeksere flest mulig av de parametre som inngår i applikasjonen (f.eks. vindhastighet og vindretning i en vindrose-applikasjon).

I det applikasjons-uavhengige arbeidslageret vet man ikke apriori hvilke parametre som bør være indeksert i tillegg til primærnøkkelen. I utgangspunktet er alle parametre like interessante med hensyn på kontroll.

2. Valg av indekser til arbeidslager

Det fins en øvre grense for hvor mange parametre man kan ta med dersom man ønsker en rimelig responstid. Erfaring tyder på at denne grensen ligger rundt 5-6 parametre.

Det naturlige valg av indekser blir da arbeidslagerets primærnøkkel (stnr, aar, mnd, dag, tim). Man må undersøke om alle kolonner i nøkkelen skal være med, og eventuelt i hvilken rekkefølge disse bør stå.

3. Optimalt indekseringsutvalg

Følgende eksperimenter er utført på en tabell m2222 (se tabell 3.4.1 nedenfor) bestående av 448953 rader. Tabellen inneholder data fra ti stasjoner, med et variabelt antall år for hver stasjon:

STNR	AAR
604	1957-91
1870	1951-91
3910	1951-91
4730	1951-91
5523	1957-78
6910	1957-91
8229	1957-91
9045	1951-91
9855	1951-91
9995	1956-89

Eksperimentet består av 326 indeks-kombinasjoner (alle distinkte utvalg fra en populasjon på 5 individer), hvor eksekveringstiden for create-index-setningen og første select-setning med ny index er registrert. Midlet eksekveringstid for de fire derpåfølgende select-setninger og den assosierte varians er også målt.

Setningen som ble testen denne gangen var "SELECT STNR, AAR, MND, DAG, TIM, TT FROM M2222 WHERE STNR=1870 AND AAR=1980 AND MND=3 AND DAG=1". Oppslaget returnerer tre rader.

En utskrift av resulatene følger i tabell 3.4.1 nedenfor.

Tabell 3.4.2 gir en sortering av de 50 beste indekseringer med hensyn på middelverdien av kolonnene First og Expect, dvs sortering med hensyn på tidsforbruk sampel-middel av oppslag 1,2,3,4 og 5.

Av tabellen ser vi at full indeksering av primærnøkkelen er å foretrekke, og kombinasjoner med kolonnen aar først kommer særlig godt ut. Alle bruker kun brøkdelen av et sekund på å gjøre de ønskede oppslag.

Kvaliteten av estimatene bekreftes gjennom variansen. Det kan leses av tabellen at det er ytterst lite variasjon

når det gjelder tidsforbruk for senere oppslag etter at det første er gjort.

Som en kuriositet kan man se at outsiderene (aar,dag), (aar, mnd) og (dag, mnd) heller ikke er å forakte når det taes i betraktning av de kun benytter seg av to kolonner. Create-tidsforbruket for disse er vesentlig mindre enn for de større indekseringer.

På basis av indekstestene har prosjektgruppen valgt å indeksere arbeidslager med (aar, mnd, stnr, tim, dag) for samtlige ni tabeller.

4. Indeksering av hovedlager

I de forbindelse med konstruksjon av tidseffektive applikasjoner kan det være naturlig å vurdere bruk av hjelpetabeller med egen indeksering (problemspesifikk indeksering). Ønsker man f.eks. å beregne vindroser på basis av synop-data kan det tenkes fornuftig å automatisk dumpe over de ønskede datamengder fra HL på hjelpetabeller, automatisk generere passende indeksering og starte opp analyserutinene.

Uttesting i forbindelse med bruk av hjelptabeller med indeksering i HL må testes og sammenliknes mot tilsvarende tester uten ekstra indeksering.

To hovedstrategier kan synes naturlig for HL:

- Hovedlagerets primærnøkkel pluss de hyppigst brukte meteorologiske parametre indekseres. Man benytter da hyppighetsentitetene i informasjonslageret omtalt i kapittel 2.

- Hovedlageret indekseres på (stnr, DATE, tim) pluss de hyppigst brukte meteorologiske parametre. Ved å bruke DATE reduserer man antall indekserte kolonner. Det bør undersøkes om det er noen effektivitets-gevinst på dette.

Hvilke parametre som bør indekseres i tillegg til nøkkelparametre er ikke mulig å avgjøre uten utstrakt eksperimentering med ferdig-implementerte applikasjoner (prosjekt 7), så kun uttesting av arbeidslageret indekser har vært mulig under prosjekt 3.

Tabell 3.4.1

dex	Create	First	Expect.	Variance
(-)	0.0000	109.0166	67.6708	1.8159
(stnr)	147.3000	264.3000	35.4333	0.2375
(aar)	162.1000	9.2833	8.4708	0.0276
(mnd)	251.7166	251.3833	30.5375	0.4468
(dag)	255.9166	398.3500	22.3083	0.0480
(tim)	349.0833	96.2000	92.5166	403.7867
(stnr,mnd)	303.5166	19.1833	3.3083	0.1360
(stnr,dag)	567.6333	3.9833	1.2541	0.0002
(stnr,tim)	336.8500	365.6000	36.9542	0.0156
(aar,stnr)	198.9666	16.4500	1.0166	0.0001
(aar,mnd)	212.4500	0.8500	0.7583	0.0001
(aar,dag)	259.2333	0.6500	0.3417	0.0001
(aar,tim)	295.3000	73.6833	8.7792	0.0002
(mnd,stnr)	304.7833	23.7500	3.1625	0.0103
(mnd,aar)	308.6000	0.9000	0.7791	0.0002
(mnd,dag)	302.2000	48.0000	1.0749	0.0001
(mnd,tim)	381.1000	364.0333	31.7416	0.0823
(dag,stnr)	356.1833	98.0500	1.4250	0.0067
(dag,aar)	346.1666	1.0166	0.3625	0.0011
(dag,mnd)	314.9500	46.5000	1.2999	0.0039
(dag,tim)	389.5500	548.7833	54.0333	2.3928
(tim,stnr)	354.5500	137.6833	93.7708	172.2992
(tim,aar)	303.8000	141.3833	89.2208	110.0571
(tim,mnd)	311.2500	88.8500	84.3749	47.7124
(tim,dag)	323.1000	89.3166	83.0666	70.7987
(stnr,aar,mnd)	484.3833	1.6000	0.1124	0.0000
(stnr,aar,dag)	524.5166	4.6000	0.0666	0.0000
(stnr,aar,tim)	542.4833	19.1000	1.2208	0.0022
(stnr,mnd,aar)	833.2666	4.6833	0.1333	0.0005
(stnr,mnd,dag)	1121.1333	21.0166	0.1542	0.0001
(stnr,mnd,tim)	879.0833	83.5666	3.9291	0.0688
(stnr,dag,aar)	820.2333	4.8500	0.0750	0.0001
(stnr,dag,mnd)	797.0666	18.0333	0.1542	0.0001
(stnr,dag,tim)	801.8000	215.9666	1.5541	0.0288
(stnr,tim,aar)	622.6166	111.9166	3.0833	0.0005
(stnr,tim,mnd)	797.9500	131.3333	5.6208	0.0454
(stnr,tim,dag)	804.5500	261.2000	3.2625	0.0002
(aar,stnr,mnd)	422.5833	1.9500	0.1208	0.0001
(aar,stnr,dag)	564.3666	4.8333	0.0708	0.0001
(aar,stnr,tim)	586.7166	18.0666	1.0333	0.0032
(aar,mnd,stnr)	521.6500	1.7666	0.1166	0.0000
(aar,mnd,dag)	477.0166	3.7333	0.0666	0.0000
(aar,mnd,tim)	525.6000	20.1333	0.8708	0.0002
(aar,dag,stnr)	557.0166	4.0833	0.0708	0.0000
(aar,dag,mnd)	522.4000	4.7166	0.0666	0.0000
(aar,dag,tim)	520.1666	39.6000	0.3791	0.0001
(aar,tim,stnr)	566.3333	20.9833	1.5250	0.0008
(aar,tim,mnd)	538.0666	16.5666	1.3124	0.0001
(aar,tim,dag)	919.3666	1.5166	0.8291	0.0001
(mnd,stnr,aar)	879.5000	1.6333	0.1208	0.0001
(mnd,stnr,dag)	827.0000	17.7833	0.1542	0.0001
(mnd,stnr,tim)	851.5833	76.3833	3.5708	0.0013
(mnd,aar,stnr)	859.7166	2.0833	0.1291	0.0001
(mnd,aar,dag)	816.2666	3.8833	0.0666	0.0000
(mnd,aar,tim)	838.4500	17.5666	0.8166	0.0002
(mnd,dag,stnr)	813.1500	19.0833	0.1417	0.0001
(mnd,dag,aar)	805.0666	3.6833	0.0666	0.0000
(mnd,dag,tim)	355.4500	127.1000	1.4375	0.1439
(mnd,tim,stnr)	1065.2500	79.5500	5.2541	0.0021

