Intern Bibliotheek Systeem (IBS)

Catalogus - module

Judith van Rijt

Intern Bibliotheek Systeem (IBS)
Catalogus Module

Judith van Rijt

Centrum voor Wiskunde en Informatica
Sector Technische Ondersteuning

Begeleiders: drs. J. Wolleswinkel

drs. H. Noot

Datum: 30 juni 1989

INHOUD

1. *INLEIDING*

1.1. Beschrijving van de invoer en de uitvoer van het catalogus-systeem

Het catalogus-systeem is een onderdeel van het bibliotheek-systeem. Het hoofddoel van het catalogus-systeem is het op micro-fiche en op papier zetten van fiches.‡ Een fiche wordt gemaakt van een document. In deze beschrijving van het catalogus-systeem wordt met een document de bibliografie van boeken, tijdschriften, artikelen, rapporten en dergelijke bedoeld. De gebruiker moet opgeven welke verschillende catalogi er zijn. Voorbeelden van catalogi zijn: een boeken-catalogus, een tijdschriften-catalogus, maar ook een wiskunde-boeken-catalogus of informatica-tijdschriften-catalogus en ook wiskunde-boeken-catalogus gesorteerd op auteur of informatica-tijdschriften-catalogus gesorteerd op titel, enzovoorts. Ook een aanwinstenlijst wordt in dit systeem gezien als een (weliswaar bijzondere) catalogus.

Met behulp van files met een soort BibTeX-functies, maakt de gebruiker duidelijk welke catalogi er zijn en wat de specificaties van iedere catalogus zijn. Uit deze specificaties moeten de volgende punten worden afgeleid:
- Welke documenten wel en welke niet tot de catalogus behoren.
- Hoe de fiches er uit moeten komen te zien (welke velden in het fiche moeten worden opgenomen en welke volgorde deze velden hebben, bijvoorbeeld eerst de auteur gevolgd door de titel gevolgd door de uitgeverij enzovoorts).
- Hoe de fiches gesorteerd moeten worden (bijvoorbeeld eerst op auteur en dan op titel).
- Op wat voor een soort uitvoer de fiches moeten komen. Mogelijkheden hiervoor zijn: micro-fiches (of een afdruk van dit bestand op papier) en via een tekst-verwerker op papier.
- De lay-out van de uitvoer (bijvoorbeeld de kopjes boven een pagina en in geval van micro-fiches of een afdruk direct op papier, de pagina-grootte en in het geval dat een deel van de sleutel-gegevens van meer dan een fiche hetzelfde is, bijvoorbeeld dezelfde eerste auteur, deze sleutel-gegevens alleen bij het eerste fiche vermelden en bij de andere fiches deze ruimte open laten, enzovoorts).

In dit systeem is gekozen om de fiches niet per keer dat er uitvoer nodig is te maken, maar om één keer een catalogus te maken en deze te bewaren. Het voordeel van deze opzet is dat het afdrukken van een catalogus per keer veel minder tijd kost. Een nadeel is dat er veel grote bestanden zijn, die bewaard moeten blijven. Deze bestanden kunnen bijvoorbeeld op tape worden gezet. Een ander nadeel is dat er mutaties op de catalogi plaats kunnen vinden. Er kunnen fiches worden toegevoegd, veranderd of verwijderd. Hiervoor moet een voorziening worden getroffen.

Het catalogus-systeem heeft twee soorten invoer. Ten eerste de zojuist genoemde BibTeX-functies† en ten tweede vijf soorten tabellen, die door een ander sub-systeem namelijk het update-module worden gemaakt. Deze tabellen zijn: Ten eerste de tabellen daTYflds. Deze tabellen bevatten de documenten. De letters TY in de naam van deze tabellen staan voor het type documenten dat de bepaalde tabel bevat. Er kan bijvoorbeeld 'bk' worden ingevuld voor een tabel die boeken bevat. De tabel heet dan dabkflds. De tabellen daTYflds worden onder meer gebruikt voor het opzetten van catalogi. Alle documenten uit de tabel worden dan, aan de hand van de catalogus-specificaties, in nul, een of meer catalogi geplaatst, in de vorm van een fiche. Naast het opzetten van een nieuwe catalogus worden deze tabellen gebruikt om documenten in op te zoeken.

---

‡ Fiches zijn document-beschrijvingen. Op één micro-fiche staan een aantal fiches.

† Dit zijn in werkelijkheid geen BibTeX-functies, maar het zijn functies die er veel op lijken. In deze beschrijving van het catalogus-systeem, wordt echter net gedaan alsof het wel echte BibTeX-functies zijn.

Ten tweede de tabellen SELIDS. Ook deze tabellen worden gebruikt voor het opzetten van nieuwe catalogi. Zij zijn bedoeld voor toepassingen waarin op een manier, die buiten de hier beschreven selectie-mechanisme om gaat, een tabel met document-id's (en type's) kan worden aangemaakt (bijvoorbeeld via SQL-statements). Elke rij uit een tabel bestaat uit een id en een type. Het id is de identificatie van het document. Het document is te vinden in een tabel daTYflds. Het type van de rij geeft het type van de tabel daTYflds aan. Is het type van de rij bijvoorbeeld 'bk' dan staat het document in de tabel dabkflds. Net als bij het opzetten van catalogi met behulp van een tabel daTYflds wordt bij het opzetten van catalogi aan de hand van een tabel SELIDS ieder document, dat via de tabel daTYflds is ingelezen, naar gelang de catalogus-specificaties, in nul, een of meer catalogi geplaatst in de vorm van een fiche.

Ten derde de tabellen upTYflds. Deze tabellen worden gebruikt voor toevoegingen en veranderingen op bestaande catalogi. Ze bevatten documenten die toegevoegd en de veranderingen van de documenten die gewijzigd moeten worden. Voor de letters TY moet weer een type worden ingevuld.

Ten vierde de tabellen up2TYflds. Deze tabellen worden gebruikt voor veranderingen op bestaande catalogi. Ze bevatten de veranderingen die op de documenten hebben plaatsgevonden, sinds de laatste verandering/toevoeging is verwerkt. Deze tabellen zijn naast de tabellen upTYflds nodig als er, nadat een document is toegevoegd/veranderd (en verwerkt door het catalogus-systeem), het document gewijzigd moet worden, zonder dat het update-module de eerste toevoeging/verandering op de tabel daTYflds heeft verwerkt. Voor de letters TY moet weer een type worden ingevuld.

Ten vijfde de tabel mutats. Deze tabel wordt gebruikt om wijzigingen in documenten door te voeren op de catalogi. Er zijn drie soorten wijzigingen, te weten: toevoegen, veranderen en verwijderen. Per rij is er een identificatie van het document, het type van de tabel waarin het document is opgenomen, het soort verwerking dat moet plaatsvinden (toevoegen, veranderen of verwijderen) en een veld dat aangeeft of de verandering wel of niet heeft plaatsgevonden. Bovendien heeft de tabel een veld which dat aangeeft hoe het oude en/of het nieuwe document te maken is. Bijvoorbeeld het oude document staat in de tabel daTYflds, of het nieuwe document staat in de tabel upTYflds, of het document staat in de tabel upTYflds en de bijbehorende wijzigingen staan in up2TYflds, enzovoorts.

Behalve de uitvoer van micro-fiches en het afdrukken van catalogi op papier (hetzij direct, hetzij via de tekst-verwerker) moet het catalogus-systeem twee tabellen bijhouden. Ten eerste een history tabel en ten tweede een account tabel. Telkens wanneer een nieuwe catalogus gemaakt wordt, of als er een mutatie plaats vindt op een fiche uit een catalogus wordt de tabel history bijgewerkt. De bedoeling van deze tabel is dat wanneer er iets verkeerd is gegaan, eenvoudig kan worden nagegaan wat er precies gebeurd is.

In de tabel account worden gegevens over het afdrukken opgenomen. Telkens als er uitvoer gegenereerd wordt, moet er worden opgenomen wanneer die uitvoer gegenereerd is, van welke catalogus, wat voor een soort uitvoer het betrof (direct op papier, op micro-fiche of via de tekst-verwerker op papier) hoeveel fiches afgedrukt zijn, om hoeveel verschillende documenten het gaat (van een document kunnen in dezelfde catalogus verschillende fiches staan, bijvoorbeeld op de plaats van de tweede auteur een verwijzing naar de eerste auteur), hoeveel verschillende fiches er zijn afgedrukt (in de catalogus kan meer dan een keer hetzelfde fiche zijn opgenomen, als bijvoorbeeld het fiche zowel op de plaats van de eerste als op de plaats van de tweede auteur wordt afgedrukt) en het aantal fiches die zijn afgedrukt. Deze tabel bevat dus de productie-statistieken.

1.2. Korte beschrijving van het catalogus-systeem

In de volgende tekst staan de namen van de procedures tussen quotes en de namen van de tabellen en bestanden tussen dubbele quotes. In bijlage A is een schets van het catalogus-systeem opgenomen. Op deze schets staat een tabel "CTSPCS". In werkelijkheid is dit een verzamelnaam van vier tabellen, die te zamen de catalogus-specificaties bevatten. Deze tabellen heten: "caspccod", "caspcstr", "caspcatt" en "caspcprd".

In de opzet van dit systeem is ervoor gekozen om sleutel- en fiche-bestanden gescheiden te houden. De sleutel-bestanden zorgen ervoor dat de fiches in de goede volgorde komen te staan. Het gescheiden houden van de sleutel- en fiche-bestanden is gedaan om het sorteren van de fiches mogelijk te maken.

In het catalogus-systeem worden de volgende procedures gebruikt:

'update catalog descript': Deze procedures leest de files met BibTeX-functies in en zet deze met behulp van Lex en Yacc om naar catalogus-specificaties in de tabellen: "caspccod", "caspcstr", "caspcatt" en "caspcprd".

'verwerk documenten tot sleutels en fiches': Deze procedure leest afhankelijk van de meegegeven parameters documenten uit de tabellen "daTYflds","upTYflds" en "up2TYflds", maakt van deze documenten, aan de hand van de in de parameters meegegeven catalogus-specificaties, nul, één of meer fiches voor het fiche- en nul, één of meer sleutels voor het sleutel-bestand. Voor iedere catalogus waaraan tenminste één fiche van het document wordt toegevoegd, gewijzigd of verwijderd, wordt een rij in de tabel "history" gezet.

'orden sleutels': Deze procedure sorteert alle sleutel-bestanden die door de procedure 'verwerk documenten tot sleutels en fiches' gemaakt zijn.

'update sleutels': Deze procedure voert een soort merge uit met het sleutel-bestand en het bestand dat door de procedure 'orden sleutels' gesorteerd is. Sleutels uit dit laatste bestand kunnen hetzij een verwijder-teken hetzij een invoeg-teken bevatten. Sleutels met een verwijder-teken worden niet meer opgenomen (de fiches worden zo als het ware uit het fiche-bestand gehaald† omdat er niet meer naar verwezen wordt) en sleutels met een invoeg-teken worden op de juitste plaats in het sleutel-bestand opgenomen (de fiches worden als het ware in het fiche-bestand gezet).

'clean fiches': Deze procedure ordent de fiche-bestanden en past het adres van het fiche in de sleutel-bestanden aan. Met deze procedure wordt bereikt dat fiches waar niet meer naar verwezen wordt, worden verwijderd.

'pagineer': Deze procedure maakt een nieuw bestand dat klaar staat hetzij voor de procedure 'genereer uitvoer' hetzij voor de procedure 'tekst verwerken'. Tevens past de procedure de tabel "nodef" (aantal documenten, entries en fiches) aan. Indien het om een nieuwe catalogus gaat wordt er een rij in deze tabel bijgezet en indien het om een bestaande catalogus gaat, worden de getallen van de rij met de juiste id en type aangepast.

'genereer uitvoer': Deze procedure maakt aan de hand van de uitvoer van de procedure 'pagineer' micro-fiches of een afdruk van het bestand op papier en past vervolgens de tabel "account" aan, met de gegevens uit de tabel "nodef".

'tekst verwerken': Deze procedure geeft het bestand "caIDTYtin", de uitvoer van de procedure 'pagineer', door aan de tekst-verwerker. Deze tekst-verwerker zorgt dat er een afdruk op papier wordt gemaakt. De procedure past nadat de opdracht door de tekst-verwerker is verwerkt de tabel "account" aan, met de gegevens uit de tabel "nodef".

Uitgebreidere procedure-specificaties staan in hoofdstuk 2 en de pseudo-code van de procedures staat in

† zie 'clean fiches'

bijlage D.

Het catalogus-systeem maakt gebruik van de volgende ingres-tabellen:

"caspcprd": Deze tabel geeft aan welke catalogi bij een bepaalde productie-set horen. Een productie-set is een groep catalogi die altijd gezamenlijk worden aangemaakt. Er is één reguliere productie-set met catalogi, die als parameter aan de procedure 'verwerk documenten tot sleutels en fiches' meegegeven wordt, om mutaties op te verrichten. Deze productie-set bevat de alle catalogi die up to date gehouden moeten worden.

"caspccod": Bij iedere catalogus horen vier soorten specificaties, te weten: selectie, sleutel, fiche en pagina. Selectie geeft aan of een bepaald document wel of niet tot de catalogus behoort. Sleutel geeft aan hoe de sleutels gemaakt moeten worden. Fiche geeft aan hoe de fiches opgebouwd moeten worden en pagina geeft de lay-out van de catalogus aan en of de uitvoer bestemd is om micro-fiches te maken (of direct op papier te zetten) of om via een tekst-verwerker op papier te komen. Alle vier de specificaties staan in deze tabel.

"caspcstr": De tabel "caspccod" verwijst naar strings die in de tabel "caspcstr" staan. Bij de specificaties heb je strings nodig, bijvoorbeeld als je alleen engelstalige boeken in een bepaalde catalogus op wilt nemen moet je de string van het document die de taal aangeeft vergelijken met de string "eng".

"caspcatt": Deze tabel bevat acht getallen. Deze getallen zijn nodig, bij het alloceren van ruimte, bij het inlezen van de catalogus-specificaties en bij het evalueren van de specificaties.

"daTYflds": Dit zijn diverse tabellen met documenten. De letters die voor TY worden ingevuld, geven het type van de documenten aan die in de betreffende tabel staan. Deze tabellen worden onder meer gebruikt om nieuwe catalogi op te zetten.

"upTYflds": Dit zijn diverse tabellen met nieuwe documenten en gewijzigde gegevens van documenten. Ook hier geven de letters die voor TY worden ingevuld aan wat voor een type documenten de betreffende tabel bevat.

"up2TYflds": Dit zijn diverse tabellen met gewijzigde gegevens van documenten. Ook hier geven de letters die voor TY worden ingevuld aan wat voor een type documenten de betreffende tabel bevat.

"mutats": Deze tabel is er voor om mutaties uit te kunnen voeren op catalogi die bij de reguliere productie-set behoren. Er zijn drie soorten mutaties mogelijk te weten: toevoegen, verwijderen of wijzigen.

