


# SQL Server en Oracle

adlib
an AXIELL product

# Axiell ALM Netherlands BV

# Inhoud

# 1 Databaseplatformen vergeleken

Vanaf Adlib 6.0 kunt u naast het Adlib-eigen .CBF-formaat, ook SQL Server en Oracle-databases benaderen via Adlib. Uw Adlib-databases kunt u daartoe laten converteren in het andere formaat.

Ofschoon het Adlib CBF-databaseformaat uitstekend geschikt is voor onze toepassingen, zijn er ook voordelen in het gebruik van SQL Server of Oracle. Dat zijn bijvoorbeeld:

- Het formaat wordt wereldwijd ondersteund.
- Vanwege de client/server-implementatie is er snellere toegang voor meerdere gebruikers tegelijk mogelijk.
- In SQL Server en Oracle-databases zijn uw gegevens zeer veilig. Zo kunnen deze typen databases niet geopend en gelezen worden vanuit het Windows bestandssysteem.
- SQL Server en Oracle-databases kunnen ook niet verwijderd worden als een gebruiker daar niet toe gemachtigd is.
- Ook heeft SQL Server/Oracle replicatievoordelen, want u kunt zo'n database zodanig instellen dat elke wijziging die een gebruiker in een record invoert en opslaat automatisch wordt gekopieerd naar een backup van de database elders.

SQL Server en Oracle zijn zogenaamde relationele databases. Een relationele database bestaat uit een verzameling tabellen met daarin meestal een (zeer) beperkt aantal veldkolommen, waarin de gebruiker kan zoeken naar gegevens, of op veel verschillende manieren data kan herschikken en samenstellen in zogenaamde views of rapporten, zonder dat die tabellen daarvoor hoeven te worden gereorganiseerd. De structuur van zo'n database is dus betrekkelijk eenvoudig. Maar dat heeft ook nadelen:

- De occurrences (veldherhalingen) bijvoorbeeld, die in een Adlib CBF-bestand veelvuldig voorkomen, hebben geen gelijkwaardig equivalent in zelfs een minimaal genormaliseerde relationele database. (Normalisatie is het efficiënt organiseren van een relationele database door een aantal richtlijnen te volgen, zoals het niet laten voorkomen van redundante data in tabellen, door die tabellen op te splitsen. Voor een relationele database bespaart dit schijfruimte en zorgt ervoor dat data logisch opgeslagen wordt. Merk trouwens op dat Adlib CBF-bestanden praktisch geen redundantie kennen, en dus standaard al een minimum aan schijfruimte benodigen.) De normalisatie die gewoonlijk op relationele databases wordt toegepast, kan voor gecompliceerd gekoppelde data met veldher-

halingen zoals in museum- of archiefdatabases, leiden tot honderden of duizenden tabellen. En al die tabellen moeten tijdens runtime weer met elkaar in verband worden gebracht, wat veel geheugenruimte en processortijd kost.

- Een ander nadeel is dat er een zeer expliciete koppeling bestaat tussen een relationele database en de programma's die er gebruik van maken. Dit betekent in de praktijk dat veranderingen in een tabelstructuur meestal ook wijzigingen en hercompilatie van de software betekenen, terwijl dat voor Adlib-databases niet nodig is: voor die laatste kunt u de structuur van een database aanpassen, eventueel de grafische interface aanpassen, en de nieuwe applicatie direct starten.

Het samengaan van deze twee technologieën heeft geleid tot de mogelijkheid om vanaf Adlib 6.0 ook (Adlib-specifieke) relationele databases te benaderen. We willen immers grotendeels dezelfde software, dezelfde grafische schil en dezelfde onderhoudsprogramma's (Adlib Designer) kunnen blijven gebruiken, terwijl we onze gebruikers wel de voordelen maar niet de nadelen van SQL Server of Oracle willen bieden.
Vandaar dat ervoor is gekozen bestaande Adlib databases (gevuld of leeg) eenmalig te converteren naar een relationele database als de klant van deze functionaliteit gebruik wil maken. (Merk op dat er aan het gebruik hiervan en ondersteuning ervoor, extra kosten verbonden zijn. Informeer vrijblijvend bij onze verkoopafdeling.)
Na die conversie zijn de Adlib-databases omgezet naar evenzovele tabellen in één relationele database. Elk indexbestand, woordenlijst, pointerfile, lockbestand en cnt-bestand heeft bovendien een eigen tabel gekregen (de laatstgenoemde drie vanaf 6.1.0). Pointerfiles worden bij de omzetting naar het nieuwe formaat overigens in XML gestructureerd, wat een grotere uitwisselbaarheid, en betere uitleesbaarheid op tabelniveau mogelijk maakt. Een heel record in een CBF-bestand zal als XML-document in één tabelcel zijn opgenomen, vergezeld van het recordnummer in een andere kolom. Dit XML-document was in SQL Server 2000 hexadecimaal gecodeerd (als zogenaamde BLOB: Binary Large OBject) omdat er in een tabelcel alleen afdrukbare tekens mochten voorkomen. Vanaf SQL Server 2005 (en na een eventuele nieuwe conversie van uw database voor compatibiliteit met Adlib 6.5.0) is dat verleden tijd en wordt elk record als leesbaar XML-document in de (Adlib SQL of Adlib Oracle) database bewaard, en wordt in database-serverbeheerprogramma's (zoals Microsoft SQL Server Management Studio Express) ook als zodanig getoond, wat zo'n database voor beheerders een stuk inzichtelijker maakt.

Met SQL (Structured Query Language, de standaard interface voor relationele databases) en/of de SQL Server Enterprise Manager of SQL

Server Management Studio Express, kunt u zowel indextabellen en woordlijsten doorzoeken (bijvoorbeeld voor integriteitcontroles en eventuele reparaties), als de records zelf (zolang de XML-documenten niet hexadecimaal gecodeerd zijn). We raden u wel sterk af op dit niveau wijzigingen in data aan te brengen. Normaal gesproken hebt u dus een Adlib-applicatie en -software nodig om uw nieuwe SQL Server of Oracle-database te kunnen bewerken, doorzoeken en vullen.

