(12) **PATENT**

(19) NO (11) **328664** (13) **B1**

NORGE (51) Int Cl.
G06F 17/30 (2006.01)

Patentstyret

(21)	Søknadsnr	20085150	(86)	Int.inng.dag og søknadsnr	
(22)	Inng.dag	2008.12.10	(85)	Videreføringsdag	
(24)	Løpedag	2008.12.10	(30)	Prioritet	
(41)	Alm.tilgj	2010.04.19			
(45)	Meddelt	2010.04.19			

(73) Innehaver Fast Search & Transfer AS, Postboks 1677 Vika, 0120 OSLO, Norge
(72) Oppfinner Petter Moe, Nordengkollen 71, 1396 BILLINGSTAD, Norge
(74) Fullmektig Bryn Aarflot AS, Postboks 449 Sentrum, 0104 OSLO, Norge

(54) Benevnelse **Fremgangsmåte for å forbedre ytelse i et system for søking og gjenfinning**

(56) Anførte publikasjoner US 2007/0239666 A1

(57) Sammendrag

I en fremgangsmåte for å forbedre søkeytelsen i et system for søking og gjenfinning av informasjon implementert på en søkemotor som benytter et søkespørsmål på en database omfattende dataposter i form av tabeller, spesifiseres en denormalisert søkeindeks. Deler av den denormaliserte indeks representeres fysisk som en søkeindeks og andre deler som separate tabeller. Et søk kan deretter finne sted forut for oppbyggingen av søkeindeksen og således forbedre en søkespørsmålsytelse ved å redusere latens så vel som oppdaterings- og lagringskostnadene.

INNLEDNING

Den foreliggende oppfinnelse angår en fremgangsmåte til å forbedre søkeytelse i et system for søking og gjenfinning av informasjon implementert på en søkemotor, hvor et søkespørsmål benyttes på en database som inneholder dataposter i form av tabeller, og hvor fremgangsmåten omfatter å spesifisere og danne en denormalisert søkeindeks.

I dag er det standard praksis å lagre operasjonelle data i et foretak i en database. Eksempler på dette innbefatter produktdatabaser, arbeidstakerdatabaser, utstyrsdatabaser, ordredatabaser osv., og enda mer typisk kan datapostene eller dokumentene selv sammenføyes i tabeller eller struktureres som tabeller i seg selv.

La en normalisert form av en databasetabell være en hvor mengder av attributtnavn, eller kort benevnt attributter, benyttes til å dele en tabell opp i flere tabeller hvor de resulterende duplikater av dataposter er fjernet. Typisk kan en mengde av normaliserte tabeller sammenføyes til forskjellige syn for å presentere informasjon på en nyttig måte. Som et eksempel kan det tas en tabell som inneholder en persons arbeidstakernummer og navn og en annen som avbilder fra arbeidstakernummer til et mobiltelefonnummer. Ved å sammenføye disse kan det dannes et syn i form av en telefonliste, hvor synet lagres aksesserbart som en virtuell tabell bestående av resultatmengden for et søkespørsmål. Gjenfinning av slike dataposter kan være utført direkte fra en database. Dette gir informasjon som er oppdatert, men ulempen er at det kan være tidkrevende og i noen tilfelle vil låse en eller flere dataposter i datatabellene, eller til og med tabellene selv. Dataposter kan også låses med tanke på å skriving eller oppdatering og kan da ikke ses av en annen part.

KJENT TEKNIKK

For å overkomme disse begrensninger er det i dag forholdsvis vanlig å mate data som fremkommer fra ekstrahering av informasjonen fra syn som beskrevet ovenfor, for å frembringe fullstendige dataposter (som sett fra søkeapplikasjonssiden) og indeksere disse i en søkemotor. En slik indeks kalles en denormalisert søkeindeks, siden den gjenspeiler et denormalisert syn, og formatet til en slik indeks kalles gjerne indeksformatet. Det kan således argumenteres for at informasjonen som er tilgjengelig fra databasen og den fra (indeks-)søkemotoren, er semantisk ekvivalente for formålene med et gitt søkespørsmål, og at å kjøre et databasespørsmål eller -søk er

ekvivalente i det minste med hensyn til medlemskap i mengden, det vil si at resultatmengden for hvert søkespørsmål vil være den samme. Imidlertid vil rangering og representasjon typisk være ulik for resultatmengden som returneres i hvert tilfelle.

Fra US publisert patentsøknad nr. 2007/0239666 A1 publisert 11. oktober 2007 (Garcia, overdratt til Enporion, Inc., Tampa, FL (US)) er det kjent et system og fremgangsmåte for søking i en denormalisert database. Fremgangsmåten omfatter å motta en søkeanmodning fra en søker og konstruere et konfigurert søkespørsmål fra en konfigurasjonstabell basert på brukers identitet og deretter eksekvere et enkelt inngrep i en innholdstabell som bruker konfigurasjonsspørsmålet for å hente innhold fra innholdstabellen. Fremgangsmåten omfatter å generere en eller flere tråder for en eller flere konfigurasjonsspørsmål og eksekvere en eller flere tråder i parallell, slik at eksekveringen kan fordeles og omfordeles over en eller flere applikasjonsklynger, og å fremlegge ett eller flere søkeresultater i en felles resultattabell.