"SELIDS": Naast de tabellen "daTYflds" worden deze tabellen gebruikt om nieuwe catalogi op te zetten. Zij bevatten de id's en types van de in één katalogus op te nemen documenten. Deze tabellen kunnen bijvoorbeeld met behulp van SQL-statements worden gemaakt.

"history": Deze tabel bevat de gegevens, wanneer een bepaald document in een bepaalde catalogus is opgenomen, veranderd of gewijzigd; dit levert aanknopingspunten voor het geval het systeem onoverhoopt niet optimaal functioneert.

"account": Deze tabel bevat gegevens over de uitvoer op micro-fiches, of op papier, of via de tekst-verwerker op papier. Het bevat wanneer er wat voor een soort uitvoer is gegenereerd en van hoeveel documenten, entries (het aantal verschillende fiches) en het aantal fiches. Met behulp van deze tabel kan de gebruiker enig inzicht krijgen in de geproduceerde uitvoer.

"nodef": Deze tabel bevat het aantal documenten, entries en fiches die in een bepaalde catalogus opgenomen zijn. Tevens bevat iedere rij naast de drie getallen ook het id en het type van de catatogus. Deze tabel wordt bijgewerkt door de

procedure 'pagineer'. Het bevat de getallen die in de tabel "account" gezet moeten worden als er uitvoer geproduceerd wordt.

Een uitgebreidere beschrijving van de ingres-tabellen staat in hoofdstuk 3 en een attribuut-beschrijving van de ingres-tabellen staat in hoofdstuk 5.

Omdat de sleutels en de fiches een onbekende lengte hebben en het in ingres erg onhandig is om met onbekende lengte te werken, maken we gebruik van UNIX-bestanden. Bij ieder van de bestanden is een veld ID waar de catalogus-identificatie moet worden ingevuld en een veld TY waarop het type van de catalogus moet worden ingevuld. Voor het ID zou bijvoorbeeld 'wia' kunnen worden ingevuld, voor wiskunde auteurs catalogus en voor TY 'bk' voor boeken catalogus. In het bestand "cawiabkcumfchs" staan dan de fiches van wiskunde-boeken gesorteerd op auteur.

Het catalogus-systeem maakt de volgende UNIX-bestanden en uitvoer-producten:

"caIDTYupkeys": Deze bestanden worden door 'verwerk sleutels en fiches' gemaakt. Deze bestanden bevatten de sleutel van een fiche, een code die aangeeft wat er met sleutel moet gebeuren (toevoegen van de sleutel of verwijderen van de oude sleutel) en het adres van het fiche als de sleutel toegevoegd moet worden.

"caIDTYkeys": Deze bestanden bevatten de sleutel en het adres van het fiche in het bestand "caIDTYcumfchs".

"caIDTYcumfchs": Deze bestanden bevatten de fiches in willekeurige volgorde. Deze bestanden worden door de procedure 'clean fiches' zo goed mogelijk op volgorde gezet. De fiches kunnen gesorteeerd benaderd worden via het bestand "caIDTYkeys".

"caIDTYcumpags": Deze bestanden worden door het proces 'pagineer' gemaakt. Deze bestanden dienen als invoer voor de procedure 'genereer uitvoer'.

"caIDTYtin": Deze bestanden worden door het proces 'pagineer' gemaakt en dienen als invoer voor een tekst-verwerker.

"Micro Fiche": Dit zijn de micro-fiches. Ze worden door de procedure 'genereer uitvoer' gemaakt.

"Hard Copy": Dit zijn de fiches afgedrukt op papier. Deze uitvoer wordt ofwel door de procedure 'genereer uitvoer' (als het bestand "caIDTYcumpags" dat anders op micro-fiche zou komen, op papier moet worden afgedrukt), ofwel door de procedure 'tekst verwerken' gemaakt, afhankelijk van de catalogus-specificataties.

Een uitgebreidere beschrijving van deze bestanden staat in hoofdstuk 4.

2. *PROCEDURE-SPECIFICATIES*

De procedures 'verwerk documenten tot sleutels en fiches', 'orden sleutels' en 'update sleutels' moeten door het systeem op een vastgesteld tijdstip kunnen worden uitgevoerd en/of door de gebruiker via een ABF- of UNIX-applicatie kunnen worden aangeroepen. Hierbij moet het mogelijk zijn om wel of niet de procedure 'pagineer' en wel of niet een van de procedures 'genereer uitvoer' en 'tekst verwerken' aan te roepen. Ook moet het mogelijk zijn om via een ABF-applicatie alleen de procedure 'pagineer' en/of een van de procedures 'genereer uitvoer' en 'tekst verwerken' aan te roepen.

Apart van andere aanroepen moet de procedure 'clean fiches' worden aangeroepen ofwel op een vastgesteld tijdstip ofwel via een ABF- of UNIX-applicatie.

In de opzet van dit catalogus-systeem is gekozen om alle catalogi, ook als de verschillen tussen twee catalogi klein zijn, gescheiden te houden. Wanneer er bijvoorbeeld een wiskunde-boeken catalogus gesorteerd op auteur en een statistiek-boeken catalogus gesorteerd op auteur worden gemaakt, waarbij de statistiek-boeken catalogus een deelverzameling van de wiskunde-boeken catalogus is, worden deze toch als geheel verschillende catalogi gezien. Dit betekent dat beide catalogi ieder een eigen catalogus-specificatie en eigen bestanden "calDTYupkeys", "calDTYkeys", "calDTYcumfchs" en een bestand "calDTYcumpags" of "calDTYtin" hebben. Ook als er een catalogus is waarvan van ieder document een uitgebreide bibliografie is opgenomen en er een 'zelfde' catalogus is waarvan van dezelfde documenten een minder uitgebreide bibliografie is opgenomen, worden deze als geheel verschillende catalogi gezien.

In dit hoofdstuk wordt omschreven wat elke procedure moet doen. Verder wordt van iedere procedure een korte uitleg gegeven hoe hij in elkaar zit. Voor meer gedetailleerde uitleg van het programma verwijs ik naar bijlage D waarin de pseudo-code van de procedures is opgenomen.

2.1. 'Update catalog descript(ion)'

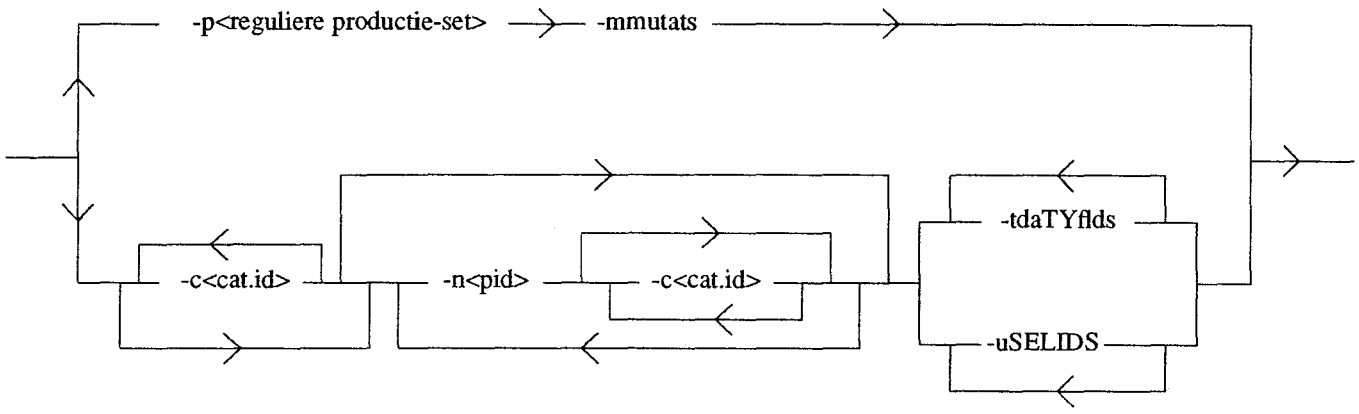
De gebruiker moet een file aanmaken met BibTeX functies. Voor iedere catalogus moeten er van die functies gemaakt worden. Deze opzet is gekozen, omdat BibTeX in de bibliotheek-wereld algemeen bekend is. De procedure 'update catalog desciption' leest deze procedures, interpreteert ze met behulp van Lex en Yacc, en zet de interpretatie in de tabellen "caspccod", "caspcstr", "caspcatt" en "caspcprd".

2.2. 'Verwerking van documenten tot sleutels en fiches'

Deze procedure moet aan de hand van de meegegeven parameters bepaalde mutaties of toevoegingen op bepaalde catalogi verrichten. Hiervoor is de procedure in vier delen gesplitst, namelijk: Inlezen van de parameters, inlezen van de catalogus-specificaties, inlezen van documenten en bekijken welke documenten betrekking hebben op welke catalogi en het maken van sleutels en fiches van deze documenten. Daarom heb ik deze paragraaf in vier sub-paragrafen gesplitst, te weten: 'wat te doen bij welke parameters', 'het inlezen van de catalogus-specificaties', 'het inlezen van documenten' en 'creeer sleutels en fiches'.

2.2.1. wat te doen bij welke parameters

Aan de procedure 'verwerk documenten tot sleutels en fiches' kunnen als volgt parameters worden meegegeven:



Wanneer de parameters -p<identificatie van de reguliere productie-set> -mmutats worden meegegeven, moeten de mutaties die in de tabel "mutats" staan worden uitgevoerd op alle catalogi die in de reguliere productie-set staan. De procedure test of de id van deze productie-set, bij de mutatie-tabel hoort. De optie -mmutats en de <identificatie van de reguliere productie set> moeten bij de -p optie worden meegegeven omdat het in de toekomst mogelijk zou moeten worden om meer dan een bibliotheek in dezelfde database onder te brengen. In dat geval heb je meer dan één tabel "mutats" en meer dan één reguliere productie-set nodig.

De procedure zal alle catalogus-specificaties die in de reguliere productie-set zijn opgenomen inlezen. Daarna zullen de mutaties die in de tabel "mutats" staan worden uitgevoerd op "alle catalogi" waarin het fiche volgens de ingelezen catalogus-specificatie moet voorkomen. (met "alle catalogi" bedoel ik de bestanden "calDTYcumfchs" en "calDTYupkeys"). Voor iedere sleutel die gemaakt is, wordt er een rij in de tabel "history" opgenomen.

Wanneer de -c<cat.id> optie wordt gekozen, worden de specificaties ingelezen van de catalogus-specificaties die de meegegeven identificatie hebben. Deze catalogus identificatie is in geen van de productie-set's opgenomen. Eventueel kan de -c<cat.id> optie herhaald worden.

Nadat nul een of meer 'losse' catalogus-specificaties zijn ingelezen, kan voor de -n<pid> optie gekozen worden. Voor <pid> moet een identificatie van een productie-set worden ingevuld. Wanneer er géén -c<cat.id> optie volgt, worden alle catalogus-specificaties van de productie-set ingelezen. Volgen er wèl één of meer -c<cat.id> opties, dan worden alléén die catalogus-specificaties ingelezen waarvan de identificatie in de -c<cat.id> optie is vermeld. Eventueel kan de -n<pid> optie worden herhaald.

Wanneer er (diverse) -t optie(s) zijn gegeven, worden alle documenten uit de gegeven tabel(len) "daTYflds" (in de vorm van een fiche met een sleutel) aan "alle catalogi" toegevoegd waaraan het document volgens de ingelezen catalogus-specificaties voldoet. (met "alle catalogi" bedoel ik de bestanden "calDTYcumfchs" en "calDTYupkeys"). Voor iedere sleutel die gemaakt is, wordt er een rij in de tabel "history" opgenomen.

Wanneer er (diverse) -u optie(s) zijn gegeven worden alle documenten via de tabellen "SELIDS" uit de tabellen "daTYflds" gelezen. In iedere rij van een tabel "SELIDS" staat de id van het document dat moet worden toegevoegd en het type van de tabel "daTYflds" waaruit het document gelezen kan worden. Verder verloopt de procedure hetzelfde als met de -t optie.

2.2.2. het inlezen van de catalogus-specificaties

Bij het inlezen van de catalogus-specificaties worden voor de procedure 'verwerk documenten tot sleutels en fiches' drie van de vier soorten specificaties ingelezen, namelijk: selectie, sleutel en fiche. De pagineer-specificatie hoeft hier niet te worden ingelezen.

In bijlage C is de header-file opgenomen van de structuren die bij het inlezen van catalogus-specificaties gebruikt worden. Bovendien staat in de bijlage een schets van de uiteindelijke structuur waarin de catalogus-specificaties na inlezen terecht zijn gekomen.

Voor iedere catalogus-id wordt er een struct gealloceerd. In deze struct komt het id van de catalogus te staan en vier pointers. De eerste pointer gaat naar een struct wijzen met selectie-specificaties, de tweede pointer gaat naar een struct wijzen met sleutel-specificaties, de derde pointer gaat naar een struct wijzen met fiche-specificaties en de vierde pointer gaat naar een struct wijzen met specificaties voor de volgende catalogus. De structs met selectie-, sleutel- en fiche-specificaties zien er hetzelfde uit. Ze hebben twee pointers en negen getallen. De eerste pointer wijst naar een code-array en de tweede pointer wijst naar een string-array. De getallen zijn de getallen uit de tabel "caspcatt". Het getal codes geeft de lengte van de code-array aan, zodat er precies genoeg ruimte voor de code-array gealloceerd kan worden. Het getal strings is de lengte van de string-array, zodat er precies genoeg ruimte voor de string-array gealloceerd kan worden. De andere zeven getallen zijn bij het inlezen van de catalogus-specificaties niet van belang.

Het code-array is een array van structs. Iedere struct bestaat uit twee integers code en value. Deze integers worden direct uit de tabel "caspccod" gelezen.

De string-array is een array van structs. Iedere struct heeft een pointer naar een string. De string waarnaar verwezen wordt, is de value uit de tabel "caspcstr". Wanneer twee of meer rijen met dezelfde id, sk en sq voorkomen, worden de value's van deze rijen als een string ingelezen, in de volgorde van het veld fr. De struct bevat naast de pointer ook een integer. Deze integer bevat de totale lengte van de string waar de pointer naar wijst.

2.2.3. het inlezen van documenten

Voordat er met het inlezen van documenten wordt begonnen, wordt het tijdstip van verwerking (sedert 1 januari 1970) in seconden bepaald. Deze tijd wordt in alle sleutels die gemaakt worden opgenomen in het veld seconden. Deze tijd is nodig als de procedure 'verwerkt documenten tot sleutels en fiches' verschillende keren wordt aangeroepen voordat de procedure 'update sleutels' wordt aangeroepen.

Wanneer als eerste de optie -n aan de procedure 'verwerk documenten tot sleutels en fiches' is meegegeven, staan de documenten die moeten worden ingelezen altijd in een tabel "daTYflds".

Is als eerste de optie -p aan de procedure 'verwerk documenten tot sleutels en fiches' meegegeven, dan hangt het van de rij uit de mutatie-tabel af, hoe het document ingelezen moet worden en waar het document staat.