# 2 SQL bestandsanalyse

Na installatie, setup en databaseconversie (zie de *Installatiegids Museum, Bibliotheek en Archief*), kunt u via onder andere Microsofts SQL Server Management Studio Express of SQL Server Enterprise Manager, een Adlib SQL Server-database eventueel openen, en via de Oracle Enterprise Manager Console een Adlib Oracle-database, om de opbouw of inhoud ervan te bekijken – op deze manier wijzigingen aanbrengen wordt aan gebruikers in het algemeen overigens afgeraden. (Voor reparatiewerkzaamheden aan bestanden door Axiell-medewerkers kan van deze richtlijn worden afgeweken.)
In SQL Server Management Studio Express bijvoorbeeld, opent u een tabel door uw database-map te openen, te klikken op de submap *Tables*, en in het linker of rechter deelvenster te rechtsklikken op de gewenste tabel, en in het snelmenu *Open table* te kiezen: alle rijen worden opgehaald. In SQL Server Management Studio (2008) rechtsklikt u op een tabel en kiest u bijvoorbeeld *Select top 1000 rows* om de eerste duizend rijen op te halen.
Rechtsklik in Microsofts SQL Server Management Studio (Express) op de map van uw database in het linker deelvenster en kies *New query* in het snelmenu om een SQL-zoekvraag voor die database op te kunnen geven, of klik voor een geselecteerde database op de knop *New query* in de werkbalk. (In de gele balk onder in het venster kunt u altijd zien voor welke database u momenteel een zoekvraag uitvoert.) Een in het middelste deelvenster opgegeven zoekvraag kunt u uitvoeren door op de knop *Execute* te klikken. Via SQL kunt u de structuur en inhoud van elke tabel onderzoeken.

> SQL (Structured Query Language) is enigszins vergelijkbaar met de Adlib zoektaal. Zo betekent:
> `SELECT * FROM collect` bijvoorbeeld dat u alle records wilt ophalen uit de tabel *collect*. En `SELECT count(*) FROM thesau_term` bijvoorbeeld, telt het aantal termen in de *term*-index van de thesaurus.

In een Adlib SQL Server-database kunt u aan de naam van een tabel zien wat de inhoud is. Een enkelvoudige naam zoals *collect*, is een voormalige Adlib-database. Een dubbele naam, in de vorm *databasenaam_indexnaam*, geeft een individuele index aan. Als de tweede helft van zo'n dubbele naam letterlijk gelijk is aan of begint met "*pointerfiles*", dan betreft het een tabel waarin de pointerfiles voor de betreffende SQL hoofdtabel (ex-Adlib-database) worden opgeslagen.

Merk op dat SQL-query's op tabellen die voormalige Adlib-databases bevatten complex kunnen zijn, omdat elk heel record in één eigen veld is gestopt. U kunt eenvoudiger zoeken in tabellen van Adlib in-

dexen, al bevatten veel indexen alleen maar recordnummers (voor geïndexeerde linkreference-velden). U moet dus onder andere weten hoe Adlib koppelingen opslaat en verwerkt, en wat het verschil tussen termindexen en de wordlist is bijvoorbeeld, om de inhoud van en het verband tussen de verschillende SQL Server-tabellen te kunnen begrijpen. Zie de Designer Help en/of het *Adlib software functionality profile*-document voor alle informatie over deze mechanismen.

## 2.1 Geheel record in tabelcel

Van een geopende databasetabel wordt de inhoud van een record dus in één veld bewaard en weergegeven. De recordinhoud is hierin gestructureerd in XML. De priref, aanmaakdatum en modificatiedatum staan in aparte kolommen, maar zijn ook attributen van het XML *<record>*-element (dat laatste is handig bij eventuele export van XML-records), namelijk als volgt:

```
<record priref="nnnn" creation="jjjj-mm-ddTuu:mm:ss" modification="jjjj-mm-ddTuu:mm:ss">

   <field tag="tt" occ="nn">data voor veld</field>
   <field>.....</field>
    ....

</record>
```

Vanaf 6.6.0 kan per veld ook de aanmaak- en wijzigingsdatum en -tijd en de naam van de huidige gebruiker automatisch als metadata in een gewijzigd record opgeslagen, als de database daarvoor via Adlib Designer is ingesteld via de optie *Store modification history*. Om precies te zijn: per gewijzigde veldoccurrence per datataal wordt de genoemde metadata als attributen van de veldnode in de XML opgeslagen. Voorbeeld:

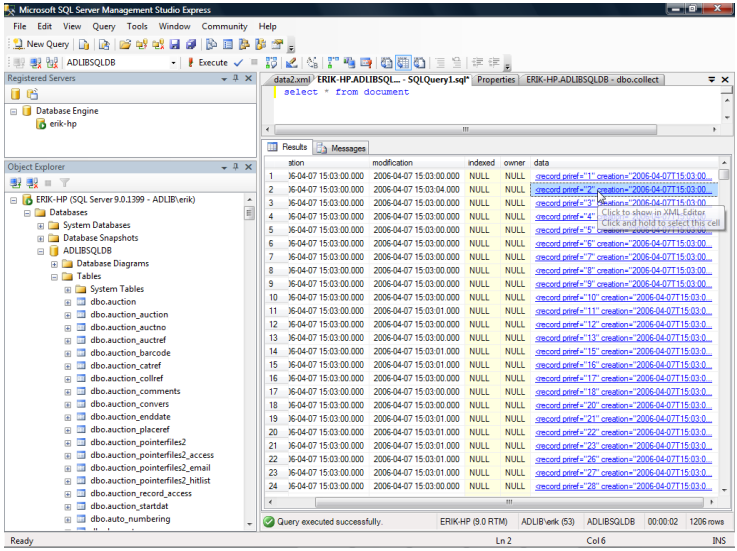
```
<field tag="T9" occ="2" cd="2011-04-11T11:26:51" cu="erik" md="2011-04-11T11:26:51" mu="erik">Bronski House journey back to Poland</field>
```

De attributen hebben de volgende betekenis: `cd` staat voor aanmaakdatum (creation date), `cu` betekent gebruiker voor aanmaak (creation user), `md` wijzigingsdatum (modification date) en `mu` is de gebruiker voor wijziging (modification user).

In uw adlwin.exe-applicatie kunt u deze metadata niet tonen, maar via Adlib Designer kunt u wel indexen op de metadata creëren en er zoekingangen voor definiëren zodat u bijvoorbeeld kunt zoeken naar records met velden die ná een bepaalde datum zijn gewijzigd en/of door een bepaalde gebruiker.