Da SQL (Structured Query Language) og et søkespørsmålspråk har signifikant overlappende funksjonalitet, er det mulig å implementere søkefrontender for databaseoppslag i et søkespørsmålspråk. Det er også mulig å implementere databaseoppslag mot en søkeindeks. Funksjonaliteten er ikke identisk og en rekke trekk vil typisk gå tapt, men resultatene vil fortsatt være anvendelige i mange tilfeller.

Bruken av nåværende og kjent teknikk har to ulemper. For å finne informasjon må det velges å bruke databaseoppslagsoperasjoner mot en database eller søkeprimitiver mot en indeks. Følgelig vil enten a) oppslaget ved å benytte søkeprimitiver fra starten av, ikke være tilgjengelig før en indeks er formatert og utfylt, eller b) dersom det benyttes databaseprimitiver, er en reimplementering av oppslagsfunksjoner ofte nødvendig for å redusere søkelatens senere.

HENSIKTEN MED OPPFINNELSEN

I lys av de ovennevnte ulemper med kjent teknikk er det en hovedhensikt med den foreliggende oppfinnelse å redusere latens inntil et søk er tilgjengelig.

En annen hensikt med den foreliggende oppfinnelse er å redusere kostnaden for å konfigurere en søkeindeks.

Endelig er det også en hensikt med den foreliggende oppfinnelse å redusere ressursforbruket til en søkeindeks.

SAMMENDRAG AV OPPFINNELSEN

De ovennevnte hensikter så vel som andre trekk og fordeler realiseres i henhold til den foreliggende oppfinnelse med en fremgangsmåte som er kjennetegnet ved å representere deler av den denormaliserte indeks fysisk som en søkeindeks og andre deler som separate tabeller, idet de separate tabeller konstrueres som én eller flere blant uavhengige søkeindekser, databasetabeller, eller indekserte databasetabeller, eller kombinasjoner av disse, hvorved eksekvering av søk kan finne sted før søkeindeksen fylles.

Ytterligere trekk og fordeler vil fremgå av de vedføyde, uselvstendige krav.

FIGURLISTE

Den foreliggende oppfinnelse skal forståes bedre når den etterfølgende detaljerte beskrivelse leses i samband med den vedføyde tegning, på hvilken

fig. 1 viser et blokkdiagram av en forenklet søkemotorarkitektur,

fig. 2 eksempler på datapost i form av tabeller, hvordan de er sammenføyd i en database, og en søkeindeks for denne databasen,

fig. 3 en enhetlig gjenfinningsarkitektur slik den kan benyttes i den foreliggende oppfinnelse og implementert på søkemotoren på fig. 1, og

fig. 4 hvordan attributter velges for sammenføyning i en indeks.

OPPFINNELSENS BAKGRUNN

Frengangsmåten i henhold til den foreliggende oppfinnelse skal implementeres på en søkemotor som kjent i teknikken. For bedre å forstå det operative miljø for den foreliggende oppfinnelse skal det gis en kort beskrivelse av en forenklet søkemotorarkitektur og med henvisning til fig. 1.

Den foreliggende oppfinnelse angår spesielt søkeapplikasjoner som implementert i søkesystemer på i og for seg kjente søkemotorer. Med tanke på å illustrere dette skal en søkemotor som kjent i teknikken og benyttet i foretaksøkesystemer nå kort drøftes med henvisning til fig. 1.