Het update-systeem moet er voor zorgen dat er geen twee rijen in de tabel "mutats" voorkomen met hetzelfde id en type en met state = 'u' (van undone). Wanneer er iets gewijzigd moet worden aan een document dat al in de mutatie-tabel voorkomt (met state = 'u'), dan moet het zodanig veranderd worden dat er in de mutatie-tabel toch maar één rij nodig blijft.

Er zijn twee andere mutatie-states mogelijk namelijk 'd' (done) en 'p' (productie). Het catalogus-systeem werkt de mutaties af met mutatie-state = 'u'. Zodra de mutatie in bewerking gaat, wordt de state op 'p' gezet. Wanneer er iets fout zou gaan tijdens het uitvoeren van deze mutatie, blijft deze mutatie-state op 'p' staan. Als de mutatie goed verlopen is, wordt de mutatie-state op 'd' gezet.

Het maximale aantal mutaties op een document met hetzelfde type en id is twee. De eerste mutatie is dan door het catalogus-systeem verwerkt en de tweede niet (de eerste heeft state = 'd' en de tweede 'u'), óf de mutaties zijn allebei door het catalogus-systeem verwerkt (beide hebben mutatie-code 'd').

Bij de normale gang van zaken staat er maximaal één rij per document in de tabel "mutats". Hier volgt een schema hoe de rijen er in dit geval uitzien:

	type	id	state	code	which	tabel waarin het document te vinden is:
1)	A	A	u	i (insert)	0	upTYflds
2)	A	B	u	d (delete)	0	daTYflds
3)	A	C	u	u (update)	0	daTYflds/upTYflds (zie §2.2)

De kolom which geeft aan in wat voor een soort situatie de rij zich bevindt. Is er in totaal één rij aanwezig van een document waarop een mutatie moet worden verricht, dan is which = 0. De kolom which is ook 0 voor de eerste mutatie-rij die van een bepaald document is opgenomen. Verder is de waarde van which afhankelijk van de mutatie-code van de twee rijen die van het document zijn opgenomen.

De volgende situaties kunnen zich *niet* voordoen in de tabel "mutats":

	type	id	state	code
1)	A	A	d	i
	A	A	u/d	i
2)	A	B	d	u
	A	B	u/d	i
3)	A	C	d	d
	A	C	u/d	u
4)	A	D	d	d
	A	D	u/d	d
5)	A	E	d	d
	A	E	u/d	i

Van de eerste vier mogelijkheden zal het duidelijk zijn waarom ze niet voor kunnen komen. De vijfde mogelijkheid kan niet voorkomen omdat het document nog niet is vrijgegeven door het update-module, zodat het id nog niet gebruikt kan worden.

De volgende situaties kunnen *wel* voorkomen:

	type	id	state	code	which	het document is te vinden in:
1)	A	A	d	i	0	upTYflds
	A	A	u/d	d	1	upTYflds
2)	A	B	d	u	0	daTYflds / upTYflds
	A	B	u/d	d	2	daTYflds / upTYflds
3)	A	C	d	u	0	daTYflds / upTYflds
	A	C	u/d	u	3	daTYflds / upTYflds / up2TYflds
4)	A	D	d	i	0	upTYflds
	A	D	u/d	u	4	upTYflds / up2TYflds

In §2.2 staat beschreven hoe de mutaties die in de tabel "upTYflds" staan gelezen moeten worden. De tabel "up2TYflds" is in feite een zelfde tabel als de tabel "upTYflds" en kan op exact dezelfde manier gebruikt worden.

In de direct hierboven genoemde gevallen drie en vier zou het mogelijk zijn dat na verwerking door het catalogus-systeem de documenten moeten worden gewijzigd of verwijderd voordat het update-module ze verder verwerkt heeft. In dit geval worden de twee verwerkte rijen door het update-module samen genomen. In het derde geval betekent dit dat er, in plaats van de twee rijen uit de mutatie-tabel, één rij komt met mutatie-code 'u', met status = 'd' en met which = 0. In dit geval moet het update-module het document inlezen zoals dat in "daTYflds" staat en het document inlezen zoals dit in "daTYflds" staat gewijzigd met de wijzigingen uit "upTYflds" én met de wijzigingen uit "up2TYflds". De verschillen tussen de twee ingelezen documenten worden in "upTYflds" gezet en de rijen uit "up2TYflds" die betrekking op het document hadden worden verwijderd. De rij uit de mutatie tabel met which = 3 wordt verwijderd en daarvoor in de plaats komt de nieuwe rij met state = 'u'.

Het vierde geval wordt op een dergelijke manier verwerkt. De rijen die in de tabel "upTYflds" staan worden vervangen door het nieuwe document. De rijen uit "up2TYflds" die op dit document betrekking hadden worden verwijderd. De rij uit de tabel "mutats" met which = 4 wordt vervangen door de nieuwe mutatie.

In bijlage C is de header-file opgenomen van de structuren die bij het inlezen van documenten gebruikt worden. Bovendien staat in de bijlage een schets van de uiteindelijke structuur waarin de documenten geplaatst zijn.

Een ingelezen document bestaat uit een struct doc die de id van het document bevat en een array van structs fldheader. De grootte van de array wordt bepaald door het aantal verschillende velden van de documenten (bijvoorbeeld een veld auteur en een veld titel en een veld uitgeverij enzovoorts). Iedere struct fldheader bevat de som hd_size van de lengte van alle waarden van het betreffende array-veld. Ook bevat deze struct het aantal voorkomens (bijvoorbeeld twee voor het auteurs-veld als het document twee auteurs heeft) van het veld. Verder bevat de struct fldheader een pointer naar een keten van structs sqfld. Deze structs sqfld bevatten de totale lengte sq_size van van alle waarden bij de betreffende sequence (bijvoorbeeld de totale lengte van een titel), het nummer sq_sqno van de betreffende sequence (wanneer een document bijvoorbeeld twee titels heeft, staat de lengte van de eerste titel in sq_size met sq_sqno = 1 en de lengte van de tweede titel in sq_size van een volgende struct sqfld met sq_sqno = 2). Verder bevat de struct sqfld twee pointers. De eerste pointer fr_first wijst naar een struct frfld. De tweede pointer wijst naar de volgende struct sqfld. De struct frfld bevat de lengte van de string waar het veld fr_value naar wijst. De string waar het veld fr_value naar wijst is maximaal 72 karakters lang. Is de value langer (bijvoorbeeld bij een lange titel) dan staat de rest van de value in de volgende struct frfld, die bereikt kan worden via de pointer fr_next.

2.2.4. 'creeer sleutels en fiches'

De procedure 'creeer sleutels en fiches' maakt gebruik van een code-evaluator (eval). Deze code-evaluator bepaalt of een document in een bepaalde catalogus is of moet worden opgenomen, kan de overige sleutel-gegevens (zie paragraaf 4.1.) en fiches maken. De procedure 'creeer sleutels en fiches' biedt het te verwerken document aan 'eval' aan om te kijken of het document aan bepaalde specificaties voldoet (of het document wel of niet thuishoort in de catalogus). Als de specificatie voldoet, dan wordt de code-evaluator weer aangeroepen, maar nu om de sleutels te maken. De functie 'eval' levert nu een lijst met structuren af, waarin een of meer overige sleutel-gegevens van de sleutels staan en het nummer van het fiche waar ze bijhoren. Als het document moet worden verwijderd hoeft de functie eval niet voor een derde keer aangeroepen te worden. Moet het document echter worden toegevoegd, dan moet de functie voor een derde keer worden aangeroepen om de fiches te maken. De functie produceert nu een of meer fiches, zet ze in het het bestand "calDTYcumfch" en zet het begin-, het eind-adres en de bijbehorende nummers van de fiches in een structuur die als parameter aan de functie was meegegeven.

De procedure 'creeer sleutels en fiches' maakt en zet de sleutels, in het geval het document aan de catalogus-specificaties voldaan heeft, in het bestand "calDTYupkeys". In het geval van toevoegen worden de volgende dingen in het bestand gezet:

- Het document-id van het ingelezen document,
- Het fchno (fiche nummer uit de structuur van gemaakte sleutels),
- Keyno (te beginnen bij één en voor iedere sleutel die voor hetzelfde fchno wordt gemaakt een hoger),
- Seconden (het tijdstip waarop de procedure verwerk documenten tot sleutels en fiches is aangeroepen),
- De mutatie-code ('i'),
- Het begin- en het eind-adres (deze zijn afkomstig uit de structuur die eval heeft afgeleverd nadat hij de fiches heeft gemaakt. Om de juiste adressen te vinden moeten het fchno dat 'eval' heeft afgeleverd na het maken van de sleutels en het het fchno dat eval heeft afgeleverd na het maken van de fiches overeenkomen),
- De overige sleutel-gegevens (deze zijn afkomstig uit de structuur die 'eval' heeft afgeleverd nadat hij de sleutels heeft gemaakt),

- Een newline.
- Tussen ieder veld in wordt er een @ naar het bestand geschreven.

De sleutels van de fiches die uit de catalogus verwijderd moeten worden, worden op dezelfde manier gemaakt, alleen wordt voor de mutatie-code een 'd' en worden voor het begin- en het eind-adres een spatie geschreven.

2.2.4.1. de code evaluator (eval)

De functie 'eval' evalueert de code uit de ingelezen catalogus-specificaties. Een van deze catalogus-specificaties wordt aangeboden te zamen met de start-waarde (de index van het code-array die als eerste geëvalueerd moet worden). Het code-array bevat twee waarden, namelijk een code en een value. Aan de hand van de code of de value wordt er een actie ondernomen. De functie 'eval' werkt met een stack om de verschillende variabelen die bij deze acties nodig zijn te bewaren. De stack wordt static, zodat hij niet bij iedere aanroep gealloceerd hoeft te worden. De stack bestaat uit drie velden, namelijk een veld dat aangeeft welk type het element van de stack heeft, een getal dat aangeeft welke lengte de string heeft in het geval het type van de stack een pointer naar een string is (bij implementatie kan blijken dat dit getal te zamen met de pointer beter in een struct gezet zou kunnen worden) en een integer of een pointer naar een string of een pointer naar een struct. De volgende types kunnen voorkomen op de stack:

type:	inhoud van het stack element
numb:	integer
strg:	pointer naar een string
numba:	index van interne integers (zie 1.interne integers).
strga:	pointer naar een struct btstrg van een ketting van interne strings (zie 2. interne strings). Btstrg is een struct met een pointer naar een string (str), een integer (lalloc) die aangeeft hoeveel ruimte er totaal gereserveerd is voor de string, een integer (lused) die aangeeft hoe lang de string is waar naar verwezen wordt en een veld nxt dat wijst naar de volgende btstr.
btstrg:	pointer naar een struct btstrg van een ketting van externe strings (zie 3. externe strings).
btint:	pointer naar een struct btint van een ketting van externe integers (zie 4. externe integers). Een struct btint bevat een integer en een pointer naar de volgende struct btint.
func:	integer. De code evaluator kan later eventueel met deze waarde aangeroepen worden. De waarde van het stack element is dan de start waarde die als parameter wordt meegegeven.
sfunc:	integer. De waarde van deze integer stelt het nummer van een BibTeX built_in of een soortgelijke functie voor. Deze functie kan later eventueel worden aangeroepen.
field:	sqno van een bepaald veld van een document.
fmember:	pointer naar een sqfld van een document.

Op de stack wordt ruimte gereserveerd voor variabelen. De stack is in vijf delen gesplitst, die hier opgesomd worden:
1. Interne integers
Ten eerste is er een aantal plaatsen gereserveerd waarvan de hoeveelheid aangegeven wordt door de variabele inints (interne integers) uit de betreffende catalogus-specificatie. De variabelen inints hebben op de stack het type numb. Er is in de functie 'eval' een variabelde iip (interne integer base pointer), die aangeeft waar de eerste inint staat. De variabele iip dient dus als een soort base-pointer. De inints zijn bedoeld om binnen de functie 'eval' integers als globale variabelen te kunnen gebruiken.

2. Interne strings
Ten tweede wordt er een aantal plaatsen gereserveerd waarvan de hoeveeldheid aangegeven wordt door de

variabele instrs (interne strings) uit de betreffende catalogus-specificatie. Deze variabelen hebben op de stack het type btstrg. In dit geval zal het veld nxt van deze struct niet gebruikt worden. Er is in de functie 'eval' een variabele isp (interne string base pointer) die aangeeft waar de eerste instr op de stack staat en dient dus als een soort base-pointer. De instrs zijn bedoeld om binnen de functie 'eval' strings als globale variabelen te kunnen gebruiken.

3. Externe strings
Als derde wordt er op de stack een aantal plaatsen (het aantal wordt bepaald door de variable exstrs (externe strings) uit de betreffende catalogus-specificatie) gereserveerd voor kettingen van structs btstrg. Het veld nxt wijst nu telkens naar de volgende btstrg. Deze externe strings zijn bedoeld om informatie aan de functie 'creeer sleutels en fiches' terug te geven, namelijk, hoort wel/niet in catalogus, of de adressen en de nummers van de fiches of de overige sleutel-gegevens van de fiches met de nummers van de fiches waar ze bijhoren.

4. Externe integers
Als vierde wordt er op de stack een aantal plaatsen (het aantal wordt bepaald door de variable exints (externe integers) uit de betreffende catalogus-specificatie) gereserveerd. Alle struct btint die niet (meer) gebruikt worden, worden net zoals de structs btstr in een ketting van vrije structs gezet. Het nut van deze variabelen is nog onduidelijk, zij danken hun ontstaan aan de analogie inint/instr vs. exint/exstr.

5. Overige deel van de stack
Het vijfde deel van de stack wordt gebruikt om variabelen op te zetten waarmee de code evaluator bezig is. Het code array van de catalogus-specificaties bevat een code en een value. Het nummer van de code uit het array komt overeen met een functie. De volgende functies kunnen op deze manier worden aangeroepen:

number: Zet value op de stack, met type numb.

string: Zet pointer naar een string op de stack, met type strg.

exint: In het code array staat een index. Zet een pointer naar de eerste struct btstr op de stack die op de plaats index + eip (externe integer base pointer) op de stack staat. Het type van dit stack element is btint.

exstr: In het code array staat een index. Zet de pointer op de stack die op de plaats index + esp (externe string base pointer) op de stack staat. Het type van dit stack element is btstrg.

inint: In het code array staat een index. De functie inint zet nu de integer op de stack die op de plaats index + iip (interne integer base pointer) op de stack staat. Het type dat op de stack komt is numb.

instr: In het code array staat een index. De functie instr zet nu de pointer op de stack die op de plaats index + isp (interne string base pointer) op de stack staat. Het type van dit stack element is btstg.

field: Zet value op de stack, met type field.

fcall: Roep evalfnc aan met value als parameter.

sfcall: Roep evalsysf aan met value als parameter. De parameter is een nummer van een BibTeX built_in of soortgelijke functie. De functie evalsysf roept de functie aan die door value wordt aangeduid.

return: Stop evalfnc.

qinint: Zet value op de stack, met type numba.

qinstr: Zet value op de stack, met type strga.

qfcall: Zet value op de stack, stack is van het type qfcall. De value die nu op de stack staat is de nummer van een functie die later eventueel kan worden aangeroepen.

qsfcall: Zet value op de stack, stack is van het type qsfcall. De value die nu op de stack staat is het nummer van een BibTeX built_in of soortgelijke functie die later kan worden aangeroepen.