## Records ophalen

Haal bijvoorbeeld alle rijen uit de *document*-tabel op, via: `select * from document`. Het resultaat wordt in bijvoorbeeld SQL Server Management Studio Express als volgt gepresenteerd:




Elke rij bevat één record. De *priref* kolom bevat het recordnummer, *creation* de aanmaakdatum en *modification* de laatste wijzigdatum van het record. De *data*-kolom bevat de records in hun geheel. De data is direct leesbaar in SQL Server 2005 of hoger wanneer u met Adlib 6.5.0 of hoger werkt.

> In SQL Server 2000 zijn records hier hexadecimaal (aangegeven door 0x) gecodeerd. (Merk op dat in de SQL Server Enterprise Manager de *data*-kolom alleen vermeldt dat de inhoud binair is, en niet de daadwerkelijke inhoud toont zoals hier in SQL Server Management Studio Express.) In principe kunt u de hele hexadecimale string (programmatisch of via een handige conversiewebsite als Translator, Binary) naar ASCII omzetten. Elke twee alfanumerieke tekens hierin representeren één ASCII-teken. *3C* bijvoorbeeld is 60 decimaal, en het 60ste ASCII-teken is "<", het begin van een XML-tag. En hex *72* bijvoorbeeld vertaalt naar "r". Elk XML-record begint dus met *<record* (hexadecimaal: 3c 72 65 63 6f 72 64)

Mocht u uw Adlib-applicaties niet als interface voor deze data willen gebruiken omdat u uw eigen applicatie wilt bouwen, dan kunt u elk opgehaald record als XML verder verwerken. Om bepaalde records te

kunnen vínden, moet u wel gebruik maken van de indextabellen die er voor veel velden bestaan.

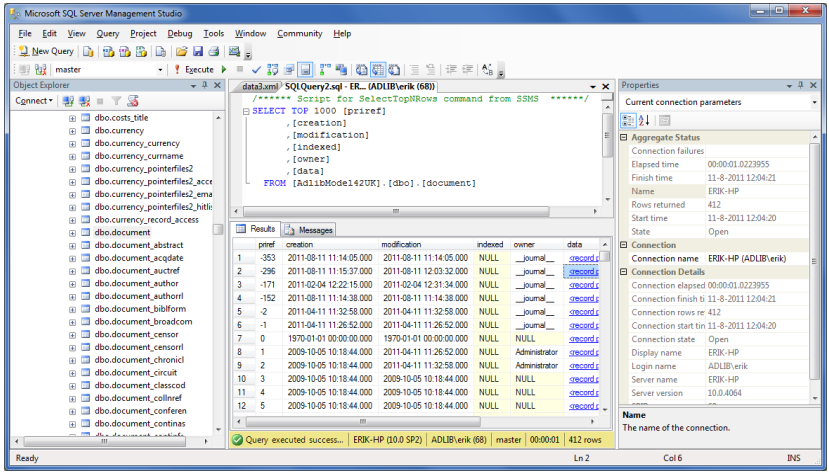
## 2.2 Wijzigingen in records bijhouden

Vanaf 6.6.0 kunt u Adlib alle opgeslagen wijzigingen in records laten bijhouden, als u een Adlib SQL Server of Adlib Oracle-database gebruikt. Zo kan de gebruiker achteraf gemakkelijk bekijken wie, wanneer, welke wijzigingen heeft ingevoerd, of kunt u bij foutieve veranderingen aan de hand van de wijzigingshistorie de oorspronkelijke data opsporen en herstellen.

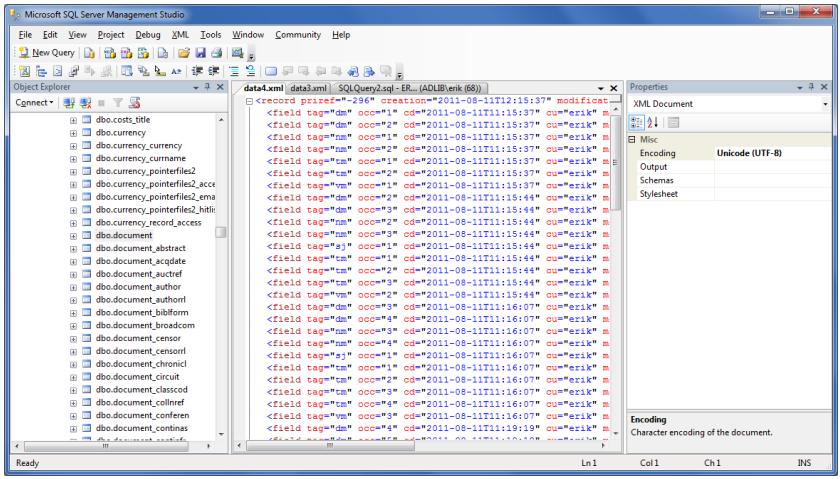
Nadat deze functionaliteit voor een Adlib-database is ingesteld (zie de Designer Help voor meer informatie), wordt er voor elk record dat een gebruiker heeft gewijzigd, in de betreffende Adlib databasetabel in de Adlib SQL Server of Adlib Oracle-database een extra record aangemaakt met de negatieve variant van het recordnummer: de eerste wijziging in bijvoorbeeld record 171 veroorzaakt dus de aanmaak van een record met nummer -171. In dit "negatieve" record worden alle* wijzigingen in het record voortaan bijgehouden: elke datawijziging komt in dit record in een nieuwe veldoccurrence te staan. Merk op dat uw database hierdoor aanzienlijk groter zal worden.
* Alleen wijzigingen in gekoppelde velden en eventueel daarbij opgehaalde velden worden niet gelogd.

### Het historierecord in XML bekijken

U kunt het historierecord zelf niet in uw adlwin.exe-applicatie bekijken maar eventueel wel in uw SQL Server managementsoftware.



In dit voorbeeld ziet u verschillende negatieve records met als owner "_journal_". Klik op de onderstreepte data om deze in een apart tabblad te openen. U ziet de opgeslagen XML in dit historierecord.