(mnd,tim,aar)	860.6833	18.5500	2.8541	0.0005
(mnd,tim,dag)	363.5000	138.6333	2.8333	0.0017
(dag,stnr,aar)	902.0500	4.7166	0.0875	0.0001
(dag,stnr,mnd)	809.9833	18.7833	0.1583	0.0001
(dag,stnr,tim)	851.3333	199.3666	1.4833	0.0003
(dag,aar,stnr)	880.9666	6.8333	0.0666	0.0000
(dag,aar,mnd)	837.2166	5.2500	0.0708	0.0000
(dag,aar,tim)	835.1166	46.1666	0.3916	0.0003
(dag,mnd,stnr)	832.8833	16.6166	0.1583	0.0001
(dag,mnd,aar)	865.6166	3.8333	0.0708	0.0001
(dag,mnd,tim)	367.1000	121.8333	1.2416	0.0001
(dag,tim,stnr)	861.3666	226.9500	2.0791	0.0004
(dag,tim,aar)	842.6000	77.2666	1.2083	0.0001
(dag,tim,mnd)	365.2833	117.9333	1.9625	0.0006
(tim,stnr,aar)	723.3666	454.7500	72.8374	0.7714
(tim,stnr,mnd)	740.8500	361.0333	71.3125	1.3470
(tim,stnr,dag)	838.0166	402.2166	72.4292	0.5418
(tim,aar,stnr)	768.6000	458.8833	81.0292	98.9264
(tim,aar,mnd)	1235.6000	92.5333	72.0041	0.5412
(tim,aar,dag)	742.8666	360.2000	71.5208	0.6073
(tim,mnd,stnr)	729.1833	386.0000	71.1958	0.4418
(tim,mnd,aar)	782.8666	344.9666	90.1125	87.0664
(tim,mnd,dag)	357.0500	232.3333	71.7042	0.4999
(tim,dag,stnr)	851.3666	369.7833	71.9291	0.2961
(tim,dag,aar)	847.1166	377.9500	71.4625	0.2997
(tim,dag,mnd)	356.9000	197.5833	71.8916	0.3725
(stnr,aar,mnd,dag)	649.8500	1.2500	0.0500	0.0000
(stnr,aar,mnd,tim)	705.3666	3.0166	0.1458	0.0001
(stnr,aar,dag,mnd)	698.0666	1.3333	0.0708	0.0003
(stnr,aar,dag,tim)	704.5166	6.8500	0.1083	0.0001
(stnr,aar,tim,mnd)	700.8166	5.7500	0.2000	0.0000
(stnr,aar,tim,dag)	1009.3333	5.7833	0.1542	0.0001
(stnr,mnd,aar,dag)	1190.0500	1.3500	0.0583	0.0001
(stnr,mnd,aar,tim)	1124.1333	3.3500	0.1458	0.0001
(stnr,mnd,dag,aar)	1149.9666	1.5000	0.0875	0.0002
(stnr,mnd,dag,tim)	837.4333	18.7166	0.1833	0.0000
(stnr,mnd,tim,aar)	1134.0500	14.2500	0.5166	0.0513
(stnr,mnd,tim,dag)	953.9666	30.6166	0.4208	0.0017
(stnr,dag,aar,mnd)	1104.7000	1.4000	0.0791	0.0006
(stnr,dag,aar,tim)	1595.9500	6.2666	0.0917	0.0001
(stnr,dag,mnd,aar)	1097.3000	1.3000	0.0666	0.0001
(stnr,dag,mnd,tim)	914.9166	15.8833	0.1750	0.0001
(stnr,dag,tim,aar)	1150.6500	11.1000	0.2167	0.0028
(stnr,dag,tim,mnd)	925.2000	24.1166	0.2708	0.0002
(stnr,tim,aar,mnd)	1054.7666	138.9000	2.5291	0.0029
(stnr,tim,aar,dag)	1106.1500	140.4000	2.4167	0.0061
(stnr,tim,mnd,aar)	1130.8167	142.1833	2.5083	0.0078
(stnr,tim,mnd,dag)	849.1500	151.2333	2.4416	0.0062
(stnr,tim,dag,aar)	1160.6000	144.0500	2.4166	0.0061
(stnr,tim,dag,mnd)	942.2333	146.6000	2.4500	0.0132
(aar,stnr,mnd,dag)	689.2500	0.5000	0.0583	0.0001
(aar,stnr,mnd,tim)	740.3500	1.9500	0.1417	0.0001
(aar,stnr,dag,mnd)	737.8000	0.4166	0.0542	0.0000
(aar,stnr,dag,tim)	737.4333	6.2166	0.1041	0.0009
(aar,stnr,tim,mnd)	743.2333	1.9333	0.2083	0.0001
(aar,stnr,tim,dag)	752.8000	5.8333	0.1708	0.0005
(aar,mnd,stnr,dag)	795.1166	0.4833	0.0583	0.0001
(aar,mnd,stnr,tim)	730.4500	1.7500	0.1625	0.0001
(aar,mnd,dag,stnr)	686.8166	0.4166	0.0791	0.0002
(aar,mnd,dag,tim)	578.3000	5.0500	0.0833	0.0000
(aar,mnd,tim,stnr)	737.5000	2.5166	0.2166	0.0000
(aar,mnd,tim,dag)	617.4333	2.9500	0.1708	0.0002
(aar,dag,stnr,mnd)	736.3833	0.4500	0.0542	0.0000
(aar,dag,stnr,tim)	742.6666	5.8666	0.1417	0.0001

(aar,dag,mnd,stnr)	1055.6500	0.5333	0.0542	0.0000
(aar,dag,mnd,tim)	619.1333	2.8500	0.0833	0.0000
(aar,dag,tim,stnr)	744.9500	5.6833	0.1166	0.0001
(aar,dag,tim,mnd)	623.5500	3.3666	0.1041	0.0001
(aar,tim,stnr,mnd)	739.4666	2.3833	0.6791	0.0002
(aar,tim,stnr,dag)	752.3000	6.0833	0.6208	0.0001
(aar,tim,mnd,stnr)	743.0833	2.4500	0.7041	0.0002
(aar,tim,mnd,dag)	629.1000	3.4333	0.6083	0.0008
(aar,tim,dag,stnr)	1294.6833	0.7666	0.6041	0.0000
(aar,tim,dag,mnd)	634.1000	3.3833	0.6125	0.0001
(mnd,stnr,aar,dag)	1166.3000	1.4333	0.0624	0.0001
(mnd,stnr,aar,tim)	1158.1500	3.0500	0.1458	0.0001
(mnd,stnr,dag,aar)	1093.8333	1.2000	0.0583	0.0001
(mnd,stnr,dag,tim)	873.1833	15.3166	0.2625	0.0230
(mnd,stnr,tim,aar)	1187.8666	14.9166	0.4125	0.0006
(mnd,stnr,tim,dag)	973.1500	29.6166	0.3708	0.0001
(mnd,aar,stnr,dag)	1149.2166	0.8500	0.0750	0.0001
(mnd,aar,stnr,tim)	1156.8167	1.8500	0.1500	0.0000
(mnd,aar,dag,stnr)	1151.3666	0.8000	0.0583	0.0001
(mnd,aar,dag,tim)	944.0333	2.0000	0.1125	0.0003
(mnd,aar,tim,stnr)	1115.0833	6.0166	0.2124	0.0000
(mnd,aar,tim,dag)	965.6666	4.0166	0.1166	0.0000
(mnd,dag,stnr,aar)	1130.2333	1.3000	0.0583	0.0001
(mnd,dag,stnr,tim)	875.6833	18.4666	0.1833	0.0000
(mnd,dag,aar,stnr)	1131.0166	1.3833	0.0750	0.0001
(mnd,dag,aar,tim)	1248.7333	3.2666	0.0791	0.0000
(mnd,dag,tim,stnr)	843.3000	18.3333	0.3666	0.0379
(mnd,dag,tim,aar)	883.6333	6.0500	0.1875	0.0001
(mnd,tim,stnr,aar)	1180.0333	114.7333	2.5375	0.0001
(mnd,tim,stnr,dag)	917.4833	98.6000	1.9583	0.0001
(mnd,tim,aar,stnr)	1176.1500	119.8500	2.7583	0.0008
(mnd,tim,aar,dag)	976.9333	96.1000	1.9208	0.0001
(mnd,tim,dag,stnr)	970.4000	107.3666	1.9583	0.0003
(mnd,tim,dag,aar)	874.6833	88.4833	1.9333	0.0002
(dag,stnr,aar,mnd)	1190.7166	0.8833	0.0624	0.0001
(dag,stnr,aar,tim)	1182.9166	6.6666	0.1000	0.0000
(dag,stnr,mnd,aar)	1170.9166	1.2500	0.0624	0.0001
(dag,stnr,mnd,tim)	939.6000	12.7666	0.2750	0.0308
(dag,stnr,tim,aar)	1177.5833	10.3166	0.3083	0.0467
(dag,stnr,tim,mnd)	987.7166	20.5833	0.2666	0.0000
(dag,aar,stnr,mnd)	1162.3833	1.2333	0.0583	0.0001
(dag,aar,stnr,tim)	1171.9666	5.7000	0.0917	0.0001
(dag,aar,mnd,stnr)	1258.0833	1.4000	0.0542	0.0001
(dag,aar,mnd,tim)	982.4166	4.2166	0.0917	0.0001
(dag,aar,tim,stnr)	1179.0333	6.9333	0.1208	0.0001
(dag,aar,tim,mnd)	976.9000	5.3500	0.1125	0.0001
(dag,mnd,stnr,aar)	1169.1833	1.2833	0.0583	0.0001
(dag,mnd,stnr,tim)	975.8500	17.5333	0.1833	0.0000
(dag,mnd,aar,stnr)	1164.1333	1.2166	0.0542	0.0001
(dag,mnd,aar,tim)	966.1833	4.2000	0.0875	0.0001
(dag,mnd,tim,stnr)	1368.0833	26.7500	0.2916	0.0001
(dag,mnd,tim,aar)	977.8500	7.8500	0.2833	0.0461
(dag,tim,stnr,aar)	1182.8666	52.9666	1.0166	0.0005
(dag,tim,stnr,mnd)	986.8666	63.2166	1.0416	0.0001
(dag,tim,aar,stnr)	1180.5333	52.7000	1.0208	0.0001
(dag,tim,aar,mnd)	984.1666	43.0833	1.5041	0.9946
(dag,tim,mnd,stnr)	980.6333	56.2666	1.1416	0.0384
(dag,tim,mnd,aar)	989.6833	39.6000	2.3750	7.0251
(tim,stnr,aar,mnd)	1036.5667	493.5833	96.9083	4385.9811
(tim,stnr,aar,dag)	1079.1333	497.3333	103.7792	4352.2796
(tim,stnr,mnd,aar)	1091.1500	492.4166	100.5916	4586.0902
(tim,stnr,mnd,dag)	892.5833	488.0500	103.9833	3972.9941
(tim,stnr,dag,aar)	1147.0333	499.1833	99.1083	4548.6061
(tim,stnr,dag,mnd)	859.0666	482.2000	98.5333	4250.4661