De meeste van de functies die door evalsysf kunnen worden uitgevoerd, zijn BibTeX built_in functies. Deze (en andere) functies zijn:

quote: Zet een pointer naar een string op de stack die een double_quote bevat. Het type van het stack element is strng.

assign: Kent de waarde die bovenop de stack staat toe aan de plaats die de op een na bovenste plaats van de stack aangeeft.

equals: Popt de twee bovenste waardes van de stack. Test of de waardes gelijk zijn. Zo ja, wordt er een 0 op de stack gezet en anders een 1 met type numb.

less: Popt de twee bovenste waardes van de stack. Als de tweede kleiner is dan de eerste, wordt er een 0 op de stack gezet, anders een 1 met type numb.

greater: Popt de twee bovenste waardes van de stack. Als de tweede groter is dan de eerste, wordt er een 0 op de stack gezet, anders een 1 met type numb.

append: Popt de twee bovenste waardes van de stack. De tweede string wordt achter de eerste string bovenop de stack gezet.

plus: Popt de twee bovenste waardes van de stack, telt ze op en pushed het resultaat.

minus: Popt de twee bovenste waardes van de stack, trekt de eerste van de tweede af en pushed het resultaat.

index: Popt de twee bovenste waardes van de stack. De tweede waarde is het nummer van veld (bijvoorbeeld het nummer van het veld "ti "). De eerste is de occurence (bijvoorbeeld sqno 1) die we willen hebben. Als resultaat wordt er een pointer naar een sqfld op de stack gezet. Een soorgelijke operatie moet er ook voor exstr's komen.

addperiod: Popt de bovenste waarde van de stack. Indien de string waarnaar verwezen wordt geen punt, vraagteken of uitroepteken bevat, wordt er een punt achter gezet. De waarde wordt daarna weer op de stack gezet.

calltype: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

changecase: Popt de twee bovenste waardes van de stack. De tweede string wordt verandert aan de hand van de eerste. Als van de eerste string de eerste letter een 'u' of een 'U' is, wordt de eerste letter van de tweede string in een upper-case letter verandert. Is de eerste letter van de eerste string een 'l' of een 'L' dan wordt de eerste letter van de tweede string verandert in een lower-case letter. Als de tweede letter van de eerste string een 'u' is of een 'U' dan wordt de tweede string, behalve de eerste letter, verandert in upper-case letters. Is de tweede letter van de eerste string echter een 'l' of een 'L', dan worden de overige letters van de tweede string verandert in lower-case letters. Het resultaat wordt op de stack gezet.

chrtoint: Popt de bovenste waarde van de stack, verandert deze in een integer en pushed het resultaat.

cite: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

duplicate: Popt de bovenste waarde van de stack en pushed deze waarde tweemaal.

formatname: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

if: Popt de drie bovenste waardes van de stack. Als de eerste waarde groter is dan 0, wordt de functie die op de tweede plaats stond aangeroepen. In het andere geval wordt de functie aangeroepen die op de eerste plaats stond.

inttochr: Popt de bovenste waarde (integer) van de stack, vervangt deze door het character dat bij de integer hoort, zet deze in een string en pushed het resultaat.

inttostr: Popt de bovenste waarde (integer) van de stack, vervangt iedere digit door een character, zet dit een een string en pushed het resultaat.

missing: Popt de bovenste waarde (sqfld) van de stack, als de pointer null is, wordt er 1 op de stack gezet en anders 0 (type numb).

newline: Schrijft een newline naar de file "calDTYcumfch".

numfields: Popt de bovenste waarde van de stack. Deze waarde bevat het type van een veld (bijvoorbeeld "ti "). Er wordt een integer op de stack gepushed die het aantal velden bevat dat het document van het type veld bezit.

numnames: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

pop: Popt de bovenste waarde van de stack en doet hier niets mee.

purify: Popt de bovenste waarde van de stack. De string waar direct of indirect naar wordt verwezen wordt geanalyseerd. Alle niet letters en cijfers worden uit de string weggelaten. Tabs, newlines en meerdere spaties worden door een spatie vervangen.

skip: Doe niets.

sortkey: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

stack: Popt de hele stack en drukt hem af. Deze functie wordt voor het debuggen gemaakt.

substring: Popt de bovenste drie waardes van de stack. De eerste twee zijn integers, terwijl de derde direct of indirect een pointer naar een string is. Het tweede getal is de index voor de eerste letter die van de string moet worden genomen, terwijl het eerste getal het totaal aantal letters is. Als het tweede getal negatief is, moeten de laatste n letters van de string worden genomen (n is hier het eerste getal). Als resultaat wordt een pointer (direct of indirect) naar de substring op de stack gezet.

swap: Popt de bovenste twee waardes van de stack, verwisselt ze en zet ze weer op de stack.

switch: Dit is een generalisatie van if.

top: Popt de bovenste waarde van de stack en drukt hem af.

type: Tijdens het implementeren zal van deze functie nog een nadere omschrijving moeten worden gemaakt.

while: Tijdens het implementeren zal van deze functie nog een specifieke. omschrijving moeten worden gemaakt.

width: Popt de bovenste waarde van de stack, telt het aantal characters dat de string waar (in)direct naar verwezen wordt en pushed het resultaat als een integer.

write: Popt de bovenste waarde van de stack en schrijft deze naar de file "calDTYcumfchs". Ook pushed hij het begin- en eind-adres naar de stack (of in plaats van naar de stack rechtstreeks naar een exstrg).

Tijdens de implementatie kan ervoor gekozen worden om de stack bij iedere aanroep van de code evaluator in te delen aan de hand van de getallen instrs, inints, exints en exstrs. Een andere mogelijkheid is dat de functie 'creeer sleutels en fiches' een keer de aantallen van alle specificaties bekijkt en aan de hand hiervan de stack in een keer goed opbouwt.

Omdat de functie 'eval' in de meeste gevallen heel vaak achter elkaar wordt aangeroepen, is het verstandig om niet telkens alle ruimte die gealloceerd is vrij te geven, maar om deze ruimte voor een volgende keer te bewaren. Alle struct btstrg die niet (meer) gebruikt worden, worden daarom in een ketting van vrije structs gezet. Onder meer hiervoor is het veld nxt nodig.

2.3. 'Orden sleutels'

De sleutels uit het bestand "calDTYupkeys" dat door de procedure 'verwerk documenten tot sleutels en fiches' is gemaakt, worden met behulp van het UNIX-commando sort gesorteerd. De sleutels uit dit bestand zien er als volgt uit:

document-id @ fchno @ keyno @ seconden @ mutatie-code @ begin-adres @ eind-adres @ overige sleutelgegevens

De sleutels worden gesorteerd in de volgorde van: overige sleutelgegevens, document-id, fchno, keyno, seconden en mutatie-code. Het sorteren van de sleutels maakt het mogelijk dat het proces 'update sleutels' een merge kan doen met de bestanden "calDTYupkeys" en "calDTYkeys". Een uitgebreide uitleg van de sleutels staat in paragraaf 4.1.

2.4. 'Update sleutels'

Het bestand "calDTYkeys" wordt bijgewerkt, aan de hand van de sleutels uit het bestand "calDTYupkeys". Er wordt als het ware een merge van deze twee bestanden gedaan. Als mutatie-code van de sleutel, uit het bestand "calDTYupkeys", een 'i' is (van insert), dan wordt deze sleutel (zonder de velden seconden en mutatie-code) geschreven naar een tijdelijk sleutel-bestand. Als mutatie-code van de sleutel een 'd' is (van delete), dan wordt noch deze noch de bijbehorende sleutel uit het bestand "calDTYkeys", naar het tijdelijke bestand geschreven. Alle andere sleutels uit het bestand "calDTYkeys" worden wel naar het tijdelijke bestand geschreven.

Als er in het bestand "calDTYupkeys" twee sleutels voorkomen met dezelfde overige sleutelgegevens, hetzelfde document-id en dezelfde fchno en keyno, waarbij de eerste sleutel een mutatie-code 'i' heeft en de tweede mutatie-code 'd', worden deze twee sleutels beide genegeerd. Als er in dit bestand een aantal sleutels voorkomen met dezelfde overige sleutelgegevens, hetzelfde document-id en dezelfde fchno en keyno (zie paragraaf 4.1.), waarbij de eerste sleutel een mutatie-code 'd' heeft, de tweede een mutatie-code 'i' heeft en er eventueel nog andere paren van dezelfde sleutels volgen met afwisselend mutatie-code 'd' en 'i', worden alleen de eerste en de laatste uit deze serie gebruikt, de tussenin liggende sleutels worden genegeerd.

Als het einde van de bestanden "calDTYkeys" en "calDTYupkeys" bereikt is, wordt het tijdelijke bestand met behulp van een move naar "calDTYkeys" geschreven.

2.5. 'Clean fiches'

De bedoeling van deze procedure is dat er in het fiche-bestand alleen fiches achterblijven, waar tenminste een sleutel uit het bijbehorende sleutel-bestand naar verwijst. De procedure zal gebruik maken van drie tijdelijke files die respectievelijk "keys.tmp" "keys2.tmp" en "fchs.tmp" heten. Deze procedure zal aan de hand van de bestanden "calDTYcumfchs" en "calDTYkeys" het bestand "calDTYcumfchs" herordenen. De sleutels uit het bestand "calDTYkeys" met de bijbehorende fiches uit het bestand "calDTYcumfchs" worden stuk voor stuk bekeken. Als keyno (zie paragraaf 4.1.) van de sleutel gelijk is aan één, wordt het fiche uit het bestand "calDTYcumfchs" naar het bestand "fchs.tmp" geschreven. Het begin- en het eind-adres (zie paragraaf 4.1.) van het fiche uit dit *nieuwe* bestand worden dan in de sleutel gezet op de plaats van begin- en eind-adres. De sleutel wordt, al dan niet veranderd, naar het bestand "keys.tmp" geschreven. Als het einde van het bestand "calDTYkeys" bereikt is, wordt het bestand "keys.tmp" gesorteerd op document-id, fchno (zie paragraaf 4.1.) en keyno. Vervolgens worden de sleutels uit het bestand "keys.tmp" gelezen en worden de begin- en eind- adressen, van de sleutels met keyno ongelijk aan één,

gelijk aan de begin- en eind-adressen van de sleutel die hetzelfde document-id en hetzelfde fchno heeft en keyno gelijk aan één. De ingelezen sleutels worden naar het bestand "keys2.tmp" geschreven. Vervolgens wordt het bestand "keys2.tmp" met behulp van het UNIX-commando sort gesorteerd op overige sleutelgegevens. Daarna wordt het bestanden "fchs.tmp" en "keys2.tmp" met behulp van een move respectievelijk naar de bestanden "caIDTYcumfchs" en "caIDTYkeys" geschreven. Als laatste wordt het bestand "keys.tmp" verwijderd.

2.6. 'Pagineer'

In bijlage C is de header-file en een schets met structs opgenomen waarin de catalogus-specificatie moet worden ingelezen. Zodra deze procedure wordt aangeroepen worden de catalogus-specificaties (van de sectie pagineer) ingelezen, van de catalogus waarvan uitvoer geproduceerd moet worden. Voor iedere catalogus waarvan uitvoer gegenereerd moet worden, wordt een struct gealloceerd. In deze struct wordt het id van de catalogus geplaatst en is er ruimte voor twee pointers. Een pointer die naar een andere struct moet wijzen, waarin een pointer naar een code-array staat, een pointer naar een string-array en waarin ruimte is voor acht getallen uit de tabel "caspcatt". De code- en de string-array zien er uit als in paragraaf 2.2.2. is beschreven.

Aan de hand van de ingelezen catalogus-specificaties, waarin de lay-out van de catalogus gespecificeerd staat en het bestand "caIDTYcumfchs" waarin de cumulatieve fiches staan, wordt ofwel het bestand "caIDTYcumpags" ofwel het bestand "caIDTYtin" gemaakt. Welk van de twee bestanden gemaakt wordt, hangt af van de catalogus-specificatie. Afhankelijk van de catalogus-specificaties moet een sleutel (of een deel ervan) wel of niet in een van de bestanden worden opgenomen. Er moet rekening worden gehouden met het inspringen van een fiche (of een deel ervan). Wanneer bijvoorbeeld twee fiches dezelfde eerste auteur hebben, wordt deze bij het tweede fiche niet geschreven maar aangehaald. Wanneer het bestand "caIDTYcumpags" gemaakt moet worden moet er bijvoorbeeld rekening mee worden gehouden dat in de uitvoer soms drie karakters gereduceerd worden tot een. Dit gebeurt in het geval van accenten. Deze tekens kunnen in de tabellen en bestanden er uitzien als een <ESCAPE TEKEN> <LETTER>. Als het bestand "caIDTYtin" gemaakt moet worden, moeten er tekens voor de tekst-verwerker worden opgenomen.

Voor ieder bestand dat door de procedure 'pagineer' verwerkt moet worden, wordt er een rij in de tabel "nodef" aangepast indien de id en het type van het bestand al in de tabel aanwezig is. Anders wordt er een nieuwe rij in de tabel opgenomen.

In de rij worden behalve de id en het type van het bestand drie getallen opgenomen. Het eerste getal geeft aan hoeveel verschillende documenten zijn gebruikt bij het maken van de catalogus. Dit getal is gelijk aan het aantal sleutels uit het sleutelbestand met fchno is één èn keyno is één. Het tweede getal geeft aan hoeveel entries de catalogus heeft (hoeveel verschillende fiches de catalogus bevat). Dit getal is te berekenen door alle sleutels uit het sleutel-bestand te tellen waarvan het fchno gelijk is aan één. Het laatste getal geeft aan hoeveel fiches de catalogus bevat. Dit getal is gelijk aan het aantal sleutels dat het sleutel-bestand bevat.

Deze getallen moeten door de procedures 'genereer uitvoer' en 'tekst verwerken' in de tabel "account" worden gezet. Deze getallen moeten in de 'pagineer' procedure geteld worden, omdat de procedures 'genereer uitvoer' en 'tekst verwerken' deze getallen niet (of in ieder geval heel moeilijk) kunnen produceren.

2.7. 'Genereer uitvoer'

Afhankelijk van het catalogus-id moet de procedure 'genereer uitvoer' of de procedure 'tekst verwerken' worden aangroepen. Bij de procedure 'genereer uitvoer' kan het bestand "caIDTYcumpags" hetzij op micro fiche hetzij op papier worden afgedrukt. Bij het afdrukken op papier moet erop worden gelet dat

verschillende printers gebruikt kunnen worden. Wanneer het bestand op micro-fiche moet komen, moet er een index worden gemaakt. Bij iedere aanroep van de procedure 'genereer uitvoer' moet de tabel "account" worden bijgewerkt. De aantallen die in deze tabel moeten worden gezet staan in de tabel "nodef".