## 2.3 Indexen

In Adlib SQL (en Adlib Oracle) zijn indexen in aparte tabellen opgeslagen. Een indextabel heeft (vanaf Adlib 6.5.0) maximaal acht kolommen. Per indexsleutel is er één rij in zo'n tabel. De kolommen zijn:

- *Term* bevat de sleutel waarop u kunt zoeken (al kan het *term*veld ook gekoppeld-recordnummers bevatten). De geïndexeerde waarden in deze kolom zijn zonder eventuele accenten en geheel in kleine letters opgeslagen.
- *Display term* bevat de geïndexeerde term zoals die door de gebruiker is ingevoerd, dus inclusief accenten en eventuele hoofdletters.
- *Priref* bevat het recordnummer van het record waaruit de geïndexeerde term afkomstig is.
- *Domain* bevat de naam van het domein waartoe een term eventueel behoort. (Vanaf Adlib 6.5.0 komt het domein en de dubbele dubbelepunt niet meer in de *Term*-kolom voor.)
- *Strippedterm* bevat de geïndexeerde term maar dan zonder eventuele leestekens zoals koppelingstekens, aanhalingstekens en komma's.
- *Dmp_primary (primary key)* kan de primaire fonetische code bevatten, zoals bepaald door het DoubleMetaphone-algoritme.

- *Dmp_secondary (secondary key)* kan de secundaire fonetische code bevatten, zoals bepaald door het DoubleMetaphone-algoritme.
- *Language* bevat voor meertalige veldinhoud de taal waarin een indexwaarde is ingevoerd.
  In meertalige Adlib SQL en Oracle-databases werden tot en met Adlib 6.3.0 alle vertalingen van meertalige veldinhoud geïndexeerd zonder taalaanduiding waardoor het nog niet mogelijk was taalspecifiek te zoeken naar een term. Vanaf 6.5.0 bevatten term- en woordindexen deze extra kolom die zoeken in een specifieke datataal dus mogelijk maakt.

Hiermee kunnen SQL-indexen dus tot zes keer zo groot zijn als in Adlib 6.3.0. Maar het maakt sneller zoeken mogelijk omdat Adlib direct de juiste indexkolom kan gebruiken zonder daar bewerkingen op te hoeven loslaten.
In de toekomst zult u bij het zoeken in een term-index expliciet kunnen aangeven hoe u wilt zoeken op de door u ingevoerde term, bijvoorbeeld in de *Stripped*-vorm, of fonetisch of in een bepaalde taal. U kunt in 6.5.0 wel al via de selectietaal zoeken in een bepaalde taal, en fonetisch zoeken via de *Zoekassistent*, maar de overige zoekmogelijkheden zijn nog niet geïmplementeerd.

## 2.4 Pointerfiles

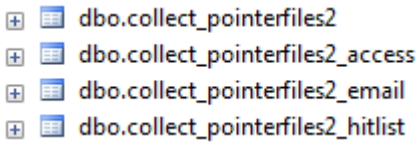
Voor Adlib 6.3.0 of ouder is er per databasetabel een <*database-tabel*>*.pointerfiles*-tabel. Elke pointerfile daarin wordt beschreven in een eigen rij van de tabel. De beschikbare kolommen zijn: *owner* (maker van de pointerfile), *title*, *number*, *selectionstatement* (de zoekvraag of selectie), *hitcount* (aantal resultaten), *modification* (datum van laatste bijwerken van pointerfileresultaten), en *data* (bevat deze pointerfile als XML-document).

In tests met zeer grote pointerfiles (met meer dan 100.000 verwijzingen naar records) bleek dat de 6.3.0-implementatie van pointerfiles onvoldoende presteerde in de SQL-gebaseerde versies van adlwin (dit geldt zowel voor de Microsoft SQL Server-implementatie als de Oracle-implementatie). Het openen van een pointerfile met meer dan 100.000 verwijzingen kon gemakkelijk meer dan een uur verwerkingstijd vragen. De implementatie in het Adlib-eigen databaseformaat presteerde daarentegen veel beter (30 seconden voor dezelfde pointerfile).

Om dit probleem in 6.5.0 op te lossen is het pointerfileformaat in de SQL-implementatie van Adlib radicaal gewijzigd, en de algoritmes voor het lezen en schrijven van pointerfiles zijn herschreven. Na de vereiste conversieprocedure bij de overstap naar Adlib 6.5.0 of hoger, zijn de wijzigingen in uw database doorgevoerd; voor gebruikers maakt de wijziging geen verschil.

## 2.5 De 6.5.0-pointerfilestructuur

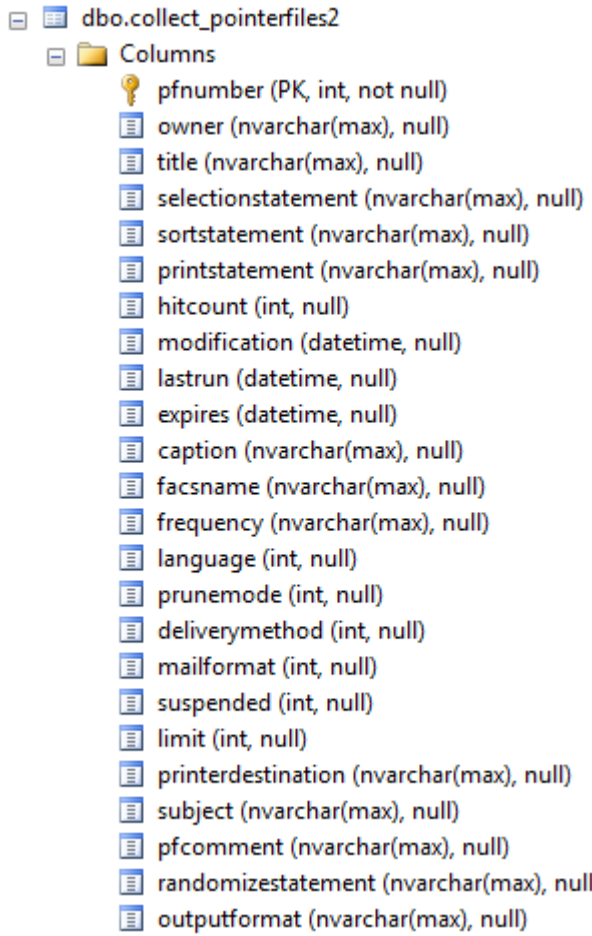
In de 6.5.0-pointerfilestructuur wordt elke pointerfile opgeslagen in vier afzonderlijke tabellen: een tabel voor de pointerfile header, een tabel voor alle recordverwijzingen (hitlist) in de pointerfile, een tabel voor de e-mailadressen voor een mogelijke SDI-component van de pointerfile en een tabel voor de toegangsrechten van de pointerfile. Alle vier de tabellen hebben dezelfde basisnaam die bestaat uit de naam van de Adlib-database met het achtervoegsel "pointerfiles2":