(tim,aar,stnr,mnd)	998.1166	508.0333	102.3083	4810.1055
(tim,aar,stnr,dag)	1122.1333	492.0333	110.9874	4933.6289
(tim,aar,mnd,stnr)	1019.9500	508.4833	108.1083	4771.0690
(tim,aar,mnd,dag)	1004.7166	461.4666	71.3333	19.3754
(tim,aar,dag,stnr)	1089.9333	493.9833	106.2250	4932.7340
(tim,aar,dag,mnd)	881.0166	500.4833	105.8833	4718.3086
(tim,mnd,stnr,aar)	1075.2833	498.4333	104.9333	4945.5469
(tim,mnd,stnr,dag)	844.6500	506.5333	106.1375	4903.2546
(tim,mnd,aar,stnr)	1158.4666	488.8500	109.8958	4676.1960
(tim,mnd,aar,dag)	914.6833	504.3500	135.1208	4035.8372
(tim,mnd,dag,stnr)	888.8000	496.8000	120.7583	5278.8646
(tim,mnd,dag,aar)	1334.9166	173.4833	90.9916	0.2280
(tim,dag,stnr,aar)	1190.0833	490.3500	126.8458	4870.5124
(tim,dag,stnr,mnd)	982.1666	492.8166	127.8833	5114.1595
(tim,dag,aar,stnr)	1186.6333	491.1000	136.0958	4216.5700
(tim,dag,aar,mnd)	1010.8333	488.2833	135.6916	4029.0153
(tim,dag,mnd,stnr)	923.0333	491.1000	130.9208	3512.3099
(tim,dag,mnd,aar)	981.8000	487.8166	132.8916	3530.3079
(stnr,aar,mnd,dag,tim)	786.2666	1.4000	0.0458	0.0000
(stnr,aar,mnd,tim,dag)	828.0333	1.8500	0.0417	0.0001
(stnr,aar,dag,mnd,tim)	826.7000	1.1333	0.0583	0.0004
(stnr,aar,dag,tim,mnd)	841.2500	1.3666	0.0458	0.0001
(stnr,aar,tim,mnd,dag)	879.6666	4.2833	0.1000	0.0000
(stnr,aar,tim,dag,mnd)	832.2500	4.5000	0.1083	0.0001
(stnr,mnd,aar,dag,tim)	1238.8666	1.3333	0.0375	0.0001
(stnr,mnd,aar,tim,dag)	1193.8333	1.1166	0.0500	0.0000
(stnr,mnd,dag,aar,tim)	1257.7833	1.3000	0.0500	0.0000
(stnr,mnd,dag,tim,aar)	1190.3000	1.8000	0.0583	0.0001
(stnr,mnd,tim,aar,dag)	1211.1666	14.5666	0.2542	0.0001
(stnr,mnd,tim,dag,aar)	1513.1166	13.4500	0.2500	0.0000
(stnr,dag,aar,mnd,tim)	1217.9833	1.2333	0.0417	0.0001
(stnr,dag,aar,tim,mnd)	1280.1000	1.1333	0.0500	0.0000
(stnr,dag,mnd,aar,tim)	1227.4833	1.2166	0.0458	0.0002
(stnr,dag,mnd,tim,aar)	1293.8167	1.2166	0.0583	0.0001
(stnr,dag,tim,aar,mnd)	1279.9000	6.2333	0.1417	0.0001
(stnr,dag,tim,mnd,aar)	1275.6833	6.3666	0.1458	0.0000
(stnr,tim,aar,mnd,dag)	1710.6666	2.9666	2.3000	0.0001
(stnr,tim,aar,dag,mnd)	1195.8333	157.7000	2.3541	0.0001
(stnr,tim,mnd,aar,dag)	1164.4000	156.3833	2.4500	0.0360
(stnr,tim,mnd,dag,aar)	1205.4666	158.1666	2.3333	0.0001
(stnr,tim,dag,aar,mnd)	1267.5833	157.5000	2.3458	0.0002
(stnr,tim,dag,mnd,aar)	1289.8833	142.8666	2.3375	0.0003
(aar,stnr,mnd,dag,tim)	777.7166	0.3333	0.0417	0.0001
(aar,stnr,mnd,tim,dag)	824.3166	0.4166	0.0458	0.0001
(aar,stnr,dag,mnd,tim)	814.1833	0.3333	0.0458	0.0001
(aar,stnr,dag,tim,mnd)	825.5000	0.4333	0.0417	0.0001
(aar,stnr,tim,mnd,dag)	830.7166	0.4500	0.1041	0.0001
(aar,stnr,tim,dag,mnd)	838.9833	0.4333	0.1041	0.0001
(aar,mnd,stnr,dag,tim)	776.3166	0.4000	0.0458	0.0000
(aar,mnd,stnr,tim,dag)	824.7000	0.3000	0.0458	0.0000
(aar,mnd,dag,stnr,tim)	778.2500	0.3833	0.0500	0.0000
(aar,mnd,dag,tim,stnr)	1035.4833	0.4833	0.0500	0.0000
(aar,mnd,tim,stnr,dag)	820.5000	0.6000	0.0917	0.0001
(aar,mnd,tim,dag,stnr)	817.3833	0.3833	0.0958	0.0001
(aar,dag,stnr,mnd,tim)	818.2833	0.3666	0.0417	0.0001
(aar,dag,stnr,tim,mnd)	818.9500	0.4000	0.0417	0.0001
(aar,dag,mnd,stnr,tim)	818.2500	0.3166	0.0417	0.0001
(aar,dag,mnd,tim,stnr)	817.6166	0.4000	0.0500	0.0000
(aar,dag,tim,stnr,mnd)	1341.1500	0.5000	0.0708	0.0001
(aar,dag,tim,mnd,stnr)	818.7500	0.3333	0.0666	0.0000
(aar,tim,stnr,mnd,dag)	826.2833	1.0666	0.5958	0.0001
(aar,tim,stnr,dag,mnd)	831.8333	1.0333	0.5958	0.0002
(aar,tim,mnd,stnr,dag)	997.7166	1.6333	0.8417	0.1163
(aar,tim,mnd,dag,stnr)	1110.5833	1.0500	0.5917	0.0001