2.8. 'Tekst verwerken'

Het doel van dit proces is een nette uitvoer van een catalogus te maken op papier. De procedure maakt deze uitvoer met behulp van de bestanden "calDTYcumpags". De uitvoer zal via een tekstverwerker verlopen. Ook deze procedure moet de tabel "account" aanpassen. De aantallen die in deze tabel moeten worden gezet staan in de tabel "nodef".

2.9. 'Recovery'

Deze procedure zal bij het opstarten van het bibliotheek-systeem worden aangeroepen. De bedoeling is, dat wanneer er om de een of andere rede iets mis is gegaan tijdens een bewerking (bijvoorbeeld omdat de computer down ging, of omdat er geen geheugen-ruimte meer was) het systeem zodanig wordt opgestart dat de fout automatisch hersteld wordt.

Voor de procedures 'orden sleutels', 'update sleutels' en 'clean fiches' betekent dit dat ze opnieuw gestart moeten worden. Voor de procedure 'verwerk documenten tot sleutels en fiches' betekent dit in het geval dat er mutaties verwerkt werden, dat de mutatie waarbij in de tabel 'mutats' bij state = 'p' (productie) staat, dat alle veranderingen die deze mutatie veroorzaakt heeft, ongedaan moeten worden gemaakt en het veld state op 'u' (undone) gezet moet worden. Daarna kan de procedure 'verwerk documenten tot sleutels en fiches' opnieuw gestart worden. In het andere geval, namelijk dat de procedure 'verwerk documenten tot sleutels en fiches' bezig was met het toevoegen van documenten via een tabel "SELIDS" of direct uit een tabel "daTYflds" is nog niet bekend wat er moet gebeuren. De vragen in dit geval zijn welke catalogus-specificaties ingelezen waren, welke tabel "daTYflds" of "SELIDS" precies in gebruik was en welk document-id en type de procedure 'verwerk documenten tot sleutels en fiches' aan het verwerken was op het moment dat er iets verkeerd ging. Het zou in dit geval jammer zijn als alles opnieuw gestart zou moeten worden.

Wat ook nog bedacht moet worden is wat er moet gebeuren in het geval het catalogus-systeem bezig was met één van de procedures 'update catalog descript(ion)', 'pagineer', 'genereer uitvoer' of 'tekst verwerken'.

3. *BESCHRIJVING VAN DE INGRES TABELLEN*

3.1. "caspcprd"

Deze tabel bevatten alle catalogus-identificaties (cat.id) en welke cat.id's bij een bepaalde productie-set (pid) behoren. Dit is nodig omdat niet altijd alle catalogi tegelijk aangemaakt of afgedrukt moeten worden of bepaalde procedures moeten volgen als 'update sleutels', 'clean fiches' en 'pagineer'. Op deze manier kan de gebruiker van één of van een bepaalde groep catalogi eenmalig uitvoer maken. Ook kan het zo zijn dat een van de procedures voor het verwerken van een catalogus zoveel tijd vraagt dat het onverstandig is om deze procedure voor alle catalogi te laten draaien. Een van deze productie-sets bevat alle catalogus-identificaties van catalogi die in het systeem bewaart en up to date moeten blijven. Deze productie-set wordt aan de procedure 'verwerk documenten tot sleutels en fiches' meegegeven achter de -p optie.

3.2. "caspccod"

Deze tabel bevat voor iedere catalogus een identificatie id en vier secties sk (selectie, sleutel, fiche en pagina). Voor iedere sectie sk en identificatie id bevat de tabel een teller sq die aangeeft de hoeveelste rij het in de tabel is met dit id en sk. Deze teller sq is nodig omdat het noodzakelijk is om de rijen in de goede volgorde in te lezen. Verder horen bij iedere rij twee integers code en value. Code geeft aan wat voor een soort getal value is. Value is een getal met twee mogelijkheden. Ofwel het is een getal dat door de functie 'creeer sleutels en fiches' op de stack gezet zal worden, ofwel het is een getal dat als het ware een functie-aanroep is.

3.3. "caspcstr"

Deze tabel bevat voor iedere catalogus een identificatie id en vier secties sk (selectie, sleutel, fiche en pagina). Voor iedere sectie sk en identificatie id bevat de tabel een teller sq die aangeeft de hoeveelste rij het in de tabel is met dit id en sk. Deze teller sq is nodig omdat het noodzakelijk is om de rijen in de goede volgorde in te lezen. Verder bevat iedere rij een value waarin de waarde van de string staat. Als de waarde niet in z'n geheel in een rij past, dan komt de rest van de waarde in een volgende rij met hetzelfde id, sk en sq. Het veld fr geeft de volgorde van deze rijen aan.

3.4. "caspcatt"

Deze tabel bevat voor iedere catalogus een identificatie id en vier secties sk (selectie, sleutel, fiche en pagina). Voor iedere sectie sk en identificatie id bevat de tabel negen waarden, te weten: start, strings, codes, exints, exstrs, inints, instrs, fields en fcalls.

Codes geeft het aantal rijen aan dat de tabel "caspccod" heeft met het bijbehorende id en sk. Met deze informatie kunnen de code-array's met juiste lengte gealloceerd worden. Strings geeft het aantal rijen aan dat de tabel "caspcstr" heeft met het bijbehorende id en sk. Met deze informatie kunnen de string-array's met juiste lengte gealloceerd worden.

Start is de waarde die de index van de code-array moet hebben als met de functie 'creeer sleutels en fiches' begonnen wordt. Het is namelijk niet zo dat de acties die in het code-array staan helemaal sequentieel afgewerkt moeten worden (er zijn namelijk call in de built-in functie die naar een andere plek in het

code-array wijzen). De functie 'creeer sleutels en fiches' moet op de plek beginnen die start aangeeft. Daarna worden de codes uit het code-array sequentieel afgewerkt, totdat er een built-in call wordt gedaan naar een ander stuk uit de code-array. Dan worden de codes op de nieuwe plaats sequentieel afgewerkt, totdat er een nieuwe built-in call wordt gedaan, of er een return is gegeven. In het laatste geval wordt de code afgewerkt waar de functie gebleven was voordat laatste de built-in call werd gedaan.

Exints en exstrs geven respectievelijk het aantal integers en het aantal strings aan die de functie 'creeer sleutels en fiches' als parameter meekrijgt. Met behulp van deze aantallen, kunnen er twee array's met juiste lengte worden gealloceerd.

Inints en instrs geven respectievelijk het aantal integers en het aantal strings aan die binnen de functie 'creeer sleutels en fiches' gebruikt worden. Binnen deze functie zijn de variabelen globaal gedefinieerd. Deze variabelen hebben een nummer. Daarom is het handig om van zowel de integers als van de strings een array te maken. Het nummer van de variabele is dan meteen de index van de array. Met behulp van de getallen inints en instrs kunnen er twee array's met de juiste lengte worden gealloceerd.

Fields geeft het aantal velden aan waaruit een bibliografie bestaat. Dit getal is ook al als een constante in een header-file opgenomen (def.h) en dient alleen als controle-middel.

Fcalls geeft het aantal functie-declaraties aan, die in de file met BibTeX-functies staan.

3.5. "daTYflds" (beschrijving van de documenten)

Deze tabellen bevatten alle informatie van een bepaald document. Het veld id geeft aan welke rijen uit deze tabel een document vormen en het veld fld geeft aan wat voor een soort informatie het veld value bevat. Het veld sqno geeft de volgorde van het veld value aan indien er meer dan een dezelfde waarde van fld is. Wanneer de inhoud die in het veld value zou moeten staan te groot is, is de rest van dit veld op de volgende rij gezet en is het veld frno van deze rij een hoger dan dat van de vorige rij. In bijlage B is een voorbeeld van een document opgenomen.

3.6. "upTYflds" en "up2TYflds"

Deze tabellen bevatten de gegevens van documenten die verandert (of in het geval van upTYflds verandert of toegevoegd) moeten worden. Van de velden die uit het document verwijderd moeten worden, bevat value de waarde "@@@@@". Van de velden die toegevoegd of gewijzigd moeten worden bevat value de nieuwe waarde. Als er een waarde tussengevoegd moet worden, staan ook alle volgende velden in deze tabel. Een voorbeeld hiervan is dat het volgende aan het document in bijlage B moet worden veranderd:
Er staat:

```
        :
        :
|M00057J |hld | 11|   1|601:47(1940)-...
|M00057J |hld | 12|   1|761:71(1964)-...
|M00057J |hld | 13|   1|831:74(1967)-...
        :
        :
```

en het moet worden:

```
        :
        :
|M00057J |hld | 11|   1|601:47(1940)-...
|M00057J |hld | 12|   1|761:71(1964)-...
|M00057J |hld | 13|   1|Dit veld moet worden tussengevoegd.
|M00057J |hld | 14|   1|831:74(1967)-...
        :
        :
```

In de tabel upTYflds staat dan het volgende:

```
        :
        :
|M00057J |hld | 13|   1|Dit veld moet worden tussengevoegd.
|M00057J |hld | 14|   1|831:74(1967)-...
        :
        :
```

3.7. "SELIDS"

Deze tabellen bevatten id's van documenten en het type van de tabel waarin het document te vinden is. Deze tabellen zijn bedoeld om de gebruiker extra selectie criteria te bieden bij het aanroepen van de functie 'verwerking documenten tot sleutels en fiches' (doordat alleen de documenten die in deze tabel zijn opgenomen worden aangeboden). Het catalogus-systeem biedt geen mogelijkheid om een dergelijke tabel te maken, maar de gebruiker kan de tabel bijvoorbeeld met behulp van SQL-statements aanmaken.

3.8. "mutats"

Deze tabel bevat de id's van de te verwerken documenten, een veld code, dat aangeeft om wat voor een soort verwerking het gaat (invoegen, veranderen of verwijderen), een veld type, dat aangeeft in welk type tabel het document te vinden is (de letters die voor TY moeten worden ingevuld), een attribuut which dat aangeeft waar het oude of nieuwe document te vinden is en/of waar de mutaties te vinden zijn. Ook is er een veld state, dat aangeeft of de verwerking wel of niet heeft plaatsgevonden. De procedure 'verwerk documenten tot sleutels en fiches' zal het veld state aanpassen als de sleutels en fiches, van het bij de mutatie horende document, zijn gemaakt.

3.9. "history"

Telkens wanneer een document in een bepaalde catalogus verwerkt wordt, wordt het document-id en de bijbehorende mutatie-code en document-type met het tijdstip van verwerking en het catalogus-id aan de tabel "history" toegevoegd. Deze tabel is opgenomen om later na te kunnen zoeken wanneer en in welke catalogus welke documenten toegevoegd of verwijderd zijn, als aanknopingspunt als er iets verkeerd is gegaan.

3.10. "account" (data die wordt opgenomen t.b.v. statistische overzichten)

Iedere keer als de procedure "genereer uitvoer" uitvoer gegenereerd heeft, worden in de tabel "account" per catalogus de volgende gegevens opgenomen:
- De datum van het moment waarop de uitvoer gegenereerd is.
- De id van de catalogus waarvan uitvoer gegenereerd is.
- Het type van de catalogus waarvan uitvoer is gegenereerd.
- Het aantal documenten waarvan uitvoer is gegenereerd.
- Het aantal entries (het verschillende aantal fiches) waarvan uitvoer is gegenereerd.
- Het aantal fiches waarvan uitvoer is gegenereerd.
- Wat voor een type uitvoer het geweest is (papier/ micro-fiche / ...).

Deze tabel is opgenomen zodat de gebruiker kan zien wat er gebeurd is. Hij zou naar aanleiding van deze tabel bijvoorbeeld kunnen beslissen, wanneer in een auteurs-catalogus de uitvoer erg groot is en het aantal fiches veel groter is dan het aantal entries, om de fiches bijvoorbeeld niet meer bij de tweede auteur te vermelden.

3.11. "nodef" (aantal documenten, entries en fiches)

Deze tabel wordt door de procedure 'pagineer' gemaakt. Deze procedure maakt de tabel omdat de getallen hier het eenvoudigst kunnen worden geproduceerd. De tabel bevat de volgende gegevens:
- De id van de catalogus.
- Het type van de catalogus.
- Het aantal documenten waarvan uitvoer is gegenereerd.
- Het aantal entries (het verschillende aantal fiches) waarvan uitvoer is gegenereerd.
- Het aantal fiches waarvan uitvoer is gegenereerd.

4. *BESCHRIJVING VAN DE UNIX-BESTANDEN EN VAN DE UITVOER-PRODUCTEN*

4.1. "caIDTYupkeys" (sleutel van fiche met pointer naar het bestand "caIDTYcumfchs")

Deze bestanden bevatten telkens de sleutel van een fiche uit het bijbehorende bestand "caIDTYupfchs". De sleutels zullen als volgt zijn opgebouwd:

document-id @ fchno @ keyno @ seconden @ mutatie-code @ begin-adres @ eind-adres @ overige sleutelgegevens

Document-id is het id van het document waarbij deze sleutel hoort. Document-id is in de sleutel opgenomen om de verschillende sleutels van elkaar te kunnen onderscheiden, omdat het voor zou kunnen komen dat de overige sleutelgegevens van meer dan een fiche hetzelfde is. Zonder document-id zou bij een mutatie de verkeerde sleutel en fiche gemuteerd kunnen worden. Sleutels met hetzelfde document-id en hetzelfde fchno hebben hetzelfde begin- en eind-adres. De procedure 'clean fiches' heeft document-id nodig om het begin- en eind-adres te kunnen bepalen van sleutels waarvan keyno groter is dan één (om te kunnen bepalen welke fiches van hetzelfde of van een ander document afkomstig zijn, omdat de overige sleutelgegevens van twee verschillende documenten niet uniek hoeven te zijn).

Fchno geeft aan naar het hoeveelste fiche van het bijbehorende document de sleutel wijst. Er kan bijvoorbeeld een eerste fiche zijn met de eerste auteur en een tweede fiche, gesorteerd op de tweede auteur, die verwijst naar het eerste fiche. Het nummer van het fiche is nodig bij de procedures 'update sleutels', 'clean fiches' en 'pagineer'.

Keyno geeft het nummer aan van de sleutel met hetzelfde document-id en hetzelfde fchno. Het kan voorkomen dat meer dan een sleutel naar hetzelfde fiche wijst, bijvoorbeeld als het fiche ook op de plaats van de tweede auteur vermeld wordt, in geval van een auteurs-catalogus. Het nummer van de sleutel is nodig bij de procedures 'update sleutels', 'clean fiches' en 'pagineer'.

Seconde geeft aan wanneer de procedure 'verwerking documenten tot sleutels en fiches' deze sleutel heeft gemaakt. Het veld seconden is nodig om de catalogus in chronologische volgorde te kunnen muteren. Wanneer de procedure 'verwerking documenten tot sleutels en fiches' meer dan een keer wordt aangeroepen voordat de procedure 'update sleutels' wordt aangeroepen, kunnen er meer dan een mutatie van hetzelfde document zijn. De mutaties moeten in dat geval in chronologische volgorde worden aangeboden aan de procedure 'update sleutels'.