En søkemotor 100 i henhold til den foreliggende oppfinnelse vil som kjent i teknikken omfatte forskjellige undersystemer 101-107. Søkemotoren kan aksessere dokument- eller innholdsmagasiner anbrakt i et innholdsdomene eller -rom, hvorfra innholdet enten aktivt kan skyves inn i søkemotoren eller via en datakobling trekkes inn i søkemotoren. Typiske magasiner innbefatter databaser, kilder som står til rådighet via ETL(Extract-Transform-Load) -verktøy så som Informatica, ethvert XML-formatert magasin, filer fra filtjenere, filer fra vevtjenere, dokumenthåndteringssystemer, innholdshåndteringssystemer, e-postsystemer, kommunikasjonssystemer, samarbeidssystemer og rike media, så som audio, bilde, og video. De gjenfunne dokumenter leveres til en søkemotor 100 via et innholds-API (Application Programming Interface) 102. Deretter blir dokumentene analysert i et innholdsanalysetrinn 103, også kalt et undersystem for forhåndsprosessering av innhold, for å forbehandle innholdet for forbedrede søke- og oppdagelsesoperasjoner. Typisk er utgangen fra dette trinn en XML-representasjon av inngangsdokumentet. Utgangen fra innholdsanalysen benyttes til å mate kjernesøkemotoren 101. Kjernesøkemotoren 101 kan typisk være anbrakt på en tjenerfarm på en desentralisert måte for å gjøre det mulig å prosessere store mengder av dokumenter og høye spørsmålsbelastninger. Kjernesøkemotoren 101 kan akseptere brukeranmodninger og danne lister av tilsvarende dokumenter. Dokumentordningen blir vanligvis bestemt i henhold til en relevansmodell som måler den sannsynlige viktighet av et gitt dokument relativt til søkespørsmålet. I tillegg kan kjernesøkemotoren 103 frembringe ytterligere metadata for resultatmengden, f.eks. sammendragsinformasjon for dokumentattributter. Kjernesøkemotoren 106 omfatter i seg selv ytterligere undersystemer, nemlig et indekseringsundersystem 101a for nedsamling ("crawling") og indeksering av innholdsdokumenter og et søkeundersystem 101b for å utføre den egentlige søking og gjenfinning. Alternativt kan utgangen fra innholdsanalysetrinnet 101 mates inn i en valgfri varselmotor 104. Varselmotoren 104 vil ha lagret en mengde søkespørsmål og kan bestemme hvilke søkespørsmål som ville ha akseptert den gitte dokumentinnmating. En søkemotor kan aksesseres fra mange forskjellige klienter og applikasjoner som typisk kan være mobile eller datamaskinbaserte klientapplikasjoner. Andre klienter innbefatter PDAer og spillinnretninger. Disse klientene, som befinner seg i et klientrom eller -domene, vil levere anmodninger til en søkemotor- eller klient-API 107. Søkemotoren 100 vil

typisk besitte et ytterligere undersystem i form av et søkespørsmålsanalysetrinn 105 for å analysere og forfine søkespørsmålet for å konstruere et avledet søkespørsmål som kan ekstrahere mer meningsfylt informasjon. Endelig kan utgangen fra kjernesøkemotoren 101 ytterligere analyseres i et annet undersystem, nemlig et resultatanalysetrinn 106 for å frembringe informasjon eller visualiseringer som benyttes av klientene. – Begge trinn 105 og 106 er forbundet mellom kjernesøkemotoren 101 og klient-API 107, og i tilfelle varselmotoren 104 foreligger, er den forbundet i parallell med kjernesøkemotoren 101 mellom innholdsanalysetrinnet 103 og søkespørsmåls og resultatanalysetrinnene 105;106.

For den foreliggende oppfinnelses formål vil begrepet dokument bli benyttet synonymt med datapost som vil brukes til å betegne objekter som utgjør en database, og dermed unngå bibetydningen av et dokument som bare en tekststørrelse. Videre vil i et foretaksmiljø en viss omfattende datapostmengde heretter primært betraktes som en database, og denne database ikke bare er strukturert, men også datapostene i seg selv vil være strukturerte eller til og med ha en kompleks struktur. Dette står i sterk kontrast til dokumentmagasiner slik de påtreffes i åpne systemer så som på World Wide Web (Verdensveven), hvor informasjon står til rådighet fra et umåtelig høyt antall høyt diversifiserte kilder hvor informasjonsleverandørene utgjør en helt heterogen masse. I tillegg er mye av denne informasjon ustrukturert og foreligger i form enten av tekstdokumenter eller forskjellige rike medier så som audio og video som velkjent for brukere av World Wide Web.

Innenfor rammen av et foretak kan informasjon som genereres eller eies av foretaket, være spredt i én eller flere databaser som typisk er fordelt over en rekke lagringsinnretninger og håndtert av tjenerne til foretaket, og de vil dessuten støtte og betjene en hvilken som helst klientgenerert applikasjon i foretaket. Databasene er vanligvis strukturert med strukturelle elementer som har et eller annet forhåndsvalgt format og som i seg selv vanligvis viser en kompleks indre struktur. Et typisk eksempel vil være en tabell eller lister som inneholder dataposter med attributtnavn og tilhørende verdier for disse attributter. Med andre ord kan en blanding av numerisk og tekstlig informasjon og med et stort antall attributter og til og med et enda større antall attributtverdier, bli tilordnet til dataposter med varierende størrelse, og disse er ytterligere tilordnet like store og til og med enda større strukturelle

elementer, f.eks. tabeller som sammen med attributtene kan betraktes å utgjøre et informasjons- eller datapostmengde i databasen.

UTFØRELSER AV OPPFINNELSEN

På fig. 2 kan tabellene 201, 202 anses å utgjøre en database. Hver tabell omfatter et par av attributter eller attributtnavn. I tabell 201 er attributtnavn *Person id* og *Navn*, i tabell 202 *Person id* og *Telefon* og hver attributt kan ha forskjellige verdier. For eksempel har attributtnavnet *Person id* i tabell 201 fire verdier 1, 2, 3, 4. I hver av tabellene 201 og 202 blir en datapost dannet av et par attributtverdier tilordnet til de respektive attributtnavn. For eksempel omfatter tabellene 201 og 202 fire dataposter. I tabell 201 kan en datapost være eksemplifisert som "1 (*Person id*); Geirr (*Navn*)". Som velkjent for fagfolk er attributtet eller attributtnavnet synonymt med felt eller feltnavn som benyttet på strukturerte dataposter i konvensjonelle databasesystemer.