(aar,tim,dag,stnr,mnd)	1075.9333	1.2833	0.7125	0.0006
(aar,tim,dag,mnd,stnr)	1483.0333	1.7833	0.8792	0.0017
(mnd,stnr,aar,dag,tim)	1341.4666	1.5500	0.0458	0.0001
(mnd,stnr,aar,tim,dag)	1357.3500	1.7000	0.0458	0.0000
(mnd,stnr,dag,aar,tim)	1205.4166	1.3500	0.0417	0.0001
(mnd,stnr,dag,tim,aar)	1200.3000	1.7500	0.0542	0.0001
(mnd,stnr,tim,aar,dag)	1260.3333	13.9000	0.2542	0.0001
(mnd,stnr,tim,dag,aar)	1210.7500	13.2000	0.2542	0.0001
(mnd,aar,stnr,dag,tim)	1246.7166	1.4666	0.0500	0.0000
(mnd,aar,stnr,tim,dag)	1411.8666	1.7833	0.0500	0.0000
(mnd,aar,dag,stnr,tim)	1251.6666	1.3500	0.0458	0.0001
(mnd,aar,dag,tim,stnr)	1257.3000	1.0000	0.0500	0.0001
(mnd,aar,tim,stnr,dag)	1176.0333	4.3166	0.1000	0.0000
(mnd,aar,tim,dag,stnr)	1196.4000	4.0000	0.1000	0.0000
(mnd,dag,stnr,aar,tim)	1248.3666	1.3000	0.0708	0.0010
(mnd,dag,stnr,tim,aar)	1182.8000	1.4500	0.0500	0.0000
(mnd,dag,aar,stnr,tim)	1729.8000	1.4000	0.0333	0.0000
(mnd,dag,aar,tim,stnr)	1236.4666	1.6666	0.0458	0.0001
(mnd,dag,tim,stnr,aar)	1182.5833	5.4500	0.1458	0.0001
(mnd,dag,tim,aar,stnr)	1186.1500	4.8833	0.1458	0.0001
(mnd,tim,stnr,aar,dag)	1274.7666	132.9333	1.9375	0.0000
(mnd,tim,stnr,dag,aar)	1242.7166	127.9666	1.9541	0.0001
(mnd,tim,aar,stnr,dag)	1260.3666	130.4833	2.0542	0.0306
(mnd,tim,aar,dag,stnr)	1269.8833	130.8000	2.0292	0.0058
(mnd,tim,dag,stnr,aar)	1229.7833	122.2166	2.0708	0.0011
(mnd,tim,dag,aar,stnr)	1201.0333	129.0000	2.0333	0.0055
(dag,stnr,aar,mnd,tim)	1289.0833	1.4000	0.0417	0.0001
(dag,stnr,aar,tim,mnd)	1302.5833	1.2000	0.0500	0.0000
(dag,stnr,mnd,aar,tim)	1320.7333	1.2500	0.0500	0.0000
(dag,stnr,mnd,tim,aar)	1314.5166	1.2500	0.0624	0.0001
(dag,stnr,tim,aar,mnd)	1306.8666	6.0666	0.1500	0.0000
(dag,stnr,tim,mnd,aar)	1509.3666	6.1000	0.1458	0.0001
(dag,aar,stnr,mnd,tim)	1289.5833	1.1000	0.0417	0.0001
(dag,aar,stnr,tim,mnd)	1298.3500	0.8000	0.0417	0.0001
(dag,aar,mnd,stnr,tim)	1289.2666	1.3833	0.0375	0.0001
(dag,aar,mnd,tim,stnr)	1283.7833	1.8833	0.0500	0.0000
(dag,aar,tim,stnr,mnd)	1290.4000	2.3666	0.0708	0.0001
(dag,aar,tim,mnd,stnr)	1304.7666	2.7166	0.0666	0.0000
(dag,mnd,stnr,aar,tim)	1800.3333	1.4166	0.0417	0.0001
(dag,mnd,stnr,tim,aar)	1316.8167	1.2666	0.1583	0.0384
(dag,mnd,aar,stnr,tim)	1304.3500	1.4000	0.0458	0.0000
(dag,mnd,aar,tim,stnr)	1285.7166	1.8500	0.0417	0.0001
(dag,mnd,tim,stnr,aar)	1309.0667	5.4666	0.1458	0.0001
(dag,mnd,tim,aar,stnr)	1310.5500	5.7166	0.1458	0.0000
(dag,tim,stnr,aar,mnd)	1307.3167	51.3333	1.4125	0.4330
(dag,tim,stnr,mnd,aar)	1310.7833	53.9500	0.9708	0.0001
(dag,tim,aar,stnr,mnd)	1289.5333	52.9000	0.9625	0.0002
(dag,tim,aar,mnd,stnr)	1287.2166	51.3833	0.9583	0.0003
(dag,tim,mnd,stnr,aar)	1292.1166	51.7333	0.9583	0.0001
(dag,tim,mnd,aar,stnr)	1284.8666	43.9166	0.9666	0.0002
(tim,stnr,aar,mnd,dag)	1147.9833	490.4500	107.4416	4356.6019
(tim,stnr,aar,dag,mnd)	1202.2333	489.0666	108.2750	4852.8001
(tim,stnr,mnd,aar,dag)	1186.4000	484.6333	108.2124	4856.2751
(tim,stnr,mnd,dag,aar)	1443.7000	456.0500	73.8833	6.7852
(tim,stnr,dag,aar,mnd)	1272.7833	487.3000	108.6875	4571.0173
(tim,stnr,dag,mnd,aar)	1284.5833	487.3666	108.1458	4714.1865
(tim,aar,stnr,mnd,dag)	1135.4500	486.2333	107.8000	4631.8503
(tim,aar,stnr,dag,mnd)	1209.9000	483.3166	107.9000	4536.5316
(tim,aar,mnd,stnr,dag)	1133.1000	488.4000	107.4166	4561.1647
(tim,aar,mnd,dag,stnr)	1134.2833	487.4666	108.1166	4591.4310
(tim,aar,dag,stnr,mnd)	1648.3167	185.9833	71.5208	0.0775
(tim,aar,dag,mnd,stnr)	1207.7000	487.7333	107.2458	4529.6761
(tim,mnd,stnr,aar,dag)	1261.2000	483.4000	107.7374	4505.1009
(tim,mnd,stnr,dag,aar)	1252.6500	483.2666	108.3666	4760.9704

(tim,mnd,aar,stnr,dag)	1263.6166	486.7166	107.2000	4533.8154
(tim,mnd,aar,dag,stnr)	1261.5833	487.7166	107.2916	4559.4150
(tim,mnd,dag,stnr,aar)	1247.5000	485.3666	108.4916	4777.5238
(tim,mnd,dag,aar,stnr)	1246.2666	484.5833	106.6833	4416.0781
(tim,dag,stnr,aar,mnd)	1269.6833	489.2833	104.4541	3810.2979
(tim,dag,stnr,mnd,aar)	1223.1500	485.4000	106.5667	4426.1960
(tim,dag,aar,stnr,mnd)	1271.8167	490.6333	107.0708	4542.7959
(tim,dag,aar,mnd,stnr)	1271.7333	489.1166	108.0708	4939.0365
(tim,dag,mnd,stnr,aar)	1277.5667	491.5166	107.5874	4447.4333
(tim,dag,mnd,aar,stnr)	1271.2166	486.0333	107.5292	4547.9723