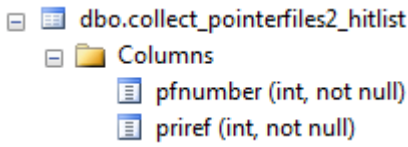

Het achtervoegsel *pointerfiles2* is bewust anders gekozen om het onderscheid tussen "oud"-formaat pointerfiles en "nieuw"-formaat pointerfiles aan te geven. Het gevolg is dat "oude" pointerfiles in adlwinversies ouder dan 6.5.0 zichtbaar zullen zijn, terwijl pointerfiles in het nieuwe formaat alleen in 6.5.0 en hoger zichtbaar zullen zijn.

De headertabelstructuur is als in de afbeelding hieronder.

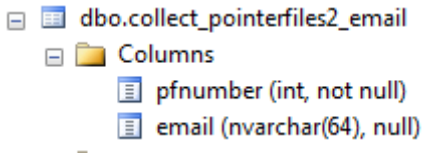
Merk op dat alle data-elementen van een pointerfile header (inclusief SDI-data) in afzonderlijke kolommen worden uitdrukt. Merk ook op dat tekstkolommen van onbepaalde lengte de `nvarchar(max)`-optie van SQL Server 2005 gebruiken, hetgeen tot gevolg heeft dat Adlib 6.5.0 niet langer op SQL Server 2000 kan worden gebruikt.




De nieuwe hitlijsttabelstructuur is als volgt:

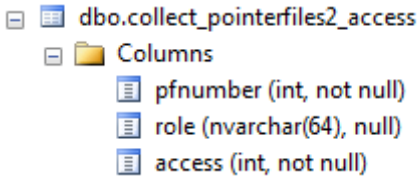



De hitlijsttabelstructuur is eenvoudig, en bestaat uit rijen met slechts twee nummers: het pointerfilenummer en de priref van het record in de pointerfile.
De andere twee tabellen zijn ook zo eenvoudig. De pointerfile e-maillijsttabelstructuur is als volgt:




(De e-mailtabel is een afzonderlijke tabel omdat elke pointerfile meerdere SDI e-mailbestemmingen kan hebben.)

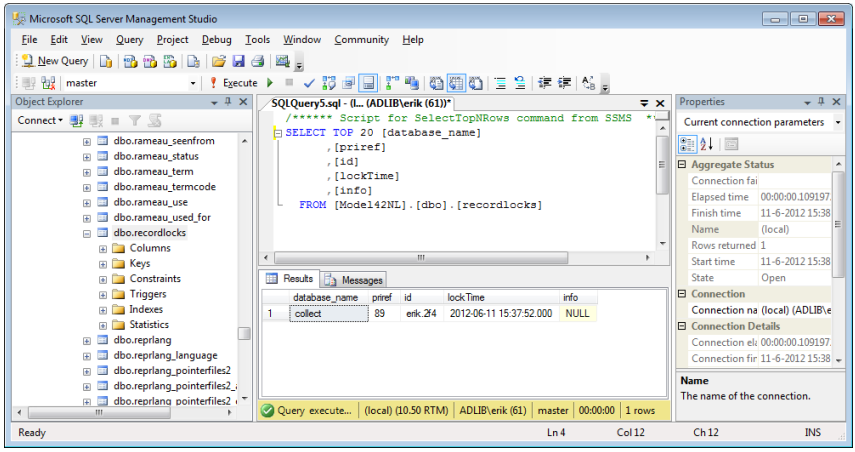
De toegangsrechtentabelstructuur is als volgt:




De *access*-kolom in deze pointerfiletabel kan de volgende waarden bevatten: 0 (niet-gedefinieerde toegangsrechten), 1 (geen toegangsrechten), 2 (alleen-lezen toegang), 3 (schrijven toegang), 4 (volledige toegang).

## 2.6 Recordlocks

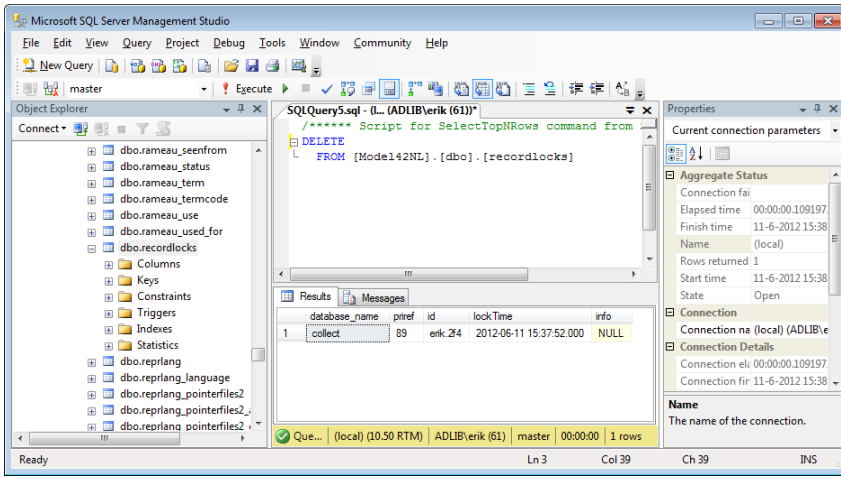
Voor de gehele SQL-database is er één *recordlocks*-tabel. Elk rij in deze tabel beschrijft één recordlock: de naam van de databasetabel waarop het lock van toepassing is, het recordnummer van het gelockte record, de lock-id en het tijdstip van locken. Haal alle rijen uit deze tabel op om te zien welke locks op dit moment bestaan.