Det er i dag vanlig praksis å generere indekser for individuelle attributter i databasen for å oppnå raskere aksess. Dette er forskjellig fra det som beskrives her, hvor indeksen erstatter et sammenføyd syn i stedet for en enkelt attributt innenfor en tabell.

I henhold til den foreliggende oppfinnelse blir dataposter sammenføyd i en database og en denormalisert indeks spesifiseres og dannes for denne databasen, som vist på fig. 2 med to tabeller 101, 102 som eksempel. Tabell 201 avbilder personidentifikasjoner *Person id* til *Navn*, og tabell 202 avbilder personidentifikasjoner *Person id* til telefonnummer *Telefon*. Resultatet av å sammenføye disse dataposter er uformelt vist som database 203 og en sannsynlig representasjon av databasen 203 i en søkeindeks i form av en denormalisert indeks 204. Bemerk at en person med flere telefonnummer ender opp med å være representert av en rekke dataposter etter sammenføyning, mens dette i mange tilfelle fortsatt kunne være én datapost i en søkeindeks fokusert på personer.

For den foreliggende oppfinnelse defineres en enhetlig datagjenfinningsarkitektur (Unified Data Retrieval Architecture (URDA)) som en kombinasjon av attributter som forhåndssammenføyes til indekser, med usammenføyde attributter i databasen. Den optimale kombinasjonen er applikasjonsspesifikk, da egenskapene og adferden til URDA avhenger av utformingen. Forhåndssammenføyning vil vanligvis øke lagringskostnaden,

men reduserer opplagslatens og omvendt, med de degenerative tilfeller ingen forhåndssammenføyning (bare databaseoppslag, med lav lagringskostnad, men også høy latens) som et første ytterpunkt og en enkelt indeks (alle felt eller attributter denormalisert og indeksert) som et annet ytterpunkt. En enhetlig datagjenfinningsarkitektur er vist skjematisk på fig. 3.

Den foreliggende oppfinnelse omfatter generering av en indeks på forespørsel. Dette skal nå forklares med henvisning til URDA-skjematikken på fig. 3 som også kan tolkes som en strukturell representasjon av et søkesystem som kan likefrem avbildes på søkemotorarkitekturen i fig. 1. Database 301 lagrer dataene, og søkeindeksen 302 er den resulterende, avledede indeks. Dataflyt 307 viser hvordan data ekstraheres fra databasen 301 for å innsettes i søkeindeks 302. SQL-grensesnitt 303 er spørsmålsgrensesnittet som uten prosessering søker i database 301, og aksessvei 305 viser hvordan SQL-grensesnittet 303 også kan avspørre indeks 302. Tilsvarende aksesserer søkegrensesnitt 304 iboende søkeindeksen 302 og aksessvei 306 viser hvordan dette søkegrensesnittet kan aksessere data fra databasen 301. Føderasjonsgrensesnittet 308 viser hvordan et spesialgrensesnitt vil trekke data fra både databasen 301 gjennom SQL-grensesnitt 303 og fra søkeindeks 302 gjennom søkegrensesnitt 304 for å gjenfinne resultater.

Før søkeindeksen 302 etableres, må alle søkespørsmål håndteres ut fra databasen 301, enten direkte gjennom SQL-grensesnittet 303 eller gjennom søkegrensesnittet 304. Straks søkeindeksen 302 er fylt og tilgjengelig, har føderasjonsgrensesnittet 308 mulighet til å optimere søkespørsmål ved å benytte denne indeksen gjennom søkegrensesnittet 304.

Som vist på fig. 2, er indeksen 204 utledet fra og funksjonelt ekvivalent med den representative del av databasen 203. Følgelig kan databasen 301 og søkeindeksen 302 som gjengitt i søkesystemet på fig. 3 sammen ses som én, med den virkning at indeksering gir lavere latensitet for oppslag.

Valget av attributter som skal forhåndssammenføyes, kan baseres på kardinaliteten til attributtene, antallet distinkte verdier for et attributt og selektiviteten til søkespørsmål, dvs. antallet resultater for et gitt søkespørsmål. Dette skal nå drøftes i noe detalj med henvisning til fig. 4, som henholdsvis viser en database 41 med fire tabeller A, B, C, D med attributtnavnene "Id", "Navn", "Land". "Høyde" og "Telefon", og et

varierende antall dataposter som utgjør rekker i tabellene som utgjør 2-tupler av hver attributtverdi. La "Id" være den primære attributt for hvilken det vil være en attributtverdi pr. datapost (eksemplifisert av attributtet "Id" i tabell A i databasen 41). Én eller flere avhengige attributter skal nå forhåndssammenføyes i en denormalisert indeks eller databasesyndokument, f.eks. eksemplifisert som attributtet "Telefon" i tabell A i databasen 41. Hvis det er en rekke avhengige attributtverdier for hver primær attributtverdi, vil forhåndssammenføyningen være mer gunstig både med hensyn til oppslagsytelse og størrelseskostnader enn hva tilfellet er hvis det bare er et avhengig attributtnavn for hver primære attributtverdi.