De mutatie-code geeft aan wat er met de sleutel moet gebeuren. Als de mutatie-code een 'i' is van insert, moet de sleutel (zonder de velden mutatie-code en seconden) worden toegevoegd aan het bestand "caIDTYkeys". Is de mutatie-code van de sleutel een 'd' van delete, dan moet de sleutel uit het bestand "caIDTYkeys" met dezelfde overige sleutelgegevens, hetzelfde document-id, hetzelfde fchno en hetzelfde keyno worden verwijderd.

Begin- en eind-adres geven aan tussen welke twee waarden het fiche precies in het bestand "caIDTYcumfchs" staat.

Ook bij de overige sleutelgegevens worden de velden gescheiden door middel van een @. Het is nodig dat de sleutelgegevens gescheiden worden, omdat bij het pagineren mogelijk een deel van de sleutel bij het fiche gezet moet worden (bijvoorbeeld auteurs-velden). Bij ieder veld zal een teken staan dat aangeeft wat voor een soort veld het is. Wat dit teken precies is zal in de catalogus-specificaties staan. Dit teken is nodig omdat het aantal velden van tevoren niet vast staat. Het ene document heeft bijvoorbeeld een auteur terwijl een ander document er vijf heeft. Wanneer bijvoorbeeld alle auteurs die in de sleutel staan, bij het fiche in het bestand "caIDTYcumpags" of "caIDTYtin" opgenomen moeten worden, maar geen andere velden uit de sleutel, moet duidelijk zijn welke van de velden wel en welke velden geen auteurs bevatten.

4.2. "calDTYcumfchs" (cumulatieve fiches)

Deze bestanden bevatten fiches waarbij de velden in vaste volgorde staan. Deze volgorde is bepaald door de catalogus-specificaties. Het bestand "calDTYcumfchs" zal meestal niet gesorteerd zijn (deze bestanden zullen gesorteerd zijn, op sleutels met keyno = 1, nadat de procedure 'clean fiches' is aangeroepen) en het zal geen sleutels bevatten, maar er is een bestand "calDTYkeys", die sleutels bevat telkens met een pointer naar het fiche in het bestand "calDTYcumfchs".

4.3. "calDTYkeys" (sleutels met pointers die wijzen naar fiches uit calDTYcumfchs)

De sleutels uit deze bestanden zullen er hetzelfde uit zien als de sleutels uit de bestanden "calDTYupkeys". Het enige verschil is dat de sleutels uit de bestanden "calDTYkeys" geen seconden en mutatie-code velden bevatten. De sleutels zullen dus als volgt zijn opgebouwd:

document-id @ fchno @ keyno @ begin-adres @ eind-adres @ overige sleutelgegevens

4.4. "calDTYcumpags" (cumulatieve pagina)

Deze bestanden zullen de fiches uit het bijbehorende bestand "calDTYcumfchs" bevatten, die gesorteerd zullen zijn als in de catalogus-specificatie aangegeven staat. De fiches zullen in pagina's zijn verdeeld en per pagina een kopje bevatten. Deze bestanden bevatten nog geen index. De index wordt gemaakt door de procedure 'genereer uitvoer'.

4.5. "calDTYtin"

Deze bestanden zullen de fiches uit het bijbehorende bestand "calDTYcumfchs" bevatten, die gesorteerd zullen zijn als in de catalogus-specificatie aangegeven staat. Tevens zijn in deze bestanden gegevens voor de tekst-verwerker opgenomen.

4.6. "hard copy"

De bestanden "calDTYcumpags" en "calDTYtin" uitgevoerd op papier. Dit zal gebeuren hetzij met behulp van de postscript-printer, hetzij met behulp van de phototypesetter, hetzij met behulp van een andere printer. De bestanden "calDTYtin" zullen via de tekst-verwerker naar de printer worden gestuurd.

4.7. "micro fiche" (gepagineerde micro-fiches)

Het bestand "calDTYcumpags" uitgevoerd op micro-fiche. Hierbij dient ook de index te worden afgedrukt.

5. *ATTRIBUUT-BESCHRIJVING VAN INGRES TABELLEN:*

"caspcprd" (een van de vier tabellen CTSPCS)

Storage structure: btree

column name	type	length	sequence key	comment	
pid	c	3	1		id van de productie-set
id	c	3	2		id van de catalogus

Secondary indices: none

"caspccod" (een van de vier tabellen CTSPCS)

Storage structure: btree

column name	type	length	sequence key	comment	
id	c	3	1		id van de catalogus
sk	c	3	2		sectie (selectie, fiche, sleutel of pagineer)
sq	integer	2	3		sequence number
cod	integer	2			code (soort getal dat value bevat)
value	integer	2			

Secondary indices: none

"caspcstr" (een van de vier tabellen CTSPCS)

Storage structure: compressed btree

column name	type	length	sequence key	comment	
id	c	3	1		id van de catalogus
sk	c	3	2		sectie (selectie, fiche, sleutel of pagineer)
sq	integer	2	3		sequence number
fr	integer	2	4		fragmentnummer van de value
value	text	72			

Secondary indices: none

"caspcatt" (een van de vier tabellen CTSPCS)

Storage structure: btree

column name	type	length	sequence key	comment
id	c	3	1	id van de catalogus
sk	c	3	2	sectie (selectie, fiche, sleutel of pagineer)
start	integer	2		start-adres van de interpretatie
strings	integer	2		aantal strings in "caspcstr"
codes	integer	2		aantal codes in "caspccod"
exints	integer	2		aantal integer-parameters van de functie creeer sleutels en fiches
exstrs	integer	2		aantal string-parameters van de functie creeer sleutels en fiches
inints	integer	2		aantal integers die als variabelen in de functie creeer sleutels en fiches gebruikt worden
instrs	integer	2		aantal strings die als variabelen in de functie creeer sleutels en fiches gebruikt worden
fields	integer	2		aantal verschillende velden van een bibliografie
fcalls	integer	2		het aantal functie-declaraties

Secondary indices: none

"daTYflds", "upTYflds" en "up2TYflds"

Storage structure: compressed btree

column name	type	length	sequence key
id	c	8	1
fld	c	3	2
sqno	integer	1	3
frno	integer	1	4
value	vchar/text	72	

Secondary indices: none

"SELIDS"

Storage structure: btree

column name	type	length	sequence key	comment
type	c	2	1	type van de tabel "daTYflds"
id	c	8	2	id van het document dat toegevoegd moet worden

Secondary indices: none

"mutats"

Storage structure: btree

column name	type	length	sequence key	comment
type	c	2	1	soort document
id	c	8	2	identificatie van het document
code	c	1		soort verwerking
state	c	1		status: wel of niet verwerkt

Secondary indices: none

"history"

Storage structure: heap

column name	type	length	comment
dat	date		datum van verwerking
id	c	8	identificatie van het document
type	c	2	soort document
code	c	1	soort mutatie
catid	c	3	catalogus-id
fchno	integer	4	nummer van het fiche
keyno	integer	4	nummer van de sleutel

Secondary indices: none

"account" (data die wordt opgenomen t.b.v. statistische verwerking)

Storage structure: heap

column name	type	length	comment
dat	date		datum van het genereren van de uitvoer
catid	c	3	catalogus waarvan uitvoer is gemaakt
type	c	2	type van de catalogus waarvan uitvoer is gemaakt
dest	c	3	destination soort uitvoer (papier/ micro-fiche / ...)
unumber	integer	4	aantal documenten
enumber	integer	4	aantal entries (aantal verschillende fiches)
fnumber	integer	4	aantal fiches

Secondary indices: none

"nodef" (aantal documenten, entries en fiches)

Storage structure: btree

column name	type	length	sequence key	comment
id	c	3	1	catalogus waarvan uitvoer is gemaakt
type	c	2	2	type van de catalogus waarvan uitvoer is gemaakt
unumber	integer	4		aantal documenten
enumber	integer	4		aantal entries (aantal verschillende fiches)
fnumber	integer	4		aantal fiches

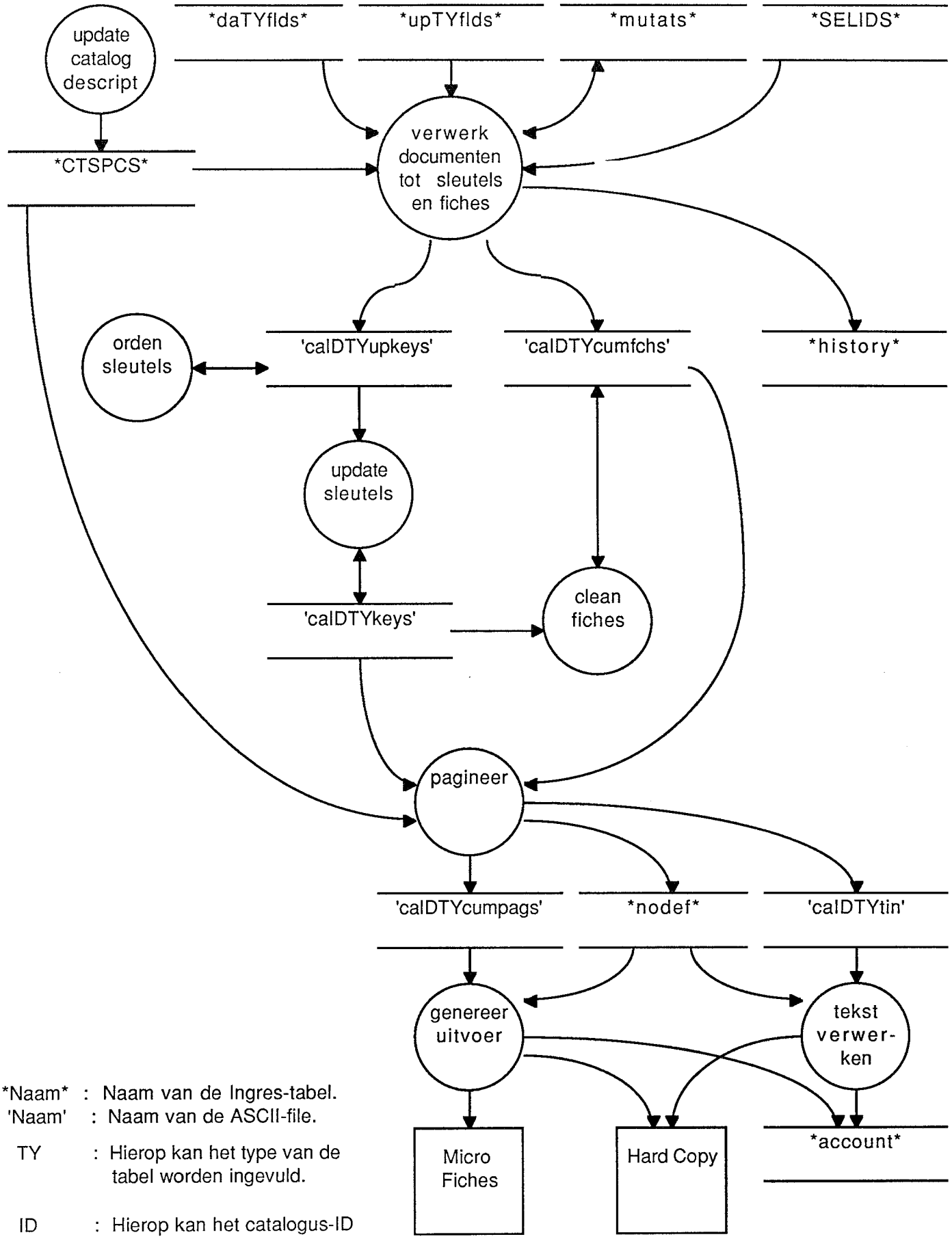
Secondary indices: none

Intern Bibliotheek Systeem (IBS)

Catalogus Module

Bijlage A

De systeem-schets



Naam : Naam van de Ingres-tabel.
'Naam' : Naam van de ASCII-file.

TY : Hierop kan het type van de tabel worden ingevuld.

ID : Hierop kan het catalogus-ID van het bestand worden ingevuld.

Module: catalogus2 vs1
Drawn By: Judith van Rijt
Revision Date: Thu, Jun 29, 1989 - 2:16 PM

```
ect * from dabkflds where id = 'M00057J';

|      |fld   |sqno  |frno  |value                                                                                    |
|------|------|------|------|-----------------------------------------------------------------------------------------|
0057J |anr   |     1|     1|Reg. 1(1894) - 80(1973)                                                                  |
0057J |awc   |     1|     1|2                                                                                        |
0057J |brm   |     1|     1|PA.09.073.35                                                                             |
0057J |ca    |     1|     1|#Mathematical Association of America                                                     |
0057J |cau   |     1|     1|ca:1:#MAA:#Mathematical Association of America                                           |
0057J |cdn   |     1|     1|AMMYA                                                                                    |
0057J |hld   |     1|     1|000:23(1916)2; 39(1932)10; 46(1939)-...                                                  |
0057J |hld   |     2|     1|011:1(1894) - 40(1933); 61(1954)-...                                                     |
0057J |hld   |     3|     1|020:77(1970)-...                                                                         |
0057J |hld   |     4|     1|021:59(1952)-...                                                                         |
0057J |hld   |     5|     1|041:29(1922)4-...                                                                        |
0057J |hld   |     6|     1|051:76(1969)-...                                                                         |
0057J |hld   |     7|     1|061:18(1911) - 22(1915), 25(1918) - 32(1925), 34(1927) - 40(1933),                       |
0057J |hld   |     7|     2|42(1935) - 44(1937), 46(1939) - 54(1947), 56(1949)-... (Ontbreekt:                      |
0057J |hld   |     7|     3|vol. 20, no. 5.)                                                                         |
0057J |hld   |     8|     1|171:72(1975)-...                                                                         |
0057J |hld   |     9|     1|201:86(1979)-...                                                                         |
0057J |hld   |    10|     1|231:28(1921); 30(1923) - 37(1930); 51(1944); 66(1959)-... (Ontbreekt:                    |
0057J |hld   |    10|     2|51(1944)7.)                                                                              |
0057J |hld   |    11|     1|601:47(1940)-...                                                                         |
0057J |hld   |    12|     1|761:71(1964)-...                                                                         |
0057J |hld   |    13|     1|831:74(1967)-...                                                                         |
0057J |ish   |     1|     1|0002-9890                                                                                |
0057J |la    |     1|     1|eng                                                                                      |
0057J |puc   |     1|     1|M.A.A.                                                                                   |
0057J |put   |     1|     1|Washington                                                                               |
0057J |rid   |     1|     1|T32804-Z                                                                                 |
0057J |sgn   |     1|     1|000:205C4                                                                                |
0057J |sgn   |     2|     1|000:204B                                                                                 |
0057J |sgn   |     3|     1|011:                                                                                     |
0057J |sgn   |     4|     1|020:                                                                                     |
0057J |sgn   |     5|     1|021:                                                                                     |
0057J |sgn   |     6|     1|041:                                                                                     |
0057J |sgn   |     7|     1|051:                                                                                     |
0057J |sgn   |     8|     1|061:                                                                                     |
0057J |sgn   |     9|     1|171:                                                                                     |
0057J |sgn   |    10|     1|201:                                                                                     |
0057J |sgn   |    11|     1|231:                                                                                     |
0057J |sgn   |    12|     1|601:                                                                                     |
0057J |sgn   |    13|     1|761:                                                                                     |
0057J |sgn   |    14|     1|831:                                                                                     |
0057J |ti    |     1|     1|#American mathematical monthly                                                           |
0057J |tis   |     1|     1|the official journal of the Mathematical Association of America                          |
0057J |ttl   |     1|     1|ti:1:#Herbert Elsworth Slaught memorial papers:#American mathematical                    |
0057J |ttl   |     1|     2|monthly                                                                                  |
-----------------------------------------------------------------------------------------------------------------|
tinue
```