Recordlocks bestaan normaal slechts tijdelijk zolang een record bewerkt wordt, om te voorkomen dat iemand anders tegelijkertijd het record gaat bewerken. Maar bij een computercrash kunnen een of meer recordlocks achterblijven in de database, waarna die records door niemand meer bewerkt kunnen worden. Gebruik dan bij voorkeur Adlib Designer om op gebruiksvriendelijke manier de overgebleven recordlocks te verwijderen. U kunt dat echter ook in MS SQL Server Management Studio doen, in de *recordlocks*-tabel. In de geopende *recordlocks*-tabel kunt u alle recordlocks tegelijk verwijderen via de volgende SQL-query:

```
DELETE FROM [<mijn-database>].[dbo].[recordlocks]
```

Vervang `<mijn-database>` door de eigenlijke naam van uw database, zonder scherpe haken. Verzeker u ervan dat momenteel niemand in Adlib werkt als u dit statement uitvoert.




Als er momenteel mensen in de database aan het werk zijn, en u wilt daarom slechts één specifiek overgebleven lock verwijderen, dan doet u dat via de statement:

```
DELETE FROM [<mijn-database>].[dbo].[recordlocks] WHERE
priref = <recordnummer>
```

Vervang `<recordnummer>` door het daadwerkelijke recordnummer, zonder scherpe haken.

## 2.7 Specifieke rechten per record in aparte tabel

Specifieke toegangsrechten per record, die voor een database via de *Authorisation type*-optie *Rights* mogelijk zijn gemaakt (zie de Designer Help voor meer informatie over dit type bescherming van data), worden vanaf 6.5.0 in een aparte tabel opgeslagen. De nieuwe tabel heeft

drie kolommen: *priref* (herhaalbaar), *role* en *rights*.
Het voordeel is dat Adlib niet meer een record hoeft te lezen om te beoordelen of de huidige gebruiker de juiste toegangsrechten voor de bewerking heeft, wat dus een prestatieverbetering van Adlib bewerkstelligt.

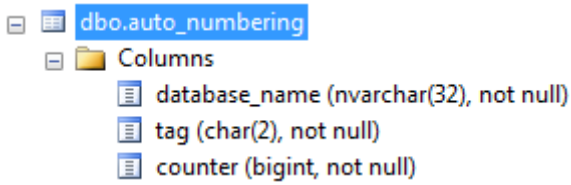
## 2.8 ISO-datumtabellen

De 6.5.0-structuur van ISO-datumindexen bevat de door de gebruiker ingevulde datums (volledig of onvolledig) in een kolom genaamd *displayterm*, en een kolom *term* die altijd automatisch complete ISO-datums zal bevatten.

Met 6.5.0 is het datatype van de *term*-kolom gewijzigd van een term-index (SQL) of nvarchar2-index (Oracle) naar een float resp. number-index om ook bijzonder grote jaartallen (verder dan - of + 9999-01-01) in uw data mogelijk te maken.

## 2.9 Een enkele tabel voor tellers

In uw Adlib SQL of Adlib Oracle-database is er een aparte tabel, genaamd *dbo.auto_numbering*, die alle tellers voor alle tabellen bevat, met zowel automatisch genummerde velden als de in 6.6.0 geïntroduceerde benoemde tellers (zie de ADAPL-functie `GetCounter`).




Er staan drie kolommen in deze tabel:

- De kolom *database_name* bevat de namen van benoemde tellers en Adlib-databasenamen voor automatisch genummerde velden. Omdat de benoemde tellers opslagen worden in een bestaande databasetabelstructuur, wordt de naam van een benoemde teller opgeslagen in de kolom met de niet geheel toepasselijke naam *database_name*. Het voordeel van het gebruik van de bestaande structuur is wel dat u geen databaseconversie nodig hebt om benoemde tellers toe te passen.
- De kolom *tag* bevat veldtags van automatisch genummerde velden.
- De kolom *counter* bevat de laatst toegekende tellerwaarden.

## 2.10 Voorbeelden

Stel u zoekt alle records in de *document* tabel waar in het auteur-veld de naam `Fruithof, Th.` voorkomt. In Adlib worden records van namen van personen en instellingen in het bestand *people* opgeslagen. De namen zelf staan in het veld *name*. Hiervan is een index beschikbaar zodat Adlib snel naar namen kan zoeken, hier de tabel *people_name*. Vanuit een record in *document* wordt naar dit veld gekoppeld via een linkreferentie: alleen het recordnummer van het gekoppeld naamrecord wordt dus in een documentatierecord bewaard. U moet dus eerst in *people_name* het recordnummer van `Fruithof, Th.` opzoeken. Omdat in deze index domeinen voorkomen – domeinen komen gewoonlijk voor in *thesau* en *people* en hun indexen – en u naar de auteur `Fruithof, Th.` zoekt, moet u ongeveer de volgende SQL-statement uitvoeren: `select * from people_name where term like 'fruit%' and domain='author'`. In onze voorbeelddatabase wordt één record gevonden, voor de betreffende auteur, met recordnummer 20. Dit recordnummer moet dus voorkomen in de index van het veld in *document* waarin auteurs worden bewaard: *document_author;* daarin komt immers niet de gezocht naam zelf voor maar alleen het recordnummer van het naamrecord, hier 20. Met de volgende zoekvraag komt u erachter in welk documentatierecord de naamrecord 20 (auteur Fruithof) voorkomt: `select * from document_author where term = '20'`. In ons voorbeeld blijkt dat toevallig record 1 te zijn, in *document* dus. Dit record kunt u bijvoorbeeld opvragen via: `select * from document where priref = '1'`.