Resultatet av en slik forhåndssammenføyning er den utledete, denormaliserte indeks 42. Her er den primære attributtet "Id" med den tilføyde, avhengige attributtet "Navn" fra tabell A forhåndssammenføyd i indeksen 42 med to andre avhengige attributtverdier fra tabellene B, C, og D. Bemerk at det er to avhengige attributtverdier av "Telefon" for verdien 2 av "Id", nemlig 8002 og 8003.

Hvis en søkespørsmålslogg analyseres, kan langt bedre, målrettede forbedringer oppnås. Anta at mange søk ser etter "Navn" og "Land" i databasen 41 på fig. 4 som eksempel, og meget få søk ser etter "Telefon" og "Høyde". I dette tilfellet blir det mer attraktivt å konstruere et indeksopplegg som angitt i del 43 på fig. 4, hvor de avhengige attributter fra tabellene C og D, nemlig "Navn" og "Land" forhåndssammenføyes til den primære attributt "Id", mens "Telefon" og "Høyde" bare vil bli ettersammenføyd etter behov, og som i forbifarten vil være denormalisert, dvs. lik indeks 42.

Når en søkespørsmålslogg er tilgjengelig, er en annen mulighet å generere alle eller en undermengde av kombinasjonen av forhåndssammenføyninger og ettersammenføyninger, og en eller flere indekser, og å kjøre søkespørsmålene på ny mot hver av disse konfigurasjoner. Denne utførelse av fremgangsmåten i henhold til oppfinnelsen vil være mer presis enn en som bare er basert på å prediktere selektivitet og definisjonsområde for hvert søkespørsmål, da den også *de facto* vil ta i betraktning den virkelige fordeling av data. For at dette skal være nøyaktig, må imidlertid både de underliggende data og spørsmålsloggen være representative for de virkelige data og søkespørsmål i en bruksmåte som har optimering som mål.

For en gitt mengde av attributter er det mulig med mange kombinasjoner av indekser og databaseoppslag. Som vist på fig. 4 dannes en indeks som omfatter alle avhengige felt eller attributtnavn i databasen 41, nemlig alle attributter "Navn", "Land", "Høyde" og "Telefon". Ved å betegne disse henholdsvis v, x, y, z og indeksen 42 Id(v, x, y, z), hvor Id er den valgte primære attributt, kan sammenføyningen J av de avhengige attributter, formelt vJxJyJz, betegnes Indeks (u, v, x, y, z), hvor u, v, x, y, z er de respektive attributtnavn. Denne indeks vil typisk være større enn den indeks som er nødvendig, da søkeindekser kan ha mengder av verdier – sammenlign tabellene 203 og 204 på fig. 2. Andre kombinasjoner kunne f.eks. innbefatte [Id (v, x), JyJz], som er eksemplifisert av den sammenføyde databaseindeks 43 på fig. 4. Her blir indeksen for primærattributtet Id dannet over v og x og resultatet fra dette søk sammenføyes med y og z fra tabellene C og D.

For å oppsummere blir i denne utførelse av den foreliggende oppfinnelse databasen 41 som er strukturert i tabeller og dataposter, indeksert som en denormalisert søkeindeks 42. Det er da ansett som fordelaktig at bare en del av søkeindeksen 42 skal representeres som en fysisk søkeindeks, mens resten av søkeindeksen 42 kan være plassert i separate tabeller. Valg av attributtnavnene i søkeindeksen 42 for sammenføyning i en fysisk søkeindeks er basert på å evaluere skjemaet for en mengde av tabeller og spesifikasjonen av denne denormaliserte søkeindeks statisk, dvs. uten å benytte en søkespørsmålslogg, og betegnes som indeksbalansering. Det skal bemerkes at skjemaet er attributtmengden for en datapostmengde eller en tabellmengde. Attributtnavnene bør velges på basis av målinger og antagelser om frekvensen med hvilken de etterspørres. Typisk kan databaseadministratoren iverksette slike målinger eller gjøre de underliggende antagelser. Mer hyppig etterspurte attributter bør være indeksert; mindre hyppig etterspurte attributter bør aksesseres fra databasen.