Tabell 5.4.2

dex	Create	First	Expect.	Variance
(aar,mnd,stnr,tim,dag)	824.7000	0.3000	0.0458	0.0000
(aar,dag,mnd,stnr,tim)	818.2500	0.3166	0.0417	0.0001
(aar,stnr,mnd,dag,tim)	777.7166	0.3333	0.0417	0.0001
(aar,stnr,dag,mnd,tim)	814.1833	0.3333	0.0458	0.0001
(aar,dag,tim,mnd,stnr)	818.7500	0.3333	0.0666	0.0000
(aar,dag,stnr,mnd,tim)	818.2833	0.3666	0.0417	0.0001
(aar,mnd,dag,stnr,tim)	778.2500	0.3833	0.0500	0.0000
(aar,mnd,tim,dag,stnr)	817.3833	0.3833	0.0958	0.0001
(aar,dag,stnr,tim,mnd)	818.9500	0.4000	0.0417	0.0001
(aar,mnd,stnr,dag,tim)	776.3166	0.4000	0.0458	0.0000
(aar,dag,mnd,tim,stnr)	817.6166	0.4000	0.0500	0.0000
(aar,stnr,mnd,tim,dag)	824.3166	0.4166	0.0458	0.0001
(aar,stnr,dag,mnd)	737.8000	0.4166	0.0542	0.0000
(aar,mnd,dag,stnr)	686.8166	0.4166	0.0791	0.0002
(aar,stnr,dag,tim,mnd)	825.5000	0.4333	0.0417	0.0001
(aar,stnr,tim,dag,mnd)	838.9833	0.4333	0.1041	0.0001
(aar,dag,stnr,mnd)	736.3833	0.4500	0.0542	0.0000
(aar,stnr,tim,mnd,dag)	830.7166	0.4500	0.1041	0.0001
(aar,mnd,dag,tim,stnr)	1035.4833	0.4833	0.0500	0.0000
(aar,mnd,stnr,dag)	795.1166	0.4833	0.0583	0.0001
(aar,stnr,mnd,dag)	689.2500	0.5000	0.0583	0.0001
(aar,dag,tim,stnr,mnd)	1341.1500	0.5000	0.0708	0.0001
(aar,dag,mnd,stnr)	1055.6500	0.5333	0.0542	0.0000
(aar,mnd,tim,stnr,dag)	820.5000	0.6000	0.0917	0.0001
(aar,dag)	259.2333	0.6500	0.3417	0.0001
(aar,tim,dag,stnr)	1294.6833	0.7666	0.6041	0.0000
(dag,aar,stnr,tim,mnd)	1298.3500	0.8000	0.0417	0.0001
(mnd,aar,dag,stnr)	1151.3666	0.8000	0.0583	0.0001
(mnd,aar,stnr,dag)	1149.2166	0.8500	0.0750	0.0001
(aar,mnd)	212.4500	0.8500	0.7583	0.0001
(dag,stnr,aar,mnd)	1190.7166	0.8833	0.0624	0.0001
(mnd,aar)	308.6000	0.9000	0.7791	0.0002
(mnd,aar,dag,tim,stnr)	1257.3000	1.0000	0.0500	0.0001
(dag,aar)	346.1666	1.0166	0.3625	0.0011
(aar,tim,stnr,dag,mnd)	831.8333	1.0333	0.5958	0.0002
(aar,tim,mnd,dag,stnr)	1110.5833	1.0500	0.5917	0.0001
(aar,tim,stnr,mnd,dag)	826.2833	1.0666	0.5958	0.0001
(dag,aar,stnr,mnd,tim)	1289.5833	1.1000	0.0417	0.0001
(stnr,mnd,aar,tim,dag)	1193.8333	1.1166	0.0500	0.0000
(stnr,dag,aar,tim,mnd)	1280.1000	1.1333	0.0500	0.0000
(stnr,aar,dag,mnd,tim)	826.7000	1.1333	0.0583	0.0004
(dag,stnr,aar,tim,mnd)	1302.5833	1.2000	0.0500	0.0000
(mnd,stnr,dag,aar)	1093.8333	1.2000	0.0583	0.0001
(stnr,dag,mnd,aar,tim)	1227.4833	1.2166	0.0458	0.0002
(dag,mnd,aar,stnr)	1164.1333	1.2166	0.0542	0.0001
(stnr,dag,mnd,tim,aar)	1293.8167	1.2166	0.0583	0.0001
(stnr,dag,aar,mnd,tim)	1217.9833	1.2333	0.0417	0.0001
(dag,aar,stnr,mnd)	1162.3833	1.2333	0.0583	0.0001
(stnr,aar,mnd,dag)	649.8500	1.2500	0.0500	0.0000
(dag,stnr,mnd,aar,tim)	1320.7333	1.2500	0.0500	0.0000

3.5 Konstruksjon av innlastingsrutine

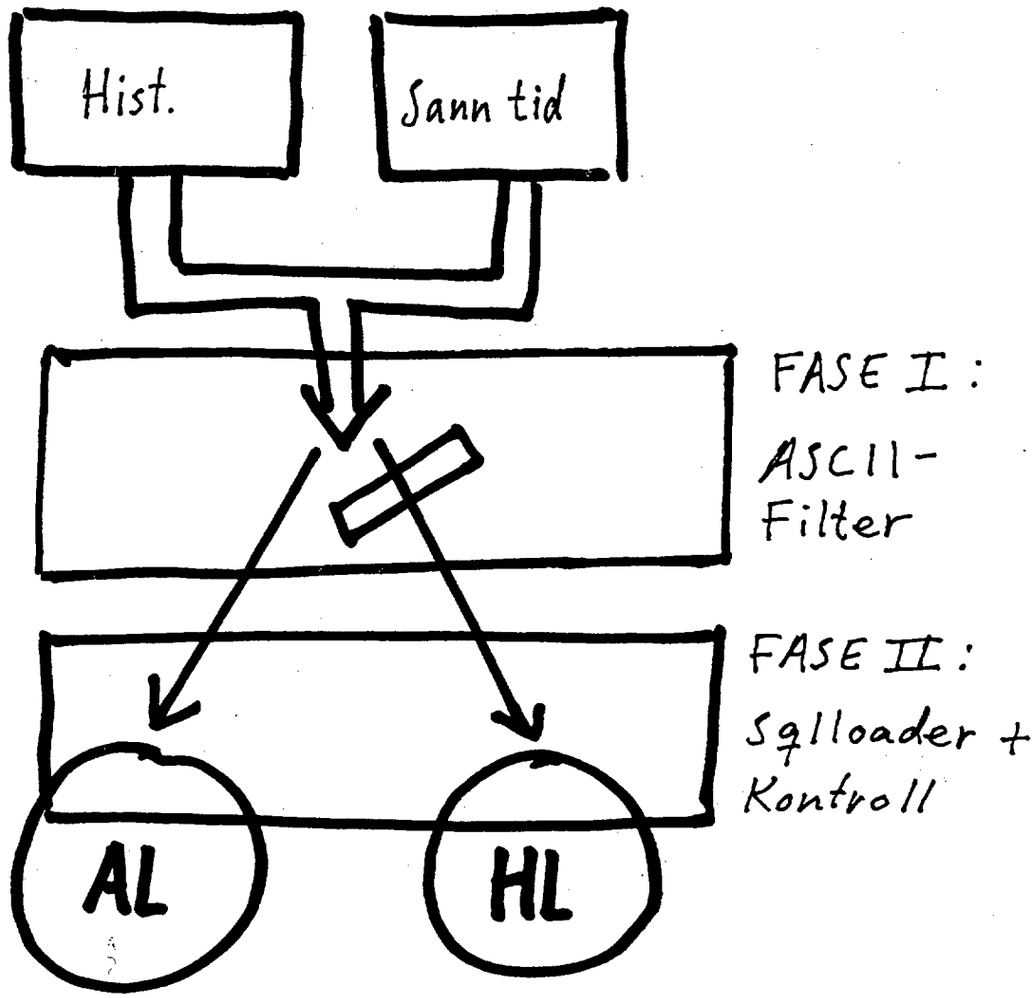
1. Innledning

Innlastingen av historisk data er tenkt som en simulering av det fremtidige innlastingssystem, og deler av den historiske data vil lastes inn via modell 1 beskrevet i kapittel 2.

Innlastingsrutinene presentert i dette delkapittelet baserer seg på dataflyt i figur 2.1.2.

2. Dataflytskisse

Under gangen fra eksternt lagringsmedium til AL/HL, må data flyte gjennom diverse poster. Figur 3.5.1 gir en skisse av hovedpostene for en slik flyt, dvs. programenhetene som styrer flyten fra ASCII til ORACLE.