Met meer geavanceerde SQL-statements kunt u deze stappen ook allemaal ineens uitvoeren (of complexere zoekvragen opgeven), bijvoorbeeld:

```
SELECT document.priref, document.data
FROM document
INNER JOIN document_author ON document.priref = docu-
ment_author.priref
INNER JOIN people_name ON document_author.term = peo-
ple_name.priref
WHERE people_name.term like 'fruit%'
and domain='author'
and (document.priref >= 0 and document.priref <= 500)
order by document.priref
```

Via de innerjoins worden de sleutelvelden aangegeven waarmee de tabellen, waarin moet worden gezocht, in Adlib aan elkaar gekoppeld zijn. Daarnaast worden hier ook de prirefgrenzen van de te doorzoeken dataset aangegeven en een sorteringscriterium.
Een vergelijkbaar voorbeeld voor zoeken op een bepaalde titel in de-

zelfde dataset (hier *Boeken*), waarbij alleen de recordnummers moeten worden opgehaald, kan er als volgt uitzien:

```
SELECT distinct document.priref
FROM document
INNER JOIN document_title ON document.priref = docu-
ment_title.priref
INNER JOIN wordlist ON document_title.term = word-
list.wordnumber
WHERE wordlist.term like 'w%'
and (document.priref >= 0 and document.priref <= 500)
order by document.priref
```

Enkele andere voorbeelden van handige queries:

- selecteer alle documentrecords waarin een veld voorkomt met occurrence 5: `select * from document where data.exist ('/record/field[@occ="5"]') = 1`
- selecteer alle documentrecords waarin een veld voorkomt met als tag 'ti': `select * from document where data.exist ('/record/field[@tag = "ti"]') = 1`
- selecteer alle documentrecords waarin een veld voorkomt met het attribuut 'creation': `select * from document where data.exist ('/record/field[@creation]') = 1`

**De wordlist gebruiken**

Results | Messages

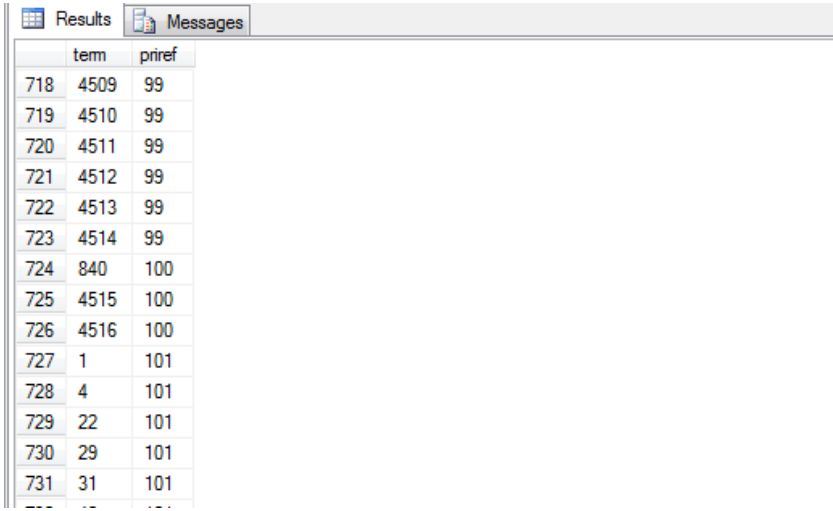
| | term | displayterm | strippedterm | language | dmp_primary | dmp_... | wordnumb |
|---|---|---|---|---|---|---|---|
| 7405 | co. | Co. | co | | K | | 7405 |
| 7406 | horizontale | Horizontale | horizontale | | HRSN | | 7406 |
| 7407 | één | één | een | | N | | 7407 |
| 7408 | cylindermachine | cylindermachine | cylindermachine | | SLNT | | 7408 |
| 7409 | één-cylindermachine | één-cylindermachine | een cylindermachine | | NSLN | | 7409 |
| 7410 | uit | uit | uit | | AT | | 7410 |
| 7411 | 1906 | 1906 | 1906 | | | | 7411 |
| 7412 | met | met | met | | MT | | 7412 |
| 7413 | een | een | een | | AN | | 7413 |
| 7414 | vermogen | vermogen | vermogen | | FRMJ | FRMK | 7414 |
| 7415 | van | van | van | | FN | | 7415 |
| 7416 | ca | ca | ca | | K | | 7416 |
| 7417 | ca. | ca. | ca | | K | | 7417 |
| 7418 | 250 | 250 | 250 | | | | 7418 |

Vermeldenswaard is hier de speciale index *wordlist*. De *wordlist* bevat alle unieke woorden die in alle op woord (vrije tekst) geïndexeerde velden voorkomen. Dit zijn velden met lange teksten, zoals titels of beschrijvingen. Er is maar één *wordlist* voor alle hoofdtabellen samen.

Welke woorden hier zoal in voorkomen, kunt u eenvoudig onderzoeken via de SQL-statement: `select * from wordlist`.

U ziet dat in deze index elke term een woordnummer heeft. En net als bij gekoppelde velden, wordt in lange-tekstvelden niet de letterlijke tekst bewaard maar alleen de woordnummers, opnieuw om schijfruimte te besparen. U kunt dit zien met de volgende query:

```
select top 1000 term, priref from document_title
```

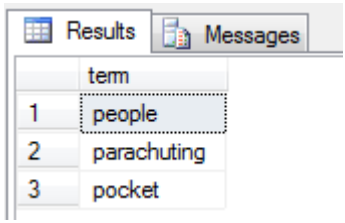


Results | Messages

| | term | priref |
|---|---|---|
| 718 | 4509 | 99 |
| 719 | 4510 | 99 |
| 720 | 4511 | 99 |
| 721 | 4512 | 99 |
| 722 | 4513 | 99 |
| 723 | 4514 | 99 |
| 724 | 840 | 100 |
| 725 | 4515 | 100 |
| 726 | 4516 | 100 |
| 727 | 1 | 101 |
| 728 | 4 | 101 |
| 729 | 22 | 101 |
| 730 | 29 | 101 |
| 731 | 31 | 101 |

Van record 100 bijvoorbeeld, zijn drie woorden geïndexeerd, met de woordnummers 840, 4515 en 4516. Met bijvoorbeeld de volgende query kunt u uit de *wordlist* deze woorden ophalen (de woorden die in de titel-index in de document-database - in SQL zijn dit natuurlijk tabellen - uit record 100 zijn geïndexeerd):

```
select wordlist.term from wordlist inner join docu-
ment_title on wordlist.wordnumber = document_title.term
where document_title.priref=100
```



Results | Messages

| | term |
|---|---|
| 1 | people |
| 2 | parachuting |
| 3 | pocket |

Om alle woorden uit de titel-index voor een set records (99-101) op te halen, en elk woord vergezeld door het recordnummer van het bron-

documentrecord terwijl de lijst wordt gesorteerd op term, gebruikt u de volgende query:

```
select wordlist.term,document_title.priref from wordlist
inner join document_title on wordlist.wordnumber = docu-
ment_title.term where document_title.priref in (99,100,101)
order by term
```

Een andere invalshoek is de volgende: als u dus bijvoorbeeld op zoek bent naar documentatierecords waarvan de titel het woord `jungle` bevat, dan moet u eerst het woordnummer ervan opzoeken, bijvoorbeeld via: `select * from wordlist where wordlist.term = 'jungle'`. In onze database heeft dat woord nummer *49054* gekregen. Nu moet u dit nummer opzoeken in de titel-index van *document*: `select * from document_title where document_title.term = '49054'`. In ons voorbeeld wordt maar één recordnummer gevonden: *655*. Dit record kunnen we vervolgens uit *document* ophalen via: `select * from document where document.priref = '655'`.