Som kjent i teknikken, er det vanlig at et søkesystem frembringer en logg av inngitte søkespørsmål, og denne loggen kan i en annen foretrukket utførelse benyttes til å identifisere og analyse søkemønsteret for å generere optimale indekskonfigurasjoner. Et eksempel vil vise det underliggende grunnlag for dette. Anta at om f.eks. 20 % av søkespørsmålstermene benyttes i 80 % av søkespørsmålene som reelt inngis til søkesystemet, er den innlysende konsekvens at indeksen konfigureres dynamisk med hensyn til oppførselen som gjenspeiler søkespørsmålet. Med andre ord kan den denormaliserte

søkeindeks 42 nå rekonfigureres som en fysisk søkeindeks som representert i del 43 på fig. 4 hvor den fysiske søkeindeks er begrenset til attributtene "Navn" og "Land". Attributtene "Høyde" og "Telefon", som forekommer som stikkord i bare en liten andel av det totale antall søkespørsmål, kan holdes som separate tabeller i del 43 på fig. 1, tilsvarende databasetabellene C og D for databasen 41. Slik det vil være innlysende for fagfolk, kunne den valgte indekskonfigurasjon velges for å være tilpasset hva som ville betraktes som akseptable ytelseskarakteristikker, og typisk ville søkespørsmålsytelse betraktes enten uttrykt som responstid, eller relevansen til dokumentene, eller så ville datapostene som returneres i resultatmengden betraktes som også å være i stand til å romme volumet av den fysiske søkeindeks. Dette vil gjenspeiles i lagrings- og oppdateringskostnadene som f.eks. i tilfelle av den fysiske søkeindeks i del 43 på fig. 4 vil være meget lavere enn lagrings- og oppdateringskostnadene for den denormaliserte søkeindeks 42. I en foretrukket utførelse av den foreliggende oppfinnelse kan kandidatkonfigurasjoner for søkeindekser genereres på basis av en opprinnelig søkeindeksspesifikasjon. For eksempel kunne dette gjøres direkte ved å benytte databasetabellene til databasen 41 eller ved å generere søkeindekskandidater som undermengder av den denormaliserte søkeindeks 42. Deretter kunne de genererte kandidatindekskonfigurasjoner testes og evalueres mot loggen for virkelige søkespørsmål, og en kandidatindekskonfigurasjon velges med hensyn til å oppfylle ytelseskarakteristikkene som ville være optimale uttrykt ved både søkeeffektivitet og søkekostnad.

Forut for den foreliggende oppfinnelse kunne søk bare utføres etter at indeksen var fylt. Med denne oppfinnelsen kan søk finne sted umiddelbart etter å ha definert formatet for søkeindeksen. Selv om fremgangsmåten i henhold til den foreliggende oppfinnelse kan resultere i en noe øket lagringskostnad, blir oppslagslatensen betraktelig redusert. Den førstnevnte kan betraktes som en marginal faktor, mens den sistnevnte vil imøtekomme brukerens forventninger om et søkesystem med høy ytelse.

Endelig skal det bemerkes at selv om den foreliggende oppfinnelse er spesielt relevant for foretakssøkesystemer, er den ikke på noen måte begrenset til disse, og som innlysende for fagfolk kunne det generelt benyttes på alle former for innhold som omfatter dataposter med attributtnavn og deres verdier.

PATENTKRAV

1. Fremgangsmåte til å forbedre søkeytelse i et system for søking og gjenfinning av informasjon implementert på en søkemotor, hvor et søkespørsmål benyttes på en database som inneholder dataposter i form av tabeller, hvor fremgangsmåten omfatter å spesifisere og danne en denormalisert søkeindeks, og hvor fremgangsmåten er k a r a k t e r i s e r t v e d å representere deler av den denormaliserte indeks fysisk som en søkeindeks og andre deler som separate tabeller, idet de separate tabeller konstrueres som én eller flere blant uavhengige søkeindekser, databasetabeller, eller indekserte databasetabeller, eller kombinasjoner av disse, hvorved eksekvering av søk kan finne sted før søkeindeksen fylles.

2. Fremgangsmåte i henhold til krav 1, k a r a k t e r i s e r t v e d å balansere forholdet mellom delene av den denormaliserte søkeindeks som henholdsvis skal representeres som den fysiske søkeindeks eller plasseres i separate tabeller.

3. Fremgangsmåte i henhold til krav 2, k a r a k t e r i s e r t v e d å velge fra den denormaliserte søkeindeks attributtnavn som skal tilordnes til den fysiske søkeindeks, og fra den denormaliserte søkeindeks attributtnavn som skal tilordnes til de separate tabeller.

4. Fremgangsmåte i henhold til krav 3, k a r a k t e r i s e r t v e d å velge attributtnavnene ved å evaluere et skjema for mengden av tabeller og spesifikasjonen av den denormaliserte søkeindeks statisk på basis av en indeksstruktur.

5. Fremgangsmåte i henhold til krav 1, k a r a k t e r i s e r t v e d å bestemme søkemønstre på basis av en søkespørsmålslogg, og å benytte søkemønstrene til en analyse av optimale indekskonfigurasjoner med hensyn til akseptable ytelseskarakteristikker.