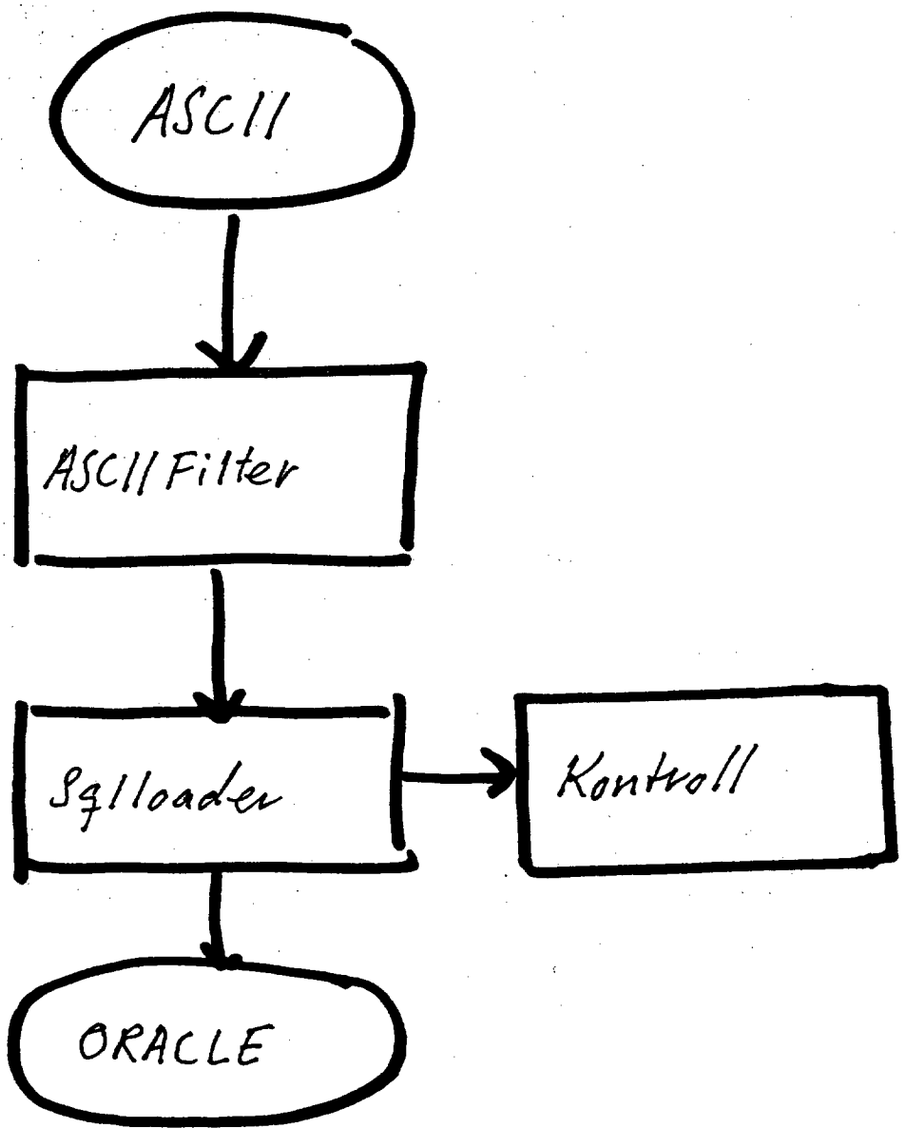
Figur 3.5.1 Innlastingsflyt - oversiktsskisse

Figuren (3.5.1) viser en flytstruktur der det skilles mellom FASE I: ASCIIfiltrering (tilrettelegging av data for innlasting) og FASE II: den faktiske innlastingen.

Man kan videre legge merke til at filteret mellom dataflytens t-kryss og HL ligger innerammet i fase I. Dette betyr med andre ord at kontrollfiltreringen mot HL foregår før innlasting, og må implementeres i et programmeringsspråk som kan behandle ASCIIformater.

Fase II beskriver grensesnittet inn mot ORACLE-databasen, som består av ORACLE-verktøyet SQL*Loader og en automatisk kontroll av at innlastingen har fungert forskriftsmessig.

Neste figur (figur 3.5.2) viser en abstrasksjon av prosessen.



Figur 3.5.2 Hovedstruktur for innlasting

3.1 Konvertering fra eksternt medium til ASCII-fil

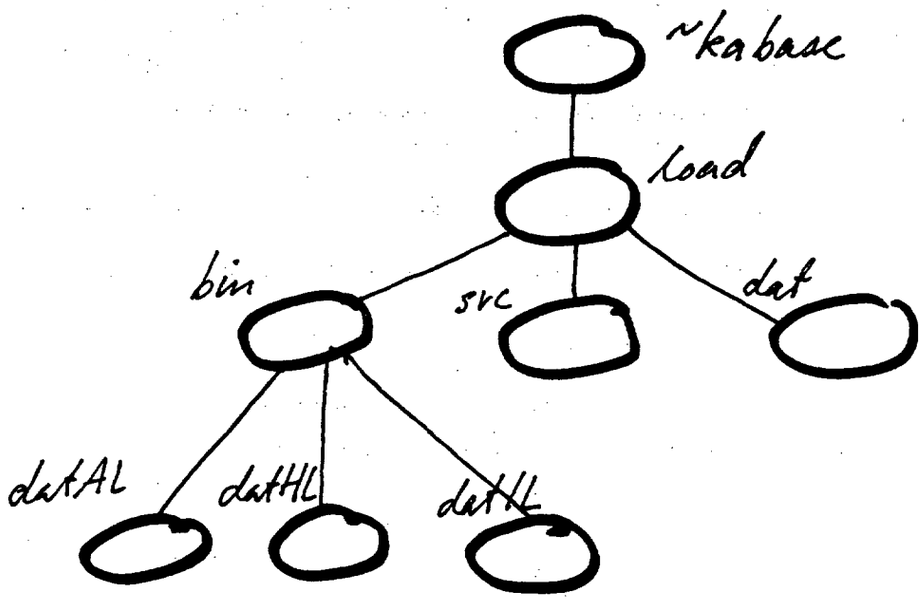
Selve konverteringen fra magnetbånd eller diskett til ASCII-fil i SG på et egnet område er omtalt i kapittel 1 og kapittel 5.

For at ASCIIfilene kan filtreres og innlasting må de underordnes følgende filnavnskonvensjoner:

1. Innlastingsfiler må ligge på området ~kabase/load/dat

2. Navn på innlastingsfil må være stasjonsnavn på formen CDDDDD.dta (en bokstav og et femsifret tall), f.eks. K18700.dta.

Arbeidsmiljøet (katalogstrukturen) for innlasting kan beskrives ved følgende figur (3.5.3):



Figur 3.5.3 Arbeidsmiljø for innlasting

Datalagringsområdet (dat) vil automatisk bli renset etter en vellykket kjøring, så man må forsikre seg om å dobbeltlagre ASCIIfilene dersom man ønsker å beholde disse.

Dersom data består av flere filer på felles format (f.eks. en fil for hver diskett) bør disse konkateneres før innlastingen starter. Konkatenerete filer går raskere gjennom systemet da man kan utfører hovedtestene en gang, dvs. for den konkatenerte filen, og ikke for hver av deldata-filene.

3.2 ASCIIfilter

Input og output for denne modulen (figur 3.5.2) er som følger:

Input: dat/CDDDDD.dta (f.eks. dat/K87110.dta)

Output: bin/datAL/AL.dat
bin/datAL/AL.ctl
bin/datHL/HL.dat
bin/datHL/HL.ctl

Scriptet som implementerer ASCIIfilteret er lagret som bin/ASCIIfilter.

Før data er klar for innlasting via SQLloader må sql-kontrollfiler genereres, datafilene kontrolleres og formateres.

ASCIIfilteret er et script bestående av C-programmer som har til hensikt å utføre de nødvendige og tilstrekkelige kontroller og prepareringer av data før data kan behandles av SQL*Loader. Prepareringen vil blant annet bestå av diverse kall inn mot databasens informasjons-lager.

ASCIIfilteret har også ansvar for forkastning av data som ikke skal lastes direkte inn i hovedlager. Selve forkastningen, eller filtreringen, gir opphav til følgende filer:

bin/filter.log (statusrapport)
bin/filter.bad (forkastet data pga formateringsfeil)
bin/filter.dsc (forkastet data pga HL-intervaller)

Filteret genrerer to filer på basis av de opprinnelige ASCII-data; en fil for AL hvor det kun er foretatt formateringsbehandling (AL.dat), og en fil for HL hvor useriøs data er luket ut (HL.dat). Filene AL.ctl og HL.ctl er informasjon til sqlloader angående formatering og innlastingsspesifikasjoner for AL.dat og HL.dat.

3.3 SQLLOADER

Sqlloader er avhengig av å bli startet opp med korrekte kontrollfiler, svarende til riktig innlastingsformat (kapittel 2.2).

Innlasting både til AL og HL skjer ved kommandoen REPLACE.

Etter en innlastingssesjon returnerer sqlloader alltid en log-, bad-, og dsc-fil, dvs.:

bin/datAL/AL.log
bin/datAL/AL.bad
bin/datAL/AL.dsc

bin/datHL/HL.log
bin/datHL/HL.bad
bin/datHL/HL.dsc

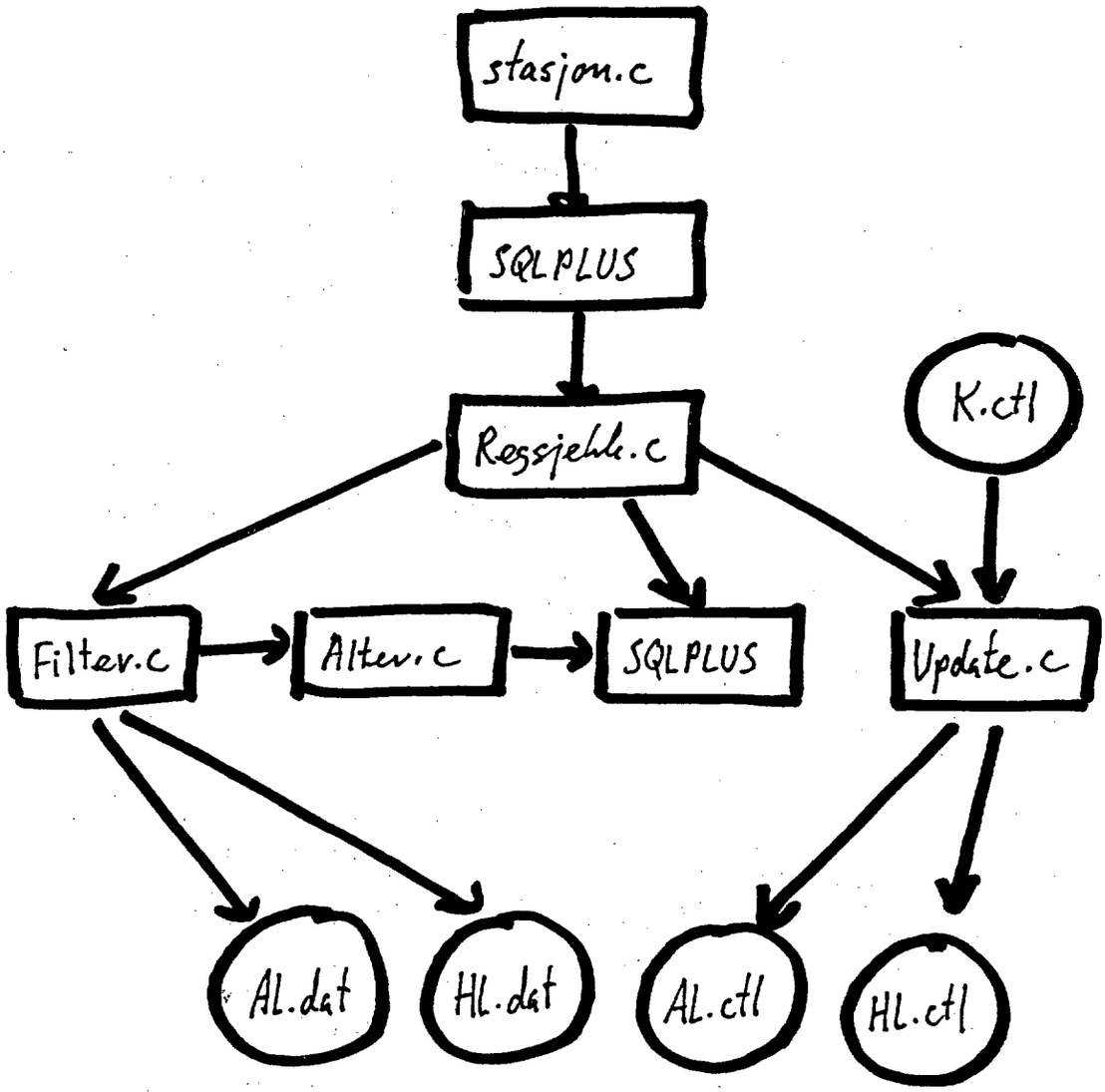
Logfilen dokumenterer overføringen inklusiv tidsforbruk og størrelsen på bad- og dsc-filene. Bad- og dsc-filene inneholder forkastet data. Dersom ASCIIfilteret har gjort en skikkelig jobb skal disse være tomme. En kontroll-rutine skissert i figur 3.5.2 vil også sjekke de sistnevnte filer.

4. Spesifikasjon av ASCIIfilter

Det kan gjentas at ASCIIfilteret kun formaterer data for innlasting. All kontroll av data for dette filteret er på basis av ASCIIrepresentasjon av meteorologiske parametre, følgelig må man benytte kontrollrutiner utviklet i FORTRAN og C, og ikke PL/SQL som blir benyttet i kontrollrutinene for data i AL.

4.1 Ordinær flyt

ASCIIfilteret kan illustreres gjennom følgende flytdiagram (figur 3.4.2)



Figur 3.5.3 Flytdiagram for ASCIIfilter

Grunnen til at HL-filteret er lagt utenfor ORACLE og hindrer data i å komme inn, istedet for å stå inne i ORACLE og sparke data ut, er hyppighets-registreringer som skal sikre at det genereres stasjonsspesifikke tabeller i HL med varierende parameterrekkefølge svarende til hva som er mest plassbesparende. Hvilket parametersett som hører til hvilken stasjon får programmet vite ved å automatisk spørre mot IL.

I flytdiagrammet kan man også legge merke til en entitet update.c. Hensikten med denne er å ta seg av formatering av innlastingsfilene AL.ctl og HL.ctl slik at disse inneholder riktig parameter-sett, riktig format og riktige tabell-angivelser. Hvordan velge riktig format får programmet oppdager programmet ved å spørre mot IL.

Nedenfor følger en spesifikasjon av program-entitetene fra figur 3.5.3:

STASJON

Input: k18700.dta (datafilen)

Output: stasjon.tmp (Stasjonens filnavn "k18700.dat" og tabellnavn "K18700" lagres på temporær fil)

IL.tmp (Informasjonsarkivopplysninger ervervet gjennom SQLPLUS)

stdout (hvis fornuftig input):
- select navn, type from parameter where lager='AL' og stasjonstype='K';
- select navn, type from parameter where lager='HL' og stasjonstype='K';
- select stnr, navn from stasjon where stasjonsnr='18700';

stderr (hvis feil oppstår): "Filnavn skal angis på formen CDDDDD.dta hvor C er en av bokstavene K, M, N, P, A, E, L, R, F og hvor D er et heltall mellom 0 og 9"

Funksjon: Programmet sjekker at filnavnets stasjonsnummer (K18700) korresponerer med filens stasjonsnummer (1870).

REGSJEKK

Input: IL.tmp (ORACLE-svar på SQL-setningene fra stasjon i "spool"-format.)

Output: startup.tmp

(a) Informasjon om en ny tabell er kreert.

(b) Informasjon om FILTER og UPDATE skal

startes opp.

Funksjon: Sjekke at angitt stasjon er registrert i databasen, og oppdatere hvis nødvendig.

FILTER

Input: stasjon.tmp
startup.tmp
../dat/K87110.dta
../datHL/K.ctl
../datAL/K.ctl
paraHL.tmp

Output: ../datAL/AL.dat
../datHL/HL.dat
filter.log (statusrapport)
filter.dsc (rader som ikke penetrerte HL)
filter.bad (formateringsfeil)
stderr: Varsling om forventet tidsforbruk for innlasting. Estimatet er beregnet på basis av frekvenslagerinformasjon (load-hastighet) og .dta-fillengde.

stdout: Output til SQLPLUS for ALTER TABLE ADD...

Funksjon: Filtrerer og kontrollerer alle data (med unntak av superkontrollerte data, se 4.2)

UPDATE

Input: stasjon.tmp

Output: ../datAL/AL.ctl
../datHL/HL.ctl

4.2 Superkontrollert flyt

Superkontrollert flyt er fundamentalt forskjellig fra ordinær flyt i følgende forstand:

1. Ingen kontroll mot HL

2. Ingen mellomlagring i AL

3. Kontrollflagging i HL

Disse elementene kan styres gjennom FILTER, som skal genrere en HL.dat der kontrollflagg settes (legges inn i ASCIIfilen), ingen kontroller utføres, og AL.dat er tom.

Da superkontrollert flyt ikke har noen betydning for etablering av datastrukturer på TYPHOON har delprosjekt 3 vedtatt å la implementasjonen av denne ligge. En

implementasjon av denne flyten avhenger også av et punche-miljø (SQL*FORMS), så videre behandling av oppgaven overlates til delprosjekt 7.1 "Data inn".

5. Sqlloader

Sqlloader (figur 3.5.1) er konstruert som et script som kaller to sqlload-jobber, enten simultant (bakgrunnsjobb) eller sekvensielt. Parallell innlasting skulle presumptivt ta kortere tid.

Eksperimenten med doble sqlloader-kjøringer resulterer imidlertid i stadige feil ORA-01562 og ORA-01547.

Sekvensiell innlastng (først HL så AL) er følgelig valgt inntil videre.

6. Kontrollscript

Etter innlastingen startes kontroll-scriptet (figur 3.5.1).

Dersom kontrollscriptet returnerer positivt startes kontrollen av data i arbeidslageret (kap 2.3).

4. Innlastingseksperimenter

1. Innledning

I det testmiljø prosjektgruppen har benyttet eksisterer det data i forskjellige format for diverse stasjonstyper lagret både tabellvis og samlet med forskjellige indekseringer.

Dette kapittelet tar for seg et par av disse datasettene for å illustrere hvordan datastrukturen på TYPHOON opprettes automatisk etterhvert som data lastes inn.

2. Innlastingsprosedyre

1. Datafiler med navn CDDDDD.dta (C bokstav, D siffer), korresponderende med stasjons-typer og nummer, plasseres i katalogen ~kabase/load/dat

2. Man utfører så kommandoen ~kabase/load/run og data lastes automatisk inn.

Status- og feil-rapporter returneres så til skjerm og fil, avhengig av kvaliteten på originaldata.