U kunt ook getrunceerd zoeken in de *wordlist* via het %-teken, bijvoorbeeld: `select * from wordlist where wordlist.term like '%sing%'`

Results | Messages

| | term | displayterm | strippedterm | language | dmp_primary | dmp_secondary | wordnumber |
|---|---|---|---|---|---|---|---|
| 1 | 'passing | 'passing | passing | | PSNK | | 3362 |
| 2 | advertising | Advertising | advertising | | ATFR | | 4072 |
| 3 | basingstoke | Basingstoke | basingstoke | | PSNK | | 6076 |
| 4 | blessing | blessing | blessing | | PLSN | | 4036 |
| 5 | casing | casing | casing | | KSNK | | 1232 |
| 6 | cleansing | cleansing | cleansing | | KLNS | | 4134 |

In een normale *wordlist* komt elk woord per data-taal (of zonder data-taal) maar één keer voor. Als om wat voor reden de *wordlist* gecorrumpeerd is geraakt, dan kunnen woorden per al dan niet gespecificeerde data-taal meer dan eens voorkomen. Een dergelijke corruptie moet worden gerepareerd. Een van de manieren om dit te doen is handmatig alle duplicaten uit de *wordlist* te verwijderen, en daarna via het hulpprogramma Indexcheck alle woordindexen (d.m.v. een specifieke lijst van alle woordgeïndexeerde tags per Adlib database) te laten herstellen. Op deze manier wordt de *wordlist* alleen gerepareerd op punten waar het nodig is, en hoeft die niet opnieuw te worden opgebouwd. Via de volgende opdracht kunt u alle duplicaten in de *wordlist* vinden:

```
select wordlist.term, count(*) as WordCount from wordlist
group by wordlist.term having COUNT(*) > 1 order by term
```

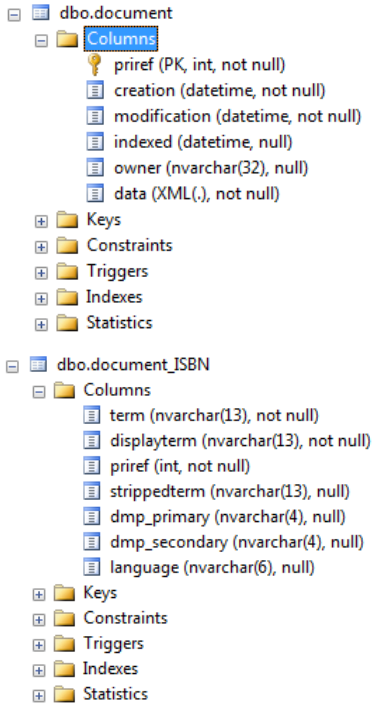
Zolang een term niet meerdere keren voorkomt met dezelfde data-taal, of een term niet meerdere keren voorkomt zonder data-taal, is er niets aan de hand. In het andere geval moet u de overbodige duplicaten verwijderen. Als u bijvoorbeeld het woord met woordnummer 186 wil verwijderen, voer dan de volgende query uit:

```
delete from wordlist where wordnumber = 186
```

Hierna moeten dus tenminste alle woordindexen waarin dat woord (eigenlijk het woordnummer) voorkomt, worden repareerd met Indexcheck. Zorg er wel eerst voor dat niemand in Adlib aan het werk is. Zie eventueel de handleiding *Adcopy, Indexcheck and Linkrefcheck* (niet beschikbaar als download) voor meer informatie over het opsporen en repareren van datacorruptie.

**De kolommen van een tabel**

Merk verder op dat het linker deelvenster van SQL Server Management Studio Express (de lijst met tabellen) ook informatie geeft over de velden (kolommen) die in de betreffende relationele tabel aanwezig zijn, over de PK (primary key) waarop de tabel kan worden gekoppeld, en het variabeletype van de genoemde velden, bijvoorbeeld:



# 3 Opmerkingen

- SQL Server Management Studio kan ook vanaf het netwerk worden gedraaid, door via een remote-desktopconnection verbinding met de SQL-server te maken.
- In *sys…*-tabellen houdt SQL Server gegevens bij voor interne doeleinden. Deze mogen nooit handmatig worden gewijzigd.
- Via het snelmenu (rechtsklikken op uw SQL Server database), kunt u onder andere back-ups maken of plannen, of een recovery uitvoeren via *Restore*.
- Zoekresultaten in SQL Server Management Studio kunt u eventueel in .csv of .txt-formaat opslaan door te rechtsklikken op de resultaattabel en in het snelmenu *Save result as* te kiezen.
- Als u de structuur van een database via Designer wilt aanpassen, maak dan eerst een backup van uw *data*-map (de .*inf*-bestanden) én van uw SQL Server-database (of Oracle-database), en haal dan bijvoorbeeld de Adlib SQL Server-database offline zodat niemand meer records kan wijzigen of invoeren. Nadat u klaar bent met wijzigen, moet u de database weer online brengen.
  In SQL Server Enterprise Manager kunt u dit doen via de opties *Take offline* en *Bring online* in het snelmenu van de database. In SQL Server Management Studio Express zijn deze "database state" opties verborgen, en is alleen de read-only optie *Database state* beschikbaar op het tabblad *Options* van de *Database properties*.
  U kunt een database offline halen via een Transact-SQL statement met bijvoorbeeld de `sp_dboption` procedure. Om bijvoorbeeld de database *AdlibSQL* offline te halen, voert u de volgende statement uit: `sp_dboption 'AdlibSQL','offline','true'`. Voordat u dit uitvoert moet u er wel zeker van zijn dat er geen processen actief zijn die de database in gebruik hebben. U kunt de database weer online maken door de volgende statement uit te voeren: `sp_dboption 'AdlibSQL','offline','false'`.