6. Fremgangsmåte i henhold til krav 5, k a r a k t e r i s e r t v e d å velge de akseptable ytelseskarakteristikker som minst én eller flere blant en søkespørsmålsytelse, lagringskostnader og oppdateringskostnader.

7. Fremgangsmåte i henhold til krav 5, k a r a k t e r i s e r t v e d å generere en mengde av kandidatkonfigurasjoner for søkeindeksen på basis av en original søkeindeksspesifikasjon, og å evaluere hver kandidatindekskonfigurasjon over søkespørsmålsloggen for å bestemme kandidatindekskonfigurasjonens egnethet med hensyn til å skaffe optimale ytelseskarakteristikker.

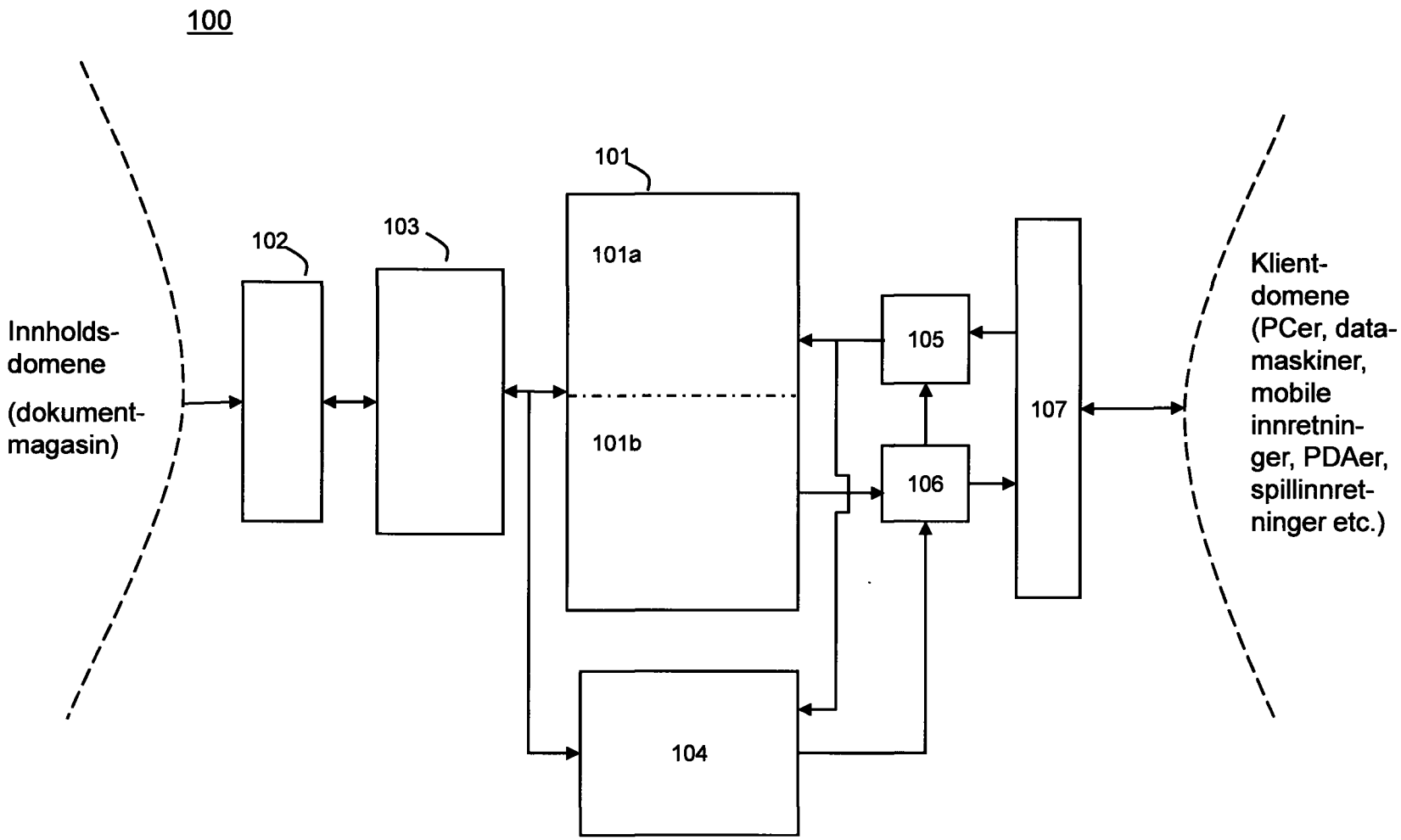
100
Innholds-
domene
(dokument-
magasin)
102
103
101
101a
101b
104
105
106
107
Klient-
domene
(PCer, data-
maskiner,
mobile
innretnin-
ger, PDAer,
spillinnret-
ninger etc.)