3. Eksempel: K87110 Andøya

Manuelt arbeid er som følger: (pkt. 1 er frivillig, men er å anbefale da programmet renser området ~kabase/load/dat etter vellykket innlasting).

1. Data hentes inn fra magnetbånd og lagres på et midlertidig område på TYPHOON (~ka/andoya.dat)

2. Data legges over på ~kabase/load/dat/k87110.dta, dvs. cd ~ka
   cp andoya.dat ~kabase/load/dat/k87110.dta

3. cd ~kabase/load og skriv run.

3.1 Utskrift til skjerm

Programmets utskrivning til skjerm er som følger:

```
>Uoverenstemmelse mellom filnavn k87110.dta og filinnhold
(stnr = 8711).
8711 vil bli erstattet med 87110.
Innlasting for K87110 ANDOYA? (j/n)
>j

Filtrering starter
500 rader kontrollert
1000 rader kontrollert
1500 rader kontrollert
2000 rader kontrollert
2500 rader kontrollert
```

```
3000 rader kontrollert
3500 rader kontrollert
...
16000 rader kontrollert
16069 rader kontrollert

Innlasting starter
SQL*loader: Version 1.0.27.0.1 - Production on Tue Oct 6
14:58:16 1992
Commit point reached - logical record count 64
Commit point reached - logical record count 128
...
Commit point reached - logical record count 16069

SQL*loader: Version 1.0.27.0.1 - Production on Tue Oct 6
15:00:05 1992
Commit point reached - logical record count 64
Commit point reached - logical record count 128
...
Commit point reached - logical record count 16069
```

3.2 Yteevne

Innlastingen av k87110.dta (16069 rader) tok ca 5.5 minutter. Arbeidsbelastningen er fordelt som følger:

```
Kontroll:      1 min 04 sek (ca. 250 rader pr. sek)
Innlasting AL: 2 min 16 sek (ca. 120 rader pr. sek)
Innlasting HL: 2 min 16 sek (ca. 120 rader pr. sek)
```

Til tross for at ASCIIfilteret har en rekke gjøremål, leser data-input-filen karaktervis, filtrerer, oppretter og fjerne filer, kommuniserer med IL osv., består flaskehalsen i bruk av SQL*Loader.

I dette eksempelet benytter SQL*Loader seg av commit-point for hver 64'de innlastede rad. Ved "tuning" er det mulig at yteevnen kan forbedres noe.

4. Eksempel: K00001 DUMMY

Eksempelet overfor illustrerer innlasting av data for en stasjon som allerede eksisterer. Nedenfor presenteres registreringsmekanismen for nye stasjoner.

4.1 Programutskrift

Samme data som ovenfor er benyttet, men filnavnet er erstattet med k00001.dta. Skjermkommunikasjonen er som følger:

```
>Uoverenstemmelse mellom filnavn k00001.dta og filinnhold
(stnr = 8711).
8711 vil bli erstattet med 87110.
ADVARSEL: Stasjonen er ikke registrert i databasen.
Skal K00001 registreres? (j/n)
>j
```

```
Sikker (j/n)
>j
Stasjonsnavn:
>DUMMY
Filtrering starter
500 rader kontrollert
1000 rader kontrollert
...
```

Før man rekker å registrere en ny stasjon går man altså gjennom tre kontroller.

Først spørsmålet om K00001 skal registreres. Her får man repetert filnavnet. Man er allerede blitt gjort oppmerksom på at filnavn er forskjellig fra filinnhold.

Deretter en verifikasjon for å hindre at det svares 'j' av ren refleks.

Endelig blir man bedt om å angi stasjonsnavn. Her vil det også være en kontroll som hindrer at stasjoner registreres dobbelt (hadde man svart ANDOYA ville programmet abortert), og også en kontroll på at tekststrengen som leses har en viss lengde.

4.2 Plassbesparelse

I det den nye stasjonen ble registrert ble det også sendt en create-forespørsel til SQLPLUS. En midlertidig tabell bestående kun av primærnøkkel (stnr, aar, mnd, dag, tim) etableres, og filtreringen startes. I filteret telles antallet forekomster av forskjellige parametre, og det genereres en parameterrekkefølge som oversendes SQLPLUS i form av en ALTER TABLE ADD-setning.

Dersom man sammenlikner tabellen konstruert i dette eksempelet med en tilsvarende tabell med vilkårlig rekkefølge (K00002) får man følgende resultat:

NAVN	BYTES	BLOCKS
K00001	5771264	1409
K00002	3833856	936

En block = 4096 (2**12) bytes.

Prosentvis plass spart er:
(1409-936)/1409=0.3357 (33.57%)

5. Tilrettelegging for endelig innlasting

KLIMAdatabasen vil bestå av både testtabeller i et utviklingsmiljø og tabeller for sluttbrukere. Testmiljøet vil være adskilt fra resten av databasen.

Dataoverføringer og tester under delprosjekt 3 er utført i testmiljøet. Dette vil fortsatt være testmiljø etterat delprosjekt 3 er avsluttet.

1. Stasjonstyper

Alle stasjonstyper er tatt i betraktning bortsett fra SONDE, E-data og METAR som det ikke er tatt noe standpunkt til med hensyn til format.

2. Krav til miljøet for en endelig dataoverføring

Det er flere krav som må være oppfylt før den endelige dataoverføringen kan igangsettes :

- generelle krav til maskin, disker, nettverk, magnetbåndstasjon, terminaler, printer, backup etc.
- ORACLE-struktur må være opprettet
- dataoverføringsmiljøet må være opprettet
- data som ikke allerede ligger på et brukbart format og medium må enten overføres til magnetbånd eller diskett hvis de ikke kan overføres direkte via FTP
- alle entiteter i IL som er nødvendige for å kunne opprette datatabeller og struktur må være definert med fullstendig attributtsett - alle attributtene bør fylles ut samtidig for å oppnå en ryddig struktur innen blokken.
  Under delprosjekt 3. er det kun tatt hensyn til nødvendige attributt for å kunne opprette tabeller og parametersett.
- et manuelt registreringsverktøy må eksistere for innlasting eller oppdateringer i IL

Da HL opprettes på grunnlag av informasjon som ligger i IL, må data lastes i IL før de meteorologiske data kan lastes.

3. DATE-funksjonen

Delprosjekt 2E har overlatt til delprosjekt 3 å avgjøre om DATE skal implementeres som et felt i HL-tabellene.

Da det skal lages en utplukksrutine som skal ivareta mange hensyn både når det gjelder datakvalitet, fornuftig bruk av

maskinressurser, konvertering av værtegn fra tallkode til en mer forståelig bokstavkode, kan også utplukksrutinen ivareta fordelene ved DATE uten å lagre DATE fysisk i tabell.

Fordelene ved DATE er beregning av tidsinformasjon og standardoppsett av dato i rapporter ol. Disse fasiliteter kan oppnås i utplukksrutinen ved å konvertere AAR, MND, DAG og TIM til DATE-format.

Ulempen med DATE er at den er plasskrevende (7 byte pr observasjon) og er redundant data.

Gruppens konklusjon er som følger :

- Da de fasiliteter som DATE innebærer kan løses i utplukks-rutinen, ser vi ingen grunn til å lagre DATE. Men hvis det skulle vise seg at DATE brukt som indeks gir en vesentlig bedring av performance generelt sett enn AAR, MND og DAG, vil det bli revurdert om DATE skal lagres. Se forøvrig kap.3.4.

4. Feltstørrelse i AL

ORACLE forkaster verdier som er større enn øvre grense for gitt felt. Feltstørrelsen defineres når tabellen blir opprettet.

Det som forekommer spesielt med sanntidsdata er at det kan forekomme en forskyvelse i datastrømmen eller kommafeil som forårsaker at parameteren blir eksempelvis 10-ganger større enn opprinnelig verdi. Disse verdiene ville bli forkastet i ORACLE hvis feltstørrelsen bare var definert til lovlige intervall.

Delprosjekt 2E har overlatt til delprosjekt 3 å avgjøre størrelsen på felt i AL.

Gruppens konklusjon er som følger :

- for å ta vare på flest mulig innkomne verdier som kan være av nyttig informasjon for eventuell interpolering/oppretting i AL, defineres maksimal feltstørrelse i AL.

Verdiene kunne isteden bli lastet i en hjelpetabell, men gruppen anser den løsningen som mer upraktisk.

Referanseliste

[1] Databaseprosjekplan revidert pr. 09.09.92

[2] Delprosjektplan 22.04.92 for delprosjekt 3

[3] Rapport 08.10.91 Delprosjekt 1B "Forslag til datamodell for informasjonsarkivet"

[4] Rapport 10.09.92 Delprosjekt 2E "Andre data/datastrukturer"

[5] Rapport 02.09.92 Delprosjekt 7.7 "Opprette arbeidsmiljø på maskinene"