Intern Bibliotheek Systeem (IBS)

Catalogus Module

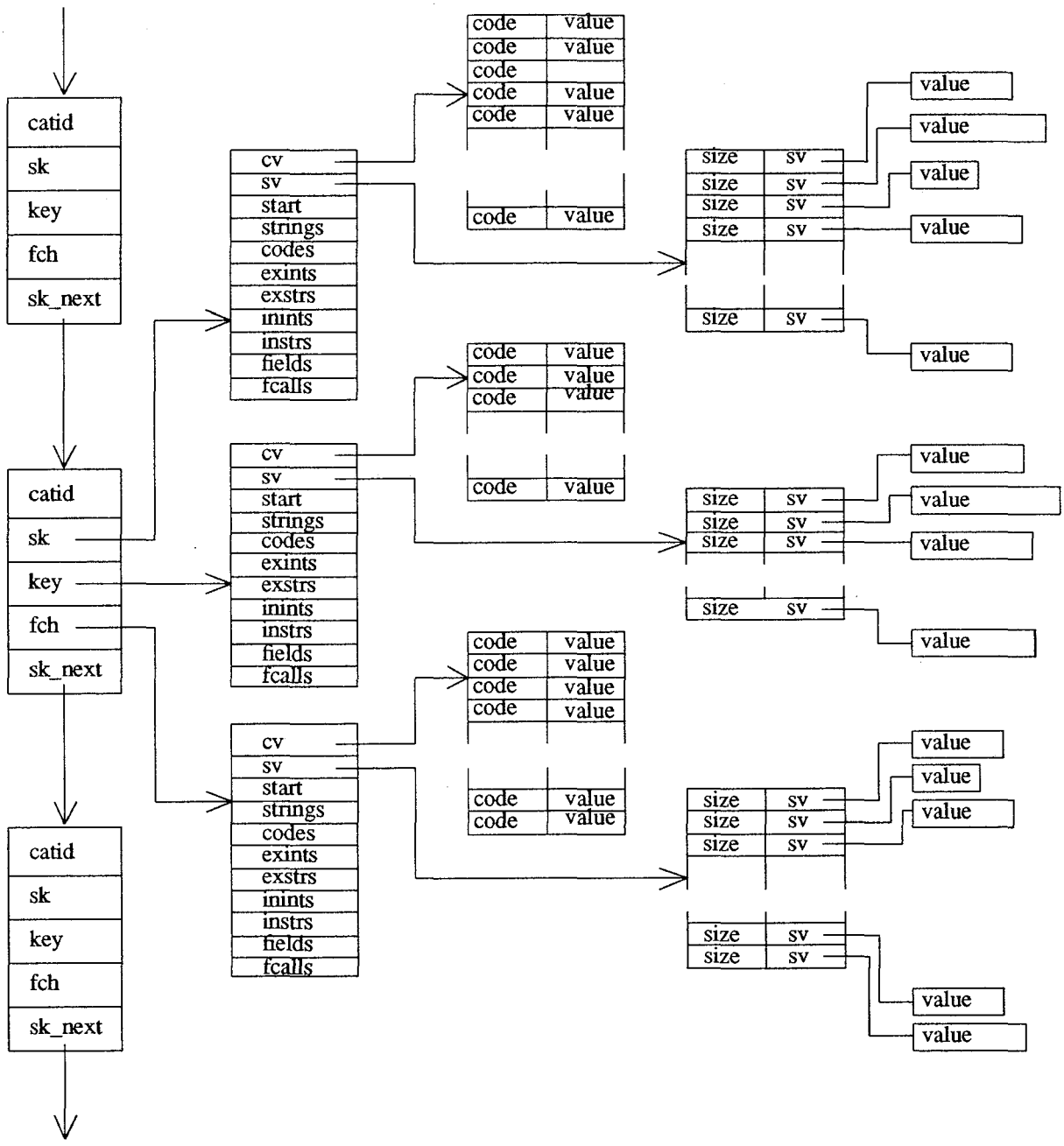
Bijlage C

De structuren van de catalogus-specificaties
en van de documenten

Na het inlezen van de catalogus-specificaties in de procedure 'creeer sleutel en fiche' staan deze in de volgende structuur:

```
struct skf {                        /* selectie, sleutel en fiche specificaties          */
        char catid[4];              /* id van de catalogus                               */
        struct section *sk;         /* pointer naar de selectie-criteria                 */
        struct section *key;        /* pointer naar de sleutel-specificaties             */
        struct section *fch;        /* pointer naar de fiche-specificaties               */
        struct skf *sk_next;        /* pointer naar de volgende catalogus-specificatie   */
};

struct section {
        struct codval *cv;          /* pointer naar het array met codes                  */
        struct strcnst *sv;         /* pointer naar het array met strings                */
        int start;                  /* start adres voor de evaluatie                     */
        int strings;                /* lengte van het strval-array                       */
        int codes;                  /* lengte van het codval-array                       */
        int exints;                 /* aantal integer-parameters dat aan de functie eval wordt meegegeven*/
        int exstrs;                 /* aantal string-parameters dat aan de functie eval wordt meegegeven*/
        int inints;                 /* aantal integer-variabelen die binnen de functie eval worden gebruikt*/
        int instrs;                 /* aantal string-variabelen die binnen de functie eval worden gebruikt*/
        int fields;                 /* aantal verschillende velden van een bibliografie  */

        int fcalls;                 /* aantal functie-declaraties in de BibTeX-procedure */
};

struct codval {
        int code;                   /* soort waarde van value                            */
        int value;
};

struct strcnst {
        int size;                   /* lengte van de string waar sv naar wijst           */
        char *sv;
};
```

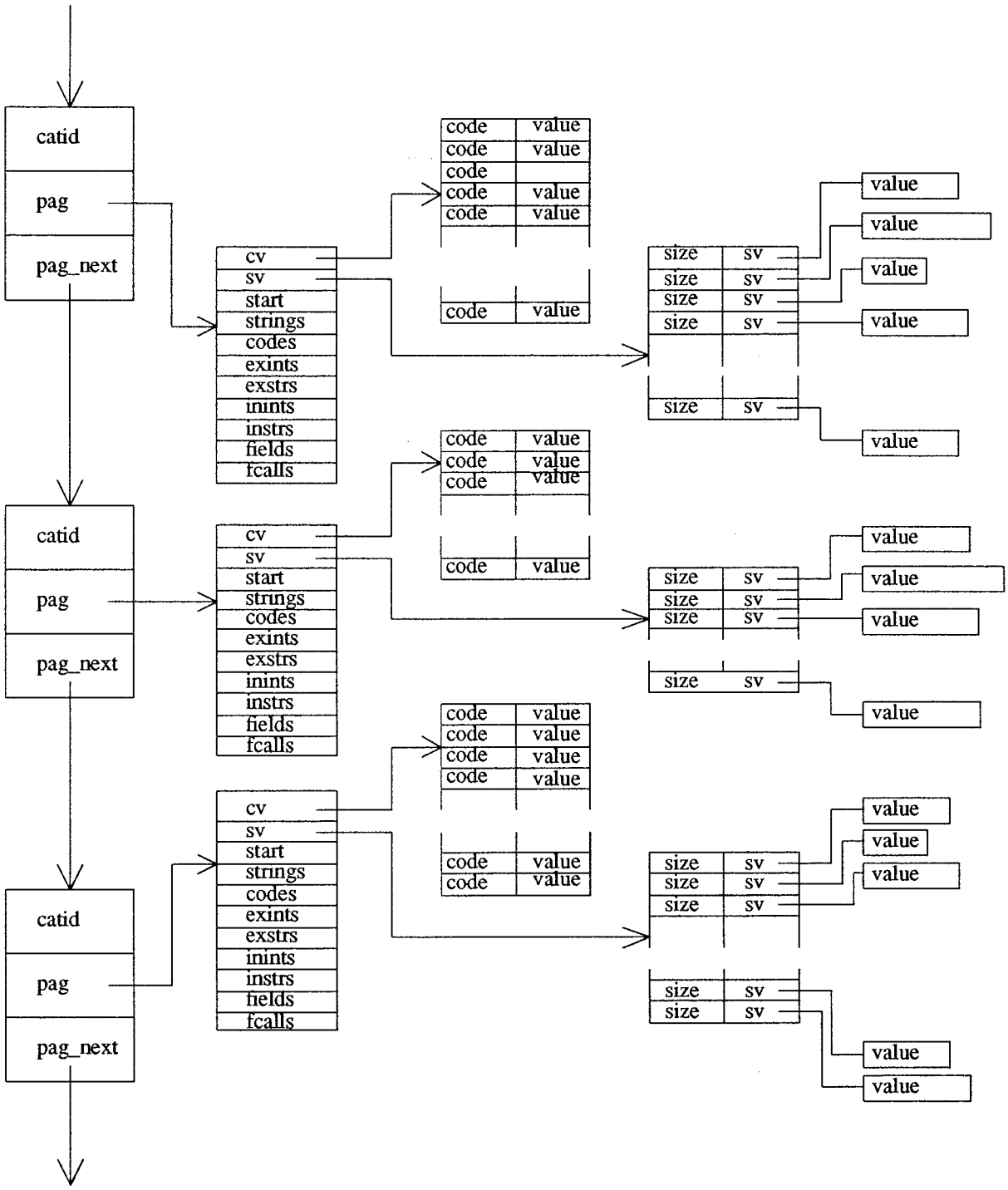
Voor de procedure 'creeer sleutel en fiche' zal dat er als volgt uitzien:


catid
sk
key
fch
sk_next
cv
sv
start
strings
codes
exints
exstrs
inints
instrs
fields
fcalls
code
value
size
sv
value

Voor de procedure 'pagineer' zal de structuur van de catalogus-specificaties er als volgt uitzien:

```
struct paging {                 /* pagineer specificaties                                   */
        char catid[4];          /* id van de catalogus                                      */
        struct section *pag;    /* pointer naar de pagineer-specificaties                   */
        struct paging *pag_next;/* pointer naar de volgende pagineer-specificatie           */
};

struct section {
        struct codval *cv;      /* pointer naar het array met codes                         */
        struct strcnst *sv;     /* pointer naar het array met strings                       */
        int start;              /* start adres voor de evaluatie                            */
        int strings;            /* lengte van het strval-array                              */
        int codes;              /* lengte van het codval-array                              */
        int exints;             /* aantal integer-parameters dat aan de functie eval wordt meegegeven*/
        int exstrs;             /* aantal string-parameters dat aan de functie eval wordt meegegeven*/
        int inints;             /* aantal integer-variabelen die binnen de functie eval worden gebruikt*/
        int instrs;             /* aantal string-variabelen die binnen de functie eval worden gebruikt*/
        int fields;             /* aantal verschillende velden van een bibliografie         */
        int fcalls;             /* aantal functie-declaraties in de BibTeX-procedure        */
};

struct codval {
        int code;               /* soort waarde van value                                   */
        int value;
};

struct strcnst {
        int size;               /* lengte van de string waar sv naar wijst                  */
        char *sv;
};
```

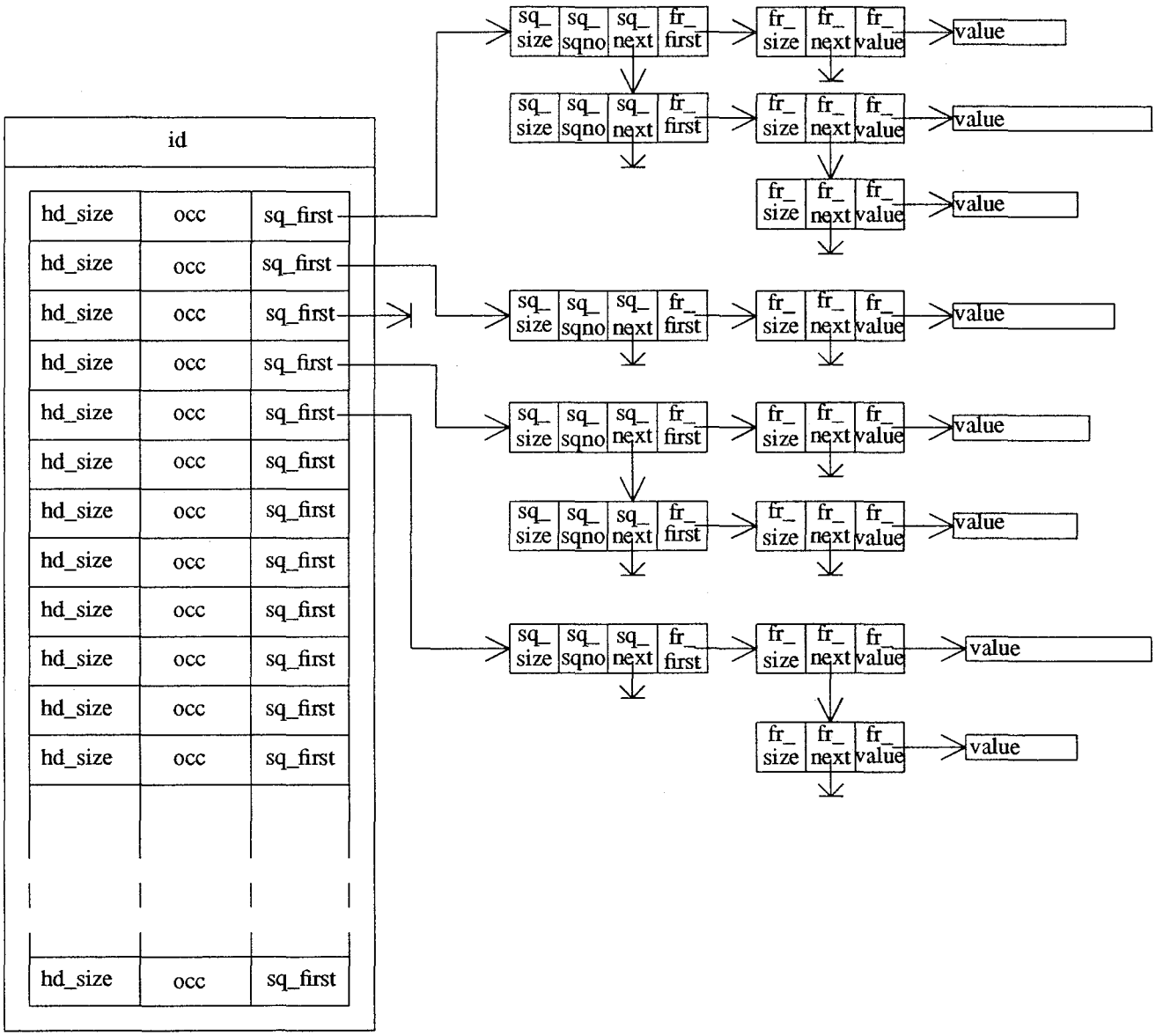
Dit zal er als volgt uitzien:


catid
pag
pag_next
cv
sv
start
strings
codes
exints
exstrs
inints
instrs
fields
fcalls
code
value
size
sv
value