Fig. 1

Fig. 2

201

Person id	Navn
1	Geirr
2	John
3	Petter
4	Tom

202

Person id	Telefon
1	1101
1	1102
2	1132
3	1156

203

Navn	Telefon
Geirr	1101
Geirr	1102
John	1132
Petter	1156

204

Navn	Telefon
Geirr	1101, 1102
John	1132
Petter	1156

Fig. 3

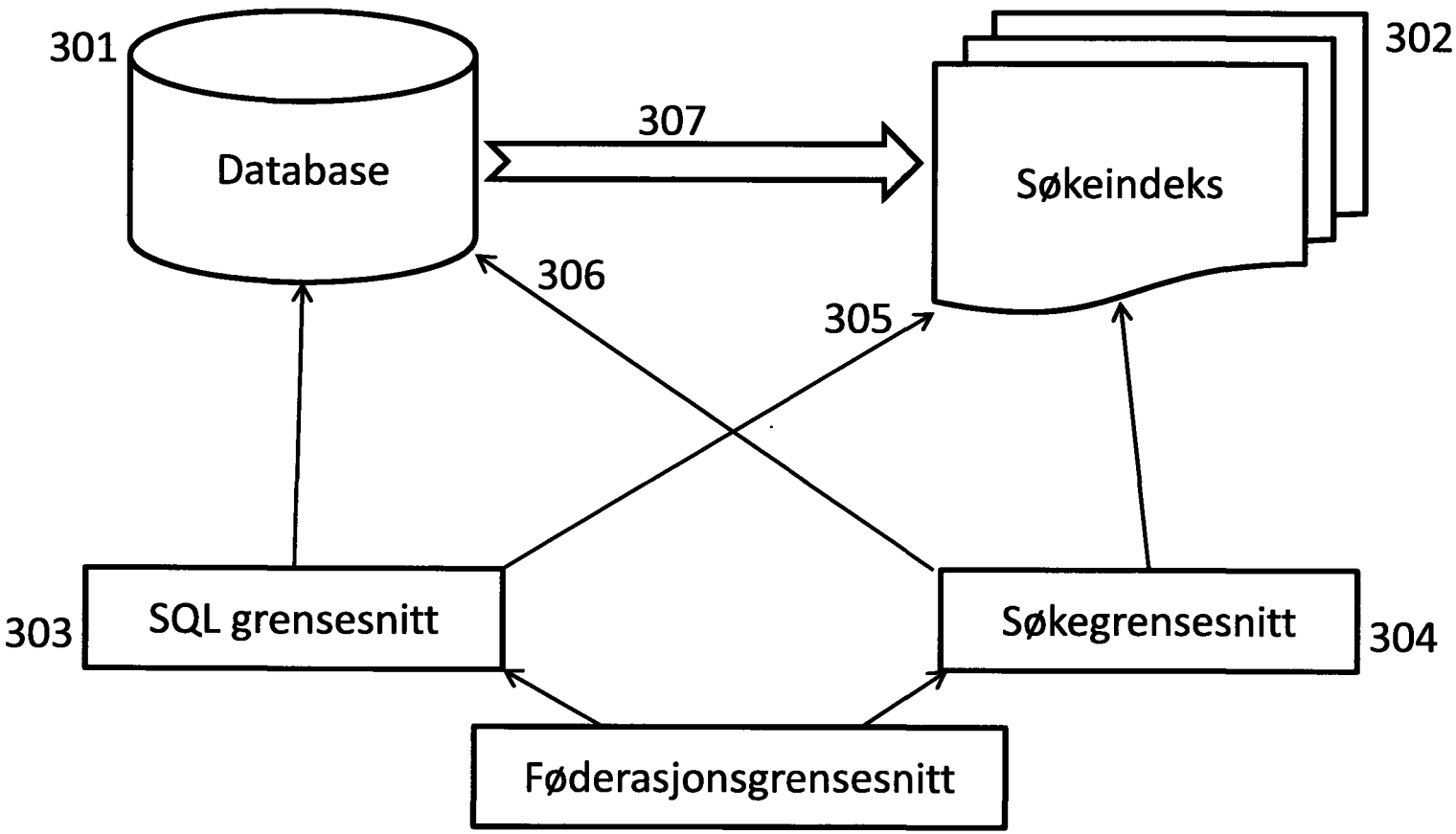
301
Database
307
302
Søkeindeks
306
305
303
SQL grensesnitt
Søkegrensesnitt
304
Føderasjonsgrensesnitt

Fig. 4

41

A

Id	Navn
1	Tom
2	John

B

Id	Land
1	Norway
2	Norway

C

Id	Høyde
1	178
2	180

D

Id	Telefon
1	8001
2	8002
2	8003

42

Id	Navn	Land	Høyde	Telefon
1	Tom	Norway	178	8001
2	John	Norway	180	8002,8003

43

Id	Navn	Land
1	Tom	Norway
2	John	Norway

C

Id	Høyde
1	178
2	180

D

Id	Telefon
1	8001
2	8002
2	8003