Na het inlezen van een document zal deze in de volgende structuur staan:

```
#define MAXFLD 35                       /* Het maximum aantal velden kan verandert worden   */

struct frfld {
        short fr_size;                  /* lengte van het veld waar fr_value naar wijst     */
        char *fr_value;                 /* value uit daTYflds of upTYflds                   */
        struct frfld *fr_next;          /* verwijzing naar het volgende frno                */
};

struct sqfld {
        short sq_size;                  /* lengte van alle value's die via fr_first te bereiken zijn  */
        short sq_sqno;                  /* het sqno zelf                                    */
        struct frfld *fr_first;         /* verwijzing naar het eerste frno                  */
        struct sqfld *sq_next;          /* verwijzing naar het volgende sqno                */
};

struct fldheader {
        short hd_size;                  /* lengte van alle value's die via sq_first te bereiken zijn  */
        short occ;                      /* aantal sqno's bij dit veld                       */
        struct sqfld *sq_first;         /* verwijzing naar het eerste sqno                  */
};

struct doc {                            /* begin van het document                           */
        char id[9];                     /* volledig id van het document                     */
        struct fldheader hd[MAXFLD];    /* array van field-headers                          */
};
```

Dat zal er als volgt uizien:


id
hd_size
occ
sq_first
sq_ size
sq_ sqno
sq_ next
fr_ first
fr_ size
fr_ next
fr_ value
value

Intern Bibliotheek Systeem (IBS)

Catalogus Module

Bijlage D

Pseudo-code van het catalogus-systeem

```
 (argc, argv) /* check parameters */

ruct spec spec;
ar ctspcs[13];
ar cid[9];
ar tablenm[13];

itch (eerste optie) {
case '-c':
     do {
         get_one_ctspcs (ctspcs, cid, &spec);
     } while ( geen -c optie meer aanwezig );
case '-n':
     while ( -n optie ) {
       if ( volgende = -c optie ) {
          do {
              get_one_ctspcs (ctspcs, cid, &spec);
          } while ( volgende optie = -c optie);
       } else
            getctspcs (ctspcs, &spec);
     }
     switch (volgende optie) {
       case '-t':
            do {
                 insdaflds (&spec, &tablenm);
            } while ( -t optie );
            break;
       case '-u':
            do {
                  insselids (&spec, &tablenm);
            } while ( -u optie );
            break;
       default  :
           geef foutmelding;
           exit;
     }
     break;
case '-p':
     if (volgende optie = -m) {
         getctspcs (ctspcs, &spec);
         mutmutats (&spec);
     } else {
         geef foutmelding;
         exit;
     }
     break;
default :
     geef foutmelding;
     exit;




tspcs (prodid, spec)

rrent_spec = (zoek) laatste specificatie;
ile ( er is een catalogus-id in de productie-set ) {
 get_one_spec (current_spec, catalogus_id);
 current_spec = current_spec->sk_next;




one_ctspcs (productie_set, catalogus_id)

f (catalogus_id is een element van de productie_set ||
       productie_set is leeg en het catalogus_id is geen element van een
       andere productie_set ) {
     current_spec = (zoek) laatste specificatie;
     get_one_spec (current_spec, catalogus_id);

   } else {
      foutmelding;
      exit;
   }
}


get_one_spec (current_spec, catalogus_id)
{
   getsectspec (catalogus_id , "sel", current_spec->sk);
   getsectspec (catalogus_id , "key", current_spec->key);
   getsectspec (catalogus_id , "fch", current_spec->fch);
}

getsectspec (catalogus_id, sektid, sekt) /* get sektie specificatie */
{
  vul sekt-> met gegevens uit "caspcatt";
  for (i = 0; i <= codes; i++) {
     sect->cv[i].code := code uit "caspccod";
     sect->cv[i].value := value uit "caspccod";
  }
  aantal = 0;
  do {
     if (frno uit "caspcstr" > 1) {
       zet value uit "caspcstr" achter de vorige value in struct;
       tel de size van de toegevoegde value bij de oude op;
     } else {
        aantal ++;
        sekt->sv[aantal].sv := pointer naar value;
     }
  } while aantal < strings uit sekt->;
}


mutmutats (spec)
{
   lees 2 rijen uit "mutats";
   if (id en type van deze rijen zijn niet gelijk) {
      if (state v.d. 1e rij is 'u' (undone) ) {
         switch (code) {
           case 'i':
                mutgetdoc (document, document_id, tablename);
                zet_state_p;
                ckfch(&spec, &doc, sec, code);
                zet_history();
                zet_state_d;
                break;
           case 'u':
                mutgetdoc (document, document_id, tablename);
                copydoc (document, new_document);
                mutnewdoc (new_document, document_id, tablename);
                zet_state_p;
                ckfch(&spec, &doc, sec, code);
                zet_history();
                ckfch(&spec, &doc, sec, code);
                zet_history();
                zet_state_d;
                break;
           case 'd':
                mutgetdoc (document, document_id, tablename);
                zet_state_p;
```

```
            ckfch(&spec, &doc, sec, code);
            zet_history();
            zet_state_d;
            break;
       default :
            foutmelding;
      }
      lees nieuwe rij in;
   }
  else {
   controleer of er niet nog meer rijen zijn met dit id en type;
   controleer of de status van de eerste rij een 'd' (done) is;
   controleer of de code en which van de rijen zijn toegestaan;
   if (bij controle is gebleken dat er iets verkeert was) {
      geef foutmelding;
      lees totdat er twee goede rijen ingelezen zijn;
   }
   if (status van de tweede rij = 'u' (undone) ) {
     lees de oude en/of de nieuwe documnenten in uit de tabellen die
     which aangeeft;
     zet_state_p;
     ckfch(&spec, &doc, sec, code);
     zet_history();  /* eval en zet_history eventueel herhalen */
     zet_state_d;
     lees 2 nieuwe rijen in;
   }



afldS ( spec, tablename )

nitialiseer document;
orige document_id = MAXID;
hile (er is een rij) {
   lees een rij in uit "daTYflds";
   if (document_id == vorige document_id) {
      zet deze rij in het document;
      lees volgende rij in;
   } else {
       if (document_id != MAXID) {
          zet_state_p;
          ckfch(&spec, &doc, sec, code);
          zet_history();
          zet_state_d;
          free_document;
          initialiseer document;
          zet de rij in het document;
       }
       zet document_id in struct;
       zet deze rij in het document;
   }

f (er was tenminste 1 document in de tabel) {
   zet_state_p;
   ckfch(&spec, &doc, sec, code);
   zet_history();
   zet_state_d;
   free_document;



elids (spec, tablename)
```

```
   while (lees rij uit "SELIDS") {
     mutgetdoc (document, id uit "SELIDS", "daTYflds" waarbij TY uit SELIDS komt):
     zet_state_p;
     ckfch(&spec, &doc, sec, code);
     zet_history();
     zet_state_d;
     free_document;
   }
}


enum stktyp {
   numb,     /* int number                                                    */
   strg,     /* char *string                                                  */
   numba,    /* index van interne int                                         */
   strga,    /* pointer naar struct btstrg met interne strings                */
   btint,    /* pointer naar struct btint met externe integers                */
   btstrg,   /* pointer naar struct btstrg met externe strings                */
   func,     /* functie-call                                                  */
   sfunc,    /* system functie-call (onder meer bibtex built-in functies)     */
   field,    /* nummer van het veld v.h. document                             */
   fmember   /* een pointer naar de occurence v.h. doc. die we willen hebben */
};

struct btstr {
   int lalloc; /* totale gealloceerde lengte                  */
   int lused;  /* de lengte van de string waar str naar wijst */
   char *str;  /* pointer naar de string                      */
   struct btstr *nxt;
}

struct btint {
   int intgr;
   struct btint *nxt;
}

struct stkelm {
   enum stktyp tp;
   int dtl;
   union {
      int numb;
      char *strg;
      struct btstr *bts;
      int numba;
      int strga;
      int xstr;
      int func;
      int sfunc;
      int field;
      struct sqfld *fmember;
   } v;
};

ckfch (spec, doc, sec, mutcode)
struct spec *spec;
struct doc *doc;
int sec;
char mutcode;
{
   eval (spec->sk, &doc, extints, in_catalogus);
   if (document in catalogus) {
      eval (&spec->key, &doc, extints, overige_sleutel_gegevens);
      if (mutcode == 'i') {
          eval (&spec->fch, &doc, extints, begin_en_eind_adressen);
      }
      zet sleutel in "caIDTYupkeys";
```

```
( cat_sct, doc, exti, exts)
truct sektion *cat_sct;
truct docum *doc;
truct btint *exti;
truct btstr *exts;

truct strcnst *sv;
truct codval *cv;
truct stkelm stk[MAXSTACK];

nt sp;
nt esp;
nt eip;
nt isp;
nt iip;

v= cat_sct->cv;
v= cat_sct->sv;
valfnc(cat_sct->start);


fnc( ip)
nt ip;

nt eofnc= FALSE;

hile (!eofnc) {
  switch(cv[ip].code) {

  case FCALL:
     evalfnc( cv[ip].value);
     break;
  case SFCALL:
     evalsysf( cv[ip].value);
     break;
  case RETURN:
     eofnc= TRUE;
     break;
  /* hier komen ook nog de andere code evaluatie functies */
  default:
     /* error melding */
     ;
  }
  ip++;



sysf(value)
nt value;

witch(value) {
  case IF:
     if (stk[sp].tp==sfunc)
        evalsysf(stk[sp].v.numb);
     else
        evalfnc( stk[sp].v.numb == 0 ? cv[sp-1].value : cv[sp-2].value);
     break;
 /* hier komen ook de ander BibTeX built_in en soortgelijke functies */
  default:
     ;
```

```
e keys (id, ty)
id[]; /* catalogus-id van de bestanden */
ty[]; /* type van de bestanden */

t fupkey;  /* pointer naar file "caIDTYupkeys"            */
t fkey;    /* pointer naar file "caIDTYkeys"              */
t fkeytmp; /* pointer naar file "keys.tmp"                */
t ffch;    /* pointer naar file "caIDTYcumfchs"           */
t first;   /* wel of niet stoppen als upkey1 == maxsleutel */
t stop;    /* wel of niet stoppen met update keys         */
rteer bestand "caIDTYupkeys" met behulp van het sort-commando;
     /* sorteer op overige sleutel-gegevens, document-id, fchno, keyno,
                  seconds en mutatie-code                              */
en de bestanden "caIDTYupkeys", "caIDTYkeys" en "caIDTYcumfchs";

es upkey1 uit "caIDTYupkeys"; /* indien geen aanwezig: stop */
es upkey2 uit "caIDTYupkeys"; /* indien geen aanwezig: maak maxsleutel */

rst = TRUE;
op = minkeys (&upkey1, &upkey2, &fupkey, &fkey, &ffch, first);
 (stop == TRUE)
 return;
rst = FALSE;
eeer "keys.tmp";
es key uit "caIDTYkeys"; /* indien geen aanwezig: maak maxsleutel */
ile (upkey1 < maxsleutel AND key < maxsleutel) {
   if (upkey1 == key AND upkey1 heeft mutatie-code == 'd') {
        lees key uit "caIDTYkeys";
                                /* indien geen aanwezig: maak maxsleutel */
        upkey1 := upkey2;
        lees upkey2 uit "caIDTYupkeys";
                                /* indien geen aanwezig: maak maxsleutel */

        stop = minkeys (&upkey1, &upkey2, &fupkey, &fkey, &ffch, first);

   } else {
       if (upkey1 < key AND upkey1 heeft mutatie-code == 'i') {
            zet upkey1 zonder mutatie-code en seconds in "keys.tmp";
            upkey1 := upkey2;
            lees upkey2 uit "caIDTYupkeys";
                                /* indien geen aanwezig: maak maxsleutel */

            stop = minkeys (&upkey1, &upkey2, &fupkey, &fkey, &ffch, first);

       } else {
           if (upkey1 > key) {
               zet key in "keys.tmp";
               lees volgende key uit "caIDTYkeys";
                               /* indien niet aanwezig maak maxsleutel */
           } else
               geef foutmelding;
       }
   }

uit de bestanden "caIDTYkeys", "caIDTYupkeys", "caIDTYcumfchs" en
                 "keys.tmp";
ve "keys.tmp" naar "caIDTYkeys";
move "caIDTYupkeys";


nimaliseer het aantal sleutels. Als er een sleutel is die toegevoegd
et worden en daarna weer verwijdert, negeer dan beide sleutels.

ys (key1, key2, fupkey, fkey, ffch, first)
*key1[];
*key2[];
int *fupkey;
int *fkey;
int *ffch;
int first;
{
   while (document-id van key1 == document-id van key2 AND
          fchno van key1 == fchno van key2 AND keyno van key1 == keyno van key2
          AND mutatie-code van key1 == 'i' AND mutatie-code van key2 == 'd' )  {
     lees key1 uit fkey; /* indien geen aanwezig: maak maxsleutel */
     if (key1 == "\0") {
        if (first == true) {
           sluit de bestanden "caIDTYkeys", "caIDTYupkeys" en "caIDTYcumfchs";
           remove "caIDTYupkeys";
           return (TRUE);
        } else
            key1 = maxsleutel;
     }
     lees key2 uit fkey;
     if (key2 == "\0")
        key2 = maxsleutel;

   }
   return (FALSE);
}
```

```
cleanfchs (id, ty)
char id[]; /* id van de bestanden */
char ty[]; /* type van de bestanden */
{
   while (niet einde "caIDTYkeys") {
     lees sleutel uit "caIDTYkeys";
     if (keyno van de sleutel == 1) {
         bepaal nieuw beginadres van het fiche;
         copieer fiche naar "cumfch.tmp";
         bepaal nieuw eindadres van het fiche;
         wijzig begin- en eind-adres van de sleutel;
     }
     zet sleutel in "keys.tmp";
   }

   sorteer "keys.tmp" op document-id, fchno en keyno met behulp van sort;
   while (niet einde "keys.tmp) {
     lees sleutel uit "keys.tmp;
     if (keyno van de sleutel == 1) {
        bewaar begin- en eind-adres;
     } else
        zet bewaarde begin- en eind-adres in de sleutel;
     zet sleutel in "keys2.tmp";
   }

   sorteer "keys2.tmp" op overige sleutel-gegevens, document-id, fchno, keyno;
   move "cumfchs.tmp" naar "caIDTYcumfchs";
   move "keys2.tmp" naar "caIDTYkeys";
   remove "keys.tmp";
   remove "cumfchs.tmp";
   remove "keys2.tmp";
}
```